HP OpenView Performance Agent

User's Manual

Edition: 13

for UNIX

Manufacturing Part Number: None
January 2003

© Copyright 2003 Hewlett-Packard Company.

Legal Notices

Warranty.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices.

©Copyright 1983-2003 Hewlett-Packard Company, all rights reserved.

No part of this document may be copied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this material is subject to change without notice.

Trademark Notices.

Adobe® is a trademark of Adobe Systems Incorporated.

HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

UNIX® is a registered trademark of The Open Group.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Contents

1. This is OpenView Performance Agent

Introduction 22
What OV Performance Agent Does 23
OV Performance Manager Major Components 24
Scopeux Data Collector 25
Collection Parameters File 25
DSI Log Files 26
Extract and Utility Programs 26
Data Sources 26
ARM Transaction Tracking Capabilities 27
Related Performance Products 28

2. Managing Data Collection

Introduction 32
Scopeux Data Collector 33
Scopeux Status 33
parm File 34
Modifying the parm File 34
parm File Parameters 35
Parameter Descriptions 37
Application Definition Parameters 42
Stopping and Restarting Data Collection 49
Stopping Data Collection 49
Restarting Data Collection 50
Automating Scopeux Startup and Shutdown 50
Effective Data Collection Management 51
Controlling Disk Space Used by Log Files 51
Setting Mainttime 51
Setting the Maximum Log File Size 52
Managing Your Resizing Processes 53
Data Archiving 54
Managing Your Archiving Processes 54
Hint 55

3. Using the Utility Program

Contents

Introduction 58
Running the Utility Program 59
Using Interactive Mode 61
  Example of Using Interactive and Batch Mode 61
Utility Command Line Interface 63
  Example of Using the Command Line Interface 65
Utility Scan Report Details 66
Scan Report Information 68
  Initial Values 68
    Initial Parm File Global Information 68
  Initial Parm File Application Definitions 68
  Chronological Detail 69
    parm File Global Change Notifications 69
    parm File Application Addition/Deletion Notifications 69
    Scopeux Off-Time Notifications 70
    Application-Specific Summary Report 70
  Summaries 71
    Process Log Reason Summary 71
    Scan Start and Stop 72
    Application Overall Summary 73
    Collector Coverage Summary 73
    Log File Contents Summary 74
    Log File Empty Space Summary 75

4. Utility Commands

Introduction 78
analyze 81
checkdef 83
detail 84
exit 85
guide 86
help 87
list 88
logfile 90
menu 92
parmfile 93

quit 94
resize 95
scan 101
sh 103
show 104
start 106
stop 108

5. Using the Extract Program

Introduction 112
Running the Extract Program 114
Using Interactive Mode 116
Extract Command Line Interface 117
Overview of the Export Function 124
How to Export Data 124
Sample Export Tasks 125
Generating a Printable CPU Report 126
Producing a Customized Export File 126
Export Data Files 126
Export Template File Syntax 128
Export File Title 132
Creating a Custom Graph or Report 132
Output of Exported Files 134
Notes on ASCII and Datafile Formats 134
Hints 135
Notes on Binary Format 135
Binary Header Record Layout 136
Binary Title Record 140
Binary Item Identification Record 140
Binary Scale Factor Record 141
Special Scale Factors 141
Application Name Record 141
Transaction Name Record 142
Disk Device Name Record 142
Logical Volume Name Record 142
Netif Name Record 143

Contents

6. Extract Commands

Introduction 146
application 153
class. 155
configuration 157
cpu. 158
disk 159
exit 160
export 161
extract. 164
filesystem 167
global. 168
guide 170
help 171
list 172
logfile. 174
lvolume 176
menu 177
monthly. 179
netif. 181
output 182
process 185
quit 187
report 188
sh. 189
shift. 190
show 192
Examples. 192
start. 194
stop 196
transaction 198
weekdays 199
weekly. 200
yearly 203

7. Performance Alarms

Introduction 208
Processing Alarms 209
How Alarms Are Processed 209
Alarm Generator 210
Sending SNMP Traps to Network Node Manager 210
Sending Messages to OpenView Operations (OVO) 210
Executing Local Actions 211
Errors in Processing Alarms 212
Analyzing Historical Data for Alarms 212
Examples of Alarm Information in Historical Data 212
Alarm Definition Components 214
Alarm Syntax Reference 215
Conventions 216
Common Elements 216
ALARM Statement 221
ALERT Statement 226
EXEC Statement 228
PRINT Statement 230
IF Statement 231
LOOP Statement 233
INCLUDE Statement 235
USE Statement 236
VAR Statement 239
ALIAS Statement 240
SYMPTOM Statement 241
Alarm Definition Examples 243
Customizing Alarm Definitions 246

8. Communicating Across a Firewall

Introduction 248
Ports Used for Communication with OVPA 249
With HTTP Proxy 250
Without HTTP Proxy 250
Overview of OVPA Communications Configuration 250
Communications default.txt Configuration File 251
Configure OVPA Server Port 252

Contents

Configure Reporter and/or OVPM 252
Configure Reporter/OVPM with HTTP Proxy 252
Configure Reporter/OVPM without HTTP Proxy 253
Other Considerations 254
About Systems with Multiple IP Addresses 254
default.txt File and Configuration Parameters. 254
default.txt Configuration File 254
default.txt Configuration Parameters 257

A. Appendix

Viewing MPE Log Files 262
Viewing and Printing Documents 263
Viewing Documents on the Web. 264
Adobe Acrobat Files 264

Glossary 265

Contents

Contents

Tables

Table 1. Typographical Conventions 15
Table 2. Changes in this Document Edition 17
Table 2-1. parm File Parameters Used by Scopeux 36
Table 3-1. Command Line Arguments 63
Table 4-1. Utility Commands: Syntax and Parameters 79
Table 4-2. Default Resizing Parameters 97
Table 5-1. Command Line Arguments 117
Table 6-1. Extract Commands: Syntax and Parameters 147
Table 6-2. Extract Commands: Extracting and Exporting Data 151
Table 8-1. OVPA Default Ports 250
Table 8-2. OV Performance Agent Documentation Set 263

Tables

Publication History

New editions are complete revisions of the manual. The publication dates for each edition are listed below. This manual is not printed. It is published in Adobe Portable Document File (.pdf) format and can be viewed online and printed as needed. No part numbers were assigned for Editions 4 and 5 and will not be assigned beginning with Edition 9.

Edition 1	11/95	B4967-90001
Edition 2	12/97	B4967-90016
Edition 3	10/99	B4967-90023
Edition 4	07/00	
Edition 5	12/00	
Edition 6	03/01	B4967-90040
Edition 7	09/01	B4967-90049
Edition 8	09/01	B4967-90052
Edition 9	01/02	
Edition 10	05/02	
Edition 11	09/02	
Edition 12	12/02	
Edition 13	01/03	

Conventions

The following typographical conventions are used in this manual.

Table 1 Typographical Conventions

Font Type	What the Font Type Represents	Example
Italic	Book or manual titles, and man page names	Refer to the *HP OpenView Operations Administrator's Reference Volume I* and the *opc(1M)* manpage for more information.
	Provides emphasis	You *must* follow these steps.
	Specifies a variable that you must supply when entering a command	At the prompt type: `rlogin` *your_name* where you supply your login name.
	Parameters to a function	The *oper_name* parameter returns an integer response.
Bold	New terms	The **monitor agent** observes...
`Computer`	Text and items on the computer screen	The system replies: `Press Enter`
	Command names	Use the `grep` command ...
	Function names	Use the `opc_connect()`function to connect ...
	File and directory names	`/opt/OV/bin/OpC/`
	Process names	Check to see if `opcmona` is running.
	Window/dialog box names	In the `Add Logfile` window...
`Computer Bold`	Text that you must enter	At the prompt, type: **`ls -l`**
Keycap	Keyboard keys	Press **Return**.
`[Button]`	Buttons on the user interface.	Click `[Operator]`. Click on the `[Apply]` button.

Table 1 Typographical Conventions (Continued) (Continued)

Font Type	What the Font Type Represents	Example
`Menu Items`	A menu name followed by a colon (:) means that you select the menu, then the item. When the item is followed by an arrow (->), a cascading menu follows.	Select `Actions:Utilities-> Reports`...

Revisions

The edition number on the title page of this document indicates the edition of this document. The print date on the title page changes each time this document is updated.

You will receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Table 2 **Changes in this Document Edition**

Chapter	Changes
The “Support” and this “Revisions” prefaces	These prefaces were added.
Chapter 1, “This is OpenView Performance Agent,” on page 21	Figure 1-1 on page 24 was updated to incorporate the changed data flow for OVPA4.0 and beyond. A note was also added to describe the changed terminology for some of the functions.
Chapter 2, “Managing Data Collection,” on page 31	Removed the “Starting the Scopeux Collector” subsection from the “Scopeux Data Collector” section.
Chapter 2, “Managing Data Collection,” on page 31	In the section “parm File” on page 34, 1) incorporated the “Sample File” section and information from the *HP OpenView Performance Agent Installation and Configuration Guide*. 2) Removed the “Default Values” subsection and table 2-2 — these values are described in the `parm` file.
Chapter 2, “Managing Data Collection,” on page 31	In Table 2-1 on page 36 and “Parameter Descriptions” on page 37, added the `subprocinterval` parameter.

Table 2 Changes in this Document Edition (Continued)

Chapter	Changes
Chapter 2, “Managing Data Collection,” on page 31	Added the `memory` option description and more information on the `shortlived` option in “Threshold” on page 39.
Chapter 2, “Managing Data Collection,” on page 31	In “Managing Your Archiving Processes” on page 54, removed references to `extract summary`, since summary data can only be exported.
Chapter 5, “Using the Extract Program,” on page 111	Added a note in the “Introduction” on page 112 that summary data can only be exported, not extracted. Any mention of extracted summary data was removed.
Chapter 5, “Using the Extract Program,” on page 111	Added a similar note in Table 5-1 on page 117 above the summary arguments.
Chapter 6, “Extract Commands,” on page 145	Added the same note about summary data in the “Introduction” on page 112.
Chapter 7, “Performance Alarms,” on page 207	1) Incorporated the new terminology throughout the chapter. 2) Removed the “Communicating Alarm Notification to OV Performance Agent” section.
Chapter 8,“Communicating Across a Firewall,” on page 247	This chapter was added — the firewall information was previously in the Install Guide.

Support

Please visit the HP OpenView web site at:

http://openview.hp.com/

There you will find contact information and details about the products, services, and support that HP OpenView offers.

You can go directly to the HP OpenView eCare web site at:

http://support.openview.hp.com/

The eCare support site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

1 This is OpenView Performance Agent

Introduction

This chapter is an introductory overview of OV Performance Agent, its components, and related products. It discusses:

- what OV Performance Agent does
- data sources
- the `scopeux` collector
- the `parm` file
- `utility` and `extract` programs
- related performance products

NOTE

OV Performance Agent (OVPA) for UNIX in this document refers only to version 4.0 and beyond, and OV Performance Manager (OVPM) refers to version 4.0 and beyond for UNIX and Windows platforms. OVPM 3.x (PerfView) will connect to OV Performance Agent 4.0 and beyond for all UNIX platforms except for OVPA for Linux. In the future, connectivity to OVPM 3.x will be discontinued.

What OV Performance Agent Does

OV Performance Agent collects, summarizes, time stamps, and detects alarm conditions on current and historical resource data across your system. It provides performance, resource, and end-to-end transaction response time measurements, and supports network and database measurement information.

Data collected outside OV Performance Agent can be integrated using data source integration (DSI) capabilities. For example, network, database, and your own application data can be brought in through DSI and is treated the same as data collected by OV Performance Agent. All DSI data is logged, time stamped, and can be alarmed on. (For details, see the *HP OpenView Performance Agent for UNIX Data Source Integration Guide*.)

All of the data collected or received by OV Performance Agent can be analyzed using spreadsheet programs, Hewlett-Packard analysis tools such as OV Performance Manager, or third-party analysis products.

The comprehensive data logged by OV Performance Agent allows you to:

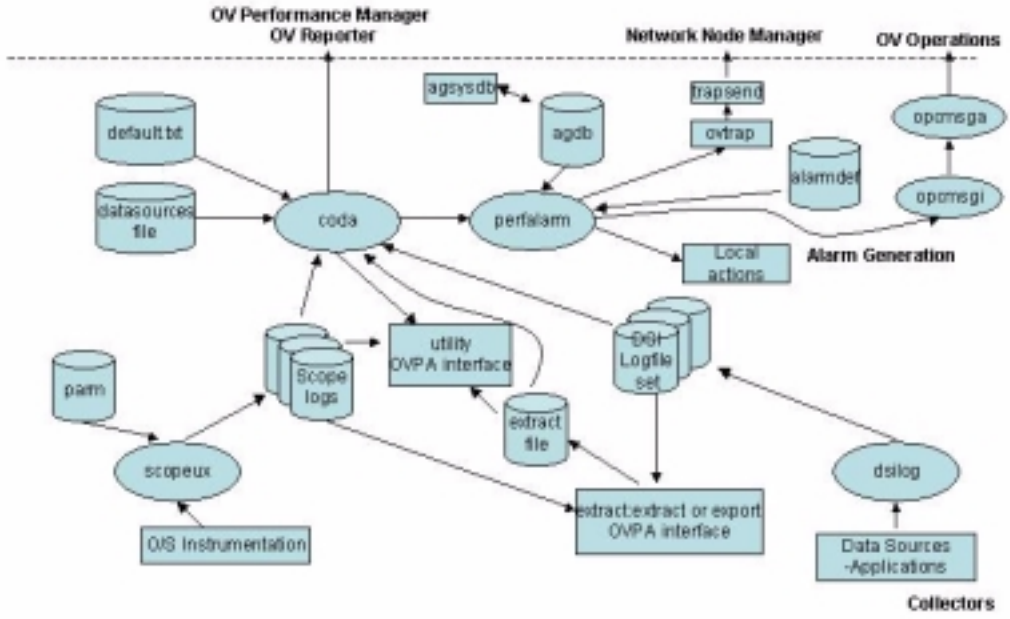
- Characterize the workloads in the environment.
- Analyze resource usage for load balancing.
- Perform trend analysis to isolate and identify bottlenecks.
- Perform service-level management based on transaction response time.
- Perform capacity planning.
- Respond to alarm conditions.
- Solve system management problems before they arise.

OV Performance Agent gathers comprehensive and continuous information on system activity without imposing significant overhead on the system. Its design offers considerable opportunity for customization. You can accept default configurations or set parameters to collect data for specific conditions.

OV Performance Manager Major Components

The following diagram shows the relationships among the major components of the OV Performance Agent system.

Figure 1-1 OVPA Major Components



NOTE

Substantial changes were made to the internal data flow of metric data in OVPA 4.0 and beyond releases. The coda daemon has replaced the function of the perflbd and rep-server daemons, the datasources file has replaced the perflbd.rc file, and the perfalarm daemon has replaced alarmgen. The OVPM 3.x (PerfView) monitor interface has been obsoleted; however, the external functionality is otherwise unchanged.

- The scopeux data collector is described in Chapter 2, “Managing Data Collection,” on page 31.
- The coda daemon and its data sources are described later in this chapter and in the *HP OpenView Performance Agent Installation & Configuration Guide*.
- The default.txt file, which is the communications configuration file used to customize communication parameters for HP OpenView applications, is described in Chapter 8, “Communicating Across a Firewall,” on page 247.
- Alarm generation components are described in Chapter 7, “Performance Alarms,” on page 207.
- Data source integration (DSI), including dsilog and other DSI components, are described in the *HP OpenView Performance Agent for UNIX Data Source Integration Guide*.

Scopeux Data Collector

The scopeux data collector collects performance data from the operating system on which OV Performance Agent resides, summarizes it, and logs it in raw log files, depending on the types of information desired.

For detailed information about scopeux, see Chapter 2, “Managing Data Collection,” on page 31.

Collection Parameters File

The type of data collected is determined by parameters set in the OV Performance Agent programs and in the collection parameters (parm) file, an ASCII file used to customize the collection environment. This file

contains instructions that tell scopeux to log specific performance measurements. The collection parameters file is commonly referred to in this manual as the parm file.

For detailed information about the parm file, see Chapter 2, “Managing Data Collection,” on page 31.

DSI Log Files

DSI log files contain self-describing data that is collected outside of OV Performance Agent. These log files are created by OV Performance Agent's DSI programs. DSI processes and the creation of DSI log files are described in detail in the *HP OpenView Performance Agent for UNIX Data Source Integration Guid*e.

Extract and Utility Programs

Two OV Performance Agent programs, extract and utility, provide the means for managing both scopeux log files and DSI log files.

The extract program lets you extract data from raw or previously extracted scopeux log files and write it to extracted log files. The extracted log files contain selected performance data for specific analysis needs. The extract program also lets you export scopeux and DSI data for use by spreadsheet programs and other analysis products. For more information about extract and extract commands, see Chapter 5, “Using the Extract Program,” on page 111 and Chapter 6, “Extract Commands,” on page 145.

The utility program lets you generate reports on raw and extracted scopeux log files, resize raw scopeux log files, and check parm file syntax. It also lets you check the syntax in your alarm definitions file and analyze alarm conditions in historical scopeux and DSI log file data. For more information about utility and utility commands, see Chapter 3, “Using the Utility Program,” on page 57 and Chapter 4, “Utility Commands,” on page 77.

Data Sources

OV Performance Agent uses the coda daemon to provide log file data to the alarm generator and the OV Performance Manager analysis product. There is one coda server for all data sources including scopeux and DSI log files.

You configure data sources in the `datasources` file that resides in the `var/opt/OV/conf/perf/` directory. This allows you to selectively make data available for alarm processing and analysis.

NOTE Previous versions of OVPA 3.x and earlier, used a DCE-based facility to serve data for analysis.

When you install OV Performance Agent, the `datasources` file contains a single entry for a data source named `SCOPE` that tells `coda` to read the `scopeux` log file set. You can add additional data sources as needed.

When you start OV Performance Agent, the `coda` daemon reads the `datasources` file and makes the data available over datacomm linkages to analysis tools for each data source it finds. For more information about configuring data sources, see the "Configuring Data Sources" section in Chapter 2 of your *HP OpenView Performance Agent Installation & Configuration Guide*.

ARM Transaction Tracking Capabilities

OV Performance Agent includes transaction tracking capabilities that allow information technology (IT) managers to measure end-to-end response time of business application transactions. To take advantage of this functionality, you must have a process running that is instrumented with the Application Response Measurement (ARM) API. For more information, see the *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.

Related Performance Products

OV Performance Agent is one of several complimentary performance products from Hewlett-Packard. Each of these products fulfills a particular need within the range of resource and performance management. This lets you purchase as much functionality as you need and add products over time without risking incompatibilities or overlapping product capabilities.

Related HP performance products include the following:

OV Performance Manager

OV Performance Manager provides integrated performance management for multi-vendor distributed networks. It gives you a single interface and a common method for centrally monitoring, analyzing, and comparing resource measurement data supplied by OV Performance Agent running on many systems.

GlancePlus

GlancePlus (or Glance) is an online diagnostic tool that displays current performance data directly to a user terminal or workstation. It is designed to assist you in identifying and troubleshooting system performance problems as they occur.

OV Reporter

OV Reporter creates web-based reports from data of targeted systems it "discovers." Discovery of a system can occur if the system is running OpenView agent and subagent software such as OV Performance Agent. Reporter can also generate reports on systems managed by OV Operations. After Reporter has run through its discovery, it gathers data based on pre-defined and user-specified lists of metrics, then formats the collected data into web page reports.

OV Operations

OV Operations (OVO) also displays and analyzes alarm information sent by OV Performance Agent. OVO is a distributed client-server software solution designed to help system administrators detect, solve, and prevent problems occurring in networks, systems, and applications in any enterprise. OVO is a scalable and flexible solution that can be configured to meet the requirements of any information technology (IT) organization and its users..

For more information about any of these products, see the product documentation on the HP OpenView Manuals web site at:

http://ovweb.external.hp.com/lpe/doc_serv

Select <*product name*> from the product list box, select the release version, and click **[Search]**.

2 Managing Data Collection

Introduction

This chapter tells you how to manage the following data collection activities that are involved in using OV Performance Agent.

- using the `scopeux` data collector
- the collection parameters (`parm`) file and its parameters
- stopping and starting data collection
- controlling the amount of disk space used by log files
- archiving data

Scopeux Data Collector

The scopeux daemon collects and summarizes performance measurements of system-resource utilization and logs the data into the following log files, depending on the data classes specified in the log line of the parm file.

- The logglob file contains measurements of system-wide, or global, resource utilization information. Global data is summarized and recorded every five minutes.
- The logappl file contains aggregate measurements of processes in each user-defined application from the parm file. Application data is summarized every five minutes and each application that had any activity during the five minute interval is recorded.
- The logproc file contains measurements of selected interesting processes. Process data is summarized every minute. However, only interesting processes are recorded. The concept of interesting processes is a filter that helps minimize the volume of data logged and is controlled via the parm file.
- The logdev file contains measurements of individual device (such as disk and netif) performance. Device data is summarized every five minutes and data from each device that had any activity during the five minute interval is recorded.
- The logtran file contains measurements of ARM transaction data. This data is summarized every five minutes and each transaction that had any activity is recorded. (For more information, see the *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.)
- A sixth log file, logindx, contains information needed to access data in the other log files.

Scopeux Status

The /var/opt/perf/status.scope file serves as a status/error log for the scopeux process. New information is appended to this file each time the scopeux collector is started, stopped, or when a warning or error is encountered. To view the most recent status and error information from scopeux, use the perfstat -t command.

parm File

The parm file is a text file containing the instructions that tell scopeux to log specific performance measurements. The latest default parm file is installed with OV Performance Agent in the /opt/perf/newconfig/ directory (/usr/lpp/perf/newconfig/ on IBM AIX) and is copied into the /var/opt/perf/ directory during installation if one does not already exist from a previous installation. scopeux reads the /var/opt/perf/parm file when it starts up.

If you haven't run the product before, you can use the /var/opt/perf/parm file to become familiar with the type of data collected. The default values for each parameter are explained in the parm file.

Once you are familiar with the OV Performance Agent environment, you should tailor the /var/opt/perf/parm file to your performance data collection needs.

The parm file is set up to collect an average amount of log file data. The maximum amount depends on your system. For more information, see "Disk Space" in Chapter 1 of your *HP OpenView Performance Agent Installation & Configuration Guide*. Also see the description of the size parameter in "Parameter Descriptions" on page 37.

If you already have experience with OV Performance Agent and are familiar with the parm file parameters, you might want to modify this file before starting the scopeux collector.

Modifying the parm File

You can modify the parm file using any word processor or editor that can save a file in ASCII format.

When you modify the parm file, or create a new one, the following rules and conventions apply:

- Any parameter you specify overrides a default. See the parm file for the default values.
- The order in which the parameters are entered into the parm file is not important except as follows:

  - If a parameter is entered more than once, the last one entered is used.
  - The `file`, `user`, `group`, `or`, and `priority` parameters must follow the `application` statement that they define.
  - Application parameters should be listed in order so that a process will be aggregated into the application when it is first matched.

- You can use uppercase letters, lowercase letters, or a combination of both for all commands and parameter statements.
- You can use blanks or commas to separate key words in each statement.
- You can comment the `parm` file. Any line starting with a comment code (/*) or pound sign (#) is ignored.

After modifying the `parm` file, you must issue the `ovpa restart` command in order for the changes to take effect. This command causes `scopeux` to `stop`, `restart`, and `reread` the `parm` file.

parm File Parameters

`Scopeux` is controlled by specific parameters in the collection parameters (`parm`) file that do the following:

- Set maximum amount of disk space for the raw `scopeux` log files.
- Specify which data types are logged.
- Specify attributes of processes to be logged.
- Define types of performance data to be collected and logged.
- Specify what user-definable sets of applications should be monitored. An application can be one or more programs that are monitored as a group.
- Specify when `scopeux` should perform daily log file maintenance activities so that they do not impact system availability.

You can modify these parameters to tell `scopeux` to log measurements that match the requirements of your particular system (see “Modifying the parm File” on page 34).

The parm file parameters listed in Table 2-1 are used by scopeux. Some of these parameters are for specific systems as indicated in the table. For detailed descriptions of these parameters, see “Parameter Descriptions” on page 37 and “Application Definition Parameters” on page 42.

NOTE The items in the following table that are applicable only to HP-UX are described in detail in Chapter 2 of the *HP OpenView Performance Agent Installation & Configuration Guide for HP-UX*.

Table 2-1 parm File Parameters Used by Scopeux

Parameter	Values or Options
id	*system ID*
log	all global application [=prm] [=all] ([=prm] onHP-UX only) process device=disk,lvm,cpu,filesystem,all (lvm on HP-UX only) transaction=correlator,resource (resource on HP-UX only)
threshold	cpu=*percent* disk=*rate* (not on Linux or Windows) memory=nn (values in MBs) nonew nokilled shortlived= [TRUE | FALSE]
wait	cpu=*percent* (HP-UX only) disk=*percent* (HP-UX only) mem=*percent* (HP-UX only) sem=*percent* (HP-UX only) lan=*percent* (HP-UX only)
application	*application name*
file =	*file name [, ...]*

Table 2-1 parm File Parameters Used by Scopeux (Continued)

Parameter	Values or Options
user =	*user login name [,]*
group =	*groupname [,]*
or	
priority =	low value-high value (range varies by platform)
size	global=nn (values are in MBs) application=nn process=nn device=nn transaction=nn
mainttime	hh:mm (24 hours time)
scopetransactions	on off
subprocinterval=	value in seconds (not on HP-UX)

Parameter Descriptions

Following are descriptions of each of the parm file parameters.

ID The system ID value is a string of characters that identifies your system. If you do not want to rely on the default ID assigned (the system's hostname), and are specifying this string explicitly, then make sure different systems have different ID strings so as not to confuse centralized analysis. This identifier is carried with the log files to identify the system on which the data was collected. You can specify a maximum of 40 characters.

Log The log parameter specifies the data types to be collected by scopeux.

- log global writes global records to the logglob file. You must have global data records to view and analyze performance data on your system. Global metrics are not affected by logging options or values of application or process data.

- `log application` will cause `scopeux` to write active application records to the `logappl` file. The default behavior is that only applications that have active processes during an interval are logged.

  — `log application=all` in the `parm` file to force `scopeux` to log all applications to the `logappl` file at every interval, regardless of whether the applications are active or not.

  The `application=all` option may be desirable in specific circumstances in relation to the use of application alarms. For example, you can alarm on an application becoming inactive (`APP_ALIVE_PROC`).

  Enabling this option causes `logappl` to fill more quickly since all applications are logged at every interval. You can use the `utility` program's `scan` function to monitor the utilization of the `scopeux` log files.

  — On HP-UX only, you may specify `log application=prm` to write active Process Resource Manager (PRM) groups to the `logappl` file. Selection of this log specification will cause `scopeux` to ignore user-defined application sets listed in the `parm` file. In addition, all application metrics collected will reflect a PRM context and be grouped by the `APP_NAME_PRM_GROUPNAME` metric.

NOTE Application logging options do not affect global or process data.

- `log process` will write information about interesting processes to the `logproc` file. A process may become interesting when it is first created, when it ends, and when it exceeds a parm-defined threshold for activity. Process threshold logging options have no affect on global or application data.
- `log device=disk,lvm,cpu,filesystem` will request that `scopeux` log information about individual disks, logical volumes (HP-UX only), cpus, and/or file systems to the `logdev` file. The default behavior is that only disks, volumes and interfaces that had I/O generated through them during an interval are logged. Note that `netif` (logical LAN device) records are always logged regardless of the selected log device options.

  For example, to request logging of records for individual disks, logical

volumes, cpus, network interfaces, but *not* individual file systems, the log parameter in the `parm` file would include `device=disk,lvm,cpu`.

— When `filesystem` is specified, all mounted local file systems are logged every interval, regardless of activity.

— `log device=all` in the `parm` file will force `scopeux` to log all disk, logical volume, cpu, and network interface devices to the `logdev` file at every interval, regardless of whether the devices are active or not.

Enabling this option causes `logdev` to fill more quickly since all devices are logged at every interval. Use the `utility` program's `scan` function to monitor logfile utilization and sizing.

- `log transaction` will write ARM transaction records to the `logtran` file. In order for `scopeux` to collect the data, a process that is instrumented with the Application Response Measurement (ARM) API must be running on your system. (For more information, see the *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.)

  The default for the `log transaction` parameter is `no resource` and `no correlator`.

  To turn on resource data collection (HP-UX only) or correlator data collection, specify `log transaction=resource` or `log transaction=correlator`. Both can be logged by specifying `log transaction=resource, correlator`.

All of the log files are created automatically regardless of logging options. If a particular type of logging is disabled, the corresponding log file is not removed.

If you specify `log` without options, the default global and process data are logged.

Threshold The `threshold` parameter is used to set activity levels to specify criteria for interesting processes. It is used only if process logging is enabled. Thresholds only affect what processes are logged and do not affect any other class of data. For example, not logging process data at all would not affect collection or values of application or global data.

Enter the options for thresholds on the same parameter line (separated by commas).

Threshold Options:

`cpu`	Sets the percentage of CPU utilization that a process must exceed to become "interesting" and be logged. The value "`percent`" is a real number indicating overall CPU use. For example, `cpu=7.5` indicates that a process is logged if it exceeds 7.5 percent of CPU utilization in a 1-minute sample.
`disk`	(Not available on Linux or Windows). Sets the rate of physical disk I/O per second that a process must exceed to become "interesting" and be logged. The value is a real number. For example, `disk=8.0` indicates that a process will be logged if it exceeds eight physical I/Os per second average in a 1-minute sample.
`memory`	Sets the memory threshold that a process must exceed to become "interesting" and be logged. The value is in megabyte units and is accurate to the nearest 100KB. If set, the memory threshold is compared with the value of the `PROC_MEM_VIRT` metric. Each process that exceeds the memory threshold will be logged, similarly to the disk and CPU process logging thresholds.
`nonew`	Turns off logging of new processes if they have not exceeded any threshold. If not specified, all new processes are logged. On HP-UX, if `shortlived` is *not* specified, then only new processes that lasted more than one second are logged.
`nokilled`	Turns off logging of exited processes if they did not exceed any threshold. If not specified, all killed (exited) processes are logged. On HP-UX, if `shortlived` is *not* specified, then only killed processes greater than one second are logged.
`shortlived`	Turns on logging of processes that ran for less than one second in an interval. (This often significantly increases the number of processes logged.) If `scopeux` finds threshold `shortlived` in the `parm` file, it logs `shortlived` processes, regardless of the `cpu` or `disk threshold`, as long as the `nonew` and `nokilled` options

are removed. The default is no `shortlived` processes will be logged. (Do not specify shortlived in the `threshold` parameter if you do not want `shortlived` processes logged.)

Scopetransactions The `scopeux` collector itself is instrumented with ARM (Application Response Measurement) API calls to log its own transactions. The `scopetransactions` flag determines whether or not `scopeux` transactions will be logged. The default is `scopetransactions=on`; `scopeux` will log two transactions, `Scope_Get_Process_Metrics` and `Scope_Get_Global_Metrics`. If you do not want these `scopeux` transactions to be logged, specify `scopetransactions=off`. A third transaction, `Scope_Log_Headers`, will always be logged; it is not affected by `scopetransactions=off`.

For more information about ARM, see your *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.

Subprocinterval The `subprocinterval` parameter, if specified, overrides the default that `scopeux` uses to sample process data. Most classes of data are logged once every 5 minutes, the exception being process data, which is logged every 1 minute. However, `scopeux` probes its instrumentation every few seconds to catch short-term activities. This instrumentation sampling interval is 5 seconds by default.

On some systems with thousands of active threads or processes, the `subprocinterval` should be made longer to reduce overall `scopeux` overhead. On other systems with many short-lived processes that you may wish to log, setting the `subprocinterval` lower could be considered, although the effect on `scopeux` overhead should be monitored closely in this case. This setting must take values that are factors of the process logging interval of 60 seconds. Therefore, valid settings include: 1, 2, 3, 4, 5 (the default if not specified or commented out), 6, 10, 12, 15, 20, and 30.

Size The `size` parameter is used to set the maximum size (in megabytes) of any raw log file. You cannot set the size to be less than one megabyte.

The `scopeux` collector reads these specifications when it is initiated. If any of these log files achieve their maximum size during collection, they will continue to grow until `maintime`, when they will be rolled back automatically. During a roll back, the oldest 25 percent of the data is removed from the log file. Raw log files are designed to only hold a

maximum of one year's worth of data if not limited by the size parameter. See “Log File Contents Summary” and “Log File Empty Space Summary” in “Utility Scan Report Details” on page 66.)

If the size specification in the parm file is changed, scopeux detects it during startup. If the maximum log file size is decreased to the point where existing data does not fit, an automatic resize will take place at the next mainttime. If the existing data fits within the new maximum size specified, no action is taken.

Any changes you make to the maximum size of a log file take effect at the time specified in the mainttime parameter.

WARNING **Partial loss of old data will occur if logs are allowed to collect more than 365 days of data. Please configure the parm file size entry so that old data is rolled out before any of the log files contain 365 days of data.**

Mainttime Log files are rolled back if necessary by scopeux only at a specific time each day. The default time can be changed using the mainttime parameter. For example, setting mainttime=8:30, causes log file maintenance to be done at 8:30 am each day.

We suggest setting mainttime to a time when the system is at its lowest utilization.

Application Definition Parameters

The following parameters pertain to application definitions: application, file, user, group, or, and priority.

OV Performance Agent groups logically related processes together into an application to log the combined effect of those processes on computing resources such as memory and CPU.

NOTE In PRM mode (for HP-UX only), active PRM groups are logged and the user-defined application sets listed in the parm file are ignored.

An application can simply be a list of files combined with `users`, `groups`, or `priority` ranges, individually or in conjunction with each other. If `user`, `file`, and `priority` parameters are all specified for the same application, a process must meet the specifications of all three, `user`, `file` and `priority`, to belong to that application.

NOTE

Any process on the system belongs to only one application. No process is counted into two or more applications. Processes are collected based on program name, not program path. Therefore, two processes with the same program name but different paths (file system locations) would be bucketed into the same application.

Application The application name defines an application or class that groups together multiple processes and reports on their combined activities.

- The application name is a string of up to 19 characters used to identify the application.
- Application names can be lowercase or uppercase and can contain letters, numbers, underscores, and embedded blanks. Do not use the same application name more than once in the `parm` file.
- An equal sign (=) is optional between the application keyword and the application name.
- The `application` parameter must precede any combination of `file`, `user`, `group`, `or`, or `priority` parameters that refer to it, with all such parameters applying against the last application workload definition.
- Each parameter can be up to 170 characters long including the carriage return character, with no continuation characters permitted. If your list of files is longer than 170 characters, continue the list on the next line after another `file`, `user`, or `group` statement.
- You can define up to 128 applications. OV Performance Agent predefines an application named `other`. The `other` application collects all processes not captured by `application` statements in the `parm` file.

  For example:

```
application Prog_Dev
file vi,cc,ccom,pc,pascomp,dbx,xdb

application xyz
file xyz*,startxyz
```

You can have a maximum of 1000 file, 300 user, and 300 group specifications for all applications combined. The previous example includes nine file specifications. (xyz* counts as only one specification even though it can match more than one program file.)

If a program file is included in more than one application, it is logged in the first application that contains it.

The default /var/opt/newconfig/parm file contains some sample applications that you can modify. The examples directory also contains other samples (in a file called parm_apps) you can copy into your parm file and modify as needed.

File The file parameter specifies which program files belong to an application. All interactive or background executions of these programs are included. It applies to the last application statement issued. An error is generated if no application statement is found.

The file name can be any of the following:

- A single UNIX program file such as vi.
- A group of UNIX program files (indicated with a wild card) such as xyz*. In this case, any program name that starts with the letters xyz is included. A file specification with wild cards counts as only one specification toward the maximum of 1000 each for all files, 300 user, and 300 group specifications.

The name in the file parameter is limited to 15 characters in length. An equal sign (=) is optional between the file keyword and the file name.

You can enter multiple file names on the same parameter line (separated by commas) or in separate file statements. File names cannot be qualified by a path name. The file specifications are compared to the specific metric PROC_PROC_NAME, which is set to a process's argv[0] value (typically its base name).

For example:

```
application = prog_dev
file = vi,vim,gvim,make,gmake,lint*,cc*,gcc,ccom*,cfront
file = cpp*,CC,cpass*,c++*
file = xdb*,adb,pxdb*,dbx,xlC,ld,as,gprof,lex,yacc,are,nm,gencat
```

```
file = javac,java,jre,aCC,ctcom*,awk,gawk

application Mail
file = sendmail,mail*,*mail,elm,xmh
```

If you do not specify a `file` parameter, all programs that satisfy the other parameters qualify.

If you want to bucket Java or shell programs into different applications, you may want to look at the `/opt/perf/examples/tools/procarg` script, which shows a way to start programs with a unique `argv[0]` value.

NOTE The asterisk (*) is the only wild card character supported by the `parm` file.

`User` The `user` parameter specifies which user login names belong to an application.

For example:

```
application Prog_Dev
file vi,xb,abb,ld,lint
user ted,rebecca,test*
```

User specifications that include wildcards count as only one specification toward the maximum of 1000 each for all `file`, 300 `user`, and 300 `group` specifications.

If you do not specify a `user` parameter, all programs that satisfy the other parameters qualify.

The name in the `user` parameter is limited to 15 characters in length.

`Group` The group parameter specifies which user group names belong to an application.

For example:

```
application Prog_Dev_Group2
file vi,xb,abb,ld,lint
user ted,rebecca,test*
group lab, test
```

If you do not specify a `group` parameter, all programs that satisfy the other parameters qualify.

The name in the `group` parameter is limited to 15 characters in length.

Or Use the `or` parameter to allow more than one application definition to apply to the same application. Within a single application definition, a process must match at least one of each category of parameters. Parameters separated by the `or` parameter are treated as independent definitions. If a process matches the conditions for any definition, it will belong to the application.

For example:

```
application = Prog_Dev_Group2
user julie
or
user mark
file vi, store, dmp
```

This defines the application (`Prog_Dev_Group2`) that consists of any programs run by the user `julie` plus other programs (`vi`, `store`, `dmp`) if they are executed by the user `mark`.

Priority You can restrict processes in an application to those belonging to a specified range by specifying values in the `priority` parameter.

For example:

```
application = swapping
priority 128-131
```

Processes can range in priority from -511 to 255, depending on which platform OV Performance Agent is running.

NOTE The `parm` file is processed in the order entered and the first match of program name and/or user login as well as `group` and `priority` (if used) defines the application to which a particular process belongs.

The priority can be changed over the life of a process. The scheduler adjusts the priority of time-share processes. You can also change priorities programmatically or while executing.

The process priority is sampled at the end of each one-minute sample interval. If the process has changed priority, it can change applications. All activity for a process during the one-minute interval is assumed to have occurred at the new priority and is attributed to the application that matches the process at the end of each one-minute sample interval.

Application Definition Examples The following examples show application definitions.

```
application Prog_Dev_Group1
file vi,dbx,abb,ld,lint
user bill,debbie

application Prog_Dev_Group2
file vi,dbx,abb,ld,lint
user julie,mark
group rdlab

application Other_Editors
file ed,sed,awk

application Compilers
file cc,ccom,xlc,c++,fe,sh

application Users
user bruce,ted,julie,anna
```

The following is an example of how several of the programs would be logged using the preceding `parm` file.

Program	User Login	Application
`vi`	bill	Prog_Dev_Group1
`awk`	dan	Other_Editors
`vi`	mark	Prog_Dev_Group2
`cc`	gene	Compilers
`dbx`	dan	other

Stopping and Restarting Data Collection

The scopeux daemon and the other daemon processes that are part of OV Performance Agent are designed to run continuously. The only time you should stop them are when any of the following occurs:

- You are updating OV Performance Agent software to a new release.
- You are adding or deleting transactions in the transaction configuration file, `ttd.conf`. (For more information, see the *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.)
- You are modifying distribution ranges or service level objectives (SLOs) in the transaction configuration file, `ttd.conf`. (For more information, see the *HP OpenView Performance Agent & Glance Plus for UNIX Tracking Your Transactions* guide.)
- You are changing the `parm` file and want the changes to take effect. Changes made to the `parm` file take effect only when scopeux is started.
- You are using the `utility` program's `resize` command to resize a OV Performance Agent log file.
- You are shutting down the system.

OV Performance Agent provides the ovpa script that includes options for stopping and restarting the daemon processes. For a description of these options, see the ovpa man page.

Stopping Data Collection

The ovpa script's `stop` option ensures that no data is lost when scopeux and the other OV Performance Agent daemon processes are stopped. To manually stop data collection, use `/opt/perf/bin/ovpa stop`.

Restarting Data Collection

You have different options for restarting data collection after the OV Performance Agent daemon processes have stopped or configuration files have been changed and you want these changes to take effect.

To start scopeux and the other OV Performance Agent processes after the system has been down, use `/opt/perf/bin/ovpa start`.

When you restart scopeux, OV Performance Agent continues to use the same log files (`logglob`, `logappl`, `logproc`, `logdev`, `logtran`, and `logindx`) used before stopping the program. New records are appended to the end of the existing files.

For more information, see "Starting and Stopping OV Performance Agent" in Chapter 2 of your *HP OpenView Performance Agent Installation & Configuration Guide*.

Automating Scopeux Startup and Shutdown

OV Performance Agent's startup can be automated to ensure that scopeux is always running while the system is operating and that any shutdown of the system includes a shutdown of scopeux without any loss of data. The process of starting OV Performance Agent and its processes automatically when the system reboots is controlled by the configuration file in the system startup directory. For more information about this file and how to modify it, see "Starting and Stopping Automatically" in Chapter 2 of your *HP OpenView Performance Agent Installation & Configuration Guide*.

Effective Data Collection Management

Efficient analysis of performance depends on how easy it is to access the performance data you collect. This section discusses effective strategies for activities such as managing log files, data archiving, and system analysis to make the data collection process easier, more effective, and more useful.

Controlling Disk Space Used by Log Files

OV Performance Agent provides for automatic management of the log files it creates. You can configure this automatic process or use alternate manual processes for special purposes. The automatic log file management process works as follows:

- Each log file has a configured maximum size. Default maximum sizes are provided when the OV Performance Agent is first installed. However, you can reconfigure these values.
- As each log file reaches its maximum size, a “roll back” is performed at `mainttime` by the `scopeux` data collector. During this roll back, the oldest 25 percent of the data in the log file is removed to make room for new data to be added.

Automatic log file maintenance is similar, but not identical, for data collected by `scopeux` and by the DSI logging process. For more information on DSI log file maintenance, see the *HP OpenView Performance Agent for UNIX Data Source Integration Guide*.

Setting Mainttime

Normally, `scopeux` will only perform log file roll backs at a specific time each day. This is to ensure that the operation is performed at off peak hours and does not impact normal system usage. The time the log files are examined for roll back is set by the `mainttime` parameter in the `parm` file.

Setting the Maximum Log File Size

Choosing a maximum log file size should be a balance between how much disk space is used and how much historical data is available for immediate analysis. Smaller log file sizes save disk space, but limit how much time can be graphed by tools such as OV Performance Manager. Some ways to reconfigure the scopeux log file sizes are discussed below.

Scopeux logs different types of data into their own log files. This is to allow you to choose how much disk space you want to dedicate to each type independently. For example, global data is fairly compact, but you will often want to go back and graph data for a month at a time. This allows a good statistical base for trending and capacity planning exercises.

Process data can consume more disk space than global data because it is possible to have many interesting processes every minute. Also, the time-value of process data is not as high as for global data. It may be very important to know details about which process was running today and yesterday. You might occasionally need to know which processes were running last week. However, it is unlikely that knowing exactly which processes were run last month would be helpful.

A typical user might decide to keep the following data online:

- Three months of global data for trending purposes
- One week of process data for troubleshooting
- Three months of application data for trending and load balancing
- Two months of device data for disk load balancing

You can edit the parm file to set the size parameters for each different log file. The sizes are specified in megabytes. For example:

```
SIZE GLOBAL=10.0 PROCESS=30.0 APPLICATION=20.0 DEVICE=5.0
```

The number of megabytes required to hold a given number of days of data can vary by data type, system configuration, and system activity. The best way to determine how big to make the log files on your system is to collect data for a week or so, then use the utility program's resize command to change your log file size. The resize command scans the log files and determines how much data is being logged each day. It then converts from days to megabytes for you. This function also updates the parm file.

Managing Your Resizing Processes

No additional activities are required once automatic log file maintenance is set up. As log files reach their configured maximum sizes, they will automatically be resized by scopeux.

Scopeux rolls back log files at the mainttime specified in the parm file. If you edit the parm file and restart scopeux, the log files will not be rolled to the new sizes until the mainttime occurs. It is important to have scopeux running at the specified mainttime time or log files may never be rolled back.

Log files may exceed their configured maximum size during the time between maintenance times without causing an immediate roll back.

A log file will never be resized so that it holds less than one full day's data. That means that the log file will be allowed to grow to hold at least one day's worth of data before it is rolled back. Normally this is not an issue, but if you set the parm file parameters to collect a large volume of process or application data or set the size to be too small, this can result in a log file significantly exceeding its configured maximum size before it is rolled back.

Every five minutes, scopeux checks the available disk space on the file system where the log files reside. If the available disk space falls below one megabyte, scopeux takes steps to ensure that it does not use any more available space by doing the following:

- Immediately performs the log file maintenance without waiting for the regular log file maintenance time. If any log files exceed their maximum sizes (and have more than one day's worth of data in them), they will be rolled back.
- If, following the log file maintenance, the available disk space is still not greater than one megabyte, scopeux writes an appropriate error message to its status.scope file and stops collecting data.

Data Archiving

Automatic log file management keeps the latest log file data available for analysis. It works on the raw log files. Process data is logged each minute and all other data is logged every five minutes. To make room for new data, older data is removed when the log files reach their maximum sizes. If you want to maintain log file data for longer periods of time, you should institute a data archiving process. The exact process you choose depends on your needs. Here are a few possibilities:

- Size the raw log files to be very large and let automatic log file maintenance do the rest. This is the easiest archiving method, but it can consume large amounts of disk space after several months.
- Extract the data from the raw log files into extracted archive files before it is removed from the raw log files. Formulate a procedure for copying the archive files to long term storage such as tape until needed.
- Extract only a subset of the raw log files into extracted archive files. For example, you may not want to archive process data due to its high volume and low time-value.
- Some combination of the preceding techniques can be used.

We recommend the following procedures for data archiving:

- Size the raw log files to accommodate the amount of detail data you want to keep online.
- Once a week, copy the detailed raw data into files that will be moved to offline storage.

Managing Your Archiving Processes

Resize your raw log files as described in the preceding section. Choose log file sizes that will hold at least two week's worth of data (assuming the archival processing will only be done once a week).

Once a week, schedule a process that runs the `extract` program twice. The first time it copies detailed data into an archive file for offline storage. The second time, it copies summary data into an online archive file. Here is a sample script file that would perform the weekly processing. It uses the special archiving features of the `extract` program.

```
#Extract detailed data into monthly archive files.
extract -gapdt -xm
```

The first `extract` command will create a series of archive log files, one per month. The log files will be named "`rxmo`" followed by four digits for the year and two more digits for the month. (For example, data for December 1999 would be in a file named "`rxmo199912`".) At the beginning of each month the previous month's log file is completed and a new log file is started. Therefore, whenever more than one "`rxmo`" log file is present, it is safe to copy all but the latest one to offline storage until its needed. When you need to access archived data, restore the desired archival file and access it using the `extract` or `utility` programs.

Depending on your system configuration and activity levels, the amount of disk space accumulated in one month may be large. If this is the case, you can break the detail archive file into smaller files by substituting the weekly command `-xw` in place of `-xm` as shown in the example.

Another alternative is to choose not to archive the detailed process data.

The detailed extraction discussed in the previous example preserves all of your collected performance data. If ever you need to investigate a situation in depth, these files can be restored to disk and analyzed.

Hint

You can use the `extract` program to combine data from multiple extracted files or to make a subset of the data for easier transport and analysis.

For example, you can combine data from several yearly extracted files in order to do multiple-year trending analysis. (See the description of the `yearly` command in Chapter 6, "Extract Commands," on page 145.)

CAUTION

Moving log files that were created on a new version of OV Performance Agent to a system using an older version of OV Performance Agent is not supported.

3 Using the Utility Program

Introduction

The utility program is a tool for managing and reporting information on log files, the collection parameters (parm) file, and the alarm definitions (alarmdef) file. You can use the utility program interactively or in batch mode to perform the following tasks.

- Scan raw or extracted log files and produce a report showing:
  - dates and times covered
  - times when the scopeux collector was not running
  - changes in scopeux parameter settings
  - changes in system configuration
  - log file disk space
  - effects of application and process settings in the collection parameters (parm) file
- Resize raw log files
- Check the parm file for syntax warnings or errors
- Check the alarmdef file for syntax warnings or errors
- Process log file data against alarm definitions to detect alarm conditions in historical data

This chapter covers the following topics:

- running the utility program
- using interactive mode
- using the command line interface
- scan report details

Detailed descriptions of the utility program's commands are in Chapter 4, "Utility Commands," on page 77.

Running the Utility Program

There are three ways to run the `utility` program:

- Command line mode - You control the `utility` program using command options and arguments in the command line.
- Interactive mode - You supply interactive commands and parameters while executing the program with `stdin` set to an interactive terminal or workstation.
  If you are an experienced user, you can quickly specify only those commands required for a given task. If you are a new user, you may want to use the `utility` program's `guide` command to get some assistance in using the commands. In guided mode, you are asked to select from a list of options to perform a task. While in guided mode, the interactive commands that accomplish each task are listed as they are executed, so you can see how they are used. You can quit and re-enter guided mode at any time.
- Batch mode - You can run the program and redirect `stdin` to a file that contains interactive commands and parameters.

The syntax for the command line interface is similar to typical UNIX command line interfaces on other programs and is described in detail in this chapter.

For interactive and batch mode the command syntax is the same. Commands can be entered in any order; if a command has a parameter associated with it, the parameter must be entered immediately after the corresponding command.

There are two types of parameters - required (for which there are no defaults) and optional (for which defaults are provided). How `utility` handles these parameters depends on the mode in which it is running.

- In interactive mode, if an optional parameter is missing, the program displays the default argument and lets you either confirm it or override it.
  If a required parameter is missing, the program prompts you to enter the argument.
- In batch mode, if an optional parameter is missing, the program uses the default values.
  If a required parameter is missing, the program terminates.

Errors and missing data are handled differently for interactive mode than for command line and batch mode. You can supply additional data or correct mistakes in interactive mode, but not in command line and batch mode.

Using Interactive Mode

Using the `utility` program's interactive mode requires you to issue a series of commands to execute a specific task.

For example, if you want to check a log file to see if alarm conditions exist in data that was logged during the current day, you issue the following commands after invoking the `utility` program:

```
checkdef /var/opt/perf/alarmdef
detail off
start today-1
analyze
```

The `checkdef` command checks the alarm definitions syntax in the `alarmdef` file and then sets and saves the file name for use with the `analyze` command. The `detail off` command causes the `analyze` command to show only a summary of alarms. The `start today-1` command specifies that only data logged yesterday is to be analyzed. The `analyze` command analyzes the raw log files in the default SCOPE data source against the `alarmdef` file.

Example of Using Interactive and Batch Mode

The following example shows the differences between how the `utility` program's `resize` command works in batch mode and in interactive mode.

The `resize` command lets you set parameters for the following functions:

- Type of log file to be resized.
- Size of the new file.
- Amount of empty space to be left in the file.
- An action specifying whether or not the resize is to be performed.

This example of the `resize` command resizes the global log file so that it contains a maximum of 120 days of data with empty space equal to 45 days. The command and its parameters are:

```
resize global days=120 empty=45 yes
```

The results are the same whether you enter this command interactively or from a batch job.

The first parameter–`global`–indicates the log file to be resized. If you do not supply this parameter, the consequent action for interactive and batch users would be the following:

- Batch users - the batch job would terminate because the `logfile` parameter has no default.
- Interactive users - you would be prompted to choose which type of log file to resize to complete the command.

The last parameter–`yes`–indicates that resizing will be performed unconditionally.

If you do not supply the `yes` parameter, the consequent action for interactive and batch users would be the following:

- Batch users - resizing would continue since `yes` is the default action.
- Interactive users - you would be prompted to supply the action before resizing takes place.

NOTE Before using the resize command in either batch mode or interactive mode, you must first stop OV Performance Agent. For details, see “Stopping and Restarting Data Collection” on page 49 in Chapter 2.

Utility Command Line Interface

In addition to the interactive and batch mode command syntax, command options and their associated arguments can be passed to the `utility` program through the command line interface. The command line interface fits into the typical UNIX environment by allowing the `utility` program to be easily invoked by shell scripts and allowing its input and output to be redirected to UNIX pipes.

For example, to use the command line equivalent of the example shown in the previous section "Using Interactive Mode" enter:

```
utility -xr global days=120 empty=45 yes
```

Command line options and arguments are listed in the following table. The referenced command descriptions can be found in Chapter 4, “Utility Commands,” on page 77.

Table 3-1 Command Line Arguments

Command Option	Argument		Description
-b	*date*	*time*	Specifies the starting date and time of an analyze or scan function. (See “start” command in Chapter 4.)
-e	*date*	*time*	Specifies the ending date and time of an analyze or scan function. (See “stop” command in Chapter 4.)
-l	*logfile*		Specifies which log file to open. (See “logfile” command in Chapter 4.)
-f	*listfile*		Specifies an output listing file. (See “list” command in Chapter 4.)
-D			Enables details for `analyze`, `scan` and `parm` file checking. (See “detail” command in Chapter 4.)
-d			Disables details for `analyze` and `parm` file for checking. (See “detail” command in Chapter 4.)

Table 3-1 Command Line Arguments (Continued)

Command Option	Argument		Description
-v			Echoes command line commands as they are executed.
-xp	*parmfile*		Syntax checks a parm file. (See "parmfile" command in Chapter 4.)
-xc	*alarmdef*		Syntax checks and sets the alarmdef file name to use with -xa (or analyze command). (See "checkdef" command in Chapter 4.)
-xa			Analyzes log files against the alarmdef file. (See "analyze" command in Chapter 4.)
-xs	*logfile*		Scans a log file and produces a report. (See "scan" command in Chapter 4.)
-xr	global application process device transaction EMPTY=*nnn* SPACE=*nnn*	SIZE=*nnn* DAYS=*nnn* YES NO MAYBE	Resizes a log file. (See "resize" command in Chapter 4.)
-? or ?			Displays command line syntax.

Example of Using the Command Line Interface

The following situation applies when you enter command options and arguments on the command line:

Errors and missing data are handled exactly as in the corresponding batch mode command. That is, missing data is defaulted if possible and all errors cause the program to terminate immediately.

Echoing of commands and command results is disabled. `Utility` does not read from its `stdin` file. It terminates following the actions in the command line.

```
utility -xp -d -xs
```

Which translates into:

`-xp`	Syntax checks the default `/opt/perf/newconfig/parm` file.
`-d`	Disables details in the scan report.
`-xs`	Performs the scan operation. No log file was specified so the default log file is scanned.

Utility Scan Report Details

The utility program's scan command reads a log file and writes a report on its contents. The report's contents depend on the commands issued prior to issuing the scan command. (For more information, see the description of the scan command in Chapter 4, “Utility Commands,” on page 77.

The following table summarizes the information contained in all scan reports and in reports that are produced only when the detail on command is used (the default) with the scan command.

Information Contained in Scan Report

Initial Values	
Initial parm file global information and system configuration information	Printed only if detail on is specified.
Initial parm file application definitions	Printed only if detail on is specified.
Chronological Detail	
parm file global changes	Printed only if detail on is specified.
parm file application changes	Printed only if detail on is specified.
Collector off-time notifications	Printed only if detail on is specified.
Application-specific summary reports	Printed only if detail on is specified.

Summaries

Process summary report	Always printed if process data was scanned.
Collector coverage summary	Always printed.
Log file contents summary	Always printed. Includes space and dates covered.
Log file empty space summary	Always printed.

Scan Report Information

The information in a `utility` scan report is divided into three types:

- Initial values
- Chronological details
- Summaries

Initial Values

This section describes the following initial values:

- Initial `parm` file global information
- Initial `parm` file application definitions

Initial Parm File Global Information

To obtain this report, use the `scan` command with its default `detail on`.

This report lists the configuration settings of the `parm` file at the time of the earliest global record in the log file. Later global information change notifications are based on the values in this report. If no change notification exists for a particular parameter, it means that the parameter kept its original setting for the duration of the scan.

The date and time listed on the first line correspond to the *first date and time* in the global log file and indicate when `scopeux` was started. Data records may have been rolled out of the global log file so the date and time on this report do not necessarily indicate the *first global record* in the log file.

Initial Parm File Application Definitions

To obtain this report, use the `scan` command with its default `detail on` and have application data in the log file.

This report lists the name and definition of each application at the time the first application record is listed in the log file. Any application addition or deletion notifications you receive are based on this initial list of applications. For example:

```
06/01/99 08:39 Application(1) = "other"
Comment=all processes not in user-defined applications

06/01/99 08:39 Application(2) = "Real_TimeSystem"
 Priority range = 0-127

06/01/99 08:39 Application(3) = "Prog_Development"
 File=vi,ed,sed,xdb,ld,lint,cc,ccom,pc,pascomp
```

NOTE

During the scan, you are notified of applications that were added or deleted. Additions and deletions are determined by comparing the spelling and case of the old application names to the new set of logged application names. No attempt is made to detect a change in the definition of an application. If an application with a new name is detected, it is listed along with its new definition.

The date and time on this record is the last time scopeux was started before logging the first application record currently in the log file.

Chronological Detail

This section describes the following chronological details:

- `parm` file global change notifications
- `parm` file application addition and deletion notifications
- `scopeux` off-time notifications
- Application-specific summary report

parm File Global Change Notifications

To obtain this report, use the `scan` command with its default `detail on`.

This report is generated any time a record is found that scopeux started.

parm File Application Addition/Deletion Notifications

To obtain this report, use the `scan` command with its default `detail on` and have application data in the log file.

User-defined applications can be added or deleted each time scopeux is started. If an application name is found that does not match the last set of applications, an application addition, deletion, or change notification is printed. If the name of an application has not changed, it is not printed.

NOTE Application definitions are not checked for changes. They are listed when an application name is changed, but any change to an existing application's definition without an accompanying name change is not detected.

Scopeux Off-Time Notifications

To obtain this report, use the scan command with its default `detail on`.

Application-Specific Summary Report

To obtain this report, use the scan command with its default `detail on` and have application data in the log file.

This report can help you define applications. Use the report to identify applications that are accumulating either too many or too few system resources and those that could be consolidated with other applications. Applications that accumulate too many system resources might benefit by being split into multiple applications.

You should define applications in ways that help you make decisions about system performance tuning. It is unlikely that system resources will accumulate evenly across applications.

The application-specific summary report is generated whenever the application definitions change to allow you to access the data of the application definitions before and after the change.

A final report is generated for all applications. This report covers only the time since the last report and not the entire time covered by the log file. For example:

Application	Records	PERCENT OF TOTAL CPU	DISK	TRANS
OTHER	22385	45.7%	20.9%	63.0%
Resource_Sharing	7531	6.0%	2.2%	17.1%
SPOOLING	13813	2.4%	0.3%	0.0%
ON_LINE_COMPILES	13119	2.9%	1.7%	0.1%
BATCH_COMPILES	8429	2.9%	0.1%	2.2%
ORDER_ENTRY	387	0.1%	0.0%	0.0%
ELECTRONIC_MAIL	6251	3.8%	1.3%	9.6%
PROGRAM_DEVELOPMENT	3141	9.1%	2.4%	0.6%
RESEARCH_DEPARMENT	3968	8.7%	2.0%	6.0%
BILL_OF_MATERIALS	336	0.6%	1.5%	0.1%
FINANCIALS	1080	5.0%	1.5%	0.5%
MARKETING_DEPT	2712	12.9%	67.3%	0.0%
GAMES	103	0.1%	0.0%	0.0%
All user applications		73.1%	54.3%	79.1% 37.0%

Summaries

This section describes the following summaries:

- Process log reason summary
- Scan start and stop actual dates and times
- Application overall summary
- scopeux coverage summary
- Log file contents summary
- Log file empty space summary

Process Log Reason Summary

To obtain this report, you must have process data in the log file.

This report helps you set the interesting process thresholds for scopeux. The report lists every reason a process might be considered interesting and thus get logged, along with the total number of processes logged that satisfied each condition.

The following example shows a process log reason summary report:

```
Process Summary Report: 04/13/99 3:32 PM to 05/04/99 6:36 PM
There were 93.8 hours of process data
Process records were logged for the following reasons:

Log Reason      Records   Percent   Recs/hr
--------------- -------   -------   -------
New Processes     17619     53.9%      44.7
Killed Processes  16047     49.1%      40.7
CPU Threshold      3169      9.7%       8.0
Disk Threshold       1093       3.3%        2.8

NOTE: A process can be logged for more than one reason at a
time. Record counts and percentages will not add up to 100%
of the process records.
```

If the `detail on` command is issued, this report is generated each time a threshold value is changed so you can evaluate the effects of that change. Each report covers the period since the last report. A final report, generated when the scan is finished, covers the time since the last report.

If the `detail off` command is issued, only one report is generated covering the entire scanned period.

You can reduce the amount of process data logged by `scopeux` by modifying the `parm` file's `threshold` parameter and raising the thresholds of the interest reasons that generate the most process log records. To increase the amount of data logged, lower the threshold for the area of interest.

In the previous example, you could decrease the amount of disk space used for the process data (at the expense of having less information logged) by raising the `CPU` threshold or setting the `nonew` threshold.

Scan Start and Stop

This summary report is printed if any valid data was scanned. It gives actual dates and times that the scan was started and stopped. For example:

```
Scan started on        03/03/99 12:40 PM
Scan stopped on        03/11/99  1:25 PM
```

Application Overall Summary

To obtain this report, you must have application data in the log file.

This report is an overall indicator of how much system activity is accumulated in user-defined applications, rather than in the other application. If a significant amount of a critical resource is not being captured by user applications, you might consider scanning the process data for processes that can be included in user applications.

For example:

```
OVERALL, USER DEFINED APPLICATIONS ACCOUNT FOR
    82534 OUT OF    112355 RECORDS     ( 73.5%)
    218.2 OUT OF     619.4 CPU HOURS   ( 35.2%)
     24.4 OUT OF      31.8 M DISC IOS  ( 76.8%)
      0.2 OUT OF       0.6 M TRANS     ( 27.3%)
```

Collector Coverage Summary

This report is printed if any valid global or application data was scanned. It indicates how well scopeux is being used to collect system activity. If the percentage of time scopeux was off is high, as in the example below, you should review your operational procedures for starting and stopping scopeux.

```
The total time covered was         108/16:14:51 out of 128/00:45:02
Time lost when collector was off   19/08:30:11  15.12%
The scopeux collector was started      45 times
```

This report will be more complete if global detail data is included in the scan. If only summary data is available, you determine the time scopeux was stopped and started only to the nearest hour. (An appropriate warning message is printed with the report if this is the case.)

The total time covered is determined by accumulating all the interval times from the logged data.The "out of" time value is calculated by subtracting the starting date and time from the ending date and time. This should represent the total time that could have been logged. The "Time lost when collector was off" value is the total time less the covered time.

The formats for the three times mentioned are:

ddd/hh:mm:ss

where *ddd* are days and *hh:mm:ss* are hours, minutes, and seconds.

In the previous example, the total time collected was 108 days, 16 hours, 14 minutes, and 51 seconds.

Log File Contents Summary

The log file contents summary is printed *if any* valid data was scanned. It includes the log file space and the dates covered. This summary is helpful when you are resizing your log files with the `resize` command.

```
------ Total- ---  --Each Full Day-- ----Dates-------    Full
Type         Records  MBytes  Records  MBytes  Start       Finish    Days
Global        1376     0.27     288.9   0.057  05/23/99 to 05/28/99  4.8
Application   6931     0.72    1455.0   0.152  05/23/99 to 05/28/99  4.8
Process       7318     1.14    1533.6   0.239  05/23/99 to 05/28/99  4.8
Disk          2748     0.07     567.6   0.014  05/23/99 to 05/28/99  4.8
Transaction  no data found
Overhead              0.29
             ------- -------- -------  ---------
TOTAL          18373      2.49     3845.0   0.461
```

The columns are described as follows:

Column	Explanation
`Type`	The general type of data being logged. One special type, `Overhead`, exists: `Overhead` is the amount of disk space occupied (or reserved) by the log file *versus* the amount actually used by the scanned data records. If less than the entire log file was scanned, Overhead includes the data records that were not scanned. If the entire file was scanned, `Overhead` accounts for any inefficiencies in blocking the data into the file *plus* any file-access support structures. It is normal for extracted log files to have a higher overhead than raw log files since they have additional support structures for quicker positioning.
`Total`	The total record count and disk space scanned for each type of data.
`Each Full Day`	The number of records and amount of disk space used for each 24-hour period that `scopeux` runs.

Dates	The first and last valid dates for the data records of each data type scanned.
Full Day	The number of full (24-hour) days of data scanned for this data type. Full Days may not be equal to the difference between the start and stop dates if scopeux coverage did not equal 100 percent of the scanned time.

The TOTAL line (at the bottom of the listed data) gives you an idea of how much disk space is being used and how much data you can expect to accumulate each day.

Log File Empty Space Summary

This summary is printed for each log file scanned. For example:

```
The Global      file is now  13.9% full with room for 61 more full days
The Application file is now  15.1% full with room for 56 more full days
The Process     file is now  23.5% full with room for 32 more full days
The Device      file is now   1.4% full with room for 2896 more full days
```

The amount of room available for more data is calculated based on the amount of unused space in the file and the scanned value for the number of megabytes of data being logged each 24-hour day (see “Log File Contents Summary”). If the megabytes-scanned-per-day values appear unrealistically low, they are replaced with default values for this calculation.

If you scan an extracted file, you get a single report line because all data types share the same extracted file.

4 Utility Commands

Introduction

This chapter describes the `utility` program's commands. It includes a syntax summary and a command reference section that lists the commands in alphabetical order.

`Utility` commands and parameters can be entered with any combination of uppercase and lowercase letters. Only the first three letters of the command name are required. For example, the `logfile` command can be entered as `logfile` or it can be abbreviated as `log` or `LOG`.

Examples of how these commands are used can be found in online help for the `utility` program.

The table on the next pages contains a summary of `utility` command syntax and parameters.

Table 4-1 Utility Commands: Syntax and Parameters

Command	Parameter
analyze	
checkdef	*alarmdef* file
detail	on off
exit e	
guide	
list	*filename* or *
logfile	*logfile*
menu ?	
parmfile	*parmfile*
quit q	
resize	global application process device transaction days=maxdays size=max MB empty=days space=MB yes no maybe
scan	*logfile* (Operation is also affected by the list, start, stop, and detail commands.
show	all

Table 4-1 Utility Commands: Syntax and Parameters (Continued)

Command	Parameter
sh !	*system command*
start	*date* [*time*] today [*-days*] [*time*] last [*-days*] [*time*] first [*+days*] [*time*]
stop	*date* [*time*] today [*-days*] [*time*] last [*-days*] [*time*] first [*+days*] [*time*]

analyze

Use the analyze command to analyze the data in a log file against alarm definitions in an alarm definitions (alarmdef) file and report resulting alarm status and activity. Before issuing the analyze command, you should run the checkdef command to check the alarm definitions syntax. Checkdef also sets and saves the alarm definitions file name to be used with analyze. If you do not run checkdef before analyze, you are prompted for an alarm definitions file name.

If you are using command line mode, the default alarm definitions file /var/opt/perf/alarmdef is used.

For detailed information about alarm definitions, see Chapter 7, “Performance Alarms,” on page 207.

Syntax

```
analyze
```

How to Use It

When you issue the analyze command, it analyzes the log files specified in the data sources configuration file, datasources, against the alarm definitions in the alarmdef file.

The analyze command allows you to evaluate whether or not your alarm definitions are a good match against the historical data collected on your system. It also lets you decide if your alarm definitions will generate too many or too few alarms on your analysis workstation.

Also, you can perform data analysis with definitions (IF statements) set in the alarm definitions file because you can get information output by PRINT statements when conditions are met. For explanations of how to use the IF and PRINT statements in an alarm definition, see Chapter 7, “Performance Alarms,” on page 207.

You can optionally run the start, stop, and detail commands with analyze to customize the analyze process. You specify these commands in the following order:

```
checkdef
start
stop
detail
analyze
```

Use the start and stop commands if you want to analyze log file data that was collected during a specific period of time. (Descriptions of the start and stop commands appear later in this chapter.)

While the analyze command is executing, it lists alarm events such as alarm start, end, and repeat status plus any text in associated print statements. Also, any text in PRINT statements is listed as conditions (in IF statements) become true. EXEC statements are not executed but are listed so you can see what would have been executed. An alarm summary report shows a count of the number of alarms and the amount of time each alarm was active (on). The count includes alarm starts and repeats, but not alarm ends.

If you want to see the alarm summary report only, issue the detail off command. However, if you are using command line mode, detail off is the default so you need to specify -D to see the alarm events as well as the alarm summary.

Example

The checkdef command checks the alarm definitions syntax in the alarmdef file and saves the name of the alarmdef file for later use with the analyze command. The start today command specifies that only data logged today is to be analyzed. Lastly, the analyze command analyzes the log file in the default SCOPE data source specified in the datasources file against the alarm definitions in the alarmdef file.

```
utility>
checkdef /var/opt/perf/alarmdef
start today
analyze
```

To perform the above task using command line arguments, enter:

```
utility -xc -D -b today -xa
```

checkdef

Use the checkdef command to check the syntax of the alarm definitions in an alarm definitions file and report any warnings or errors that are found. This command also sets and saves the alarm definitions file name for use with the analyze command.

For descriptions of the alarm definitions syntax and how to specify alarm definitions, see Chapter 7, “Performance Alarms,” on page 207.

Syntax

```
checkdef [/directorypath/alarmdef]
```

Parameters

alamdef	The name of any alarm definitions file. This can be a user-specified file or the default alarmdef file. If no directory path is specified, the current directory will be searched.

How to Use It

When you have determined that the alarm definitions are correct, you can process them against the data in a log file using the analyze command.

In batch mode, if no alarm definitions file is specified, the default alarmdef file is used.

In interactive mode, if no alarm definitions file is specified, you are prompted to specify one.

Example

The checkdef command checks the alarm definitions syntax in the alarmdef file and then saves the name of the alarmdef file for later use with the analyze command.

```
utility>
checkdef /var/opt/perf/alarmdef
```

To perform the above task using command line arguments, enter:

```
utility -xc
```

detail

Use the detail command to control the level of detail printed in the analyze, parmfile, and scan reports.

The default is detail on in interactive and batch modes and detail off in command line mode.

Syntax

```
detail      [on]
            [off]
```

Parameters

on	Prints the effective contents of the parm file as well as parm file errors. Prints complete analyze and scan reports.
off	In the parm file report, application definitions are *not* printed. In the scan report, scopeux collection times, initial parm file global information, and application definitions are *not* printed. In the analyze report, alarm events and alarm actions are *not* printed.

How to Use It

For explanations of how to use the detail command with the analyze, scan, and parmfile commands, see the analyze, parmfile, and scan command descriptions in this chapter.

Examples

For examples of using the detail command, see the descriptions of the analyze, parmfile, and scan commands in this chapter.

exit

Use the `exit` command to terminate the `utility` program. The `exit` command is equivalent to the `utility` program's `quit` command.

Syntax

```
exit
e
```

guide

Use the guide command to enter guided commands mode. The guided command interface leads you through the various utility commands and prompts you to perform the most common tasks that are available.

Syntax

```
guide
```

Hot to Use It

- To enter guided commands mode from utility's interactive mode, type **guide** and press **Return**.
- To accept the default value for a parameter, press **Return**.
- To terminate guided commands mode and return to interactive mode, type **q** at the guide> prompt.

This command does not provide all possible combinations of parameter settings. It selects settings that should produce useful results for the majority of users.

help

Use the help command to access the utility program's online help facility.

Syntax

```
help [keyword]
```

How to Use It

You can enter parameters to obtain information on utility commands and tasks, or on help itself. You can navigate to different topics by entering a key word. If more than one page of information is available, the display pauses and waits for you to press **Return** before continuing. Type **q** or **quit** to exit the help system and return to the utility program.

You can also request help on a specific topic. For example,

```
help tasks
```

or

```
help resize parms
```

When you use this form of the help command, you receive the help text for the specified topic and remain in the utility command entry context. Because you do not enter the help subsystem interactively, you do not have to type **quit** before entering the next utility command.

list

Use the `list` command to specify the output file for all `utility` reports. The contents of the report depends on which other commands are issued after the `list` command. For example, using the `list` command before the `logfile`, `detail on`, and `scan` commands produces the list file for a detailed summary report of a log file.

Syntax

```
list [filename]|*
```

where * sets the output back to `stdout`.

How to Use It

There are two ways to specify the list file for reports:

- Redirect `stdout` when invoking the `utility` program by typing:

  `utility > utilrept`

- Or, use the `list` command when `utility` is running by typing:

  `list utilrept`

In either case, user interactions and errors are printed to `stderr` and reports go to the file specified.

The *`filename`* parameter in the `list` command must represent a valid filename to which you have write access. Existing files have the new output appended to the end of existing contents. If the file does not exist, it will be created.

To determine the current output file, issue the `list` command without parameters:

If the output file is not `stdout`, most commands are echoed to the output file as they are entered.

Example

The `list` command produces a summary report on the extracted log file `rxlog`. The `list utilrept` command directs the scan report listing to a disk file. `Detail off` specifies less than full detail in the report. The `scan` command reads `rxlog` and produces the report.

The `list *` command sets the list device back to the default `stdout`. `!lp utilrept` sends the disk file to the system printer.

```
utility>
logfile rxlog
list utilrept
detail off
scan
list *
!lp utilrept
```

To perform the above task using command line arguments, enter:

```
utility -l rxlog -f utilrept -d -xs print utilrept
```

logfile

Use the `logfile` command to open a log file. For many `utility` program functions, a log file must be opened. You do this explicitly by issuing the `logfile` command or implicitly by issuing some other command. If you are in batch or command line mode and do not specify a log file name, the default `/var/opt/perf/datafiles/logglob` file is used. If you are in interactive mode and do not specify a log file name, you are prompted to provide one or accept the default `/var/opt/perf/datafiles/logglob` file.

Syntax

```
logfile [logfile]
```

How to Use It

You can specify the name of either a raw or extracted log file. If you specify an extracted log file name, all information is obtained from this single file. You do not need to specify any of the raw log files other than the global log file, `logglob`. Opening `logglob` gives you access to all of the data in the other logfiles.

Raw log files have the following names:

`logglob`	global log file
`logappl`	application log file
`logproc`	process log file
`logdev`	device log file
`logtran`	transaction log file
`logindx`	index log file

Once a log file is opened successfully, a report is printed or displayed showing the general content of the log file (or log files).

You can verify the log file you opened with the `show` command, as described later.

You can open another log file at any time by entering another `logfile` command. Any currently opened log file is closed before the new log file is opened.

The `resize` and `scan` commands require a log file to be open. If no log file is currently open, an implicit `logfile` command is executed.

CAUTION

Do not rename raw log files! Access to these files assumes that the standard log file names are in effect.

You may have more than one set of raw log files on the same system, in separate directories for each set of files. If you want to resize the log files in any way, you must have read/write access to all the log files.

menu

Use the menu command to print a list of the available utility commands.

Syntax

menu

Example

```
utility> menu
Command  Parameters   Function
HELP     [topic]      Get information on commands and options
GUIDE                 Enter guided commands mode for novice
users
LOGFILE  [logname]    Specify a log file to be processed
LIST    [filename|*]  Specify the listing file

START  [startdate time] Set starting date & time for SCAN or
ANALYZE
STOP   [stopdate  time] Set ending   date & time for SCAN or
ANALYZE
DETAIL [ON|OFF]       Set report detail for SCAN, PARMFILE, or
ANALYZE
SHOW   [ALL]          Show the current program settings

PARMFILE [parmfile]   Check parsing of a parameter file

SCAN [logname]  Read the log file and produce a summary report
RESIZE [GLOB|APPL|PROC|DEV|TRAN][DAYS=][EMPTY=] Resize raw log
files

CHECKDEF [alarmdef]   Check parsing and set the alarmdef file
ANALYZE               Analyze the log file using the alarmdef
file

! or Sh [command]    Execute a system command
MENU or ?             List the commands menu (This listing)
EXIT or Q             Terminate the program
utility>
```

parmfile

Use the parmfile command to view and syntax check the OV Performance Agent parm file settings that are used for data collection.

Syntax

```
parmfile [/directorypath/parmfile]
```

How to Use It

You can use the parmfile command to do any of the following:

- Examine the parm file for syntax warnings and review the resulting settings. All parameters are checked for correct syntax and errors are reported. After the syntax check is completed, only the applicable settings are reported.
- Find out how much room is left for defining applications.
- If detail on is specified, print the effective contents of the parm file plus any default settings that were not overridden, and print application definitions.

In batch mode, if no parm file name is specified, the /var/opt/perf/parm file is used.

In interactive mode, if no parm file name is supplied, you are prompted to supply one.

Example

The parmfile command checks the syntax of the current parm file and reports any warnings or errors. Detail on lists the logging parameter settings.

```
utility>
detail on
parmfile parm
```

To perform the above task using command line arguments, enter:

```
utility -xp -D
```

quit

Use the quit command to terminate the utility program. The quit command is equivalent to the utility program's exit command.

Syntax

```
quit
q
```

resize

Use the `resize` command to manage the space in your raw log file set. This is the *only* program you should use to resize the raw log files in order to preserve coordination between the files and their internal control structures. If you use other tools you might remove or destroy the validity of these control structures.

The `utility` program *cannot* be used to resize extracted files. If you want to resize an extracted file, use the `extract` program to create a new extracted log file.

Syntax

```
resize  [global     ] [days=maxdays] [empty=days] [yes  ]
        [application] [size=maxMB  ] [space=MB  ] [no   ]
        [process    ]                             [maybe]
        [device     ]
        [transaction]
```

Parameters

`log file type`	Specifies the type of raw data you want to resize: global, application, process, device, or transaction, which correspond to the raw log files `logglob`, `logappl`, `logproc`, `logdev`, and `logtran`. If you do not specify a data type and are running `utility` in batch mode, the batch job terminates. If you are running `utility` interactively, you are prompted to supply the data type based on those log files that currently exist.
`days & size`	Specify the maximum size of the log file. The actual size depends on the amount of data in the file.
`empty & space`	Specify the minimum amount of room required in the file after the resizing operation is complete. This value is used to determine if any of the data currently in the log file must be removed in the resizing process.

You might expect that a log file would not fill up until the specified number of days after a resizing operation. You may want to use this feature of the resize command to minimize the number of times a log file must be resized by the scopeux collector because resizing can occur any time the file is filled. Using resize to force a certain amount of empty space in a log file causes the log file to be resized when you want it to be.

The days and empty values are entered in units of days; the size and space values are entered in units of megabytes. Days are converted to megabytes by using an average megabytes-per-day value for the log file. This conversion factor varies depending on the type of data being logged and the particular characteristics of your system.

More accurate average-megabytes-per-day conversion factors can be obtained if you issue the scan command on the existing log file before you issue the resize command. A scan measures the accumulation rates for your system. If no scan is done or if the measured conversion factor seems unreasonable, the resize command uses a default conversion factor for each type of data.

yes	Specifies that resizing should be unconditionally performed. This is the default action if utility is not running interactively. If no action is specified when utility is running interactively, you are prompted to supply the action.
no	Specifies that resizing should not be performed. This parameter can be specified as an action if you want to see the resizing report but do not want to perform the resizing at that time.
maybe	Specifies that utility should decide whether or not to resize the file. This parameter forces utility to make this decision based on the current amount of empty space in the log file (before any resizing) and the amount of space specified in the resize command. If the current log file contains at least as much empty space as specified, resizing does *not* occur. If the current log file contains less than the specified empty space, resizing occurs.

`maybe` (*continued*)	If the resizing can be made without removing any data from the log file (for example, increasing the maximum log file size, or reducing the maximum log file size without having to remove any existing data), resizing occurs. The `maybe` parameter is intended primarily for use by periodic batch executions. See the "Examples" subsection below for an explanation of how to use the resize command in this manner.

Default resizing parameters are shown in the following table.

Table 4-2 Default Resizing Parameters

Parameter	If Executed Interactively	If Executed in Batch
`log file type`	You are prompted for each available log file type.	No default. This is a required parameter.
`days` `size`	The current file size.	The current file size.
`empty space`	The current amount of empty space or enough empty space to retain all data currently in the file, whichever is smaller.	The current amount of empty space or enough empty space to retain all data currently in the file, whichever is smaller.

`yes`
`no`
`maybe`
	You are prompted following the reported disk space results.	Yes. Resizing will occur.

How to Use It

Before you resize a log file, you *must* stop OV Performance Agent using the steps under "Stopping and Restarting Data Collection" on page 49 in Chapter 2, "Managing Data Collection".

A raw log file must be opened before resizing can be performed. Open the raw log file with the `logfile` command before issuing the `resize` command. The files cannot be opened by any other process.

The resize command creates the new file /tmp/scopelog before deleting the original file. Make sure there is sufficient disk space in the /var/tmp directory (/tmp on IBM AIX 4.1 and later) to hold the original log file before doing the resizing procedure.

After resizing, a log file consists of data plus empty space. The data retained is calculated as the maximum file size minus the required empty space. Any data removed during the resizing operation is lost. To save log file data for longer periods, use extract to copy this data to an extracted file *before* doing the resize operation.

Resize Command Reports

One standard report is produced when you resize a raw log file. It shows the three interrelated disk space categories of maximum file size, data records, and empty space, before and after resizing. For example:

```
resize global days=120;empty=10
empty space raised to match file size and data records

final resizing parameters:
file: logglob                        megabytes / day: 0.101199
                 ---currently-----    --after resizing---
maximum size:   65 days (  6.6 mb)   120 days ( 12.1 mb)  83%
                                  increase
data records:   61 days (  6.2 mb)    61 days (  6.2 mb) no data
                                  removed
empty space:     4 days (  0.5 mb)    59 days (  6.0 mb) 1225%
                                    increase
```

The megabytes per day value is used to convert between days and megabytes. It is either the value obtained during the scan function or a default for the type of data being resized.

The far right-hand column is a summary of the net change in each category of log file space. Maximum size and empty space can increase, decrease, or remain unchanged. Data records have either no data removed or a specified amount of data removed during resizing.

If the resize is done interactively and one or more parameters are defaults, you can get a preliminary resizing report. This report summarizes the current log file contents and any parameters that were provided. The report is provided to aid in answering questions on the unspecified parameters. For example:

```
resize global days=20

file resizing parameters (based on average daily
space estimates and user resizing parameters)
file: logglob                      megabytes / day: 0.101199
                -----currently-----   --after resizing---
maximum size:   65 days (  6.6 mb)    20 days (  2.0 mb)
data records:   61 days (  6.2 mb)    ??
empty space:     4 days (  0.5 mb)    ??
```

In this example, you are prompted to supply the amount of empty space for the file before the final resizing report is given. If no action parameter is given for interactive resizing, you are prompted for whether or not to resize the log file immediately following the final resizing report.

Examples

The following commands are used to resize a raw process log file. The scan is performed before the resize to increase the accuracy of the number-of-days calculations.

```
logfile /var/opt/perf/datafiles/logglob
detail off
scan
resize process days=60 empty=30 yes
```

`days=60` specifies holding a maximum of 60 days of data. `empty=30` specifies that 30 days of this file are currently empty. That is, the file is resized with no more than 30 days of data in the file to leave room for 30 more days out of a total of 60 days of space. `yes` specifies that the resizing operation should take place regardless of current empty space.

The next example shows how you might use the `resize` command in batch mode to ensure that log files do not fill up during the upcoming week (forcing `scopeux` to resize them). You could schedule a `cron` script using the `at` command that specifies a minimum amount of space such as 7 days - or perhaps 10 days, just to be safe.

The following shell script accomplishes this:

```
echo detail off                          > utilin
echo scan                               >> utilin
echo resize global      empty=10 maybe >> utilin
echo resize application empty=10 maybe >> utilin
echo resize process     empty=10 maybe >> utilin
echo resize device      empty=10 maybe >> utilin
echo quit                               >> utilin
utility < utilin > utilout 2> utilerr
```

Specifying `maybe` instead of `yes` avoids any resizing operations if 10 or more days of empty space currently exist in any log files. Note that the maximum file size defaults to the current maximum file size for each file. This allows the files to be resized to new maximum sizes without affecting this script.

NOTE If you use the script described above, remember to stop scopeux before running it. See the "Starting & Running OV Performance Agent" chapter in your *HP OpenView Performance Agent Installation & Configuration Guide* for information about stopping and starting scopeux.

scan

Use the scan command to read a log file and write a report on its contents. (For a detailed description of the report, see “Utility Scan Report Details” on page 66 in Chapter 3, “Using the Utility Program”.

Syntax

scan

How to Use It

The scan command requires a log file to be opened. The log file scanned is the first of one of the following:

- The log file named in the scan command itself.
- The last log file opened by any previous command.
- The default log file.

  In this case, interactive users are prompted to override the default log file name if desired.

The following commands affect the operation of the scan function:

detail	Specifies the amount of detail in the report. The default, detail on, specifies full detail.
list	Redirects the output to another file. The default is to list to the standard list device.
start	Specifies the date and time of the first log file record you want to scan. The default is the beginning of the log file.
stop	Specifies the date and time of the last log file record you want to scan. The default is the end of the log file.

For more information about the detail, list, start, and stop commands, see their descriptions in this chapter.

The scan command report consists of 12 sections. You can control which sections are included in the report by issuing the detail command prior to issuing scan.

The following four sections are always printed (even if `detail off` is specified):

- Scan start and stop actual dates and times
- Collector coverage summary
- Log file contents summary
- Log file empty space summary

The following sections are printed if `detail on` (the default) is specified:

- Initial parm file global information and system configuration information
- Initial `parm` file application definitions
- `parm` file global changes
- `parm` file application addition/deletion notifications
- Collector off-time notifications
- Application-specific summary reports

The following section is always printed if application data was scanned (even if `detail off` is specified):

- Application overall summary

The following section is always printed if process data was scanned (even if `detail off` is specified):

- Process log reason summary

Example

The scan of the current default global log file starts with records logged from June 1, 1999 at 7:00 AM until the present date and time.

```
utility>
logfile /var/opt/perf/datafiles/logglob
detail on
start 6/1/99 7:00 am
scan
```

To perform the above task using command line arguments, enter:

```
utility -D -b 6/1/99 7:00 am -xs
```

sh

Use sh to enter a shell command without exiting utility by typing **sh** or an exclamation point (**!**) followed by a shell command.

Syntax

```
sh or ! [shell command]
```

Parameters

sh ls	Executes the ls command and returns to utility.
!ls	Same as above.

How to Use It

Following the execution of the single command, you automatically return to utility. If you want to issue multiple shell commands without returning to utility after each one, you can start a new shell. For example,

```
sh ksh
```

or

```
!ksh
```

show

Use the show command to list the names of the files that are open and the status of the utility parameters that can be set.

Syntax

```
show [all]
```

Examples

Use show to produce a list that may look like this:

```
Logfile:  /var/opt/perf/datafiles/logglob
List:     "stdout"
Detail:   ON for ANALYZE, PARMFILE and SCAN functions

The default starting date & time = 10/08/99 08:17 AM (FIRST + 0)
The default stopping date & time = 11/20/99 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM
```

NOTE The default shift time is shown for information only. Shift time cannot be changed in utility.

Use show all to produce a more detailed list that may look like this:

```
Logfile: /var/opt/perf/datafiles/logglob

    Global      file: /var/opt/perf/datafiles/logglob
    Application file: /var/opt/perf/datafiles/logappl
    Process     file: /var/opt/perf/datafiles/logproc
    Device      file:  /var/opt/perf/datafiles/logdev
    Transaction file: /var/opt/perf/datafiles/logtran
    Index       file: /var/opt/perf/datafiles/logindx
    System ID:  homer
    System Type 9000/715 S/N 66677789   OS/ HP-UX B.10.20 A
    Data Collector: SCOPE/UX C.02.30
    File created:   10/08/99
    Data Covers:    44 days to 11/20/99
    Shift is:       All Day
Data records available are:
  Global Application Process Disk Volume Transaction

Maximum file sizes:
  Global=10.0 Application=10.0 Process=20.0 Device=10.0 Transaction
10.0 MB
```

```
List    "stdout"
Detail  ON for ANALYZE, PARMFILE and SCAN functions

The default starting date & time = 10/08/99 11:50 AM (FIRST + 0)
The default stopping date & time = 11/20/99 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM
```

start

Use the `start` command to specify the beginning of the subset of a log file that you want to scan or analyze. `Start` lets you start the scan or analyze process at data that was logged at a specific date and time.

The default starting date and time is set to the date and time of the first record of any type in a log file that has been currently opened with the `logfile` command.

Syntax

```
                [date     [time]]
start           [today    [-days]   [time]]
                [last     [-days]   [time]]
                [first    [+days]   [time]]
```

Parameters

date — The date format depends on the native language configured on the system being used. If you do not use native languages or have the default language set to C, the date format is *mm/dd/yy* (month/day/year) or 06/30/99 for June 30, 1999.

time — The time format also depends on the native language being used. For C, the format is *hh:mm* am or *hh:mm* pm (hour:minute in 12-hour format with the am/pm suffix) such as 07:00 am for 7 o'clock in the morning. Twenty-four hour time is accepted in all languages. For example, 23:30 would be accepted for 11:30 pm.
If the date or time is entered in an unacceptable format, an example in the correct format is shown.
If no start time is given, a midnight (12 am) is assumed. A starting time of midnight for a given day starts at the *beginning* of that day (00:00 on a 24-hour clock).

today — Specifies the current day. The parameter `today-`*days* specifies the number of days prior to today's date. For example, `today-1` indicates yesterday's date and `today-2`, the day before yesterday.

`last`	Can be used to represent the last date contained in the log file. The parameter `last-`*`days`* specifies the number of days prior to the last date in the log file.
`first`	Can be used to represent the first date contained in the log file. The parameter `first+`*`days`* specifies the number of days after the first date in the log file.

How to Use It

The `start` command is useful if you have a very large log file and do not want to scan or analyze the entire file. You can also use it to specify a specific time period for which data is scanned. For example, you can scan a log file for data that was logged for a period beginning 14 days before the present date by specifying `today-14`.

You can use the `stop` command to further limit the log file records you want to scan.

If you are not sure whether native language support is installed on your system, you can force `utility` to use the C date and time formats by issuing the following statement before running `utility`:

```
LANG=C; export LANG
```

or in C Shell

```
setenv LANG C
```

Example

The scan of the default global log file starts with records logged from August 5, 1999 at 8:00 AM until the present date and time.

```
utility>
logfile /var/opt/perf/datafiles/logglob
detail on
start 8/5/99 8:00 AM
scan
```

To perform the above task using command line arguments, enter:

```
utility -D -b 8/5/99 8:00 am -xs
```

stop

Use the `stop` command to specify the end of a subset of a log file that you want to scan or analyze. `Stop` lets you terminate the scan or analyze process at data that was logged at a specific date and time.

The default stopping date and time is set to the date and time of the last record of any type in a log file that has been currently opened with the `logfile` command.

Syntax

```
                [date     [time]]
stop            [today    [-days]   [time]]
                [last     [-days]   [time]]
                [first    [+days]   [time]]
```

Parameters

date The date format depends on the native language configured on the system being used. If you do not use native languages or have the default language set to C, the date format is *mm/dd/yy* (month/day/year) or 06/30/99 for June 30, 1999.

time The time format also depends on the native language being used. For C, the format is *hh:mm* am or *hh:mm* pm (hour:minute in 12-hour format with the am/pm suffix) such as 07:00 am for 7 o'clock in the morning. Twenty-four hour time is accepted in all languages. For example, 23:30 would be accepted for 11:30 pm.
If the date or time is entered in an unacceptable format, an example in the correct format is shown.
If no stop time is given, one minute before midnight (11:59 pm) is assumed. A stopping time of midnight (12 am) for a given day stops at the *end* of that day (23:59 on a 24-hour clock).

`today`	Specifies the current day. The parameter `today-`*`days`* specifies the number of days prior to today's date. For example, `today-1` indicates yesterday's date and `today-2`, the day before yesterday.
`last`	Can be used to represent the last date contained in the log file. The parameter `last-`*`days`* specifies the number of days prior to the last date in the log file.
`first`	Can be used to represent the first date contained in the log file. The parameter `first+`*`days`* specifies the number of days after the first date in the log file.

How to Use It

The `stop` command is useful if you have a very large log file and do not want to scan the entire file. You can also use it to specify a specific time period for which data is scanned. For example, you can scan a log file for seven-days worth of data that was logged starting a month before the present date.

If you are not sure whether native language support is installed on your system, you can force `utility` to use the C date and time formats by issuing the following statement before running `utility`:

```
LANG=C; export LANG
```

or in C Shell

```
setenv LANG C
```

Example

The scan of 14 days worth of data starts with records logged from July 5, 1999 at 8:00 AM and stops at the last record logged July 18, 1999 at 11:59 pm.

```
utility>
logfile /var/opt/perf/datafiles/logglob
detail on
start 7/5/99 8:00 am
stop 7/18/99 11:59 pm
scan
```

To perform the above task using command line arguments, enter:

```
utility -D -b 7/5/99 8:00 am -e 7/18/99 11:59pm -xs
```

5 Using the Extract Program

Introduction

The extract program has two main functions: it lets you extract data from raw log files and write it to extracted log files. Extract also lets you export log file data for use by analysis products such as spreadsheets.

NOTE After the initial installation of OV Performance Agent, services must be started for file installation to complete, before extract will function.

The extract and export functions *copy* data from a log file; *no* data is removed.

Three types of log files are used by OV Performance Agent:

- scopeux log files, which contain data collected in OV Performance Agent by the scopeux collector.
- extracted log files, which contain data extracted from raw scopeux log files.
- DSI (data source integration) log files, which contain user-defined data collected by external sources such as applications and databases. The data is subsequently logged by OV Performance Agents DSI programs.

Use the extract program to perform the following tasks:

- Extract subsets of data from raw scopeux log files into an extracted log file format that is suitable for placing in archives, for transport between systems, and for analysis by OV Performance Manager. Data *cannot* be extracted from DSI log files.
- Manage archived log file data by extracting or exporting data from extracted format files, appending data to existing extracted log files, and subsetting data by type, date, and shift (hour of day).
- Export data from raw or extracted scopeux log files and DSI log files into ASCII, binary, datafile, or WK1 (spreadsheet) formats suitable for reporting and analysis or for importing into spreadsheets or similar analysis packages.

NOTE The `extract` function cannot produce summarized data. Summary data can only be produced by the `export` function.

Examples of how various tasks are performed and how `extract` commands are used can be found in online help for the `extract` program.

This chapter covers the following topics:

- running the `extract` program
- using interactive mode
- command line interface
- overview of the export function

Running the Extract Program

There are three ways to run the `extract` program:

- Command line mode - You control the `extract` program using command options and arguments in the command line.
- Interactive mode - You supply interactive commands and parameters while executing the program with `stdin` set to an interactive terminal or workstation.
  If you are an experienced user, you can quickly specify only those commands required for a given task. If you are a new user, you may want to specify guided mode to receive more assistance in using `extract`. In guided mode, you are asked to select from a list of options in order to perform a task. While in guided mode, the interactive commands that accomplish each task are listed as they are executed, so you can see how they are used. You can quit or re-enter guided mode at any time.
- Batch mode - You can run the program and redirect `stdin` to a file that contains interactive commands and parameters.

The syntax for the command line interface is similar to standard UNIX command line interfaces on other programs and is described in detail in this chapter.

For interactive and batch mode the command syntax is the same: a command followed by one or more parameters. Commands can be entered in any order; if a command has a parameter associated with it, the parameter must be entered immediately after the corresponding command.

There are two types of parameters - *required* (for which there are no defaults) and *optional* (for which defaults are provided). How the extract program handles these parameters depends on the mode in which it is running.

- In interactive mode, if an *optional* parameter is missing, the program displays the default parameter and lets you either confirm it or override it.
  If a *required* parameter is missing, the program prompts you to enter the parameter.

- In batch mode, if an *optional* parameter is missing, the program uses the default values.
  If a *required* parameter is missing, the program terminates.

Errors and missing data are handled differently for interactive mode than for command line and batch mode, because you can supply additional data or correct mistakes in interactive mode, but not in command line and batch mode.

Using Interactive Mode

Using the `extract` program's interactive mode requires you to issue a series of commands to execute a specific task.

For example, if you want to export application data collected starting May 15, 2002, from the default global log file, you issue the following commands after invoking the `extract` program

```
logfile /var/opt/perf/datafiles/logglob
application detail
start 5/15/2001
export
```

The `logfile` command opens `/var/opt/perf/datafiles/logglob`, the default global log file. The `start` command specifies that only data logged after 5/15/01 will be exported. The `export` command starts the exporting of the data.

Extract Command Line Interface

In addition to the interactive and batch mode command syntax, command options and arguments can be passed to the `extract` program through the command line interface. The command line interface fits into the typical UNIX environment by allowing the `extract` program to be easily invoked by shell scripts and allowing its input and output to be redirected into UNIX pipes.

For example, the command line equivalent of the example shown in the previous section “Using Interactive Mode” on page 116 is:

```
extract -l -a -b 5/15/02 -xp
```

In command line mode, the global log file `/var/opt/perf/datafiles/logglob` is the default; you do not have to specify it.

Command line options and arguments are listed in the following table. The referenced command descriptions can be found in Chapter 6, “Extract Commands,” on page 145.

Table 5-1 Command Line Arguments

Command Option	Argument		Description
`-b`	*date*	*time*	Specifies starting date and time of an extract or export function. (See `start` command in Chapter 6.)
`-B`		`UNIX` *start time*	Specifies starting time in UNIX format for an extract or export function.
`-e`	*date*	*time*	Specifies ending date and time of an extract or export function. (See `stop` command in Chapter 6.)

Table 5-1 Command Line Arguments (Continued)

Command Option	Argument		Description
-E		UNIX *stop time*	Specifies stopping time in UNIX format for an extract or export function.
-s	*time-time*	*noweekends*	Specifies start and end time for specific periods excluding weekends. (See "shift" command in Chapter 6.)
-l	*logfile*		Specifies input log file. (See "logfile" command in Chapter 6.) /var/opt/perf/datafiles/logglob is the default.
-r	*export template file*		Specifies an export template file for export function. (See "report" command in Chapter 6.)
-C	*classname*	opt	Specifies scopeux data to extract or export, or self-describing (DSI) data to export. (See "class" command in Chapter 6.) opt = detail (default) summary both off

Table 5-1 Command Line Arguments (Continued)

Command Option	Argument		Description
gapkcdzntuy GADZNTUY			Specifies types of data to extract/export: g = global detail. (See "global" command in Chapter 6.) global detail is off by default. a = application detail. (See "application" command in Chapter 6.) p = process detail (See "process" command in Chapter 6.) k = process killed. (See "process" command in Chapter 6.) c = configuration detail (See "configuration" command in Chapter 6.) d = disk device detail (See "disk" command in Chapter 6.) z = lvolume detail (See "lvolume" command in Chapter 6.) n = netif detail (See "netif" command in Chapter 6.)

Table 5-1 Command Line Arguments (Continued)

Command Option	Argument		Description
gapkcdzntuy GADZNTUY (*continued*)			t = transaction detail (See "transaction" command in Chapter 6.) u = CPU detail (See "cpu" command in Chapter 6.) y = filesystem detail (See "filesystem" command in Chapter 6.) **NOTE**: The following summary options are for export only; the extract function does not support data summarization. G = global summary (See "global" command in Chapter 6.) Global summary is off by default. A = application summary (See "application" command in Chapter 6.) D = disk device summary (See "disk" command in Chapter 6.) Z=lvolume summary (See "lvolume" command in Chapter 6.) N = netif summary (See "netif" command in Chapter 6.)

Table 5-1 Command Line Arguments (Continued)

Command Option	Argument		Description
gapkcdzntuy GADZNTUY (*continued*)			T = transaction summary (See "transaction" command in Chapter 6.) U = CPU summary (See "cpu" command in Chapter 6.) Y = filesystem summary (See "filesystem" command in Chapter 6.)
-v			Generates verbose output report formats.
-f	*filename*	,new ,append ,purge	Sends extract or export data to a file. If no filename, sends data to default output files. (See "output" command in Chapter 6.)
-ut			Shows date and time in UNIX format in exported DSI log file data.
-we	1.....7		Specifies days to exclude from export; 1=Sunday. (See "weekdays" command description.)
-xp	xopt		Exports data to external format files. (See "export" command in Chapter 6.)

Table 5-1 Command Line Arguments (Continued)

Command Option	Argument		Description
-xt	xopt		Extracts data in system internal format. (See "extract" command in Chapter 6.) xopt = *dwmy* (Day Week Month Year) *dwmy*-[offset] *dwmy* [absolute]
-xw	week		Extracts a calendar week's data. (See "weekly" command in Chapter 6.)
-xm	month		Extracts a calendar month's data. (See "monthly" command in Chapter 6.)
-xy	year		Extracts a calendar year's data. (See "yearly" command in Chapter 6.)
-? or ?			Displays command line syntax.

When you are evaluating arguments and entering command options on the command line, the following rules apply:

- Errors and missing data are handled exactly as in the corresponding batch mode command. That is, missing data will be defaulted if possible and all errors cause the program to terminate immediately.
- Echoing of commands and command results is disabled unless the -v argument is used to enable verbose mode.
- If no valid action is specified (-xp, -xw, -xm, -xy, or -xt), extract starts reading commands from its stdin file after all parameters have been processed.

- If an action is specified (`-xp`, `-xw`, `-xm`, `-xy`, or `-xt`), the program will execute those command options after all other parameters are evaluated, regardless of where they were positioned in the list of parameters.
- If an action is specified in the command line, the `extract` program will not read from its `stdin` file; instead it will terminate following the action:

```
extract -f rxdata -r /var/opt/perf/rept1 -xp d-1 -G
```

Which translates into:

`-f rxdata`	Outputs to a file named `rxdata` in current directory
`-r rept1`	File `/var/opt/perf/rept1` contains the desired export format
`-xp d-1`	Exports data for this day minus 1 (yesterday)
`-G`	Exports global summary data.

Note that the actual exporting is not done until the end so the `-G` parameter is processed before the export is done.

Also notice that the log file was not specified so it uses the default `logglob` file.

Because an action was specified (`-xp`), once the export is finished the `extract` program terminates without reading from its `stdin` file. In addition, verbose mode was not set with the `-v` command option so all extraneous output to `stdout` is eliminated.

Overview of the Export Function

The `extract` program's `export` command converts OV Performance Agent raw, extracted, or DSI log file data into exported files. The export command writes files in any one of four possible formats: ASCII, datafile, binary, and WK1 (spreadsheet). Exported files can be used in a variety of ways, such as reports, custom graphics packages, databases, and user-written analysis programs.

How to Export Data

In the simplest form, you can export data by:

- specifying the default global log file, `/var/opt/perf/datafiles/logglob`, from which you want to export data
- specifying the default export template file, `/var/opt/perf/reptfile`, that defines the format of the exported data
- starting the export function.

The exported data is placed in a default output file named `xfrdGLOBAL.asc` in your current directory. The output file's ASCII format is suitable for printing.

If you want to export something other than this default set of data, you can use other commands and files in conjunction with the `export` command.

You can export the following types of data:

global	5-minute and hourly summaries
application	5-minute and hourly summaries
process	One-minute details
disk device	5-minute and hourly summaries
lvolume	5-minute and hourly summaries
transaction	5-minute and hourly summaries

configuration	One record containing parm file information, and system configuration information, for each time the data collector started.
any DSI class	Intervals and summaries for DSI log files
netif	5-minute and hourly summaries
cpu	5-minute and hourly summaries
filesystem	5-minute and hourly summaries

- You can specify which data items (metrics) are needed for each type of data.
- You can specify starting and ending dates for the time period in which the data was collected along with shift and weekend exclusion filters.
- You can specify the desired format for the exported data in an export template file. This file can be created using any text editor or word processor that lets you save a file in ASCII (text) format.
- You can also use the default export template file, `/var/opt/perf/reptfile`. This file specifies the following output format settings:
  - ASCII file format
  - a 0 (zero) for the missing value
  - a blank space as the field separator
  - 60-minute summaries
  - column headings are included
  - a recommended set of metrics for a given data type is included in the export

Sample Export Tasks

Two sample export template files, `repthist` and `reptall`, are furnished with OV Performance Agent. These files are located in the `/var/opt/perf/` directory. You can use `repthist` and `reptall` to perform common export tasks or as a starting point for custom tasks, such as the task described next.

Generating a Printable CPU Report

The repthist export template file contains the specifications to generate a character graph of CPU and disk usage for the system over time. This graph consists of printable characters that can be printed on any device capable of 132 column printing. For example, you could use the following extract program commands to generate a graph of the last seven days and should produce approximately two pages (34 pages if 5-minute detail is specified instead of hourly summaries).

```
logfile /var/opt/perf/datafiles/logglob
report /var/opt/perf/repthist
global summary
start today-7
export
```

The exported data is in an export file named xfrsGLOBAL.asc. To print it, type:

```
lp xfrsGLOBAL.asc
```

Producing a Customized Export File

If you want to create a totally new export template file, copy the export template file and customize it using the extract program's guide command. In guided mode, you copy the reptall file from the /var/opt/perf/ directory and read the scopeux or DSI log file specified to dynamically create the list of data types and metric names.

The reptall file contains every possible metric for each type of scopeux log file data so you need only uncomment those metrics that are of interest to you. This is easier than retyping the entire export template file.

Export Data Files

If you used the output command to specify the name of an output file prior to issuing the export command, all exported data will be written to this single file. If you are running the extract program interactively and want to export data directly to your workstation (standard output file), specify the extract command output stdout prior to issuing the export command.

If the output file is set to the default, the exported data is separated into as many as 14 different default output files depending on the type of data being exported.

The default export log file names are:

`xfrdGLOBAL.ext`	Global detail data file
`xfrsGLOBAL.ext`	Global hourly summary data file
`xfrdAPPLICATION.ext`	Application detail data file
`xfrsAPPLICATION.ext`	Application hourly summary data file
`xfrdPROCESS.ext`	Process detail data file
`xfrdDISK.ext`	Disk device detail data file
`xfrsDISK.ext`	Disk device hourly summary data file
`xfrdVOLUME.ext`	Logical volume detail data file
`xfrsVOLUME.ext`	Logical volume summary data file
`xfrdNETIF.ext`	Netif detail data file
`xfrsNETIF.ext`	Netif summary detail data file
`xfrdCPU.ext`	CPU detail data file
`xfrsCPU.ext`	CPU summary data file
`xfrdFILESYSTEM.ext`	Filesystem detail data file
`xfrsFILESYSTEM.ext`	Filesystem summary data file
`xfrdTRANSACTION.ext`	Transaction detail data file
`xfrsTRANSACTION.ext`	Transaction summary data file
`xfrdCONFIGURATION.ext`	Configuration data file

where `ext`= `asc` (ASCII), `bin` (binary), `dat` (datafile), or `wk1` (spreadsheet).

NOTE

No output file is created *unless* you specify the type and associated items that match the data in the export template file prior to issuing the export command.

Export Template File Syntax

The export template file can contain all or some of the following information, depending on how you want your exported data to be formatted and what you want the export file to contain:

```
report      "export file title"

format      [ASCII]
            [datafile]
            [binary]
            [WK1] or
            [spreadsheet]

headings    [on]
            [off]
separator= "char"
summary=value
missing=value
layout=single | multiple
output=filename
data type datatype
items
```

Parameters

`report` — Specifies the title for the export file. For more information, see the following section, “Export File Title” on page 132.

`format` — Specifies the format for the exported data.

ASCII

ASCII (or text) format is best for copying files to a printer or terminal. It does not enclose fields with double quotes (").

Datafile

The datafile format is similar to ASCII format except that non-numerical fields are enclosed in double quotes. Because double quotes prevent strict column alignment, files in datafile format are not recommended for direct printing. The datafile format is the easiest format to import into most spreadsheets and graphics packages.

Binary

The binary format is more compact because numerical values are represented as binary integers. It is the most suitable format for input into user-written analysis programs because it needs the least conversion, and it maintains the highest metric accuracy. It is not suitable for direct printing.

WK1 (spreadsheet)

The WK1 (spreadsheet) format is compatible with Microsoft Excel and other spreadsheet and graphics programs.

headings — Specifies whether or not to include column headings for the metrics listed in the export file. If headings off is specified, no column headings are written to the file. The first record in the file is exported data. If headings on is specified, ASCII and datafile formats place the export title plus column headings for each column of metrics written *before* the first data records. Column headings in binary format files contain the description of the metrics in the file. WK1 formats always contain column headings.

separator — Specifies the character that is printed between each field in ASCII or datafile formatted data. The default separator character is a blank space. Many programs prefer a comma as the field separator. You can specify the separator as any printing or nonprinting character.

`summary` Specifies the number of minutes for each summary interval. The value determines how much time is included in each record for summary records. The default interval is 60 minutes. The summary value can be set between 5 and 1440 minutes (1 day).

`missing` Specifies the data value to be used in place of missing data. The default value for missing data is zero. You can specify another value in order to differentiate missing from zero data.
A data item may be missing if it was:

- not logged by a particular version of the `scopeux` collector
- not logged by `scopeux` because the instance (application, disk, transaction, netif) it belongs to was not active during the interval, or
- in the case of DSI log files, no data was provided to the `dsilog` program during an interval, resulting in "missing records".

Missing records are, by default, excluded from exported data.

`layout` Specifies either single or multiple layouts (per record output) for multi-instance data types such as application, disk, transaction, lvolume, or netif.

Single layout writes one record for every application (disk, transaction, etc.) that was active during the time interval. Multiple layout writes one record for each time interval, with part of that record reserved for each configured application.

`output` Specifies where exported data is to be output. It can be specified for each class or data type exported by placing `output` *`filename`* just after the line indicating the *`data type`* that starts the list of exported data items. Any valid file name can be specified with `output`.

You can also override the default output file name by specifying the name interactively using the `output` command.

data type	Specifies one of the exportable data types: global, application, process, disk, transaction, lvolume, netif, configuration, or DSI class name. This starts a section of the export template file that lists the data items to be copied when this type of data is exported.
items	Specifies the metrics to be included in the exported file. Metric names are listed, one per line, in the order you want them listed in the resulting file. You must specify the proper *data type* before listing *items*. The same export template file can include item lists for as many data types as you want. Each *data type* will be referenced *only* if you choose to export that type of data.

The output and layout parameters can be used more than once within an export template file. For example:

```
data type global
   output=myglobal
   gbl_cpu_total_util

data type application
   output=myapp
   layout=multiple
   app_cpu_total_util
```

You can have more than one export template file on your system. Each one can define a set of exported file formats to suit a particular need. You use the report command to specify the export template file to be used with the export function.

NOTE You cannot specify different layouts within a single data type. For example, you cannot specify data type disk once with layout = multiple and again with layout = single within the same export file.

Export File Title

The following items can be substituted in the `export file title` string:

`!date`	The date the export function was performed.
`!time`	The time the export function was performed.
`!logfile`	The fully qualified name of the source log file.
`!class`	The type of data requested.
`!collector`	The name and version of the collector program. (Not valid with DSI log files.)
`!system_id`	The identifier of the system that collected the data. (Not valid with DSI log files.)

For example, the string

```
report "!system_id data from !logfile on !date !time"
```

generates an export file title similar to

```
barkley data from logglob on 02/02/99 08:30 AM
```

Creating a Custom Graph or Report

Suppose you want to create a custom graph or report containing exported global and application data. You would do the following:

1. Determine which data items (metrics) are needed from each data type and in what format you will access them.

   For this example, you want an ASCII file without headings and with fields separated by commas.

2. Create and save the following ASCII export template file in the `/var/opt/perf/` directory. Name the file `report1`.

```
REPORT "sample export template file (report1)"
FORMAT ASCII
HEADINGS OFF

DATA TYPE GLOBAL
   GBL_CPU_TOTAL_UTIL
   GBL_DISK_PHYS_IO_RATE
```

```
DATA TYPE APPLICATION
   APP_CPU_TOTAL_UTIL
   APP_DISK_PHYS_IO_RATE
   APP_ALIVE_PROCESSES
```

3. Run the `extract` program.
4. Issue the `report` command to specify the export template file you created.

   `report /var/opt/perf/report1`

5. Specify global summary data and application summary data using the `global` and `application` commands.

   ```
   global summary
   application summary
   ```

6. Issue the `export` command to start the export.

   `export`

7. Because you did not specify where the program should get the performance data from, you are prompted to do so. In this example, since the default log file is correct, just press **Enter**.
8. The output looks like this:

   ```
   exporting global data ........50%......100%
   exporting application data .....50%......100%

   The exported file contains 31 days of data from 01/01/99
   to 01/31/99

                              examined  exported
   data type                   records   records     space
   ------------------------  --------- --------- ---------
   global      summaries                   736   0.20 Mb
   application summaries                  2560   0.71 Mb
                                                 ---------
                                                   0.91 Mb
   ```

The two files you have just created — `xfrsGLOBAL.asc` and `xfrsAPPLICATION.asc` — contain the global and application summary data in the specified format.

Output of Exported Files

The contents of each exported file are:

`export tittle line`	If `export title` and `headings on` were specified.
`Names (application, netif, lvolume, or transaction)`	If `headings on` was specified along with a multiple layout file.
`Heading line1`	If `headings on` was specified.
`Heading line2`	If `headings on` was specified.
`first data record`	
`second data record`	
`...`	
`last data record`	

Report title and heading lines are not repeated in the file.

Notes on ASCII and Datafile Formats

The data in these format files is printable `ASCII` format. `ASCII` and `datafile` formats are identical except that in the latter, all non-numeric fields are enclosed with double quotes. Even the `datafile` header information is enclosed with double quotes.

The `ASCII` file format does not enclose fields with double quotes. Therefore, the data in `ASCII` files will be properly aligned when printed.

Numerical values are formatted based on their range and internal accuracy. Since all fields will not be the same length, be sure to specify the separator you want to use to start each field.

The user-specified separator character (or the default blank space) separates the individual fields in `ASCII` and `datafile` formats. Blank spaces, used as separators, can be visually more attractive if you plan to print the report. Other characters can be more useful as separators if you plan to read the export template file with another program.

Using the comma as a separator is acceptable to many applications, but some data items may contain commas *that are not separators*. These commas can confuse analysis programs. The date and time formats can contain different special characters based on the native language specified when you execute the extract program.

NOTE To use a nonprinting special character as a separator, enter it into your export template file immediately following the first double quote in the `separator` parameter.

Hints

- Most spreadsheets accept files in `datafile` format using `separator=",".`
- Many spreadsheet packages accept a maximum of 256 columns in a single sheet. Use care when exporting multiple layout types of data because it is easy to generate more than 256 total items. You can use the output of the `report` *`reportfile`*`,show` command to determine if you are likely to see this problem.
- If you have a printer that supports underlining, you can create a more attractive printout by specifying `ASCII` format and the vertical bar character (`separator=|`) and then printing the file with underlining turned on.

Notes on Binary Format

In `binary` format files, numerical values are written as 32-bit integers. This can save space by reducing the overall file size, but your program must be able to read `binary` files. We do not recommend copying a `binary` format file to a printer or a terminal.

In `binary` format, non-numerical data is written the same as it was in the `ASCII` format except separator characters are not used. To properly use a `binary` format file, you should use the record layout report printed by `extract` when you specify `report` *`reportfile`*`,show`. This report gives you the starting byte for each item specified.

To maintain maximum precision and avoid nonstandard, binary floating-point representations, all numerical values are written as scaled, 32-bit integers. Some items might be multiplied by a constant before they are truncated into integer format.

For example, the number of seconds the CPU was used is multiplied by 1000 before being truncated. To convert the value in the exported file back to the actual number of seconds, divide it by 1000. For ease in conversion, specify headings on to write the scale factors to the exported file. The report title and special header records are written to binary format files to assist in programmatic interpretation.

Binary integers are written in the format that is native to the system on which the extract program is being run. For example, Intel systems write “little endian” integers while HP-UX, IBM AIX, and Sun systems write “big endian” integers. Use care when transporting binary exported files to systems that use different “"endians”.

Binary Header Record Layout

Each record in a binary format exported file contains a special 16-byte record header preceding any user-specified data. The report *reportfile*, show command includes the following four fields that make up this record header:

Binary Record Header Metrics

Record Length	Number of bytes in the record, including the 16 byte record header.
Record ID	A number to identify the type of record (see below).
Date_Seconds	Time since January 1, 1970 (in seconds).
Number_of_vars	Number of repeating entries in this record.

The Record ID metric uniquely identifies the type of data contained in the record. Current Record ID values are:

```
-1   Title Record
-2   First header Record       (Contains Item Numbers)
-3   Second header Record       (Contains Item Scale Factors)
-4   Application Name Record   (for Multiple Instance Application
                                Files)
-5   Transaction Name Record   (for Multiple Instance Transaction
```

```
                                  Files)
 -7   Disk Device Name Record   (for Multiple Instance Disk Device
                                  Files)
 -8   Logical Volume Name Record (for Multiple Instance Lvolume
                                  Files)
 -9   Netif Name Record (for Multiple Instance Netif Files)
-10   Filesystem Name Record (for Multiple Instance Netif Files)

-11   CPU Name Record (for Multiple Instance Netif Files)


  1   Global Data Record      ( 5 minute detail  record)
101   Global Data Record      (60 minute summary record)
  2   Application Data Record ( 5 minute detail  record)
102   Application Data Record (60 minute summary record)
  3   Process Data Record     ( 1 minute detail record)
  4   Configuration Data Record
  7   Disk Device Data Record ( 5 minute detail record)
107   Disk Device Data Record (60 minute summary record)
  8   Logical Volume Data Record ( 5 minute detail record)
108   Logical Volume Data Record (60 minute summary record)

  9   Filesystem Data Record ( 5 minute detail record)
109   Filesystem  Data Record (60 minute summary record)
 11   Netif Data Record       ( 5 minute detail record)
111   Netif Data Record       (60 minute summary record)
 12   Transaction Data Record ( 5 minute detail record)
112   Transaction Data Record (60 minute summary record)
 13   CPU Data Record ( 5 minute detail record)
113   CPU Data Record ( 60 minute summary record)

ClassID +1,000,000 Class Data Record ( 5 minute detail record)
ClassID +1,000,000+100 Class Data Record (60 minute summary record)
```

The Date_Seconds metric is identical to the user selectable Date_Seconds metric and is provided to ensure that records can be scanned easily for desired dates and times.

The Number_of_vars metric indicates how many groups of repeating fields are contained in the record. For single instance data types, this value is zero.

For Multiple Instance application records, the Number_of_vars metric is the number of applications configured. For Multiple Instance disk device records, the Number_of_vars metric is the number of disk devices configured. For all header records, this metric is the maximum number of repeating groups allowed.

`Binary` format files have special formats for the title and header records. These records contain the information needed to describe the contents of the file so that a program can properly interpret it. If `headings off` is specified, only data records will be in the file. If `headings on` is specified, the following records will precede all data records.

Binary Header Records

`Title Record`	This record (Record ID -1) is written whenever headings on, regardless of whether the user specified a report title. It contains information about the log file and its source.
`First Header Record`	The first header record (Record ID -2) contains a list of unique item identification numbers corresponding to the items contained in the log file. The position of the item ID numbers can be used to determine the position and size of each exported item in the file.
`Second Header Record`	The second header record (Record ID -3) contains a list of scale factors which correspond to the exported items. For more details, see the discussion of "Scale Factors" later in this section.
`Application Name Record`	This record (Record ID -4) will only be present in application data files that utilize the Multiple Layout format. It lists the names of the applications that correspond to the groups of application metrics in the rest of the file.
`Transaction Name Record`	This record (Record ID -5) will only be present in transaction tracking data files that utilize the Multiple Layout format. It lists the names of the transactions that correspond to the groups of transaction metrics in the rest of the file.

Binary Header Records

Disk Device Name Record	This record (Record ID -7) will only be present in disk device data files that utilize the Multiple Layout format. It lists the names of disk devices that correspond to the groups of disk device metrics in the rest of the file.
Logical Volume Name Record	This record (Record ID -8) will only be present in logical volume data files that utilize the Multiple Layout format. It lists the names of logical volumes that correspond to the groups of logical volume metrics in the rest of the file.
Netif Name Record	This record (Record ID -9) will only be present in netif (LAN) data files that utilize the Multiple Layout format. It lists the names of netif devices that correspond to the groups of netif device metrics in the rest of the file.
Filesystem Name Record	This record (Record ID -12) will only be present in filesystem data files that utilize the Multiple Layout format. It lists the names of filesystems that correspond to the groups of filesystem metrics in the rest of the file.
Cpu Name Record	This record (Record ID -13) will only be present in CPU data files that utilize the Multiple Layout format. It lists the names of CPUs that correspond to the groups of CPU metrics in the rest of the file.

Binary Title Record

The Title Record for BINARY files contains information designed to assist programmatic interpretation of the exported file's contents. This record will be written to the exported file whenever headings on is specified.

The contents of the Binary Title Record are:

```
Record Length          4 byte Int    Length of Title Record
Record ID              4 byte Int    -1
Date_Seconds           4 byte Int    Date exported file was created
Number_of_vars         4 byte Int    Maximum number of repeating
                                     variables
Size of Fixed Area     4 byte Int    Bytes in nonvariable part of
                                     record
Size of Variable Entry 4 byte Int    Bytes in each variable entry
GMT Time Offset        4 byte Int    Seconds offset from Greenwich
                                     Mean Time
Daylight Savings Time  4 byte Int    =1 indicates Daylight Savings
                                     Time
System ID             40 Characters, System Identification
Collector Version     16 Characters, Name & version of the data
                                     collector
Log File Name         72 Characters, Name of the source log file
Report Title          100 Characters, User specified report title
```

The Date_Seconds, GMT Time Offset, and Daylight Savings Time metrics in the Binary Title Record apply to the system and time when the export file was created. If this is not the same system that logged the data, these fields cannot properly reflect the data in the file.

Binary Item Identification Record

The first header record (record ID -2) in the binary file contains the unique item numbers for each item exported. Each Item ID is a 4-byte integer number that can be cross referenced using the rxitemid file supplied with this product. The Item ID fields are aligned with the data fields they represent in the rest of the file. All binary exported data items will occupy a multiple of 4 bytes in the exported file and each will start on a 4-byte boundary. If a data item requires more then 4 bytes of space, its corresponding item ID field will be zero filled on the left.

For example, the process metric Program requires 16 bytes. Its data and item ID records would be:

```
Header 1 (Item Id Record) ...|   0|   0|   0|12012|
Process Data Record          |Prog|ram_|name| _aaa|
```

Binary Scale Factor Record

The second header record (record ID -3) in the binary file contains the scale factors for each of the exported items. Numeric items are exported to binary files as 32-bit (4-byte) integers in order to minimize problems with the way in which different computer architectures implement floating point. Before being truncated to fit into the integer format, most items are multiplied by a fixed scale factor. This allows floating point numbers to be expressed as a fraction, using the scale factor as a denominator.

Each scale factor is a 32-bit (4-byte) integer to match the majority of data items. Special values for the scale factors are used to indicate non-numeric and other special valued metrics.

Special Scale Factors

Non-numeric metrics, such as ASCII fields, have zero scale factors. A negative 1 scale factor should not occur, but if it does it indicates an internal error in the extract program and should be reported.

The DATE format is MPE CALENDAR format in the least significant 16 bits of the field (the 16 bits farthest right). The scale factor for date is 512. Scaling this as a 32-bit integer (dividing by 512) isolates the year as the integer part of the date and the day of the year (divided by 512) as the fractional part.

TIME is a 4-byte binary field (hour, minute, second, tenths of seconds). The scale factor for time is 65536. Dividing it by 65536 forms a number where the integer part is the (hour * 256) + minute.

It is easier to handle a `Date_Seconds` value in a binary file.

Application Name Record

When application data is exported in the Multiple Layout format, a special Application Name Record is written to identify the application groups. For binary format files, this record has record ID -4. It consists of the binary record 16-byte header and a 20-byte application name for each application which was defined at the starting date of the exported data.

If applications are added or deleted during the time covered in the data extraction, the Application Name Record is repeated with the new application names.

Transaction Name Record

When transaction data is exported in the Multiple Layout format, a special Transaction Name Record is written to identify the application-transaction name. For binary format files, this record has a record ID -5. It consists of the binary record 16-byte header and a 60-byte truncated application-transaction name for each transaction that was configured at the starting date of the exported data. If transactions are added during the time covered in the data extraction, the Transaction Name Record will be repeated with the new application-transaction name appended to the end of the original list. Transactions that are deleted after the start of the data extraction will not be removed from the Multiple Layout data record. For more information, see the *HP OpenView Performance Agent & GlancePlus for UNIX: Tracking Your Transactions* guide.

Disk Device Name Record

When disk device data is exported in the Multiple Layout format, a special Disk Device Name Record is written to identify the disk device name. For binary format files, this record has a record ID -7. It consists of the binary record 16-byte header and a 20-byte disk device name for each disk device that was configured at the starting date of the exported data.

If disk devices are added during the time covered in the data extraction, the Disk Device Name Record will be repeated with the new disk device name appended to the end of the original list. Disk devices that are deleted after the start of the data extraction will not be removed from the Multiple Layout data record.

Logical Volume Name Record

When logical volume data is exported in the Multiple Layout format, a special Logical Volume Name Record is written to identify the logical volume name. For binary format files, this record has a record ID -8. It consists of the binary record 16-byte header and a 20-byte disk device name for each logical volume that was configured at the starting date of the exported data.

If logical volumes are added during the time covered in the data extraction, the Logical Volume Name Record will be repeated with the new logical volume name appended to the end of the original list. Logical volumes that are deleted after the start of the data extraction will not be removed from the Multiple Layout data record.

Netif Name Record

When netif data is exported in the Multiple Layout format, a special Netif Name Record is written to identify the netif device name. For binary format files, this record has a record ID -11. It consists of the binary record 16-byte header and a 20-byte netif device name for each netif device that was configured at the starting date of the exported data.

If netif devices are added during the time covered in the data extraction, the Netif Name Record will be repeated with the new netif device name appended to the end of the original list. Netif devices that are deleted after the start of the data extraction will not be removed from the Multiple Layout data record.

6 Extract Commands

Introduction

This chapter describes the extract program's commands. It includes a table showing command syntax, a table of commands for extracting and exporting data, and a command reference section describing the commands in alphabetical order.

Commands and parameters for extract can be entered with any combination of uppercase and lowercase letters. Only the first three letters of the command's name are required, *except* for the weekdays and weekly commands that require you to enter the whole name. For example, the command application detail can be abbreviated as app det.

Examples of how these commands are used can be found in online help for the extract program.

The table on the following pages summarizes the syntax of the extract commands and their parameters.

NOTE The extract function cannot produce summarized data. Summary data can only be produced by the export function.

Table 6-1 Extract Commands: Syntax and Parameters

Command	Parameter
application	on detail summary (export only) both (export only) off (default)
class	detail (default) summary (export only) both (export only) off
cpu	detail summary (export only) both (export only) off (default)
configuration	on detail off (default)
disk	on detail summary (export only) both (export only) off (default)
exit e	
export	day[*ddd*] [*-days*] week [*ww*] [*-weeks*] month[*mm*] [*-months*] year [*yy*] [*-years*]
extract	day[*ddd*] [*-days*] week [*ww*] [*-weeks*] month[*mm*] [*-months*] year [*yy*] [*-years*]

Table 6-1 Extract Commands: Syntax and Parameters (Continued)

Command	Parameter
filesystem	detail summary (export only) both (export only) off (default)
global	on detail (default) summary (export only) both (export only) off
guide	
help	
list	*filename* *
logfile	*logfile*
lvolume	on detail summary (export only) both (export only) off (default)
menu	
monthly	*yyymm* *mm*
netif	on detail summary (export only) both (export only) off (default)
output	*outputfile* ,new ,purgeboth ,append

Table 6-1 Extract Commands: Syntax and Parameters (Continued)

Command	Parameter
process	on detail [app#[-#],...] off (default) killed
quit q	
report	*[export template file]* ,show
shift	*starttime - stoptime* all day noweekends
sh !	*shell command*
show	all
start	*date[time]* today[*-days*][*time*] last[*-days*][*time*] first[*+days*][*time*]
stop	*date[time]* today[*-days*][*time*] last[*-days*][*time*] first[*+days*][*time*]
transaction	on detail summary (export only) both (export only) off (default)
weekdays	1.....7
weekly	*yyww* *ww*

Table 6-1 Extract Commands: Syntax and Parameters (Continued)

Command	Parameter
`yearly`	*yyyy* *yy*

The following table lists the commands that are used for extracting and exporting data and the types of log files used (scopeux log files or DSI log files).

Table 6-2 Extract Commands: Extracting and Exporting Data

Command	Extract Data	Export Data	Scopeux Log Files	DSI Log Files
application	x	x	x	
class	x	x	x	x
configuration		x	x	
cpu	x	x	x	
disk	x	x	x	
export		x	x	x
extract	x		x	
filesystem	x	x	x	
global	x	x	x	
logfile	x	x	x	x
lvolume	x	x	x	
monthly	x		x	
netif		x	x	
output	x	x	x	x
process	x	x	x	
report		x	x	x
shift	x		x	x
start	x	x	x	x
stop	x	x	x	x

Table 6-2 Extract Commands: Extracting and Exporting Data (Continued)

Command	Extract Data	Export Data	Scopeux Log Files	DSI Log Files
transaction	x	x	x	
weekdays		x	x	x
weekly	x		x	
yearly	x		x	

application

Use the `application` command to specify the type of application data that is being extracted or exported.

The default is `application off`

Syntax

```
                    [on]
                    [detail]
       application  [off]
```

Parameters

`on` or `detail`	Specifies that raw, 5-minute detail data should be extracted or exported.
`summary` (`export` only)	Specifies that data should be summarized by: • the number of minutes specified with the *summary* parameter in the specified export template file (`export` only) • the default summary interval of one hour (`export` or `extract`) Summarization can significantly reduce the size of the resulting extracted or exported data, depending on the summarization interval used. For example, hourly summary data is about one-tenth the size of 5-minute detail data.
`both` (`export` only)	Specifies that detail data and summary data are to be extracted or exported.
`off`	Specifies that no data of this type is to be extracted or exported.

NOTE If you are using OV Performance Manager, detail data must be included in an extracted file before drawing application graphs with points every 5 minutes.

Example

In this example, the `application` command causes detailed application log file data to be exported: The output export file contains the application metrics specified in the `myrept` export template file.

```
logfile /var/opt/perf/datafiles/logglob
global off
application detail
report /var/opt/perf/myrept
export
```

To perform the above task using command line arguments, enter:

```
extract -a -r /var/opt/perf/myrept -xp
```

class

Use the `class` command to specify the class of DSI data to be exported, or scopeux data to be extracted or exported.

The default is `class detail`.

Syntax

```
                                [detail]
class      [classname]          [summary]
                                [both]
                                [off]
```

Parameters

classname	Name of a group similarly classified metrics.
`detail`	For DSI log files, specifies how much detail data is exported according to the time set in DSI log file. (For more information, see the *HP OpenView Performance Agent for UNIX Data Source Integration Guide*.) For `scopeux` log files, specifies that raw, 5-minute detail should be extracted or exported.
`summary` `bothoff`	See “Parameters” on page 153 in the description of the `application` command at the beginning of this chapter.

Examples

To export summary data in a DSI log file that contains a class named `acctg_info`, issue the following command:

```
class acctg_info summary
```

Once the log file is specified by the user and opened by the `extract` program, the `acctg_info` class is verified to exist in the log file and can subsequently be exported.

Other variations of this command are:

```
CLASS ACCTG_INFO SUMMARY
class ACCTG_INFO summary
class acctg_info sum
```

Commands can be either uppercase or lowercase. Class names are always upshifted and then compared.

In the following example, summary data in a class named `fin_info` is exported.

```
extract>
class fin_info summary
export
```

To perform the above task using command line arguments, enter:

```
extract -C fin_info summary -xp
```

configuration

Use the configuration command to specify whether or not to export system configuration information.

The default is configuration off.

Syntax

```
                  [on]
configuration     [detail]
                  [off]
```

Parameters

on or detail	Specifies that all configuration records should be exported.
off	Specifies that no configuration data is to be exported.

All configuration information available in the log file is exported. Any begin, end, shift, start, stop or noweekends commands that are used with the configuration command are ignored.

NOTE The configuration command affects only the export function. The extract function is not affected because it always extracts system configuration information.

Example

In this example, the configuration command causes system configuration information to be exported. The output export file contains the configuration metrics specified in the myrept export template file.

```
logfile /var/opt/perf/datafiles/logglob
configuration on
report /var/opt/perf/myrept
export
```

To perform the above task using command line arguments, enter:

```
extract -c -r /var/opt/perf/myrept -xp
```

cpu

Use the cpu command to specify the summarization level of CPU.

The default is cpu off.

Syntax

```
cpu [detail]
    [summary]
    [both][off]
```

Parameters

detail	Extracts or exports 5-minute detail records.
summary	Exports summary records.
both	Exports both detail and summary records.
off	Extracts or exports no CPU data.

Example

In this example, the cpu command causes CPU detail data that was collected starting July 26, 2001 to be exported. Because no export template file is specified, the default export template file, reptfile, is used. All disk metrics are included in the output file as specified by reptfile.

```
logfile /var/opt/perf/datafiles/logglob
global off
cpu detail
start 7/26/01
export
```

To perform the above task using command line arguments, enter:

```
extract -u -b 7/26/01 -xp
```

disk

Use the disk command to specify the type of disk device data that is being extracted or exported.

The default is disk off.

Syntax

```
              [on]
              [detail]
       disk   [summary]
              [both]
              [off]
```

Parameters

on or detail summary bothoff	See “Parameters” on page 153 in the description of the application command at the beginning of this chapter.

Example

In this example, the disk command causes disk detail data that was collected starting July 5, 1999 to be exported. Because no export template file is specified, the default export template file, reptfile, is used. All disk metrics are included in the output file as specified by reptfile.

```
logfile /var/opt/perf/datafiles/logglob
global off
disk detail
start 7/5/99
export
```

To perform the above task using command line arguments, enter:

```
extract -D -b 7/5/99 -xp
```

exit

Use the `exit` command to terminate the `extract` program. The `exit` command is equivalent to the `extract` program's `quit` command.

Syntax

```
exit
e
```

export

Use the export command to start the process of copying data into an exported file format.

Syntax

```
                [day      [ddd] [yyddd] [-days]]
export          [week     [ww] [yyww] [-weeks]]
                [month    [mm] [yymm] [-months]]
                [year     [yy] [yyyy] [-years]]
```

Parameters

Use one of the following parameters to export data for a particular interval.

day	Represents a single day
week	Represents a single week, Monday through Sunday
month	Represents a single month, first through last calendar day
year	Represents a single year, first through last calendar day

If no parameters are used with the export command, the interval used for the exported data is set by the start and stop commands.

How to Use It

There are four ways to specify a particular interval (day, week, month, year).

- Current interval - Specify the parameter only. For example, month means the current month.
- Previous interval - Specify the parameter, a minus, and the number of intervals before the current one desired. For example, day-1 is yesterday, week-2 is two weeks prior to the current week.
- Absolute interval - Specify the parameter and a positive number. The number indicates the absolute interval desired in the current year. For example, day 2 is January 2 of the current year.

- Absolute interval plus year - Specify the parameter and a large positive number. The number should consist of the last two digits of the year and the absolute interval number in that year. In this format the absolute day would have 5 digits (99002 means January 2, 1999) and all other parameters would have four digits (month 9904 means April of 1999).

If you have not previously specified a log file or an export template file, the `logfile` command uses the default global log file `logglob` and the `report` command uses the default export template file `reptfile`.

The settings or defaults for all other parameters are used. For details on their actions, see descriptions of the `application`, `configuration`, `global`, `process`, `disk`, `lvolume`, `netif`, `CPU`, `filesystem`, `transaction`, `output`, `shift`, `start`, and `stop` commands.

The `export` command creates up to 16 different default output files based on the types of data and level of summarization specified.

`xfrdGLOBAL.ext`	Global detail data file
`xfrsGLOBAL.ext`	Global hourly summary data file
`xfrdAPPLICATION.ext`	Application detail data file
`xfrsAPPLICATION.ext`	Application hourly summary data file
`xfrdPROCESS.ext`	Process detail data file
`xfrdDISK.ext`	Disk device detail data file
`xfrsDISK.ext`	Disk device summary data file
`xfrdVOLUME.ext`	Logical volume detail data file
`xfrsVOLUME.ext`	Logical volume summary data file
`xfrdNETIF.ext`	Netif detail data file
`xfrsNETIF.ext`	Netif summary data file
`xfrdCPU.ext`	CPU detail data type
`xfrsCPU.ext`	CPU summary data type
`xfrdFILESYSTEM.ext`	Filesystem detail data type
`xfrsFILESYSTEM.ext`	Filesystem summary data type

xfrdTRANSACTION.ext	Transaction detail data file
xfrsTRANSACTION.ext	Transaction summary data file
xfrdCONFIGURATION.ext	Configuration detail data file

where ext = asc, dat, bin, or wk1

The default file names are created from the data type name. The prefix is either xfrd or xfrs depending if the data is detailed or summary data. The extension is the specified asc (ASCII), bin (binary), dat (datafile), or wk1 (spreadsheet) data format.

For example, classname = ACCTG_INFO would have export file names of:

xfrdACCTG_INFO.wk1	detailed spreadsheet data for ACCT_INFO
xfrsACCTG_INFO.asc	summarized ASCII data for ACCT_INFO

For more information about exporting data, see "Overview of the Export Function" in Chapter 5, Using the Extract Program,.

Example

In this example, the export command causes log file data collected yesterday from 8:00 am to 5 pm to be exported. Because no export template file is specified, the default export template file, reptfile, is used. All global metrics are included in the output file as specified by reptfile

```
logfile /var/opt/perf/datafiles/logglob
start today-1 8:00 am
stop today-1 5:00 pm
global both
export
```

To perform the above task using command line arguments, enter:

```
extract -gG -b today 8:00 am -e today 5:00 pm -xp
```

extract

Use the `extract` command to start the process of copying data from raw log files into an extracted file format. Extracted files can be used for archiving or for analysis by analyzer programs such as OV Performance Manager. You can extract data from raw log files and from extracted files.

The `extract` command cannot be used to process data from DSI log files.

Syntax

```
                [day      [ddd] [yyddd] [-days]]
extract         [week     [ww] [yyww] [-weeks]]
                [month    [mm] [yymm] [-months]]
                [year     [yy] [yyyy] [-years]]
```

Parameters

Use one of the following parameters to extract data for a particular interval:

day	Represents a single day
week	Represents a single week, Monday through Sunday
month	Represents a single month, first through last calendar day
year	Represents a single year, first through last calendar day

If no parameters are used with the `extract` command, the interval used for data extraction is set by the `start` and `stop` commands.

How to Use It

There are four ways to specify a particular interval (`day, week, month, year`).

- Current interval - Specify the parameter only. For example, `month` means the current month.

- Previous interval - Specify the parameter, a minus, and the number of intervals before the current one desired. For example, `day-1` is yesterday, `week-2` is two weeks prior to the current week.
- Absolute interval - Specify the parameter and a positive number. The number indicates the absolute interval desired in the current year. For example, `day 2` is January 2 of the current year.
- Absolute interval plus year - Specify the parameter and a large positive number. The number should consist of the last two digits of the year and the absolute interval number in that year. In this format, the absolute day would have five digits (99002 means January 2, 1999) and all other parameters would have four digits (month 99904 means April of 1999).

The `extract` command starts data extraction. If not previously specified, the `logfile` and `output` commands assume the following defaults when the `extract` command is executed:

```
log file = /var/opt/perf/datafiles/logglob
output file = rxlog,new
```

The settings or defaults for all other parameters are used. For details on their actions, see descriptions of the `application`, `global`, `process`, `disk`, `lvolume`, `netif`, `CPU`, `filesystem`, `transaction`, `shift`, `start`, and `stop` commands.

The size of an extracted log file cannot exceed 64 megabytes.

Example

In the first example, data collected from March 1, 2000 to June 30, 2000 during the hours 8:00 am to 5:00 pm on weekdays is extracted. Only global and application detail data is extracted.

```
logfile /var/opt/perf/datafiles/logglob
start 03/01/00
stop 06/30/00
shift 8:00 am - 5:00 pm noweekends
global detail
application detail
extract
```

To perform the above task using command line arguments, enter:

```
extract -ga -b 03/01/00 -e 6/30/00 -s 8:00 am - 5:00
noweekends -xt
```

In the second example, a new extracted log file named `rxjan00` is created. Any existing file that has this name is purged. All raw log file data collected from January 1, 2000 through January 31, 2000 is extracted:

```
logfile /var/opt/perf/datafiles/logglob
output rxjan00,purge
start 01/01/00
stop 01/31/00
global detail
application detail
transaction detail
process detail
disk detail
lvolume detail
netif detail
filesystem detail
cpu detail
extract
```

To perform the above task using command line arguments, enter:

```
extract -f rxjan00,purge -gatpdznyu -b 01/01/00 -e 01/31/00
-xt
```

filesystem

Use this command to specify the summarization level of filesystem data to extract or export.

The default is `filesystem off`.

Syntax

```
                [detail]
filesystem      [summary]
                [both]
                [off]
```

Parameters

`detail`	Extracts or exports 5-minute detail records.
`summary`	Exports summary records.
`both`	Exports both detail and summary records.
`off`	Extracts or exports no filesystem data.

Example

In this example, the `filesystem` command causes filesystem detail data that was collected starting July 26, 2001 to be exported. Because no export template file is specified, the default export template file, `reptfile`, is used. All filesystem metrics are included in the output file as specified by `reptfile`.

```
logfile /var/opt/perf/datafiles/logglob
global off
filesystem detail
start 7/26/01
export
```

To perform the above task using command line arguments, enter:

```
extract -y -b 7/26/01 -xp
```

global

Use the `global` command to specify the amount of global data to be extracted or exported.

The default is `global detail`. (In command line mode, the default is `global off`.)

Syntax

```
                [on]
                [detail]
       global   [summary]
                [both]
                [off]
```

Parameters

`detail` or `on`
`summary`
`both`
`off`
	See "Parameters" on page 153 in the description of the `application` command at the beginning of this chapter.

How to Use It

Detail data must be extracted if you want to draw OV Performance Manager global graphs with points every 5 minutes.

Summarized data is graphed by OV Performance Manager more quickly since fewer data records are needed to produce a graph. If only global summaries are extracted, OV Performance Manager global graphs cannot be drawn with data points every 5 minutes.

The `both` option maintains the access speed gained with the hourly summary records while permitting you to draw OV Performance Manager global graphs with points every 5 minutes.

The `off` parameter is not recommended if you are using OV Performance Manager because you must have global data to properly understand overall system behavior. OV Performance Manager global graphs cannot be drawn unless the extracted file contains at least one type of global data.

Example

The `global` command is used here to specify that *no* global data is to be exported (`global detail` is the default). Only detailed transaction data is exported. The output export file contains the transaction metrics specified in the `myrept` export template file.

```
extract>
      logfile /var/opt/perf/datafiles/logglob
      global off
      transaction detail
      report /var/opt/perf/myrept
      export
```

To perform the above task using command line arguments, enter:

```
extract -l -t -r /var/opt/perf/myrept -xp
```

guide

Use the guide command to enter guided commands mode. The guided command interface leads you through various extract commands and prompts you to perform some of the most common tasks that are available.

Syntax

```
guide
```

How to Use It

- To enter guided commands mode from extract's interactive mode, type **guide**.
- To accept the default value for a parameter, press **Return**.
- To terminate guided commands mode and return to interactive mode, type **q** at the guide> prompt.

This command does not provide all possible combinations of parameter settings. It selects settings that should produce useful results for the majority of users. You can obtain full control over extract's functions through extract's interactive command mode.

NOTE If you are exporting DSI log file data, we recommend using guided commands mode to create a customized export template file and export the data.

help

Use the help command to access online help.

Syntax

```
help [keyword]
```

How to Use It

You can enter parameters to obtain information on extract commands and tasks, or on help itself. You can navigate to different topics by entering a key word. If more than one page of information is available, the display pauses and waits for you to press **Return** before continuing. Type **q** or **quit** to exit the help system and return to the extract program.

You can also request help on a specific topic. For example,

```
help tasks
```

or

```
help resize parms
```

When you use this form of the help command, you receive the help text for the specified topic and remain in the extract command entry context. Because you do not enter the help subsystem interactively, you do not have to type **quit** before entering the next extract command.

list

Use the `list` command to specify the list file for all extract program reports.

Syntax

```
list     [file]
         [*]
```

How to Use It

You can use `list` at any time while using `extract` to specify the list device. It uses a file name or list device name to output the user-specified settings. If the list file already exists, the output is appended to it.

The data that is sent to the list device is also displayed on your screen.

While `extract` is running, type:

```
list outfilename
```

To return the listing device to the user terminal, type:

```
list stdout
```

OR

```
list *
```

To determine the current list device, type the `list` command without parameters as follows:

```
list
```

If the list file is not `stdout`, most commands are echoed to the list file as they are entered.

Example

The following example, the list device is set to `mylist`. The results of the next commands are printed to `mylist` and displayed on your screen.

```
extract>
logfile /var/opt/perf/datafiles/logglob
list mylist
global detail
shift 8:00 AM - 5:00 PM
extract
```

logfile

Use the `logfile` command to open a log file. You must open a log file for all `extract` program functions. You can do this explicitly by issuing the `logfile` command, or implicitly by issuing the `extract` command or `export` command. If you do not specify a log file name, the `extract` program prompts you for a log file name and displays the default global log file `/var/opt/perf/datafiles/logglob`. You can either accept the default or specify a different log file.

Syntax

```
logfile [logfile]
```

How to Use It

To open a log file, you can specify the name of either a raw or extracted log file. You cannot specify the name of a file created by the `export` command. If you specify an extracted log file name, all information is obtained from this single file. If you specify a raw log file name, you must specify the name of the global log file before you can access the raw log file. This is the only raw log file name you should specify.

If the log file is not in your current working directory, you must provide its path.

The global log file and other raw log files must be in the same directory. They have the following names:

`logglob`	global log file
`logappl`	application log file
`logproc`	process log file
`logdev`	device log file
`logtran`	transaction log file
`logindx`	index log file

The general contents of the log file are displayed when the log file is opened.

CAUTION Do not rename raw log files! When accessing these files, the program assumes that the standard log file names are in effect. If you must rename log files to place log files from multiple systems on the same system for analysis, you should first extract the data and then rename the extracted log files.

Example

This is a typical listing of the default global log file.

```
Global      file: /var/opt/perf/datafiles/logglob, version D
Application file: /var/opt/perf/datafiles/logappl
Process     file: /var/opt/perf/datafiles/logproc
Device      file: /var/opt/perf/datafiles/logdev
Transaction file: /var/opt/perf/datafiles/logdev
Index       file: /var/opt/perf/datafiles/logindx

System ID:  homer
System Type 9000/715/ S/N 2223334442 O/S HP-UX B.10.20 A
Data collector: SCOPE/UX C.02.30
File Created:  10/08/99
Data Covers:   44 days to 11/20/99
Shift is:      All Day

Data records available are:
Global Application Process Disk Volume Transaction

Maximum file sizes:
Global=10.0 Application=10.0 Process=20.0 Device=10.0
Transaction=10.0 MB

The first GLOBAL      record is on 10/08/99 at 08:17 AM
The first NETIF       record is on 10/08/99 at 08:17 AM
The first APPLICATION record is on 11/17/99 at 12:15 PM
The first PROCESS     record is on 10/08/99 at 08:17 AM
The first DEVICE      record is on 10/31/99 at 10:45 AM
The Transaction data file is empty
The default starting date & time = 10/08/99 11:50 AM   (LAST
-30)
The default stopping date & time  = 11/20/99 11:59 PM   (LAST
-0)
```

lvolume

Use the lvolume command to specify the type of logical volume data that is being extracted or exported. (This command is used only on HP-UX systems.)

The default is lvolume off.

Syntax

```
                [on]
                [detail]
lvolume         [summary]
                [both]
                [off]
```

Parameters

on or detail summary both off	See "Parameters" on page 153 in the description of the application command at the beginning of this chapter.

Example

In this example, a new extracted log file named rx899 is created and any existing file that has that name is purged. All logical volume data collected from August 1 through August 31 is extracted.

```
logfile /var/opt/perf/datafiles/logglob
output rx899,purge
start 08/01/99
stop 08/31/99
global detail
lvolume detail
month 9908
```

To perform the above task using command line arguments, enter:

```
extract -f rx899,purge -gz -xm 9908
```

menu

Use the menu command to print a list of the available extract commands.

Syntax

menu

Example

```
Command   Parameters   Function
HELP      [topic]      Get information on commands and options
GUIDE     Enter guided commands mode for novice users
LOGFILE   [logname]    Specify a log file to be processed
LIST      [filename|*] Specify the listing file
OUTPUT    [filename]
          [,NEW/PURGE/APPEND] Specify a destination file
REPORT    [filename][,SHOW]   Specify an Export Format file for
                              "EXPORT"
GLOBAL    [DETAIL/SUMMARY/BOTH/OFF] Extract GLOBAL records
APPLICATION [DETAIL/SUMMARY/BOTH/OFF] Extract APPLICATION
                              records
PROCESS   [DETAIL/OFF/KILLED][APP=] Extract PROCESS records
DISK      [DETAIL/SUMMARY/BOTH/OFF] Extract DISK DEVICE records
LVOLUME   [DETAIL/SUMMARY/BOTH/OFF] Extract Logical VOLUME
                              records
NETIF     [DETAIL/SUMMARY/BOTH/OFF] Extract Logical NETIF
                              records
CPU       [DETAIL/SUMMARY/BOTH/OFF] Extract CPU records
FILESYSTEM [DETAIL/SUMMARY/BOTH/OFF] Extract FILESYSTEM
                              records
CONFIG    [DETAIL/OFF]    Export CONFIGURATION records
CLASS     classname[DETAIL/SUMMARY/BOTH/OFF]  Export classname
                              records
TRANSACTION [DETAIL/SUMMARY/BOTH/OFF] Extract TRANSACTION
                              records
START     [startdate time]  Specify a starting date and time for
                              SCAN
STOP      [stopdate  time]  Specify an ending  date and time for
                              SCAN
SHIFT     [starttime - stoptime] [NOWEEKENDS] Specify daily
                              shift times
SHOW      [ALL]             Show the current program settings
```

```
EXPORT    [d/w/m/y][-offset] Copy log file records to HOST
                              format files
EXTRACT   [d/w/m/y][-offset] Copy selected records to output
                              (or append) file
WEEKLY    [ww/yyww]  Extract one calendar  week's data with
                              auto file names
MONTHLY   [mm/yymm]  Extract one calendar month's data with
                              auto file names
YEARLY    [yy/yyyy]  Extract one calendar  year's data with
                               auto file names

WEEKDAYS  [1...7]   Set days to exclude from export 1=Sunday
! or SH   [command] Execute a system command
MENU or ? List the command menu (this listing)
EXIT or Q Terminate the program
```

monthly

Use the monthly command to specify data extraction based on a calendar month. During execution, this command sets the start and stop dates to the proper dates, based on the month and year of the data extracted.

The name of the output file consists of the letters rxmo followed by the four digits of the year and the two-digit number of the month being extracted. For example, data extracted in March 1999 would be output to a file named rxmo199903.

Syntax

```
monthly     [yymm]
            [mm]
```

Parameters

monthly	Extracts data from the current (default) month.
monthly *mm*	Extracts data for a specific month from the current year's data (where *mm* is a number from 01 to 12).
monthly *yymm*	Extracts data for a specific month and year (where *yymm* is a single number consisting of the last two digits of the year and two-digit month number). For example, to extract data for February 1999, specify monthly 9902.

If you do not specify the log file before executing the monthly command, the default logglob file is used.

How to Use It

Use the monthly command when you are extracting data for archiving on a monthly basis.

The type of data extracted and summarized follows the normal rules for the extract command and can be set before executing the monthly command. These settings are honored unless a monthly output file already exists. If it does, data is appended to it based on the type of data that was originally specified.

The monthly command has a feature that opens the previous month's extracted file and checks to see if it is filled--whether it contains data extracted up to the last day of the month. If not, the monthly command appends data to this file to complete the previous month's extraction.

For example, a monthly command is executed on May 7, 1999. This creates a log file named rxmo199905 containing data from May 1 through the current date (May 7).

On June 4, 1999, another monthly command is executed. Before the rxmo199906 file is created for the current month, the rxmo199905 file from the previous month is opened and checked. When it is found to be incomplete, data is appended to it to complete the extraction through May 31, 1999. Then, the rxmo199906 file is created to hold data from June 1, 1999 to the current date (June 4).

As long as you execute the monthly command at least once a month, this feature will complete each month's file before creating the next month's file. When you see two adjacent monthly files--for example, rxmo199905 (May) and rxmo199906 (June)--you can assume that the first file is complete for that month and it can be archived and purged.

NOTE

The monthly and extract month commands are similar in that they both extract one calendar month's data. The monthly command ignores the setting of the output command, using instead predefined output file names. It also attempts to append missing data to the previous month's extracted log file if it is still present on the system. The extract month command, on the other hand, uses the settings of the output command. It cannot append data to the previous month's extracted file since it does not know its name.

Example

In this example, detail application data logged during May 1999 is extracted.

```
logfile /var/opt/perf/datafiles/logglob
global off
application detail
monthly 9905
```

To perform the above task using command line arguments, enter:

```
extract -a -xm 9905
```

netif

Use the netif command to specify the type of logical network interface (LAN) data to extract or export. Netif data is logged in the logdev file.

The default is netif off.

Syntax

```
                [on]
                [detail]
        netif   [summary]
                [both]
                [off]
```

Parameters

on or detail summary both off	See "Parameters" on page 153 in the description of the application command at the beginning of this chapter.

Example

In this example, netif detail data collected from March 1, 2000 to June 30, 2000 during the hours 8:00 am to 5:00 pm on weekdays is extracted.

```
logfile /var/opt/perf/datafiles/logglob
start 03/01/00
stop 06/30/00
shift 8:00 AM - 5:00 PM noweekends
netif detail
extract
```

To perform the above task using command line arguments, enter:

```
extract -n -b 03/01/00 -e 6/30/00 -s 8:00 am - 5:00
noweekends -xt
```

output

Use the `output` command to specify the name of an output file for the extract or export functions.

The optional second parameter specifies the action to be taken if an output file with the same name exists.

Syntax

```
                                [,new]
output        [filename         [,purge]
              ]                 [,append]
```

Parameters

`,new`	Specifies that the output file must be a new file. This is the default action in batch mode. If a file with the same name exists, the batch job terminates.
`,purge`	Specifies that any existing file should be purged to make room for the new output file.
`,append`	Specifies that an existing extracted file should have data appended to it. If no file exists with the output file name specified, a new file is created.

How to Use It

If you do not specify an action in batch mode, the default action `,new` is used. In interactive mode, you are prompted to enter an action if a duplicate file is found.

If you do not specify an output file, default output files are created. The default output file names are:

For `extract`: `rxlog`

For `export`:

```
xfrdGLOBAL.ext
xfrsGLOBAL.ext
xfrdAPPLICATION.ext
xfrsAPPLICATION.ext
xfrdPROCESS.ext
```

```
xfrdDISK.ext
xfrsDISK.ext
xfrdLVOLUME.ext
xfrsLVOLUME.ext
xfrdNETIF.ext
xfrsNETIF.ext
xfrdCPU.ext
xfrsCPU.ext
xfrdFILESYSTEM.ext
xfrsFILESYSTEM.ext
xfrdTRANSACTION.ext
xfrsTRANSACTION.ext
xfrdCONFIGURATION.ext
```

where `ext = asc` (ASCII), `dat` (datafile), `bin` (binary), or `wk1` (spreadsheet).

A special file name, `stdout` (or *), can be used with the export operation to direct the output to the `stdout` file (normally your terminal or workstation, although this can be redirected using shell commands).

```
output stdout
```

or

```
output *
```

To return the output to its default settings, type:

```
output default
```

or

```
output -
```

NOTE

You can override the default output file names for exported files using the *output* parameter in the export template file.

You should not output extract operation files to `stdout`, because they are incompatible with ASCII devices. You should also not output binary or WK1 formats of the export operation to the `stdout` file for the same reason.

Care should be taken to avoid appending *extracted* data to an existing *exported* data file and to avoid appending exported data to an existing extracted file. Attempts to append the wrong data type will result in an error condition.

Example

In this example, no output file is specified so the default output file, `rxlog` is used for the application summary data being extracted. The `,purge` option specifies that any existing output file should be purged.

```
extract>
logfile /var/opt/perf/datafiles/logglob
output rxlog,purge
global off
application summary
extract month 9905
```

To perform the above task using command line arguments, enter:

```
extract -f rxlog,purge -A -xm 9905
```

process

Use the process command to specify whether or not to extract or export process data.

The default is process off.

Syntax

```
                [on]
process         [detail]   [application=#[-#] ,...]
                [off]
                [killed]
```

Parameters

on	Specifies that process data *should* be extracted or exported.
detail	Specifying process detail is the same as specifying process on.
off	Specifies that process data *should not* be extracted or exported.
killed	Specifies only processes that have an interest reason that includes killed. (Processes that terminated in the measurement interval.)
application	Specifies only processes that belong to selected applications. An application can be entered as a single number or as a range of application numbers (7-9 means applications 7, 8, and 9). The application number is determined by the order of the application definition in the parm file when the data was collected. If you are specifying multiple applications, separate each one with a comma.

NOTE Process data can increase the size of an extracted log file significantly. If you plan to copy the log file to a workstation for analysis, you might want to limit the amount of process data extracted.

Example

In this example, the process command specifies processes that terminated during an interval and belong to applications 1, 4, 6, 7, 8, or 10. Use the `utility` program's `scan` command to find the application numbers for specific applications.

```
process killed applications=1,4,6-8,10
```

quit

Use the `quit` command to terminate the `extract` program. The `quit` command is equivalent to the `extract` program's `exit` command.

Syntax

```
quit
q
```

report

Use the `report` command to specify the export template file to be used by the export function. If no export template file is specified, the default export template file, `reptfile`, is used. The export template file is used to specify various output format attributes used in the export function. It also specifies which metrics will be exported.

If you are in interactive mode and specify no export template file, all metrics for the data types requested will be exported in ASCII format.

Syntax

```
report [exporttemplatefile] [,show]
```

Parameters

`,show` — Specifies that the field positions and starting columns should be listed for all metrics specified in the export template file. This listing can be used when export files are processed by other programs.

How to Use It

When you issue this command, you are prompted by a message that asks whether or not you want to validate metrics in the export template with the previously specified log file. Validation ensures that the metrics specified in the export template file exist in the log file. This allows you to check for possible errors in the export template file. If no validation is performed, this action is deferred until you perform an export.

NOTE The `,show` parameter of the report command discussed here is different from the show command discussed later.

sh

Use sh to enter a shell command without exiting extract by typing sh or an exclamation point(!) followed by a UNIX shell command.

Syntax

sh or ! [shell command]

Parameters

sh ls	Executes the ls command and returns to extract. The shell command is any system command.
!ls	Same as above.
!ksh	Starts a Korn shell. Does not return immediately to extract. Type **exit** or **CTRL-d Return** to return to the extract program.

How to Use It

Following the execution of the single command, you automatically return to extract. If you want to issue multiple shell commands without returning to extract after each one, you can start a new shell.

If you issue the sh command without the name of the shell command, you are prompted to supply it. For example,

```
sh
enter SYSTEM command: ls
```

shift

Use the `shift` command to limit data extraction to certain hours of the day corresponding to work shifts and to exclude weekends (Saturday and Sunday).

The default is `shift all day` to extract data for all day, every day including weekends.

Syntax

```
                [starttime-stoptime]
        shift   [all day]
                [noweekends]
```

Parameters

The *starttime* and *stoptime* parameters are entered in the same format as the time in the `start` command. Shifts that span midnight are permitted. If *starttime* is scheduled *after* the *stoptime*, the shift will start at the start time and proceed past midnight, ending at the *stoptime* of the next day.

`all day` — Specifies the default shift of 12:00 am - 12:00 am (or 00:00 -00:00 on a 24-hour clock).

`noweekends` — Specifies the exclusion of data which was logged on Saturdays and Sundays. If `noweekends` is entered in conjunction with a shift that spans midnight, the weekend will consist of those shifts that *start* on Saturday or Sunday.

Example

In this example, disk detail data collected between 10:00 am and 4:00 pm every day starting June 15, 1999 is extracted.

```
extract>
logfile /var/opt/perf/datafiles/logglob
global off
disk detail
shift 10:00 am - 4:00 PM
start 6/15/99
extract
```

To perform the above task using command line arguments, enter:

```
extract d -b 6/15/99 -s 10:00 AM-4:00 PM -xt
```

show

Use the show command to list the names of the opened files and the status of the extract parameters that can be set.

Syntax

```
show [all]
```

NOTE The show command discussed here is different from the ,show parameter of the report command discussed earlier.

Examples

Use show by itself to produce a list that may look like this:

```
Logfile:  /var/opt/perf/datafiles/logglob

Output:   Default
Report:   Default
List:     "stdout"

The default starting date & time = 10/08/99 12:00 AM (LAST -30)
The default stopping date & time = 11/20/99 11:59 PM (LAST -0)
The default shift = 12:00 AM - 12:00 PM

GLOBAL           DETAIL           records will be processed
APPLICATION. . . . . . . . . . NO records will be processed
PROCESS . . . . . . . . . . .  NO records will be processed

DISK DEVICE. . . . . . . . . . NO records will be processed

LVOLUME. . . . . . . . . . . . NO records will be processed

TRANSACTION. . . . . . . . . . NO records will be processed

NETIF . . . . . . . . . . . . .NO records will be processed

CPU . . . . . . . . . . . . . .NO records will be processed

FILESYSTEM. . . . . . . . . . .NO records will be processed
Configuration . . . . . . . . . .NO records will be processed
```

Use `show all` to produce a more detailed list that may look like this:

```
  Logfile:          /var/opt/perf/datafiles/logglob
  Global      file: /var/opt/perf/datafiles/logglob,version D
  Application file: /var/opt/perf/datafiles/logappl
  Process     file: /var/opt/perf/datafiles/logproc
  Device      file: /var/opt/perf/datafiles/logdev
  Transaction file: /var/opt/perf/datafiles/logdev
  Index       file: /var/opt/perf/datafiles/logindx
  System ID:  homer
  System Type 9000/715/ S/N 2223334442 O/S HP-UX B.10.20 A
  Data collector: SCOPE/UX C.02.30
  File Created:   10/08/99
  Data Covers:    44 days to 11/20/99
  Shift is:       All Day

Data records available are:
  Global Application Process Disk Volume Transaction

Maximum file sizes:
  Global=10.0 Application=10.0 Process=20.0 Device=10.0
  Transaction=10.0 MB

Output:  Default
Report:  Default
List:    "stdout"

The default starting date & time = 10/08/99 11:50 AM (LAST -30)
The default stopping date & time = 11/20/99 11:59 PM(LAST - 0)
The default shift = 12:00 AM - 12:00 PM

GLOBAL...........DETAIL..........records will be processed
APPLICATION.....................NO records will be processed
PROCESS.........................NO records will be processed
DISK DEVICE.....................NO records will be processed
LVOLUME.........................NO records will be processed
TRANSACTION.....................NO records will be processed
NETIF...........................NO records will be exported
CPU.............................NO records will be processed
FILESYSTEM......................NO records will be processed
Configuration ..................NO records will be exported

Export Report Specifications:
   Interval = 3600, Separator = " "
   Missing data will not be displayed
   Headings will be displayed
   Date/time will be formatted
   Days to exclude: None
```

start

Use the `start` command to set a starting date and time for the extract and export functions. The default starting date is the date 30 full days before the last date in the log file, *or* if less than 30 days are present, the date of the earliest record in the log file.

Syntax

```
                [date [time]]
        start   [today [-day][time]]
                [last [-days][time]]
                [first [+days][time]]
```

Parameters

`date` — The *date* format depends on the native language that is configured for your system. If you do not use native languages or you have set C as the default language, the data format is mm/dd/yy (month/day/year) such as 09/30/99 for September 30, 1999, for the extract or export function.

`time` — The *time* format also depends on the native language used. For the C language, the format is hh:mm am or hh:mm pm (hour:minute in a 12-hour format with the am or pm suffix). For example, 07:00 am is 7 o'clock in the morning.

Twenty-four hour time is accepted in all languages. For example, 23:30 would be accepted for 11:30 pm.

If the format of the date or time is unacceptable, you are prompted with an example in the correct format.

If no start time is given, midnight (12:00 am) is assumed. A starting time of midnight for a given day starts at the *beginning* of that day (00:00 on a 24-hour clock).

today	Specifies the current day. The qualification of the parameter, such as today-days, specifies the number of days *prior* to today's date. For example, today-1 indicates yesterday's date and today-2, the day before yesterday.
last	Can be used to represent the last date contained in the log file. The parameter last-days specifies the number of days *prior* to the last date in the log file.
first	Can be used to represent the first date contained in the log file. The parameter first+days specifies the number of days *after* the first date in the log file.

How to Use It

The following commands override the starting date set by the start command.

- weekly
- monthly
- yearly
- extract (If day, week, month, or year parameter is used)
- export (If day, week, month, or year parameter is used)

Example

In this example, the start command specifies June 5, 1999 8:00 am as the start time of the first interval to be extracted. The output command specifies an output file named myout.

```
logfile /var/opt/perf/datafiles/logglob
start 6/5/99 8:00 am
output myout
global detail
extract
```

To perform the above task using command line arguments, enter:

```
extract -g -b 06/05/99 8:00 AM -f myout -xt
```

stop

Use the `stop` command to terminate an extract or export function at a specified date and time.

The default stopping date and time is the *last* date and time recorded in the log file.

Syntax

```
              [date [time]]
       start  [today [-day][time]]
              [last [-days][time]]
              [first [+days][time]]
```

Parameters

`date` The *date* format depends on the native language that is configured for your system. If you do not use native languages or you have set C as the default language, the data format is mm/dd/yy (month/day/year) such as 09/30/99 for September 30, 1999, for the extract or export function.

`time` The *time* format also depends on the native language used. For the C language, the format is hh:mm am or hh:mm pm (hour:minute in a 12-hour format with the am or pm suffix). For example, 07:00 am is 7 o'clock in the morning.

Twenty-four hour time is accepted in all languages. For example, 23:30 would be accepted for 11:30 pm.

If the format of the date of time is unacceptable, you are prompted with an example in the correct format.

If no stop time is given, one minute before midnight (11:59 pm) is assumed. A stopping time of midnight (12:00 am) for a given day stops at the *end* of that day (23:59 on a 24-hour clock).

today	Specifies the current day. The qualification of the parameter, such as today-days, specifies the number of days prior to today's date. For example, today-1 indicates yesterday's date and today-2 the day before yesterday.
last	Can be used to represent the last date contained in the log file. The parameter last-days specifies the number of days *prior* to the last date in the log file.
first	Can be used to represent the first date contained in the log file. The parameter first+days specifies the number of days *after* the first date in the log file.

How to Use It

The following commands override the stopping date set by the stop command.

- weekly
- monthly
- yearly
- extract (If day, week, month, or year parameter is used)
- export (If day, week, month, or year parameter is used)

Example

In this example, the stop command specifies June 5, 1999 5:00 pm as the stopping time of the last interval to be extracted. The output command specifies an output file named myout.

```
extract>
logfile /var/opt/perf/datafiles/logglob
start 6/5/99 8:00 AM
stop 6/5/99 5:00 PM
output myout
global detail
extract
```

To perform the above task using command line arguments, enter:

```
extract -g -b 6/5/99 8:00 AM -e 5:00 PM -f myout -xt
```

transaction

Use the transaction command to specify the type of transaction data that is being extracted or exported.

Syntax

```
                  [on]
                  [detail]
transaction       [summary]
                  [both]
                  [off]
```

Parameters

on or detail summary both off	See “Parameters” on page 153 in the description of the application command at the beginning of this chapter.

Example

A new extracted log file called rxmay99 is created on June 1, 1999. Any existing file that has this name is purged. All raw transaction log file data collected from May 1, 1999 to May 31, 1999 is extracted.

```
extract>
logfile /var/opt/perf/datafiles/logglob
output rxmay99,purge
global detail
transaction detail
month 9905
```

To perform the above task using command line arguments, enter:

```
extract -gt -f rxmay99,purge -xm 9905
```

weekdays

Use the weekdays command to exclude data for specific days from being exported (day 1 = Sunday).

Syntax

```
weekdays [1|2.....7]
```

How to Use It

If you want to export data from only certain days of the week, use this command to exclude the days from which you *do not* want data. Days have the following values:

```
Sunday      =1
Monday      =2
Tuesday     =3
Wednesday   =4
Thursday    =5
Friday      =6
Saturday    =7
```

For example, if you want to export data that was logged only on Monday through Thursday, *exclude* data from Friday, Saturday, and Sunday from your export.

Example

In this example, any detailed global data logged on Tuesdays and Thursdays is excluded from the export. The output export file contains the global metrics specified in the myrept export template file.

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
report myrept
weekdays 35
export
```

weekly

Use the `weekly` command to specify data extraction based on a calendar week. A week is defined as seven days starting on Monday and ending on Sunday.

During execution, this command sets the start and stop dates to the proper dates, based on the week and year of the extracted data.

Syntax

```
weekly      [yyww]
            [ww]
```

Parameters

`weekly`	Extracts the current week's data (the default).
`weekly` *ww*	Extracts data for a specific week from this year's data (where `ww` is any number from 01 to 53).
`weekly` *yyww*	Extracts data for a specific week *and* year (where `yyww` is a single number consisting of the last two digits of the year and the two-digit week-of-the-year number). For example, the 20th week of 1999 would be `weekly 9920`.

If you do not specify the log file before executing the `weekly` command, the default `logglob` file in the `datafiles` directory is used.

How to Use It

Use the `weekly` command when you are extracting data for archiving on a weekly basis.

The name of the output file consists of the letters `rxwe` followed by the last two digits of the year, and the two-digit week number for the week being extracted. For example, the 12th week of 1999 (from Monday, March 22 to Sunday, March 29) would be output to a file named `rxwe9912`.

The type of data extracted and summarized follow the normal rules for the `extract` command and can be set before executing the `weekly` command. These settings are honored unless a weekly output file already exists. If it does, data is appended to it, based on the type of data selected originally.

The `weekly` command has a feature that opens the *previous* week's extracted file and checks to see if it is filled--whether it contains data extracted up to the last day of the week. If not, the `weekly` command appends data to this file to complete the previous week's extraction.

For example, a `weekly` command is executed on Thursday, May 20, 1999. This creates a log file named `rxwe199920` containing data from Monday, May 17 through the current date (May 20).

On Wednesday, May 26, 1999, another `weekly` command is executed. Before the `rxwe199921` file is created for the current week, the `rxwe199920` file from the previous week is opened and checked. When it is found to be incomplete, data is appended to it to complete the extraction through Sunday, May 23, 1999. Then, the `rxwe199921` file is created to hold data from Monday, May 24, 1999 to the current date (May 26).

As long as you execute the `weekly` command at least once a week, this feature will complete each week's file before creating the next week's file. When you see two adjacent weekly files (for example, `rxwe199920` and `rxwe199921`), you can assume that the first file is complete for that week, and it can be archived and purged.

NOTE

The weeks are numbered based on their starting day. Thus, the first week of the year (week 01) is the week starting on the *first* Monday of that year. Any days before that Monday belong to the last week of the previous year. The weekly and extract week commands are similar in that they both extract one calendar week's data. The weekly command ignores the setting of the output command, using instead predefined output file names. It also attempts to append missing data to the previous week's extracted log file if it is still present on the system. The `extract week` command, on the other hand, uses the settings of the output command. It cannot append data to the previous week's extracted file because it does not know its name. The output file is named `rxwe` followed by the current year (*yyyy*) and week of the year (*ww*).

Example

In this example, the `weekly` command causes the current week's data to be extracted and complete the previous week's extracted file, if it is present.

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
application detail
process detail
weekly
```

To perform the above task using command line arguments, enter:

```
extract -gap -xw
```

yearly

Use the `yearly` command to specify data extraction based on a calendar year.

During execution, the command sets the start and stop dates to the proper dates, based on the year being extracted.

Syntax

```
yearly      [yyyy]
            [yy]
```

Parameters

`yearly`	Extracts the current year's data (the default).
`yearly` *yy*	Extracts a specific year's data (where yy is a number from 00 to 99). The specifications 00 to 27 assume the years 2000 to 2027, whereas 71 to 99 assume the years 1971 to 1999.
`yearly` *yyyy*	Extracts a specific year's data (where `yyyy` is the full-year numbered 1971 to 2027).

If you do not specify the log file before executing the `yearly` command, the default `logglob` file is used.

How to Use It

Use the `yearly` command when you are extracting data for archiving on a yearly basis.

The name of the output file consists of the letters `rxyr` followed by the four digits of the year being extracted. Thus, data from 1999 would be output to a file named `rxyr1999`.

The type of data extracted and summarized follow the normal rules for the `extract` command and can be set before executing the `yearly` command. These settings are honored unless a yearly output file already exists. If it does, data is appended to it, based upon the type of data selected originally.

The `yearly` command has a feature that opens the *previous* year's extracted file and checks to see if it is filled--whether it contains data extracted up to the last day of the year. If not, the `yearly` command appends data to this file to complete the previous year's extraction.

For example, a `yearly` command was executed on December 15, 1998. This created a log file named `rxyr1998` containing data from January 1, 1998 to the current date (December 15).

On January 5, 1999, another `yearly` command is executed. Before the `rxyr1999` file is created for the current year, the `rxyr1998` file from the previous year is opened and checked. When it is found to be incomplete, data is appended to it to complete its extraction until December 31, 1998. Then, the `rxyr1999` file is created to hold data from January 1, 1999 to the current date (January 5).

As long as you execute the `yearly` command at least once a year, this feature will complete each year's file before creating the next year's file. When you see two adjacent yearly files (for example, `rxyr1998` and `rxyr1999`), you can assume that the first file is complete for that year, and it can be archived and purged.

The previous paragraph is true *only* if the raw log files are sized large enough to hold *one full year* of data. It would be more common to size the raw log files smaller and execute the `yearly` command more often (such as once a month).

NOTE

The yearly and extract year commands are similar in that they both extract one calendar year's data. The yearly command ignores the setting of the output command, using instead predefined output file names. It also attempts to append missing data to the previous year's extracted log file if it is still present on the system. The extract year command, on the other hand, will use the settings of the output command. It cannot append data to the previous year's extracted file since it does not know its name.

Example

In this example, application and global detail data is appended to the existing yearly summary file (or creates it, if necessary). The output file is rxy*yyyy* (where *yyyy* represents the current year).

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
application detail
process off
yearly
```

To perform the above task using command line arguments, enter:

```
extract -ga -xy
```

7 Performance Alarms

Introduction

This chapter describes what an alarm is, the syntax used to define an alarm, how an alarm works, and how alarms can be used to monitor performance.

You can use OV Performance Agent to define alarms. These alarms notify you when scopeux or DSI metrics meet or exceed conditions that you have defined.

To define alarms, you specify conditions on each OV Performance Agent system that when met, trigger an alert or action. You define alarms in the OV Performance Agent alarm definitions file, `alarmdef`.

As data is logged by scopeux or DSI, it is compared to the alarm definitions to determine if a condition is met. When this occurs an alert or action is triggered.

With the real time alarm generator you can configure where you want alert notifications sent and whether you want local actions performed. SNMP traps can be sent to HP OpenView Network Node Manager. Alert notifications can be sent to OpenView Operations (OVO). Local actions can be performed on your OV Performance Agent system.

You can analyze historical log file data against the alarm definitions and report the results using the `utility` program's `analyze` command.

Processing Alarms

As performance data is collected by OV Performance Agent, it is compared to the alarm conditions defined in the `alarmdef` file to determine whether the conditions have been met. When a condition is met, an alarm is generated and the actions defined for alarms (ALERTs, PRINTs, and/or EXECs) are performed. You can set up how you want the alarm information communicated once an alarm is triggered. For example, you can:

- send SNMP traps to Network Node Manager
- send messages to OVO
- execute a UNIX command on the local system. For example, to send yourself a message

How Alarms Are Processed

When you first start up OV Performance Agent, the coda daemon looks for each data source configured in the `datasources` configuration file and then starts the alarm generator. Every data source mentioned in your alarm definitions must have a corresponding entry in the `datasources` file. For more information about the `datasources` file and starting and stopping the alarm generator, see Chapter 2 of the *HP OpenView Performance Agent Installation & Configuration Guide*.

As data is collected in the log files, it is compared to the alarm definitions in the `alarmdef` file. When an alarm condition is met, the actions defined in the alarm definition are carried out. Actions can include:

- local actions performed via UNIX commands
- messages sent to Network Node Manager or OVO

Alarm Generator

The OV Performance Agent alarm generator handles the communication of alarm notifications. The alarm generator consists of the alarm generator server (perfalarm), the alarm generator database server (agdbserver), the alarm generator database (agdb), and the utility program agsysdb.

The agdb contains a list of OV Performance Manager analysis nodes (if any) to which you communicate alarm notifications and various on/off flags that you set to define if and where you want the alarm notifications sent. It also contains a list of SNMP trap destination. The agsysdb program is used for displaying and changing the actions taken by alarm events.

Use the following command line option to see a list showing where alert notifications are being sent:

```
agsysdb -l
```

Sending SNMP Traps to Network Node Manager

To send SNMP traps to Network Node Manager, you must add your system name to agdb in OV Performance Agent using the command:

```
agsysdb -add systemname
```

Every ALERT generated will cause an SNMP trap to be sent to the system you defined. The trap text will contain the same message as the ALERT.

To stop sending SNMP traps to a system, you must delete the system name from agdb using the command:

```
agsysdb -delete systemname
```

Sending Messages to OpenView Operations (OVO)

You can have alert notifications sent to OVO if there is an OVO agent on the same system as OV Performance Agent. The OVO agent communicates with the central OVO system.

By default, if the OVO agent is running on the OV Performance Agent system, the alarm generator does *not* execute local actions that are defined in any alarms in the EXEC statement. Instead, it sends a

message to OVO's event browser. If the OVO agent is *not* running on the OV Performance Agent system, the alarm generator does not try to send alert notifications to OVO and local actions are executed.

You can change the default to stop sending information to OVO, even though an OVO agent is running on the OV Performance Agent system, using the command:

```
agsysdb -ovo OFF
```

Executing Local Actions

Without an OVO agent running on the OV Performance Agent system, local actions in EXEC statements will be executed.

You can change the default to turn off local actions as follows:

```
agsysdb -actions off
```

If you want local actions to *always* execute even if the OVO agent is running, type:

```
agsysdb -actions always
```

The following table lists the settings for sending information to OVO and for executing local actions:

Flags	OVO Agent Running	OVO Agent Not Running
OVO Flag		
off	No alert notifications sent to OVO.	No alert notifications sent to OVO.
on	Alert notifications sent to OVO.	No alert notifications sent to OVO.
Local Actions Flag		
off	No local actions executed.	No local actions executed.

Flags	OVO Agent Running	OVO Agent Not Running
always	Local actions executed even if OVO agent is running.	Local actions executed.
on	Local actions sent to OVO.	Local actions executed.

Errors in Processing Alarms

The last error that occurred when sending an alarm is logged in agdb. To view the contents of agdb, type:

```
agsysdb -l
```

The following information is displayed:

```
OVPA alarming status:
OVO messages : on   Last Error : <error number>
  Exec Actions : on   (See status.perfalarm file for errors)

  Analysis system:  <hostname>, Key=<ip address>
    SNMP     : yes     Last Error : <error number>
```

Analyzing Historical Data for Alarms

You can use the utility program's analyze command to find alarm conditions in log file data (see Chapter 4, “Utility Commands,” on page 77). This is different from the processing of real-time alarms explained earlier because you are comparing historical data in the log file to the alarm definitions in the alarmdef file to determine what alarm conditions would have been triggered.

Examples of Alarm Information in Historical Data

The following examples show what is reported when you analyze alarm conditions in historical data.

For the first example, START, END, and REPEAT statements have been defined in the alarm definition. An alarm-start event is listed every time an alarm has met all of its conditions for the specified duration. When

these conditions are no longer satisfied, an alarm-end event is listed. If an alarm condition is satisfied for a period long enough to generate another alarm without having first ended, a repeat event is listed.

Each event listed shows the date and time, alarm number, and the alarm event. EXEC actions are *not* performed, but they are listed with any requested parameter substitutions in place.

```
05/10/99 11:15  ALARM [1] START
  CRITICAL: CPU test 99.97%

05/10/99 11:20  ALARM [1] REPEAT
  WARNING: CPU test 99.997%

05/10/99 11:25  ALARM [1] END
  RESET: CPU test  22.86%
  EXEC: end.script
```

If you are using a color workstation, the following output is highlighted:

CRITICAL statements are RED

MAJOR statements are MAGENTA

MINOR statements are YELLOW

WARNING statements are CYAN

NORMAL statements are GREEN

The next example shows an alarm summary that is displayed after alarm events are listed. The first column lists the alarm number, the second column lists the number of times the alarm condition occurred, and the third column lists the total duration of the alarm condition.

```
Performance Alarm Summary:

  Alarm   Count   Minutes
      1     574      2865
      2       0         0

Analysis coverage using "alarmdef":

Start: 05/04/99 08:00    Stop: 05/06/99 23:59

Total time analyzed:  Days: 2  Hours: 15  Minutes: 59
```

Alarm Definition Components

An alarm occurs when one or more of the conditions you define continues over a specified duration. The alarm definition can include an action to be performed at the start or end of the alarm.

A condition is a comparison between two or more items. The compared items can be metric names, constants, or variables. For example:

```
ALARM gbl_cpu_total_util > 95 FOR 5 MINUTES
```

An action can be specified to be performed when the alarm starts, ends, or repeats. The action can be one of the following:

- an ALERT, which sends a message to OV Performance Manager or OVO or an SNMP trap to Network Node Manager
- an EXEC, which performs a UNIX command, or
- a PRINT, which sends a message to `stdout` when processed using the `utility` program.

For example:

```
ALARM gbl_swap_space_util > 95 FOR 5 MINUTES
   START
      RED ALERT "Global swap space is nearly full"
   END
      RESET ALERT "End of global swap space full condition"
```

You can create more complex actions using Boolean logic, loops through multiple-instance data such as applications, and variables. (For more information, see the next section, “Alarm Syntax Reference”).

You can also use the INCLUDE statement to identify additional alarm definition files you want used. For example, you may want to break up your alarm definitions into smaller files.

Alarm Syntax Reference

This section describes the statements that are available in the alarm syntax. You may want to look at the `alarmdef` file for examples of how the syntax is used to create useful alarm definitions.

Alarm Syntax

```
ALARM  condition [[AND,OR]condition]
  FOR duration [SECONDS, MINUTES]

  [TYPE="string"]
  [SERVICE="string"]
  [SEVERITY=integer]
  [START action]
  [REPEAT EVERY duration [SECONDS, MINUTES] action]
  [END action]

[RED, CRITICAL, ORANGE, MAJOR, YELLOW, MINOR, CYAN, WARNING,
  GREEN, NORMAL, RESET] ALERT message

EXEC "UNIX command"

PRINT message
IF condition
   THEN action
   [ELSE action]

{APPLICATION, PROCESS, DISK, LVOLUME, TRANSACTION, NETIF, CPU,
   FILESYSTEM} LOOP action

INCLUDE "filename"

USE "data source name"

[VAR] name =  value

ALIAS name = "replaced-name"

SYMPTOM  variable [ TYPE = {CPU, DISK, MEMORY, NETWORK}]
  RULE condition  PROB probability
  [RULE condition PROB probability]
    .
    .
```

Conventions

- Braces ({ }) indicate that one of the choices is required.
- Brackets ([ ]) indicate an optional item.
- Items separated by commas within brackets or braces are options. Choose only one.
- Italics indicate a variable name that you replace.
- All syntax keywords are in uppercase.

Common Elements

The following elements are used in several statements in the alarm syntax and are described below.

- comments
- compound statements
- conditions
- constants
- expressions
- metric names
- messages

Comments

You can precede comments either by double forward slashes (//) or the pound sign (#). In both cases, the comment ends at the end of the line. For example:

```
# any text or characters
```

or

```
// any text or characters
```

Compound Statements

Compound statements allow a list of statements to be executed as a single statement. A compound statement is a list of statements inside braces ({}). Use the compound statement with the IF statement, the LOOP statement, and the START, REPEAT, and END clauses of the ALARM statement. Compound statements cannot include ALARM and SYMPTOM statements.

```
{
 statement
 statement
}
```

In the example below, `highest_cpu = 0` defines a variable called `highest_cpu`. The `highest_cpu` value is saved and notifies you only when that `highest_cpu` value is exceeded by a higher `highest_cpu` value.

```
highest_cpu = 0
IF  gbl_cpu_total_util > highest_cpu  THEN
  // Begin compound statement
  {
  highest_cpu = gbl_cpu_total_util
  ALERT "Our new high CPU value is ", highest_cpu, "%"
  }
  // End compound statement
```

Conditions

A condition is defined as a comparison between two items.

```
item1 {>, <, >=, <=, ==, !=}item2
  [AND, OR[item3 {>, <, >=, <=, ==, !=}item4]]
```

where "==" means "equal", and "!=" means "not equal".

Conditions are used in the ALARM, IF, and SYMPTOM statements. An item can be a metric name, a numeric constant, an alphanumeric string enclosed in quotes, an alias, or a variable. When comparing alphanumeric items, only == or != can be used as operators.

Constants

Constants can be either numeric or alphanumeric. An alphanumeric constant must be enclosed in double quotes. For example:

```
345
345.2
"Time is"
```

Constants are useful in expressions and conditions. For example, you may want to compare a metric against a constant numeric value inside a condition to generate an alarm if it is too high, such as

```
gbl_cpu_total_util > 95
```

Expressions

Arithmetic expressions perform one or more arithmetic operations on two or more operands. You can use an expression anywhere you would use a numeric value. Legal arithmetic operators are:

```
+, -, *, /
```

Parentheses can be used to control which parts of an expression are evaluated first.

For example:

```
Iteration + 1
gbl_cpu_total_util - gbl_cpu_user_mode_util
( 100 - gbl_cpu_total_util ) / 100.0
```

Metric Names

When you specify a metric name in an alarm definition, the current value of the metric is substituted. Metric names must be typed exactly as they appear in the metric definition, except for case sensitivity. Metrics definitions can be found in the *HP OpenView Performance Agent Dictionary of Operating Systems Performance Metrics*. If you are using OV Performance Manager, choose On Metrics from the OV Performance Manager help menu to display a list of metrics by platform.

It is recommended that you use fully-qualified metric names if the metrics are from a data source other than the SCOPE data source (such as DSI metrics).

The format for specifying a fully qualified metric is:

```
data_source:instance(class):metric_name
```

A global metric in the SCOPE data source requires no qualification. For example:

```
metric_1
```

An application metric, which is available for each application defined in the SCOPE data source, requires the application name. For example,

```
application_1:metric_1
```

For multi-instance data types such as `application`, `process`, `disk`, `netif`, `transaction`, `lvolume`, `cpu` and `filesystem`, you must associate the metric with the data type name, except when using the LOOP statement. To do this, specify the data type name followed by a colon, and then the metric name. For example, `other_apps:app_cpu_total_util` specifies the total CPU utilization for the application `other_apps`.

NOTE

When specifying fully qualified multi-instance metrics and using aliases within aliases, if one of the aliases has a class identifier, we recommend you use the syntax shown in this example:

```
alias my_fs="/dev/vg01/lvol1(LVOLUME)"
alarm my_fs:LV_SPACE_UTIL > 50 for 5 minutes
```

If you use an application name that has an embedded space, you must replace the space with an underscore (_). For example, `application 1` must be changed to `application_1`. For more information on using names that contain special characters, or names where case is significant, see “ALIAS Statement” on page 240.

If you had a disk named “other” and an application named “other”, you would need to specify the class as well as the instance:

```
other (disk):metric_1
```

A global metric in an extracted log file (where `scope_extract` is the data source name) would be specified this way:

```
scope_extract:application_1:metric_1
```

A DSI metric would be specified this way:

```
dsi_data_source:dsi_class:metric_name
```

NOTE Any metric names containing special characters (such as asterisks) must be aliased before they are specified.

Messages

A message is the information sent by a PRINT or ALERT statement. It can consist of any combination of quoted alphanumeric strings, numeric constants, expressions, and variables. The elements in the message are separated by commas. For example:

```
RED ALERT "cpu utilization=", gbl_cpu_total_util
```

Numeric constants, metrics, and expressions can be formatted for width and number of decimals. `Width` specifies how wide the field should be formatted; `decimals` specifies how many decimal places to use. Numeric values are right-justified. The - (minus sign) specifies left-justification. Alphanumeric strings are always left-justified. For example:

```
metric names [|[-]width[|decimals]]

gbl_cpu_total_util|6|2   formats as '100.00'
(100.32 + 20)|6          formats as '   120'
gbl_cpu_total_util|-6|0  formats as '100   '
gbl_cpu_total_util|10|2  formats as '    99.13'
gbl_cpu_total_util|10|4  formats as '  99.1300'
```

ALARM Statement

The ALARM statement defines a condition or set of conditions and a duration for the conditions to be true. Within the ALARM statement, you can define actions to be performed when the alarm condition starts, repeats, and ends. Conditions or events that you might want to define as alarms include:

- when global swap space has been nearly full for 5 minutes
- when the memory paging rate has been too high for 1 interval
- when your CPU has been running at 75 percent utilization for the last ten minutes

Syntax

```
ALARM condition [[AND,OR]condition]
  FOR duration{SECONDS, MINUTES}
  [TYPE="string"]
  [SERVICE="string"]
  [SEVERITY=integer]
  [START action]
  [REPEAT EVERY duration {SECONDS, MINUTES} action]
  [END action]
```

- The ALARM statement must be a top-level statement. It cannot be nested within any other statement. However, you can include several ALARM conditions in a single ALARM statement. If the conditions are linked by AND, all conditions must be true to trigger the alarm. If they are linked by OR, any one condition will trigger the alarm.
- TYPE is a quoted string of up to 38 characters. If you are sending alarms to OV Performance Manager, you can use TYPE to categorize alarms and to specify the name of a graph template to use. OV Performance Manager can only accept up to eight characters, so up to eight characters are shown.
- SERVICE is a quoted string of up to 200 characters. If you are using OV Navigator, you can link your OV Performance Agent alarms with the services you defined in that product.

  ```
  SERVICE="Service_id"
  ```

- SEVERITY is an integer from 0 to 32767. If you are sending alarms to OV Performance Manager, you can use this to categorize alarms.

- START, REPEAT, and END are *keywords* used to specify what action to take when alarm conditions are met, met again, or stop. You should always have at least one of START, REPEAT, or END in an ALARM statement. Each of these keywords is followed by an *action*.
- *action* – The action most often used with an ALARM START, REPEAT, or END is the ALERT statement. However, you can also use the EXEC statement to mail a message or run a batch file, or a PRINT statement if you are analyzing historical log files with the `utility` program. Any syntax statement is legal except another ALARM.

  `START, REPEAT,` and `END` actions can be compound statements. For example, you can use compound statements to provide both an ALERT and an EXEC.
- Conditions – A condition is defined as a comparison between two items.

  ```
  item1 {>, <, >=, <=, ==, !=}item2
     [AND, OR[item3 {>, <, >=, <=, ==, !=}item4]]
  ```

  where "==" means "equal", and "!=" means "not equal"

  An item can be a metric name, a numeric constant, an alphanumeric string enclosed in quotes, an alias, or a variable. When comparing alphanumeric items, only `==` or `!=` can be used as operators.

  You can use compound conditions by specifying the “OR” and “AND” operator between subconditions. For example:

  ```
  ALARM gbl_cpu_total_util > 90 AND
    gbl_pri_queue > 1 for 5 minutes
  ```
- You also can use compound conditions *without* specifying the “OR” and “AND” operator between subconditions. For example:

  ```
  ALARM gbl_cpu_total_util > 90
    gbl_cpu_sys_mode_util > 50 for 5 minutes
  ```

  will cause an alarm when both conditions are true.

  `FOR duration SECONDS, MINUTES` specifies the time period the condition must remain true to trigger an alarm.

Use caution when specifying durations of less than one minute, particularly when there are multiple data sources on the system. Performance can be seriously degraded if each data source must be polled for data at very small intervals. The duration must be a multiple of the longest collection interval of the metrics mentioned in the alarm condition.

For `scopeux` data, the duration is five minutes; however, the duration for process data is one minute. For DSI data, the duration is five seconds or longer

- `REPEAT EVERY` *`duration`* `SECONDS, MINUTES` specifies the time period before the alarm is repeated.

How It Is Used

The alarm cycle begins on the first interval that all of the ANDed, or one of the ORed alarm conditions have been true for at least the specified duration. At that time, the alarm generator executes the START *action*, and on each subsequent interval checks the REPEAT condition. If enough time has transpired, the *action* for the REPEAT clause is executed. (This continues until one or more of the alarm conditions becomes false.) This completes the alarm cycle and the END *statement* is executed if there is one.

In order for OV Performance Manager to be notified of the alarm, you should use the ALERT statement within the START and END statements. If you do not specify an END ALERT, the alarm generator will automatically send one to OV Performance Manager and OVO and send an SNMP trap to Network Node Manager.

Examples

The following ALARM example sends a red alert when the swap utilization is high for 5 minutes. It is similar to an alarm condition in the default `alarmdef` file. Do not add this example to your `alarmdef` file without removing the default alarm condition, or your subsequent alert messages may be confusing.

```
ALARM  gbl_swap_space_util > 90 FOR 5 MINUTES
    START
      RED ALERT "swap utilization is very high "
   REPEAT EVERY 15 MINUTES
      RED ALERT "swap utilization is still very high "
    END
      RESET ALERT "End of swap utilization condition"
```

This ALARM example tests the metric gbl_swap_space_util to see if it is greater than 90. Depending on how you configured the alarm generator, the ALERT can be sent to the Alarms window in OV Performance Manager, to Network Node Manager via an SNMP trap, or as a message to OVO. If you have OV Performance Manager configured correctly, the RED ALERT statement places the "swap utilization still very high" message in the OV Performance Manager Alarms window.

The REPEAT statement checks for the gbl_swap_space_util condition every 15 minutes. As long as the metric remains greater than 90, the REPEAT statement will send the message "swap utilization is still very high" every 15 minutes.

When the gbl_swap_space_util condition goes below 90, the RESET ALERT statement with the "End of swap utilization condition" message is sent.

The following example defines a compound action within the ALARM statement. This example shows you how to cause a message to be mailed when an event occurs.

```
ALARM gbl_cpu_total_util > 90 FOR 5 MINUTES
  START
  {
    RED ALERT "Your CPU is busy."
    EXEC "echo 'cpu is too high'| mailx root"
  }
  END
     RESET ALERT "CPU no longer busy."
```

The ALERT can trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to OVO. The EXEC can trigger a mail message to be sent as a local action on your OV Performance Agent system, depending on how you configured your alarm generator. If you set up OV Performance Manager to receive alarms from this system, the RED ALERT statement places the "Your CPU is busy" message in the OV Performance Manager Alarms window and causes a message to be sent.

By default, if the OVO agent is running, the local action will not execute. Instead, it will be sent as a message to OVO.

The following two examples show the use of multiple conditions. You can have more than one test condition in the ALARM statement. In this case, each statement must be true for the ALERT to be sent.

The following ALARM example tests the metric gbl_cpu_total_util and the metric gbl_cpu_sys_mode_util. If both conditions are true, the RED ALERT statement sends a red alert. When either test condition becomes false, the RESET is sent.

```
ALARM gbl_cpu_total_util > 90
  AND gbl_cpu_sys_mode_util > 50 FOR 5 MINUTES
START
  RED ALERT "CPU busy and Sys Mode CPU util is high."
END
  RESET ALERT "The CPU alert is now over."
```

The next ALARM example tests the metric gbl_cpu_total_util and the metric gbl_cpu_sys_mode_util. If either condition is true, the RED ALERT statement sends a red alert.

```
ALARM gbl_cpu_total_util > 90
  OR
      gbl_cpu_sys_mode_util > 50 FOR 10 MINUTES
START
  RED ALERT "Either total CPU util or sys mode CPU high"
```

WARNING

Do not use metrics that are logged at different intervals in the same alarm. For example, you should not loop on a process (logged at 1-minute intervals) based on the value of a global metric (logged at 5-minute intervals) in a statement like this:

```
IF global_metric THEN
   PROCESS LOOP...
```

The different intervals cannot be synchronized as you might expect, so results will not be valid.

ALERT Statement

The ALERT statement allows a message to be sent to OV Performance Manager, Network Node Manager, or OVO. It also allows the creation and deletion of the alarm symbols in the Network Node Manager map associated with OV Performance Manager and controls the color of the alarm symbols, depending on the severity of the alarm. (For more information, see OV Performance Manager online Help.)

The ALERT statement is most often used as an action within an ALARM. It could also be used within an IF statement to send a message as soon as a condition is detected instead of after the duration has passed. If an ALERT is used outside of an ALARM or IF statement, the message will be sent at every interval.

Syntax

```
[RED, CRITICAL, ORANGE, MAJOR, YELLOW, MINOR, CYAN,
    WARNING, GREEN, NORMAL, RESET] ALERT message
```

- RED is synonymous with CRITICAL, ORANGE is synonymous with MAJOR, YELLOW is synonymous with MINOR, CYAN is synonymous with WARNING, and GREEN is synonymous with NORMAL. These keywords turn the alarm symbol to the color associated with the alarm condition in the Network Node Manager map associated with OV Performance Manager. They also send the message with other information to the OV Performance Manager Alarms window. CYAN is the default. However, if you are using version C.00.08 or earlier of OV Performance Manager, YELLOW is the default.
- RESET records the *message* in the OV Performance Manager Alarms window and deletes the alarm symbol in the Network Node Manager map associated with OV Performance Manager. A RESET ALERT without a message is sent automatically when an ALARM condition ENDs if you do not define one in the alarm definition.
- *message* — A combination of strings and numeric values used to create a message. Numeric values can be formatted with the parameters `[|[-]width[|decimals]]`. `Width` specifies how wide the field should be formatted; `decimals` specifies how many decimal places to use. Numeric values are right-justified. The `-` (minus sign) specifies left-justification. Alphanumeric strings are always left-justified.

How It Is Used

The ALERT can also trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to OVO, depending on how you configured your alarm generator. If you configured OV Performance Manager to receive alarms from this system, the ALERT sends a message to the OV Performance Manager Alarms window.

If an ALERT statement is used outside of an ALARM statement, the alert message will show up in the OV Performance Manager Alarms window but no symbol will be created in the Network Node Manager map.

For alert messages sent to OVO, the WARNINGS will be displayed in blue in the message browser

Example

An typical ALERT statement is:

```
RED ALERT "CPU utilization = ", gbl_cpu_total_util
```

If you have OV Performance Manager and Network Node Manager, this statement creates a red alarm symbol in the Network Node Manager map associated with OV Performance Manager and sends a message to the OV Performance Manager Alarms window that reads:

```
CPU utilization = 85.6
```

EXEC Statement

The EXEC statement allows you to specify a UNIX command to be performed on the local system. For example, you could use the EXEC statement to send mail to an IT administrator each time a certain condition is met.

EXEC should be used within an ALARM or IF statement so the UNIX command is performed only when specified conditions are met. If an EXEC statement is used outside of an ALARM or IF statement, the action will be performed at unpredictable intervals.

Syntax

```
EXEC "UNIX command"
```

- *UNIX command* — a command to be performed on the local system.

Do not use embedded double quotes (") in EXEC statements. Doing so causes `perfalarm` to fail to send the alarm to OVO. Use embedded single (') quotes instead. For example:

```
EXEC "echo 'performance problem detected' "
```

How It Is Used

The EXEC can trigger a local action on your local system, depending on how you configured your alarm generator. For example, you can turn local actions on or off. If you configured your alarm generator to send information to OVO, local actions will not usually be performed.

Examples

In the following example, the EXEC statement performs the UNIX `mailx` command when the `gbl_disk_util_peak` metric exceeds 20.

```
IF gbl_disk_util_peak > 20 THEN
  EXEC "echo 'high disk utilization detected'| mailx root"
```

The next example shows the EXEC statement sending mail to the system administrator when the network packet rate exceeds 1000 per second average for 15 minutes.

```
ALARM gbl_net_packet_rate > 1000 for 15 minutes
    TYPE = "net busy"
    SEVERITY = 5
    START
      {
      RED ALERT "network is busy"
      EXEC "echo 'network busy condition detected'| mailx root"
      }
    END
    RESET ALERT "NETWORK OK"
```

NOTE

Be careful when using the EXEC statement with commands or scripts that have high overhead if it will be performed often.

The alarm generator executes the command and waits until it completes before continuing. We recommend that you not specify commands that take a long time to complete.

PRINT Statement

The PRINT statement allows you to print a message from the `utility` program using its `analyze` function. The alarm generator ignores the PRINT statement.

Syntax

```
PRINT message
```

- *message* — A combination of strings and numeric values that create a message. Numeric values can be formatted with the parameters `[|[-]width[|decimals]]`. `Width` specifies how wide the field should be formatted; `decimals` specifies how many decimal places to use. Alphanumeric components of a message must be enclosed in quotes. Numeric values are right-justified. The - (minus sign) specifies left-justification. Alphanumeric strings are always left-justified.

Example

```
PRINT "The total time the CPU was not idle is",
              gbl_cpu_total_time |6|2, "seconds"
```

When executed, this statement prints a message such as the following:

```
The total time the CPU was not idle is 95.00 seconds
```

IF Statement

Use the IF statement to define a condition using IF-THEN logic. The IF statement should be used within the ALARM statement. However, it can be used by itself or any place in the `alarmdef` file where IF-THEN logic is needed.

If you specify an IF statement outside of an ALARM statement, you do not have control over how frequently it gets executed.

Syntax

```
IF condition THEN action [ELSE action]
```

- IF *condition* — A condition is defined as a comparison between two items.

  ```
  item1 {>, <, >=, <=, ==, !=}item2
     [AND, OR[item3 {>, <, >=, <=, ==, !=}item4]]
  ```

  where "==" means "equal", and "!=" means "not equal".

  An item can be a metric name, a numeric constant, an alphanumeric string enclosed in quotes, an alias, or a variable. When comparing alphanumeric strings, only == or != can be used as operators.

- *action* — Any action, or set a variable. (ALARM is not valid in this case.)

How It Is Used

The IF statement tests the *condition*. If the *condition* is true, the *action* after the THEN is executed. If the *condition* is false, the *action* depends on the optional ELSE clause. If an ELSE clause has been specified, the *action* following it is executed; otherwise the IF statement does nothing.

Example

In this example, a CPU bottleneck symptom is calculated and the resulting bottleneck probability is used to define cyan or red ALERTs. If you have OV Performance Manager configured correctly, the message "`End of CPU Bottleneck Alert`" is displayed in the OV Performance Manager Alarms window along with the percentage of CPU used.

The ALERT can also trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to OVO, depending on how you configured your alarm generator.

```
SYMPTOM CPU_Bottleneck > type=CPU
     RULE gbl_cpu_total_util  > 75  prob 25
     RULE gbl_cpu_total_util  > 85  prob 25
     RULE gbl_cpu_total_util  > 90  prob 25
     RULE gbl_cpu_total_util  >  4  prob 25

     ALARM CPU_Bottleneck > 50 for 5 minutes
       TYPE="CPU"
       START
         IF CPU_Bottleneck > 90 then
           RED ALERT "CPU Bottleneck probability= ",
                     CPU_Bottleneck, "%"
         ELSE
           CYAN ALERT "CPU Bottleneck probability= ",
                     CPU_Bottleneck, "%"
       REPEAT every 10 minutes
         IF CPU_Bottleneck > 90 then
           RED ALERT "CPU Bottleneck probability= ",
                     CPU_Bottleneck, "%"
         ELSE
           CYAN ALERT "CPU Bottleneck probability= ",
                     CPU_Bottleneck, "%"
       END
          RESET ALERT "End of CPU Bottleneck Alert"
```

WARNING

Do not use metrics that are logged at different intervals in the same statement. For instance, you should not loop on a process (logged at 1-minute intervals) based on the value of a global metric (logged at 5-minute intervals) in a statement like this:

```
IF global_metric THEN
  PROCESS LOOP ...
```

The different intervals cannot be synchronized as you might expect, so results will not be valid.

LOOP Statement

The LOOP statement goes through multiple-instance data types and performs the *action* defined for each instance.

Syntax

```
{APPLICATION, PROCESS, LVOLUME, DISK, CPU, FILESYSTEM,
TRANSACTION, NETIF}
LOOP
   action
```

- APPLICATION, PROCESS, LVOLUME, DISK, CPU, FILESYSTEM, TRANSACTION, NETIF — OV Performance Agent data types that contain multi-instance data.
- *action* — PRINT, EXEC, ALERT, set variables.

How It Is Used

As LOOP statements iterate through each instance of the data type, metric values change. For instance, the following LOOP statement prints the name of each application to `stdout` if you are using the `utility` program's `analyze` command.

```
APPLICATION LOOP
  PRINT app_name
```

A LOOP can be nested within another LOOP statement up to a maximum of five levels.

In order for the LOOP to execute, the LOOP statement must refer to one or more metrics of the same data type as the type defined in the LOOP statement.

Example

You can use the LOOP statement to cycle through all active applications.

The following example shows how to determine which application has the highest CPU at each interval. When the statement "`highest_cpu = highest_cpu`" is executed during the first interval, `highest_cpu` will be initialized to 0. During subsequent intervals, `highest_cpu` will be initialized to the value from the previous interval.

```
highest_cpu = 0
 APPLICATION loop
   IF app_cpu_total_util > highest_cpu THEN
    {
      highest_cpu = app_cpu_total_util
      big_app = app_name
      ALERT "Application ", app_name, " has the highest cpu
util at ",highest_cpu_util|5|2, "%"

     }

  ALARM highest_cpu > 50 for 15 minutes
    START
      RED ALERT  big_app, " is the highest CPU user at ",
                           highest_cpu, "%"
    REPEAT EVERY 15 minutes
      CYAN ALERT big_ap, " is the highest CPU user at ",
                          highest_cpu, "%"
  END
    RESET ALERT "No applications using excessive cpu"
```

INCLUDE Statement

Use the INCLUDE statement to include another alarm definitions file along with the default `alarmdef` file.

Syntax

```
INCLUDE "filename"
```

where *filename* is the name of another alarm definitions file. The file name must always be fully qualified.

How It Is Used

The INCLUDE statement could be used to separate logically distinct sets of alarm definitions into separate files.

Example

For example, if you have some alarm definitions in a separate file for your transaction metrics and it is named

```
trans_alarmdef1
```

You can include it by adding the following line to the alarm definitions in your `alarmdef1` file:

```
INCLUDE "/var/opt/perf/trans_alarmdef1"
```

USE Statement

You can add the USE statement to simplify the use of metric names in the `alarmdef` file when data sources other than the default SCOPE data source are referenced. This allows you to specify a metric name without having to include the data source name.

The data source name must be defined in the `datasources` file. The `alarmdef` file will fail its syntax check if an invalid or unavailable data source name is encountered.

NOTE The appearance of a USE statement in the `alarmdef` file does not imply that all metric names that follow will be from the specified data source.

Syntax

USE "*datasourcename*"

How It Is Used

As the alarm generator (`perfalarm`) checks the `alarmdef` file for valid syntax, it builds an ordered search list of all data sources that are referenced in the file. `perfalarm` sequentially adds entries to this data source search list as it encounters fully-qualified metric names or USE statements. This list is subsequently used to match metric names that are not fully qualified with the appropriate data source name. The USE statement provides a convenient way to add data sources to `perfalarm`'s search list, which then allows for shortened metric names in the `alarmdef` file. For a discussion of metric name syntax, see “Metric Names” on page 218.

`perfalarm`'s default behavior for matching metric names to a data source is to look first in the SCOPE data source for the metric name. This implied `USE "SCOPE"` statement is executed when `perfalarm` encounters the first metric name in the `alarmdef` file. This feature enables a default search path to the SCOPE data source so that SCOPE metrics can be referenced in the `alarmdef` file without the need to fully qualify them. This is shown in the following example on the next page.

```
ALARM gbl_cpu_total_util > 80 FOR 10 MINUTES
     START RED ALERT "CPU utilization too high"

USE "ORACLE7"

ALARM ActiveTransactions >= 95 FOR 5 MINUTES
      START RED ALERT "Nearing limit of transactions for
                       ORACLE7"
```

When perfalarm checks the syntax of the alarmdef file containing the above statements, it encounters the metric "gbl_cpu_total_util" and then tries to find its data source. perfalarm does not yet have any data sources in its search list of data sources, so it executes an implied USE "SCOPE" statement and then searches the SCOPE data source to find the metric name. A match is found and perfalarm continues checking the rest of the alarmdef file.

When perfalarm encounters the USE "ORACLE7" statement, it adds the ORACLE7 data source to the search list of data sources. When the "ActiveTransactions" metric name is encountered, perfalarm sequentially searches the list of data sources starting with the SCOPE data source. SCOPE does not contain that metric name, so the ORACLE7 data source is searched next and a match is found.

If perfalarm does not find a match in any data source for a metric name, an error message is printed and perfalarm terminates.

To change the default search behavior, a USE statement can be added to the beginning of the alarmdef file before any references to metric names. This will cause the data source specified in the USE statement to be added to the search list of data sources before the SCOPE data source. The data source(s) in the USE statement(s) will be searched before the SCOPE data source to find matches to the metrics names. This is shown in the following example.

Once a data source has been referenced with a USE statement, there is no way to change its order or to remove it from the search list.

```
USE "ORACLE7"

ALARM gbl_cpu_total_util > 80 FOR 10 MINUTES
      START RED ALERT "CPU utilization too high"

ALARM ActiveTransactions >= 95 FOR 5 MINUTES
      START RED ALERT "Nearing limit of
      transactions for ORACLE7"
```

In the example above, the order of the statements in the alarmdef file has changed. The USE "ORACLE7" statement is defined before any metric names are referenced, therefore the ORACLE7 data source is added as the first data source in the search list of data sources. The implied USE "SCOPE" statement is executed when perfalarm encounters the first metric name "gbl_cpu_total_util." Because the GBL_CPU_TOTAL_UTIL metric name is not fully-qualified, perfalarm sequentially searches through the list of data sources starting with ORACLE7. ORACLE7 does not contain that metric name so the SCOPE data source is searched next and a match is found.

perfalarm continues checking the rest of the alarmdef file. When perfalarm encounters the "ActiveTransactions"" metric, it sequentially searches the list of data sources starting with ORACLE7. A match is found and perfalarm continues searching the rest of the alarmdef file. If perfalarm does not find a match in any data source for a metric name (that is not fully-qualified), an error message will be printed and perfalarm terminates.

Be careful how you use the USE statement when multiple data sources contain the same metric names. perfalarm sequentially searches the list of data sources. If you are defining alarm conditions from different data sources that use the same metric names, you must qualify the metric names with their data source names to guarantee that the metric value is retrieved from the correct data source. This is shown in the following example where the metric names in the alarm statements each include their data sources.

```
ALARM ORACLE7:ActiveTransactions >= 95 FOR 5 MINUTES
  START RED ALERT "Nearing limit of transactions for
              ORACLE7"

ALARM FINANCE:ActiveTransactions >= 95 FOR 5 MINUTES
  START RED ALERT "Nearing limit of transactions for
              FINANCE"
```

VAR Statement

The VAR statement allows you to define a variable and assign a value to it.

Syntax

```
[VAR] name = value
```

- *name* — Variable names must begin with a letter and can include letters, digits, and the underscore character. Variable names are not case-sensitive.
- *value* — If the value is an alphanumeric string, it must be enclosed in quotes.

How It Is Used

VAR assigns a value to the user variable. If the variable did not previously exist, it is created.

Once defined, variables can be used anywhere in the `alarmdef` file.

Examples

You can define a variable by assigning something to it. The following example defines the numeric variable `highest_CPU_value` by assigning it a value of zero.

```
highest_CPU_value = 0
```

The next example defines the alphanumeric variable my_name by assigning it an empty string value.

```
my_name = ""
```

ALIAS Statement

The ALIAS statement allows you to substitute an alias if any part of a metric name (class, instance, or metric) has a case-sensitive name or a name that includes special characters. These are the only circumstances where the ALIAS statement should be used.

Syntax

ALIAS *name* = "*replaced-name*"

- *name* — The name must begin with a letter and can include letters, digits, and the underscore character.
- *replaced-name* — The name that must be replaced by the ALIAS statement to make it uniquely recognizable to the alarm generator.

How It Is Used

Because of the way the `alarmdef` file is processed, if any part of a metric name (class, instance, or metric name) can be identified uniquely only by recognizing uppercase and lowercase, you will need to create an alias. You will also need to create an alias for any name that includes special characters. For example, if you have applications called "BIG" and "big," you'll need to alias "big" to ensure that they are viewed as different applications. You must define the alias somewhere in the `alarmdef` file before the *first* instance of the name you want substituted.

Examples

Because you cannot use special characters or upper and lower case in the syntax, using the application name "AppA" and "appa" could cause errors because the processing would be unable to distinguish between the two. You would alias "AppA" to give it a uniquely recognizable name. For example:

```
ALIAS appa_uc = "AppA"
 ALERT "CPU alert for AppA.util is",appa_uc:app_cpu_total_util
```

If you are using an alias for an instance with a class identifier, include both the instance name and the class name in the alias. The following example shows the alias for the instance name 'other' and the class name 'APPLICATION.'

```
ALIAS my_app="other(APPLICATION)"
   ALERT my_app:app_cpu_total_util > 50 for 5 minutes
```

SYMPTOM Statement

A symptom provides a way to set a single variable value based on a set of conditions. Whenever any of the conditions is true, its probability value is added to the value of the symptom variable.

Syntax

```
SYMPTOM variable
  RULE condition PROB probability
  [RULE condition PROB probability]
       .
       .
       .
```

- The keywords SYMPTOM and RULE are used exclusively in the SYMPTOM statement and cannot be used in other syntax statements. The SYMPTOM statement must be a top-level statement and cannot be nested within any other statement. No other statements can follow SYMPTOM until all its corresponding RULE statements are finished.
- *variable* is a variable name that will be the name of this symptom. Variable names defined in the SYMPTOM statement can be used in other syntax statements, but the variable value should not be changed in those statements.
- RULE is an option of the SYMPTOM statement and cannot be used independently. You can use as many RULE options as needed within the SYMPTOM statement. The SYMPTOM variable is evaluated according to the rules at each interval.
- *condition* is defined as a comparison between two items.

  ```
  item1 {>, <, >=, <=, ==, !=}item2
  [item3 {>, <, >=, <=, ==, !=}item4]
  ```

  where "==" means "equal" and "!=" means "not equal".

  An item can be a metric name, a numeric constant, an alphanumeric string enclosed in quotes, an alias, or a variable. When comparing alphanumeric items, only == or != can be used as operators.
- *probability* is a numeric constant. The probabilities for each true SYMPTOM RULE are added together to create a SYMPTOM value.

How It Is Used

The sum of all probabilities where the condition between measurement and value is true is the probability that the symptom is occurring.

Example

```
SYMPTOM CPU_Bottleneck
RULE gbl_cpu_total_util > 75  PROB 25
RULE gbl_cpu_total_util > 85  PROB 25
RULE gbl_cpu_total_util > 90  PROB 25
RULE gbl_run_queue      >  3  PROB 50
IF CPU bottleneck > 50 THEN
CYAN ALERT "The CPU symptom is: ", CPU_bottleneck
```

Alarm Definition Examples

The following examples show typical uses of alarm definitions.

Example of a CPU Problem

If you have OV Performance Manager configured correctly, this example turns the alarm symbol CYAN in the Network Node Manager map (whenever CPU utilization exceeds 90 percent for 5 minutes and the CPU run queue exceeds 3 for 5 minutes), and sends a message to the OV Performance Manager Alarms window.

```
ALARM gbl_cpu_total_util > 90 AND
  gbl_run_queue > 3 FOR 5 MINUTES
START
  CYAN ALERT "CPU too high at", gbl_cpu_total_util, "%"
REPEAT EVERY 20 MINUTES
{
  RED ALERT "CPU still to high at ", gbl_cpu_total_util, "%"
  EXEC "/usr/bin/pager -n 555-3456"
}
END
  RESET ALERT "CPU at ", gbl_cpu_total_util, "% - RELAX"
```

The ALERT could also trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to OVO, depending on how you configured the alarm generator.

If both conditions continue to hold true after 20 minutes, a red alert is generated, the alarm symbol turns red in the Network Node Manager map, and another message is sent to the OV Performance Manager Alarms window. A program is then run to page the system administrator.

When either one of the alarm conditions fails to be true, the alarm symbol is deleted and a message is sent to the OV Performance Manager Alarms window showing the global CPU utilization, the time the alert ended, and a note to RELAX.

Example of Swap Utilization

If you have OV Performance Manager configured correctly, this example turns the alarm symbol red in the Network Node Manager map (whenever swap space utilization exceeds 95 percent for 5 minutes) and a message is written to the OV Performance Manager Alarms window.

```
ALARM gbl_swap_space_util > 95 FOR 5 MINUTES
  START
    RED ALERT "GLOBAL SWAP space is nearly full "
  END
    RESET ALERT "End of GLOBAL SWAP full condition"
```

The ALERT can trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to OVO, depending on how you configured your alarm generator.

Example of Time-Based Alarms

You can specify a time interval during which alarm conditions can be active. For example, if you are running system maintenance jobs that are scheduled to run at regular intervals, you can specify alarm conditions for normal operating hours and a different set of alarm conditions for system maintenance hours.

In this example, the alarm will only be triggered during the day from 8:00AM to 5:00PM.

```
  start_shift = "08:00"
  end_shift = "17:00"

  ALARM gbl_cpu_total_util > 80
      TIME > start_shift
      TIME < end_shift for 10 minutes
    TYPE = "cpu"
    START
      CYAN ALERT "cpu too high at ", gbl_cpu_total_util, "%"
    REPEAT EVERY 10 minutes
      RED ALERT"cpu still too high at ", gbl_cpu_total_util,
              "%"
    END
      IF time == end_shift then
      {
      IF gbl_cpu_total_util > 80 then
         RESET ALERT "cpu still too high, but at the end of
                     shift"
      ELSE
         RESET ALERT  "cpu back to normal"
      }
      ELSE
         RESET ALERT "cpu back to normal"
```

Example of Disk Instance Alarms

Alarms can be generated for a particular disk by identifying the specific disk instance name and corresponding metric name.

The following example of alarm syntax generates alarms for a specific disk instance. Aliasing is required when special characters are used in the disk instance.

```
ALIAS diskname="2/0/1.5.0"
ALARM diskname:bydsk_phys_read > 1000 for 5 minutes
TYPE="Disk"
  START
     RED ALERT "Disk 2/0/1.50 red alert"
  REPEAT EVERY 10 MINUTES
     CYAN ALERT "Disk 2/0/1.5.0 cyan alert"
  END
     RESET ALERT "Disk 2/0/1.5.0 reset alert"
```

Customizing Alarm Definitions

You specify the conditions that generate alarms in the alarm definitions file `alarmdef`. When OV Performance Agent is first installed, the `alarmdef` file contains a set of default alarm definitions. You can use these default alarm definitions or customize them to suit your needs.

You can customize your `alarmdef` file as follows:

1. Revise your alarm definition(s) as necessary. You can look at examples of the alarm definition syntax elsewhere in this chapter.
2. Save the file.
3. Validate the alarm definitions using the OV Performance Agent `utility` program:
   a. Type **`utility`**
   b. At the prompt, type

      `checkdef`

      This checks the alarm syntax and displays errors or warnings if there any problems with the file.
4. In order for the new alarm definitions to take affect, type:

   `ovpa restart alarm`

   This causes the alarm generator to stop, restart, and read the customized `alarmdef` file.

You can use a unique set of alarm definitions for each OV Performance Agent system, or you can choose to standardize monitoring of a group of systems by using the same set of alarm definitions across the group.

The best way to learn about performance alarms is to experiment with adding new alarm definitions or changing the default alarm definitions.

8 Communicating Across a Firewall

Introduction

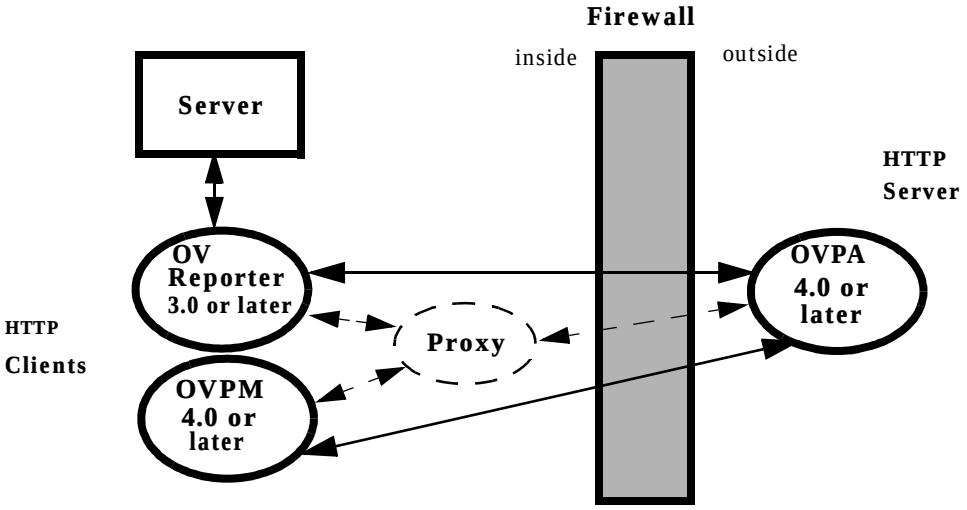
OV Performance Agent (OVPA) now uses HTTP 1.1 based communications interface for data access between client and server applications. This interface is flexible, since it can use proxies, requires fewer ports and is firewall friendly. This replaces the functions that were implemented in previous versions of OVPA by the `perflbd` and `rep_server` daemons. These daemons, and their dependency on the DCE subsystem can still be used by all UNIX systems except for OVPA for Linux.

In general, a firewall may be defined as a method for filtering the flow of data communications between one network and another. There are different ways to configure the HTTP communication in a firewall environment. The standard recommended way is to use HTTP proxies for OVPA datacomm through a firewall. This simplifies the configuration by using proxies that are often already in use in your environment. The firewall must be open for exactly one port if proxies are to be used in both directions.

In a typical remote communication a client, using the source port, connects to a server that is listening on the destination port on a remote system. For firewall configuration it is important to know which system initiates the communication **client**) and which receives communication requests s**erver**), so that the firewall rules can be set up accordingly.

Figure 8-1 shows how OVPA communicates with Reporter (version 3.0 or later) and OV Performance Manager (OVPM version 4.0 or later) through a firewall. OVPA is an HTTP server. Reporter and OVPM are HTTP clients. If an HTTP proxy is used, Reporter and OVPM communicate with OVPA via the proxy.

Figure 8-1 Communicating with OVPA in a Firewall Environment



OVPM version 3.x uses the PerfView technology. PerfView does not use the new datacomm components, but it will connect to OV Performance Agent 4.0 and later for all UNIX platforms except for OVPA for Linux. In the future, OVPM 3.x will be discontinued.

NOTE For Linux users, if you cannot upgrade OVPM 3.x to OVPM 4.0 or later, you must access the OVPA for Linux data locally via the `extract` program's `export` function. Likewise, OVPA for Linux will not send alarm notifications to the PerfView monitor.

Ports Used for Communication with OVPA

To access data collected by OVPA, ports for the HTTP server (OVPA) and the HTTP client (Reporter and OVPM) need to be opened. There are two ways to configure HTTP clients in a firewall environment: with an HTTP proxy and without.

With HTTP Proxy

The recommended way is to use HTTP proxies when communicating through a firewall. This simplifies the configuration because proxies are often in use and the firewall needs to be opened only for the proxy system and for a smaller number of ports. It is recommended that you do not change the default 383 port.

Default Ports for OVPA (with proxies) are shown in Table 8-1.

Table 8-1 OVPA Default Ports

Source	Destination	Protocol	Source Port	Destination Port	Description
PROXY	MGD NODE	HTTP	Defined by the proxy	383	Local Location Broker
PROXY	MGD NODE	HTTP	Defined by the proxy	381	OVPA

Without HTTP Proxy

If HTTP proxies are not available, additional ports need to be opened for both source and destination ports and additional configuration settings are required on the Reporter and OVPM system.

Overview of OVPA Communications Configuration

To configure communications with OVPA in a firewall environment, follow these guidelines:

- Understand your firewall environment including the client and server data flow.
- For Linux systems, if an OVO agent (version 7.0 or later) is already running on the Linux system then check to see if communications across a firewall to Reporter and OVPM is already configured. If it is, then you don't need to complete the rest of the steps documented here. Refer to the *HP OpenView Operations for UNIX Firewall White Paper* on the HP OpenView Manuals web site at

  `http://ovweb.external.hp.com/lpe/doc_serv`

Select operations for hpux, version 7.x and click [**Search**].)

- Check the port usage in your environment.
- Configure the HTTP Server Port on the OVPA system using the default.txt communications configuration file.
- If proxies are used, configure Reporter and OVPM to know the proxy to be used to contact OVPA.
- If proxies are not used, configure the HTTP Client port or port range for Reporter and OVPM.
- Test the communications across the firewall.

Refer to the following sections for details on configuring communications across firewalls:

- "Configure OVPA Server Port"
- "Configure Reporter and/or OVPM"
- "Other Considerations"
- "default.txt File and Configuration Parameters"

Communications default.txt Configuration File

Enter parameters into the default.txt file to set up the OVPA communications configuration. For more information about the default.txt file and its parameters, see "default.txt File and Configuration Parameters" on page 254.

- For Linux, the default.txt file is located in:

  /var/opt/OV/conf/BBC/

- For Windows systems with Reporter and OVPM, the default.txt file is located in:

  <OVDataDir>/conf/BBC/

  where <OVDataDir> is defined by the registry setting:

  HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\HP OpenView\DataDir

- For UNIX systems with Reporter and OVPM, refer to the Reporter and OVPM system documentation for the location of the default.txt file.

Configure OVPA Server Port

If you want to override the default server port of 381 for OVPA, use the `default.txt` file parameter `SERVER_PORT` to set a different port to be used by the HTTP server (OVPA):

1. On the OVPA system, locate the `default.txt` file.
2. Locate the line, `SERVER_PORT = 381`, and change the port number. For example:

   ```
   SERVER_PORT = 1100
   ```

3. Restart OVPA processes:

   ```
   /opt/perf/bin/ovpa stop
   /opt/perf/bin/ovpa start
   ```

   OR

   ```
   /opt/perf/bin/ovpa restart
   ```

Configure Reporter and/or OVPM

You can configure the HTTP clients (Reporter or OVPM for Windows or UNIX) in a firewall environment in one of two ways:

- With HTTP Proxy – This is the recommended way. See the section “Configure Reporter/OVPM with HTTP Proxy” .
- Without HTTP Proxy – This is *not* the recommended way. See the section “Configure Reporter/OVPM without HTTP Proxy” .

Configure Reporter/OVPM with HTTP Proxy

When an HTTP proxy is used, Reporter and/or OVPM for Windows and UNIX need to be configured to specify the proxy to be used to contact OVPA. Configure Reporter/OVPM as follows:

Edit the `default.txt` configuration file:

In the `[DEFAULT]` section of the `default.txt` file, locate the lines that relate to the `PROXY` and set the `PROXY` parameter as follows.

PROXY web-proxy.hp.com:8088- (localhost, *.hp.com) + (*)

In this example, the proxy `web-proxy` will be used with port 8088 for every server (*) except requests for the local machine (localhost) and requests internal to HP (matching `*.hp.com`, for example `www.hp.com`.

NOTE Any settings defined in the OpenView Operations `nodeinfo` communications configuration file will take precedence over the settings defined in the `default.txt` file.

Configure Reporter/OVPM without HTTP Proxy

If your firewall environment does not have proxies then you may want to specify the HTTP client ports directly if you want to filter based on both source and destination.

If Reporter and OVPM for Windows are installed on the same system and both access OVPA in parallel, you would specify a port range as described in this section. If they are running on different systems, you can instead specify a single port for each.

Edit the `default.txt` file as follows:

1. Locate the lines that apply to `CLIENT_PORT` and uncomment the line `;CLIENT_PORT =`.
2. Specify the port range for the `CLIENT_PORT` parameter. For example:

   `CLIENT_PORT = <port range>`

   Where `<port range>` is the range of ports you want to use. For example:

   `CLIENT_PORT = 14000-14003`

NOTE If OVO is installed, any settings defined in the OVO `nodeinfo` communications configuration file will take precedence over the settings defined in the `default.txt` file.

Other Considerations

About Systems with Multiple IP Addresses

If your environment includes systems with multiple network interfaces and IP addresses and you want to use a dedicated interface for the HTTP-based communication, then you can use the `nodeinfo` (or `default.txt`) parameter `CLIENT_BIND_ADDR` and `SERVER_BIND_ADDR` to specify the IP address that should be used. See the "default.txt Configuration Parameters" on page 257 for more information on these parameters.

default.txt File and Configuration Parameters

default.txt Configuration File

Following is the `default.txt` file included with OVPA:

```
;
; HP OpenView communications configuration file: default.txt
;
; This file is to be used to customize communication parameters
; for HP OpenView applications.
;
; All lines beginning with a ';' character are considered comments
; and will be ignored. Leading spaces are allowed and ignored for
; all entries in this file. Trailing spaces are also ignored.
;
; Sections are denoted using '[]' characters. All applications
; loading this file will recognize the [DEFAULT] section and
; sections with their application name. All other sections are
; ignored by the application loading this file. Application names
; are case sensitive.
;
; Key/value pairs define configuration parameters recognized by the
; application. Syntax is:
; KEY = value
; The '=' sign is required. Spaces before and after the equals sign
; are ignored. Keys are case sensitive.
;
; Keys may be defined multiple times. The last definition read is
; the one that will become active. Previous entries are
; overwritten.
;
```

```
; The key name 'INCLUDE' is reserved and has a special meaning. At
; this point the value will define a new configuration file to be
; read. The defined configuration file is read and then the
; remainder of the current configuration file is processed. The
; file to be read takes the form of a URI, e.g. file:
; /var/opt/OV/conf/BBC/CODAConfig.txt
; Protocols http:// and file:/ are currently supported.
; If no protocol is specified, file:/ is assumed.
; The filename "nodeinfo" is automatically expanded to the full
; pathname of the nodeinfo file on the local node.
;

[DEFAULT]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The PROXY parameter may be set in the DEFAULT section.
; Defines which proxy and port number to use for a given hostname.
; Format is proxy:port +(a)-(b); proxy2:port2 +(c)-(d); ...
; 'a', 'b', 'c' and 'd' are comma separated lists of hostnames,
; networks, and/or IP addresses that apply to the proxy. Multiple
; proxies may be defined for one PROXY key. '-' before the list
; denotes those entities that do not use this proxy, '+' before the
; list denotes entities that do use this proxy. The first matching
; proxy is used.
;
;PROXY = web-proxy.hp.com:8088-(localhost,*.hp.com)+(*)
;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The following are examples of parameters that may be set
; by a client application. These should be set in the client
; specific section (see below).
;
; Local port number the client will bind to.
; CLIENT_PORT = 16000
; Range of local ports the client may bind to.
; NOTE: Range syntax is only valid from BBC 2.5.2.0 or
; greater
;CLIENT_PORT = 16000-16010
; Local address the client will bind to.
;CLIENT_BIND_ADDR = 10.10.10.10
;
;
; The following are examples of parameters that may be set
; by a server application. These should be set in the server
```

```
; specific section (see below).
;
; Local port number the server will bind to.
;SERVER_PORT = 80
; Local address the server will bind to.
;SERVER_BIND_ADDR = 10.10.10.10


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; The following settings are for HP internal use only
; or are used for local communication.
; These settings don't have to be changed.
;
[com.hp.openview.OvAgency.OvAgencyCommand]
SERVER_PORT = 6600

[com.hp.openview.OvAgency.OvPolicyFileObject]
SERVER_PORT = 6600

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; reserved for future use - not used at all in OVOW 7.0
;
[com.hp.openview.bbc.FxServer]
SERVER_PORT = 6500

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Local Location Broker (LLB): used by all components
;
; NOTE: The LLB 'SERVER_PORT' value must
;       be the same on all nodes.
;
[com.hp.openview.bbc.LLBServer]
SERVER_PORT = 383
;SERVER_BIND_ADDR =

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; The following settings are the default settings for all HP
; components that use the HP OpenView HTTP-based communication
;
; OVO embedded performance component
;
[com.hp.openview.Coda]
SERVER_PORT = 381
;SERVER_BIND_ADDR =
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; used by various clients of the embedded performance component:
; OVOW embedded reporter and grapher, OV Reporter, OV Performance
; Manager
;
[com.hp.openview.CodaClient]
;CLIENT_PORT =
;CLIENT_BIND_ADDR =

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; used by service discovery agent and server component
;
[com.hp.openview.OvDiscoveryCore.OvDiscoveryInstanceXML]
SERVER_PORT = 6602
;SERVER_BIND_ADDR =
;CLIENT_PORT =
;CLIENT_BIND_ADDR =


[DEFAULT]

; It's important to note that the following statement includes
; parameters defined in the nodeinfo file. Those parameters will
; take precedence over any parameters defined up to this point.
INCLUDE = nodeinfo

; EOF
```

default.txt Configuration Parameters

The `default.txt` configuration parameters are listed and described in the following sections.

PROXY

Usage: HTTP client (Reporter and/or OVPM)

Description: Sets the proxy for any OpenView HTTP clients running on the system. Clients can be Reporter or OV Performance Manager. The variables are comma-separated lists of hostnames, networks, and IP addresses that apply to the proxy. Multiple proxies may be defined for one PROXY key. A dash (-) before the list indicates that those entities do not use this proxy, a plus (+) before the list indicates that those entities do use this proxy. The first matching proxy is used.

Values: `proxy:port +(a) – (b) ; proxy2:port2 + (c) – (d) ; …`

Default: not set

Examples:

```
PROXY web-proxy.hp.com:8088
```

Meaning: the proxy `web-proxy` will be used with port 8088 for every server.

```
PROXY web-proxy.hp.com:8088 –(localhost, *.veg.com)
+(*.lettuce.veg.com)
```

Meaning: the proxy `web-proxy` will be used with port 8088 for every server except the local host and hosts that match `*.veg.com`, for example, `www.veg.com`. The exception is hostnames that match `*.lettuce.veg.com`, for example, for `romain.lettuce.veg.com` the proxy server will be used.

CLIENT_BIND_ADDR(app_name)

Usage: HTTP client (Reporter and/or OVPM)

Description: Sets the IP address for the specified application's OpenView HTTP client. Valid application name is `com.hp.openview.CodaClient`.

Values: <*IP_address*>

Default: not set

Example:

```
[com.hp.openview.CodaClient]
CLIENT_BIND_ADDR = 10.10.10.10
```

CLIENT_PORT(app_name)

Usage: HTTP client (Reporter and/or OVPM)

Description: Sets the port number or a range of ports for the specified application's OpenView HTTP client. Valid application name is `com.hp.openview.CodaClient`.

Values: <*port_range*>

Default: not set

Example:

```
[com.hp.openview.CodaClient]
CLIENT_PORT = 14000-14003
```

SERVER_BIND_ADDR(app_name)

Usage: HTTP server (OVPA)

Description: Sets the IP address for the specified application's OpenView HTTP server. Valid application name is `com.hp.openview.Coda`.

Values: <IP_address>

Default: not set

Example:

```
[com.hp.openview.Coda]
SERVER_BIND_ADDR = 10.10.10.10
```

SERVER_PORT(app_name)

Usage: HTTP server (OVPA)

Description: Sets the port number or a range of ports for the specified application's OpenView HTTP server. Valid application name is `com.hp.openview.Coda`.

Values: <*port_number*>

Default:

```
[com.hp.openview.bbc.LLBServer]
SERVER_PORT = 381
```

```
[com.hp.openview.bbc.LLBServer]
SERVER_PORT = 383
```

Example:

```
[com.hp.openview.Coda]
SERVER_PORT = 1100
```

A Appendix

Viewing MPE Log Files

MPE log file data collected by the scopeXL collector can be viewed with OV Performance Manager. Before viewing the data, you must first extract it and then load the log files as a local data source on your OV Performance Manager system.

To view your MPE log file data using OV Performance Manager, follow these steps:

1. Login to your HP 3000 system as MANAGER.SYS,SCOPE.
2. Run the Performance Collection Software (for MPE Systems) extract program, EXTRACT.SCOPE.SYS.
3. Extract the scopeXL log file data that you want to view. (For more information about extracting log file data, see the *HP Performance Collection Software User's Manual (for MPE Systems)* or online Help for the extract program.)
4. Using binary mode, ftp the extracted log file to a system where OV Performance Agent and OV Performance Manager are running.
5. Login to your OV Performance Manager system (if you have not already done so). Make sure that you have the system name and path to the file that you just downloaded from your MPE system. You cannot access the data through NFS.
6. Run OV Performance Manager.
7. Add the extracted MPE log file data as a local data source. (For more information, see “Add a Local Data Source” in OV Performance Manager's online Help.)
8. View the data.

Viewing and Printing Documents

OV Performance Agent software includes the standard OV Performance Agent documentation set in viewable and printable file formats. You can view the Adobe Acrobat format (*.pdf) documents online and print as needed. The ASCII text (*.txt) documents are printable. However, you can view a text file on your screen using any UNIX text editor such as vi.

The documents are listed in the following table along with their file names and online locations.

Table 8-2 OV Performance Agent Documentation Set

Document	Filename	Location
HP OpenView Performance Agent Installation & Configuration Guide	install.pdf	/opt/perf/paperdocs/ovpa/C/
HP OpenView Performance Agent for UNIX User's Manual	users.pdf	/opt/perf/paperdocs/ovpa/C/
HP OpenView Performance Agent for UNIX Data Source Integration Guide	dsi.pdf	/opt/perf/paperdocs/ovpa/C/
HP OpenView Performance Agent for UNIX & GlancePlus Tracking Your Transactions	tyt.pdf	/opt/perf/paperdocs/arm/C/
Application Response Measurement (ARM) API Guide	armapi.pdf	/opt/perf/paperdocs/arm/C/
OV Performance Agent Metrics Definitions for a UNIX platform	met<*platform*>.txt met<*platform*>.htm	/opt/perf/paperdocs/ovpa/C/

Table 8-2 OV Performance Agent Documentation Set (Continued)

Document	Filename	Location
OVPA metrics list by Data Class for all operating systems	`mettable.txt`	`/opt/perf/paperdocs/ovpa/C/`

Viewing Documents on the Web

The listed `.pdf` and `.htm` documents can be viewed on the HP OpenView Manuals web site at:

http://ovweb.external.hp.com/lpe/doc_serv

Select `performance agent for` *<platform>* from the `product` list box, select the release version and click **[Search]**.

Adobe Acrobat Files

The Adobe Acrobat files were created with Acrobat 4.0 and are viewed with the Adobe Acrobat Reader versions 4.0 and higher. If the Acrobat Reader is not in your Web browser, you can download it from Adobe's web site:

http://www.adobe.com

While viewing a document in the Acrobat Reader, you can print a single page, a group of pages, or the entire document.

From Linux, you can read a `.pdf` file by entering:

```
acroread filename.pdf
```

Glossary

alarm An indication of a period of time in which performance meets or exceeds user-specified alarm criteria. Alarm information can be sent to an OV Performance Manager analysis system and to HP OpenView Network Node Manager and OV Operations (OVO). Alarms can also be identified in historical log file data.

alarm generator The service that handles the communication of alarm notification. It consists of `perfalarm` (alarm generator server), `agdbserver` (alarm generator database server) and `agdb` (alarm generator database) that is managed by `agdbserver`. The `agsysdb` program uses a command line interface for displaying and changing the actions taken by alarm events.

alarmdef file An OV Performance Agent text file containing the alarm definitions in which alarm conditions are specified.

application A user-defined group of related processes or program files. Applications are defined so that performance software can collect performance metrics for and report on the combined activities of the processes and programs.

application log file *See logappl.*

coda daemon A daemon that provides collected data to the alarm generator and analysis product data sources including `scopeux` log files or DSI log files. `coda` reads the data from the `datasources` configuration file.

data source A data source consists of one or more classes of data in a single `scopeux` or DSI log file set. For example, the default OV Performance Agent data source, is a `scopeux` log file set consisting of global data. *See also datasources file.*

datasources file A configuration file residing in the `var/opt/OV/conf/perf/` directory. Each entry in the file represents a `scopeux` or DSI data source consisting of a single log file set. *See also coda and data source.*

data source integration (DSI) The technology that enables OV Performance Agent to receive, log, and detect alarms on data from external sources such as applications, databases, networks, and other operating systems.

data type A particular category of data collected by a data collection process. Single-instance data types, such as global, contain a single set of metrics that appear only once in any data source. Multiple-instance data types, such as application, disk, and transaction, may have many occurrences in a single data source, with the same set of metrics collected for each occurrence of the data type.

default.txt A communications configuration file used to customize communication parameters for HP OpenView applications.

device A device is an input and/or output device connected to a system. Common devices include disk drives, tape drives, printers, and user terminals.

device log file *See logdev.*

DSI *See data source integration.*

dsilog The OV Performance Agent process that logs self-describing data received from `stdin`.

DSI log files Log files, created by the `dsilog` process, that contain self-describing data collected outside of OV Performance Agent. *See also dsilog.*

empty space The difference between the maximum size of a log file and its current size.

extract An OV Performance Agent program that allows you to extract (copy) data from raw or previously extracted log files, summarize it, and write it to extracted log files. It also lets you export data for use by analysis programs.

extracted log file An OV Performance Agent log file containing a user-defined subset of data extracted (copied) from a raw or previously extracted log file. Extracted log files are also used for archiving performance data. *See also rxlog.*

global A qualifier that implies the whole system. Thus, "global metrics" are metrics that describe the activities and states of each system. Similarly, application metrics describe application activity; process metrics describe process activity.

global log file *See logglob.*

interesting process A process becomes interesting when it is first created, when it ends, and when it exceeds user-defined thresholds for CPU use, disk use, response time, and other resources.

logappl The raw log file that contains summary measurements of the processes in each user-defined application.

logdev The raw log file that contains measurements of individual device (such as disk) performance.

logglob The raw log file that contains measurements of the system-wide, or global, workload.

logindx The raw log file that contains additional information required for accessing data in the other log files.

logproc The raw log file that contains measurements of selected interesting processes. *See also interesting process.*

logtran The raw log file that contains measurements of transaction data.

ovpa script The OV Performance Agent script that has options for starting, stopping and restarting OV Performance Agent processes such as data collection, alarms, and repository servers. *See also the ovpa man page.*

OV Performance Manager OV Performance Manager provides integrated performance management for multi-vendor distributed networks. It uses a single

workstation to monitor environment performance on networks that range in size from tens to thousands of nodes.

parm file An OV Performance Agent file that contains the collection parameters used by scopeux to customize data collection.

performance alarms *See alarms*

process Execution of a program file. It can represent an interactive user (processes running at normal, nice, or real-time priorities) or an operating system process.

process log file *See logproc.*

process resource manager (PRM)

Stand-alone resource management tool developed by Hewlett-Packard that is used to control the amount of resources that processes use during a peak system load. PRM can guarantee both a minimum and, depending on the resource, a maximum amount of resources available to a group of processes.

PRM *See process resource manager.*

raw log file A file containing summarized measurements of system data. The scopeux data collector collects and logs data into raw log files. *See also logglob, logappl, logproc, logdev, logtran, and logindx.*

real time The actual time in which an event takes place.

repeat time An action that can be specified for performance alarms. Repeat time designates the amount of time that must pass before an activated and continuing alarm condition triggers another alarm signal.

resize Changing the overall size of a raw log file using the utility program's resize command.

roll back Deleting one or more days worth of data from a log file, oldest data deleted first. Roll backs are performed when a raw log file exceeds its maximum size parameter.

RUN file The file created by the scopeux collector to indicate that the collection process is running. Removing the RUN file causes scopeux to terminate.

rxlog The default output file created when data is extracted from raw log files.

scopeux The OV Performance Agent collector program that collects performance data and writes (logs) this raw measurement data to raw log files for later analysis or archiving.

scopeux log files The six log files that are created by the scopeux collector: logglob, logappl, logproc, logdev, logtran, and logindx.

status.scope The file created by the scopeux collector to record status, data inconsistencies, or errors.

transaction tracking The OV Performance Agent capability that allows information technology (IT) resource managers to measure end-to-end response time of

business application transactions. To collect transaction data, OV Performance Agent must have a process running that is instrumented with the Application Response Measurement (ARM) API.

utility An OV Performance Agent program that lets you open, scan, and generate reports on raw and extracted log files. You can also use it to resize raw log files, check `parm` file syntax, check the `alarmdef` file syntax, and obtain alarm information from historical log file data.

Index

A

accessing help
  extract program, 171
  utility program, 87
agdb, 210
agdb database, 210
agdbserver, 210
agsysdb, 210
alarm conditions in historical log file data, 81, 212
alarm definitions, 208
  application metrics, 218
  components, 214
  customizing, 246
  examples, 243
  file, 81, 208
  metric names, 218
  syntax checking, 83
alarm generator, 208, 210
alarm processing errors, 212
ALARM statement, alarm syntax, 221
alarm syntax, 215
  ALARM statement, 221
  ALERT statement, 226
  ALIAS statement, 240
  comments, 216
  common elements, 216
  compound statements, 217
  conditions, 217, 222, 231
  constants, 218
  conventions, 216
  EXEC statement, 228
  expressions, 218
  IF statement, 231
  INCLUDE statement, 235
  LOOP statement, 233
  messages, 220
  metric names, 218
  PRINT statement, 230
  reference, 215
  SYMPTOM statement, 241
  USE statement, 236
  VAR statement, 239
  variables, 239
alarmdef file, 81, 83, 208, 209, 236, 246
alarms, 208
  local actions, 211
  processing, 209
  sending messages to OVO, 210
alert notifications, 208
ALERT statement, alarm syntax, 226
ALIAS statement, alarm syntax, 240
analyze command, utility program, 81, 212
analyzing
  historical log file data, 81, 212
  log files, 81, 212
application command, extract program, 153
application definition parameters, parm file, 42
application LOOP statement, alarm syntax, 233
application metrics, in alarm definitions, 218
application name parameter, parm file, 43
application name record, 141
Application Response Measurement (ARM), 39
archiving log file data, 54, 179, 200, 203
archiving processes, managing, 54
ASCII format, export file, 128
ASCII record format, 134

B

binary format, export file, 129
binary header record layout, 136
binary record format, 135

C

checkdef command, utility program, 83
class command, extract program, 155
client, firewall, 248
CLIENT_BIND_ADDR(app_name) parameter, 258
CLIENT_PORT(app_name) parameter, 258
coda daemon, 26, 209
collection parameters, 26, 36
command abbreviations
  extract, 146
  utility, 78
command line arguments
  extract program, 117
  utility program, 63
command line interface
  extract program, 114, 117
  utility program, 59, 63

commands
  extract program, 146
  perfstat, 33
  utility program, 78
comments, using in alarm syntax, 216
compound actions in ALARM statement, 224
compound statements in alarm syntax, 217
conditions
  alarm syntax, 217, 231
  in alarm syntax, 222
configuration command, extract program, 157
constants, in alarm syntax, 218
controlling disk space used by log files, 51
conventions
  typographical, 15
conventions, alarm syntax, 216
cpu command, extract program, 158
cpu option, 40
creating custom graphs or reports, 132
customized export template files, 126

D

daemons
  coda, 209
data collection, 25, 32
  management, 32, 51
  stopping, 49
data source integration (DSI), 23, 26
data sources, 26, 209, 236
data type parameter, export template file, 131
data types, 124
datafile format, export file, 129
datafile record format, 134
datasources configuration file, 27
datasources file, 27, 209
default values,parm file, 34
default.txt configuration file
  location, 251
  parameters, 257
  printout, 254
detail command, utility program, 84
disk command, extract program, 159
disk device name record, 142
disk option, 40
disk space used by log files, controlling, 51
documentation, viewing and printing, 263
DSI log files, 26, 151, 164, 170

E

errors, alarm processing, 212
EXEC statement, alarm syntax, 228
executing local actions, 211
exit command, extract program, 160
exit command, utility program, 85
export command, extract program, 124, 161
export data types, 124
export default output files, 162
export file
  title, 132
export function
  data files, 126
  export template file syntax, 128
  export template files, 125
  overview, 124
  process, 124
  sample tasks, 125
  using, 132
export template file
  data type, 131
  export file title, 132
  format, 128
  headings, 129
  items, 131
  layout, 130
  missing, 130
  output, 130
  parameters, 128
  report, 128
  separator, 129
  summary, 130
  syntax, 128
exporting DSI log file data, 170
exporting log file data, 161
expressions, in alarm syntax, 218
extract command, extract program, 164
extract commands
  application, 153
  class, 155
  configuration, 157
  cpu, 158
  disk, 159
  exit, 160
  export, 124, 161

extract, 164
filesystem, 167
global, 168
guide, 170
help, 171
list, 172
lvolume, 176
menu, 177
monthly, 179
output, 182
process, 185
quit, 187
report, 188
sh, 189
shift, 190
show, 192
start, 194
stop, 196
weekdays, 199
weekly, 200
yearly, 203
extract program, 26, 112
command line arguments, 117
command line interface, 117
commands, 146
interactive versus batch, 114
running, 114
extracting log file data, 164

F

file parameter, parm file, 44
files
alarm definitions, 81, 208
alarmdef, 81, 83, 208, 209, 246
datasources, 27, 209
default.txt, 251
default.txt configuration file, 251, 254, 257
export template, 125
logappl, 33, 38
logdev, 33, 38, 39
logglob, 33, 37
logindx, 33
logproc, 33, 38
logtran, 33, 39
parm, 26, 36
reptall, 125, 126
reptfile, 125, 188
repthist, 126
status.scope, 33
filesystem command, extract program, 167
firewall
client, 248
communicatiing across, 248
communicating across, 248
configure OVPA server port, 252
configure Reporter/OVPM with HTTP proxy, 252
configure Reporter/OVPM without HTTP proxy, 253
default ports for communication with OVPA, 250
default.txt file printout, 254
default.txt location, 251
overview of OVPA communications configuration, 250
server, 248
systems with multiple IP addresses, 254
format parameter
export template file, 128

G

GlancePlus, 28
global command, extract program, 168
group parameter, parm file, 45
guide command, extract program, 170
guide command, utility program, 86
guided mode
extract, 170
utility, 86

H

headings parameter, export template file, 129
help command, extract command, 171
help command, utility program, 87
HTTP
client, 249
clients, 249
proxies, 248

I

ID parameter
parm file, 37
IF statement, alarm syntax, 231
INCLUDE statement, alarm syntax, 235
interactive mode
extract program, 116
utility program, 61
interesting processes, 38, 52

items parameter, export template file, 131

L

layout parameter, export template file, 130
list command, extract program, 172
list command, utility program, 88
local actions
  alarms, 228
  executing, 211
log file data
  analyzing for alarm conditions, 212
  archiving, 179, 200, 203
  exporting, 161
  extracting, 164
log files
  archiving data, 54
  controlling disk space, 51
  DSI, 26, 151, 164
  MPE, 262
  resizing, 95
  rolling back, 51, 53
  scanning, 101
  scopeux, 151
  setting maximum size, 41, 52
log parameter, parm file, 37
logappl file, 33, 38
  PRM groups, 38
logdev file, 33, 38, 39
logfile command, utility program, 90
logglob file, 33, 37, 203
logical volume name record, 142
logindx file, 33
logproc file, 33, 38
logtran file, 33, 39
LOOP statement, alarm syntax, 233
lvolume command, extract program, 176

M

maintenance time, parm file, 42
mainttime parameter, parm file, 42, 51
managing data collection, 32
memory option, 40
menu command
  extract program, 177
  utility program, 92
messages in alarm syntax, 220
metric names in alarm syntax, 218, 240
missing parameter, export template file, 130
modifying
  collection parameters, 34
  parm file, 34
monthly command, extract program, 179
MPE log files, viewing, 262

N

netif name record, 143
nokilled option, 40

O

OpenView Operations (OVO), 208, 210
or parameter, parm file, 46
output command, extract program, 182
output parameter, export template file, 130
OV Network Node Manager, 208, 210
OV Operations, 28
OV Performance Agent
  components, 24
  data collection, 25
  description, 23
  extract program, 26, 112
  utility program, 26, 58
OV Performance Manager, 23, 28
OV Reporter, 28
ovpa script, 49, 50
OVPM
  configure with HTTP proxy for firewall communication, 252
  configure without HTTP proxy for firewall communication, 253

P

parameter
  subprocinterval, 41
parameters, 36
  CLIENT_BIND_ADDR(app_name), 258
  CLIENT_PORT(app_name), 258
  PROXY, 257
  SERVER_BIND_ADDR(app_name), 259
  SERVER_PORT(app_name), 259
parm file, 26, 36
  application definition parameters, 42
  default values, 34
  modifying, 34

parameters, 36, 37
subprocinterval parameter, 41
syntax check, 93
parm file parameters
application name, 43
file, 44
group, 45
ID, 37
log, 37
mainttime, 42, 51
or, 46
priority, 47
scopeprocinterval, 41
scopetransactions, 41
size, 41
threshold, 39
user, 45
parmfile command, utility program, 93
perfalarm, 210, 236, 237
performance alarms, 208
perfstat command, 33
PRINT statement, alarm syntax, 230
printing documentation, 263
priority parameter, parm file, 47
PRM application logging mode, 42
PRM groups
APP_NAME_PRM_GROUPNAME, 38
process command, extract program, 185
processing alarms, 209
proxies, HTTP, 248
PROXY parameter, 257

Q

quit command
extract program, 187
utility program, 94

R

raw log files
managing space, 95
names, 90
record formats
ASCII, 134
binary, 135
datafile, 134
report command, extract program, 188
report parameter, export template file, 128
Reporter
configure with HTTP proxy for firewall communication, 252, 253
reptall file, 125, 126
reptfile file, 125, 188
repthist file, 126
resize command
default resizing parameters, 97
reports, 98
utility program, 61, 95
resizing
log files, 95
tasks, 53
rolling back log files, 53
running
extract program, 114
utility program, 59

S

scan command, utility program, 101
scanning a log file, 101
SCOPE default data source, 27, 218, 236, 237
scopeprocinterval parameter, parm file, 41
scopetransactions parameter, parm file, 41
scopeux, 25, 33
log files, 26, 151
stopping, 49
sending alarm messages, 210, 226
sending SNMP traps, 208, 210
separator parameter, export template file, 129
server, firewall, 248
SERVER_BIND_ADDR(app_name) parameter, 259
SERVER_PORT(app_name) parameter, 259
setting maximum size of log files, 52
sh command
extract program, 189
utility program, 103
shift command, extract program, 190
shortlived option, 41
show command
extract program, 192
utility program, 104
size parameter, parm file, 41
SNMP
nodes, 210
service, 209
traps, 208, 210
start command
extract program, 194
parameters, 106
utility program, 106
status.scope file, 33

stop command
  extract program, 196
  parameters, 108
  utility program, 108
stopping
  data collection, 49
  scopeux, 49
summary parameter, export template file, 130
SYMPTOM statement, alarm syntax, 241

T

terminating
  extract program, 160, 187
  utility command, 94
  utility program, 85
threshold parameter, parm file, 39
  cpu option, 40
  disk option, 40
  memory option, 40
  nokilled option, 40
  nonew option, 40
  shortlived option, 41
transaction name record, 142
transaction tracking, 27

U

USE statement, alarm syntax, 236
user parameter, parm file, 45
utility commands
  analyze, 81, 212
  checkdef, 83
  detail, 84
  exit, 85
  guide, 86
  help, 87
  list, 88
  logfile, 90
  menu, 92
  parmfile, 93
  quit, 94
  resize, 61, 95
  scan, 101
  sh, 103
  show, 104
  start, 106
  stop, 108
utility program, 26, 58, 78, 212
  batch mode, 59
  batch mode example, 61
  command line arguments, 63
  command line interface, 59, 63
  entering shell commands, 103
  interactive mode, 61
  interactive program example, 61
  interactive versus batch, 59
  running, 59
utility scan report
  application overall summary, 73
  application-specific summary report, 70
  collector coverage summary, 73
  initial parm file application definitions, 68
  initial parm file global information, 68
  log file contents summary, 74
  log file empty space summary, 75
  parm file application addition/deletion notifications, 69
  parm file global change notifications, 69
  process log reason summary, 71
  scan start and stop, 72
  scopeux off-time notifications, 70

V

VAR statement, alarm syntax, 239
variables, alarm syntax, 239
viewing
  documentation, 263
viewing MPE log files, 262

W

weekdays command, extract program, 199
weekly command, extract program, 200
WK1 format, export file, 129

Y

yearly command, extract program, 203

Index