HP Performance Agent

For the Windows® Operating System

Software Version: 5.00

User's Manual

Document Release Date: August 2009
Software Release Date: August 2009

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

UNIX® is a registered trademark of The Open Group.

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Windows® and Microsoft Windows ® are U.S. registered trademarks of Microsoft Corporation.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

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

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

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

Support

Visit the HP Software Support web site at:

www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

1 Overview of HP Performance Agent

This chapter provides an overview of Performance Agent. This chapter also contains information about the following:

- How Performance Agent Works
- Performance Agent on Windows Operating Systems
- Brief overview of Components such as, the `scopent` collector, the Extended Collection Builder and Manager, the `parm` file and `utility` and `extract` programs, transaction tracking
- Related Performance Products

How Performance Agent Works

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 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 treated the same as data collected by Performance Agent. All DSI data is logged, time stamped, and can be alarmed on. (For details, see the *HP Performance Agent for Windows: Data Source Integration Guide*.)

The data collected or received by Performance Agent can be analyzed using spreadsheet programs, analysis tools such as Performance Manager, and other third-party analysis products.

The comprehensive data logged by Performance Agent enables 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.

Performance Agent gathers comprehensive and continuous information on system activity without imposing significant overhead on the system. The design also provides scope for customization. You can accept default configurations or set parameters to collect data for specific conditions.

Performance Agent on Windows Operating Systems

There are two ways of using Performance Agent on Windows:

- Performance Agent's graphical user interface can perform the most commonly used tasks in a typical Windows environment.
- Performance Agent's `extract` and `utility` programs in the Windows Command Prompt window can perform a broader range of tasks when you use interactive commands or arguments in the Command Prompt.

As you become familiar with Performance Agent, you may find that using both methods for different specific tasks can be the most efficient way to use the product.

- Performance Agent supports HTTP data communication on Microsoft Windows 64-bit platforms.

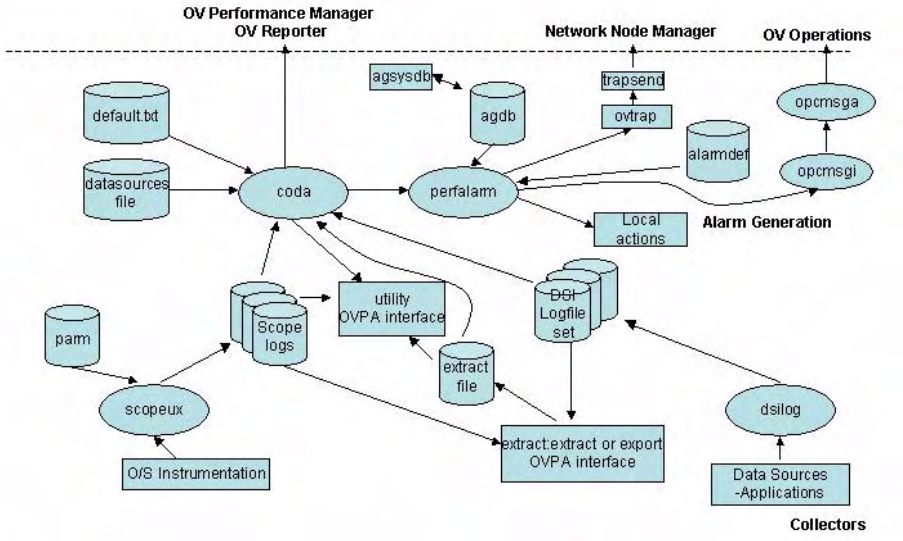
Components

The main components of Performance Agent are the following:

- scopent Data Collector
- Collection Parameters File
- DSI Log Files
- Extract and Utility Programs
- Data Sources
- ARM Transaction Tracking Capabilities

The following diagram illustrate how the Performance Agent components interact.

Figure 1 Component interactions



scopent Data Collector

The `scopent` data collector collects performance data from the operating system on which Performance Agent resides. The data is summarized, and recorded into raw log files, according to the types of information required.

For detailed information about `scopent`, see scopent Data Collector on page 22.

Collection Parameters File

The type of data collected is determined by parameters set in the Performance Agent programs and in the collection parameters (parm) file, an ASCII file used to customize the collection environment. This file contains instructions for the 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 parm File on page 23.

DSI Log Files

DSI log files contain data that is collected outside of Performance Agent. These log files contain data from external sources such as applications and databases. Performance Agent programs create these log files. DSI processes are described in detail in the *HP Performance Agent for Windows: Data Source Integration Guide*. Instructions for exporting DSI log file data can be found in Chapter 3, Using HP Performance Agent and in the online help from the Performance Agent graphical interface.

Extended Collection Builder & Manager

The Performance Agent graphical interface provides the Extended Collection Builder & Manager (ECBM), which enables you to build and manage collections of Windows performance counters. The Extended Collection Builder & Manager log file locations are described in its online help. See Building Collections of Performance Counters on page 100.

Extract and Utility Programs

Two Performance Agent programs, `extract` and `utility`, provide the means for managing both `scopent` log files and DSI log files. You use the `extract` and `utility` programs in the Windows Command Prompt window.

The `extract` program enables you to extract data from raw or previously extracted `scopent` log files, summarize it, and write it to extracted log files. The extracted log files contain selected performance data for specific analysis needs. The `extract` program's `export` function enables you to export `scopent` and DSI data that can be used by spreadsheet programs and other analysis products. For more information about `extract` program, refer to Chapter 6, Extract Program.

The `utility` program enables you to generate reports on raw and extracted `scopent` log files, resize raw `scopent` log files, and check `parm` file syntax. You can also check the syntax in your alarm definitions file and analyze alarm conditions in historical `scopent` and DSI log file data. For more information about `utility` program, see Chapter 4, Utility Program.

Performance Agent's graphical interface provides `extract` and `utility` tasks such as extracting, exporting, and archiving log file data, scanning and resizing log files, and analyzing log file data for alarm activity.

Data Sources

Performance Agent uses a set of repository servers that provide previously collected data to clients such as alarm generator, Performance Manager, and Reporter. There is a repository server for every data source such as `scopent log` files or DSI log files. Each data source consists of a single log file set. The data source is configured in the `datasources` file available in the `<DataDir>\conf\perf` directory. When you start Performance Agent for the first time after installation, a default data source named `SCOPE` is already configured and provides a `scopent` log file set.

To add other data sources, configure them into the datasources file. When you restart Performance Agent, the coda daemon reads the datasources file and is ready to serve any requests in the data source.

For more information about data sources, see Configuring Data Sources on page 87 in Chapter 3.

ARM Transaction Tracking Capabilities

Performance Agent includes transaction tracking capabilities that enable information technology (IT) managers to measure end-to-end response times of business application transactions. To collect transaction data, Performance Agent must have an application running that is instrumented with the Application Response Measurement (ARM) API. For details, see *HP Performance Agent for Windows: Tracking Your Transactions* manual.

Related Performance Products

Performance Agent is one of four related compatible performance products from Hewlett-Packard. These products are required for different purposes within the range of resource and performance management. You can 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:

Performance Manager

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

Performance Manager in this document refers only to versions 4.0 and later. The name Performance Manager 3.x is used throughout this document to refer to the product that was formerly known as PerfView.

GlancePlus

GlancePlus 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. GlancePlus is a UNIX-based product that would be applicable to a multi-platform environment; it is not currently available on Windows platforms.

Reporter

HP Reporter creates web-based reports from data of targeted systems it "discovers." Discovery of a system can occur if the system is running HP Operations agent and subagent software such as Performance Agent. Reporter can also generate reports on systems managed by HP Operations Manager.

After Reporter has run through its discovery, it gathers data based on pre-defined and user-specified lists of metrics, and then formats the collected data into web page reports.

HP Operations Manager

Operations Manager also displays and analyzes alarm information sent by Performance Agent. Operations Manager 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. Operations Manager is a scalable and flexible solution that can be configured to meet the requirements of any IT organization and its users.

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

http://h20230.www2.hp.com/selfsolve/manuals

Select **<product name>** from the product list box, select the release version, select the OS, and select the manual title. Click **[Open]** to view the document online, or click **[Download]** to place the file on your computer.

Related Documentation

Data source integration (DSI), including `dsilog` and other DSI components, is described in the *HP Performance Agent for Windows: Data Source Integration Guide*.

2 Managing Data Collection

This chapter provides instructions to perform the following data collection activities:

- The scopent Data Collector for collecting and recording performance data.
- Managing data collection through the collection parameters file (parm File) and Application Definition Parameters
- Steps to Configure Data Logging Intervals
- Starting and Stopping Data Collection
- Effective Data Collection Management by controlling the amount of disk space used by log files and archiving data

scopent Data Collector

The `scopent` data collector collects and summarizes performance measurements of system-resource utilization and logs the data into the following log files, according to the information specified in the `parm` file.

- The `logglob` file contains measurements of system-wide, or global, resource utilization information. Global data is summarized and recorded periodically at intervals specified in the parm file. For more information, see Configure Data Logging Intervals on page 41.
- The `logappl` file contains aggregate measurements of processes in each user-defined application. Application data is summarized periodically at intervals specified in the `parm` file, and each application that had any activity during the intervals specified in the `parm` file is recorded. For more information, see Configure Data Logging Intervals on page 41.
- The `logproc` file contains measurements of selected “interesting” processes. Process data is summarized periodically at intervals specified in the `parm` file. For more information, see Configure Data Logging Intervals on page 41. However, only interesting processes are recorded.The logproc log file can store data up to maximum size of 4GB.

The concept of interesting processes is a filter that helps minimize the volume of data logged and is controlled from the `parm` file.

- The `logdev` file contains measurements of individual device (such as disk and `netif`) performance. Device data is summarized periodically at intervals specified in the `parm` file, and data from each device that had any activity during that interval is recorded. For more information, see Configure Data Logging Intervals on page 41.
- The `logtran` file contains measurements of transaction data. Transaction data is summarized periodically at intervals specified in the `parm` file, and each transaction that had any activity is recorded. For more information, see Configure Data Logging Intervals on page 41. For more information about transactions, refer to the *HP Performance Agent for Windows: Tracking Your Transactions* manual.
- The `logindx`, contains information needed to access data in the other log files. The `logindx` file does not grow appreciably over time.

- The `logls` file contains information about the logical systems. Data of logical systems is summarized periodically at intervals specified in the `parm` file. For more information see Configure Data Logging Intervals on page 41.

Other Files Created by scopent

In addition to the log files, two other files are created when `scopent` is started. They are the `RUN` file and the `status.scope` file.

The `RUN` file is created to indicate that the `scopent` process is running. Removing this file causes `scopent` to terminate.

The `status.scope` file serves as a status/error log for the `scopent` process. New information is appended to this file each time the `scopent` collector is started. To view the most recent status and error information from `scopent`, use either the **Status** command in the Agent menu on the Performance Agent main window or the `perfstat` command in the Windows Command Prompt.

You may want to view the `status.scope` file on a regular basis to ensure that the collection process is running correctly.

parm File

The collection parameters file, also known as the `parm` file, is a text file containing the instructions for `scopent` to log specific performance measurements. When you start `scopent`, it looks for the `parm` file in the `<datadir>\` directory, or in a user-defined `\data\` directory. If the `parm` file is not available or if a specific parameter is not included in the `parm` file, `scopent` uses the default options.

Once you are familiar with Performance Agent, you can customize the `parm` file to your performance data collection needs. For more information about `parm` file parameters, see parm File Parameters on page 25. For instructions on modifying the `parm` file, see Modifying the parm File on page 24. In Performance Agent running on Windows, the `parm` file's file name is `parm.mwc`.

A sample `parm` file is provided with Performance Agent in `<datadir>\newconfig\` and is copied into the `<datadir>\data\` directory during installation if `parm` file is not already present from a previous installation. `scopent` reads the `parm` file when it starts.

If you have not run the product before, you can use the sample `parm` file to become familiar with the type of data collected. Once you are familiar with the Performance Agent environment, you should tailor the `parm` file to your performance data collection needs.

The sample `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 Performance Agent for Windows: Installation and Configuration Guide.* Also see the description of the `size` parameter in the Parameters Descriptions on page 29 .

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

Modifying the parm File

You can modify the `parm` file using any word processor or editor that can save a file in ASCII text format. We recommend that you use the Collection Parameters command from the Configure menu in the Performance Agent main window to modify your `parm` file. (For more information, see Configuring Collection Parameters on page 81.)

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

- Any parameter you specify overrides a default.
- 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`, `or`, and `priority` parameters must follow the `application` statement that they define.
  - Application parameters should be listed in order of priority.
- 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 restart `scopent` in order for the changes to take effect. To activate the changes, follow these steps:

1. Choose **Start/Stop** from the Agent menu to open the MeasureWare Services window.
2. Select the **Data Collection** check box.
3. Click **Refresh**.
4. Click **Close** to return to Performance Agent main window.

parm File Parameters

The scopent data collector is controlled by specific parameters in the `parm` file.

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

You can modify these parameters to tell `scopent` to log measurements that match the requirements of your system. The following table lists `parm` file parameters are used by `scopent` and the default values for each parameter. Detailed descriptions of these parameters are in Parameters Descriptions on page 29.

Table 1 parm File Parameters Used by scopent

Parameter	Values or Options	Default Values
`id`	*computer name* (from the network applet in the WindowsNT Control Panel or from the System applet on Windows 2000 and Windows XP)	*computer name* (from the Network applet in the Windows NT Control Panel or the System applet on Windows 2000, Windows 2003 and Windows XP)

`log`	`global`
	`application`
	`process`
	`device=disk, cpu, filesystem`
	`transaction`
	`logicalsystem`
	(On Windows Server 2008 where Hyper-V role is enabled, logical system is supported for enabling BYLS class)
		`global`, `process`

`procthreshold`
(same as `threshold`)
	`cpu=`*percent*
	`memory`
	`nonew`
	`nokilled`
		`cpu=10.0`
		`memory=500`
		New processes and and all killed processes are not logged.

Table 1 parm File Parameters Used by scopent

Parameter	Values or Options	Default Values
`appthreshold`	`cpu=`*percent*	`cpu=0` If the value is not set, all applications with cpu utilization greater than 0% are logged for each interval.
`diskthreshold`	`util=`*rate*	util=0 If the values are not specified, the disks that are not idle are logged.
`bynetifthreshold`	`iorate=`*rate*	iorate=0 If the value is not specified, the network interfaces that are not idle are logged for each interval.
`fsthreshold`	`util=`*rate*	util=0 If the value is not specified, all the filesystems are logged.
`bycputhrershold`	`cpu=`*percent*	cpu=0 If the value is not specified, cpu's that are not idle will be logged.
`Application`	*application name*	Applicable only when applications are specified.
`File`	*filename* [, ...]	Applicable only when applications are specified.

Table 1 parm File Parameters Used by scopent

Parameter	Values or Options	Default Values
`Or`		Applicable only when applications are specified.
`Priority`	low value-high value (range is 0 to 31)	Applicable only when applications are specified.

`size`	`global=`*nn* (values are in megabytes)
	`application=`*nn*
	`process=`*nn(the maximum value is 4096)*
	`device=`*nn*
	`transaction=`*nn*
	`logicalsystem=`*nn*
		`global=10`
		`application=10`
		`process=20`
		`device=10`
		`transaction=10`
		`logicalsystem=10`

`days`	`global=`*nn* (values are in days) `application=`*nn* `process=`*nn* `device=`*nn* `transaction=`*nn* `logicalsystem=`*nn*	
`maintweekday`	`Sun|Mon|Tue|Wed|Thu|Fri|Sat`	Sun Applicable only when `maintweekday` is not specified.
`scopetransactions`	on off	on
`Mainttime`	*hh:mm* (24 hour time)	23:30
`user`	*user login name [,]*	

Table 1 parm File Parameters Used by scopent

Parameter	Values or Options	Default Values
collectioninterval	process=nn (values in seconds) global=nn	process=60 global=300
gapapp	blank UnassignedProcesses ExistingApplicationName other	
flush	ss(values in seconds) 0 (disables data flush)	3600

Parameters Descriptions

Following are descriptions of each of the `parm` file parameters.

- ID
- Log
- Procthreshold
- appthreshold
- diskthreshold
- bynetifthreshold
- fsthreshold
- bycputhreshold
- Scopetransactions
- Size
- Mainttime
- Days
- Maintweekday
- Flush

- gapapp

ID

The system ID value is a string of characters that identifies your system. If you have multiple systems, use different ID strings on each one. 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.

The default system ID is your system's computer name as specified in the Network applet on the Windows NT Control Panel or the System applet on Windows 2000 and Windows XP.

Log

The `log` parameter specifies the data types to be collected by `scopent`.

- Specify **`log global`** to record global records to the `logglob` file.

You must log global data in all cases. Global data records are required in order to view and analyze performance data on your system.

- Specify **`log application`** to record active application data to the `logappl` log file.
- Specify **`log process`** to write information about interesting processes to the `logproc` log file.
- Specify log **`dev=disk,cpu,filesystem`** to record information about individual disks, CPUs, and filesystems to the `logdev` log file. Enabling the options may result in higher utilization of the `logdev` file.
- Specify **`log transaction`** to record transaction records to the `logtran` log file. For `scopent` 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 Performance Agent for Windows: Tracking Your Transactions* manual.)

  The default for the `log transaction` parameter is `no correlator`. To turn on resource data collection or correlator data collection, specify **`log transaction=correlator`**.
- **`log logicalsystem`** enables scopent to record information about the logical systems to the `logls` file. Data for logical system is summarized periodically at intervals specified in the parm file.

All of the log files are created automatically if logging to them is specified and they do not already exist. 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.

Procthreshold

The `procthreshold` parameter is used to set the threshold values for interesting processes. The `procthreshold` parameter is used to set activity levels to specify criteria for interesting processes. To use this parameter, you must enable process logging. `procthreshold` affects only processes that are logged and does not affect any other class of data.

The `procthreshold` parameter is same as the `threshold` parameter. The `threshold` parameter is also supported in Performance Agent version 4.60. If both the parameters are enabled, the latest value specified overrides the previous value.

You can enter the options for thresholds on the same parameter line (separated by commas) or on separate lines. You must include the word `threshold` on each line.

Threshold Options for Process Data

`CPU Option`

This option sets the percentage of CPU utilization that a process must exceed to become interesting and be logged. It is used only if process logging is enabled.

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. The default is cpu=10.0.

`Nonew Option`	This option turns off logging of new processes if they did not exceeded any threshold; for example, threshold nonew. The default is all new processes are logged.
`Nokilled Option`	This option turns off logging of exited processes if they did not exceed any threshold; for example, threshold nokilled. The default is all killed (exited) processes are logged.
`Memory Option`	The virtual memory in MB that a process must use to become interesting during an interval. Every process that exceeds the memory threshold is logged. The default for this threshold is 500 MB.

appthreshold

The `appthreshold` parameter is used to specify threshold values for the APPLICATION data class (the `APP_CPU_TOTAL_UTIL` metric). The threshold criteria is based on the percentage of CPU utilization that an application must exceed for the application to be recorded in the log files. The default threshold values records applications that use more than 0% of CPU.

diskthreshold

The `diskthreshold` parameter is used to specify the threshold values for `DISK` class. The threshold criteria for `DISK` class is based on the percentage of time duration a disk performs I/Os (`BYDSK_UTIL` metric). The default values record the disks that are busy performing I/Os for more than 10% of the time duration.

bynetifthreshold

The `bynetifthreshold` specifies the thresholds for `NETIF` class. `Netif` data class threshold criteria is based on the number of packets transferred by the network interface per second (`BYNETIF_PACKET_RATE` metric). The default values records network interfaces that transfer more than 60 packets per second.

fsthreshold

The `fsthreshold` specifies the thresholds for `FILESYSTEM` class. The file system data class threshold criteria is based on the percentage of disk space used by the filesystems (`FS_SPACE_UTIL` metric). The default values records filesystems that use more than 70% of disk space.

bycputhreshold

The `bycputhreshold` specifies the thresholds for CPU class. CPU data class thresholds is based on the percentage of time the cpu was busy (`BYCPU_CPU_TOTAL_UTIL`). The default value records CPUs that are busy more than 90% of time.

Scopetransactions

The `scopent` collector is instrumented with ARM (Application Response Measurement) API calls that generate transactions. The `scopetransactions` flag determines whether the `scopent` transactions are logged or not.

The default is `scopetransactions=on`. `scopent` logs two transactions, `Scope_Get_Process_Metrics` and `Scope_Get_Global_Metrics`.

If you do not want these `scopent` transactions to be logged, specify **`scopetransactions=off`**.

A third transaction, `Scope_Log_Headers`, is always logged. It is not affected by `scopetransactions=off`.

For additional information, see Chapter 7 of the *HP Performance Agent for Windows: Tracking Your Transactions* manual.

Size

The `size` parameter is used to set the maximum size (in MB) of any raw log file. By default, `logglob` (the global log file), `logappl` (the application log file), and `logdev` (the device log file), and `logtran`, (the transaction log file) can grow in size to 10 MB each, while `logproc` (the process log file) has a size of 20 MB. You cannot set the size to be less than 1 MB.

The `scopent` collector reads these specifications when it is started up. If any of these log files achieve their maximum size during collection, they continue to grow until `mainttime`, when they are 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 hold not more than one year's worth (365 days) of data (When there are several weeks' worth of data in a raw log file, you should use the `utility` program's `scan` function to set your file size to hold less than a year's worth of data. See Log File Contents Summary on page 123 and Log File Empty Space Summary on page 125.)

If the size specification in the `parm` file is changed, `scopent` detects it during startup. If the maximum log file size is decreased to the point where existing data does not fit, an automatic resize takes place during the `scopent` startup. 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.

Mainttime

Log files are rolled back by `scopent` only at a specific time each day. The default time is 11:30 pm or 23:30, but it 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. The `scopent` checks the available disk space on the file system where the log files reside periodically at the logging interval specified in the `parm` file. If the available disk space is less than 1 MB, `scopent` immediately performs the log file maintenance without waiting for the scheduled 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 are rolled back. If there is still less than 1 MB of free space, `scopent` terminates with an appropriate message in the `status.scope` file.

Days

The `days` parameter specifies the maximum number of days of data, any raw data log file can store at a given point of time. The value for this parameter must be in the range of 1 to 365. This parameter enables `scopent` data collector to maintain log files.

During data collection, if the number of days of data in the log file reaches the days specified in the `days` parameter, data collection continues until the day specified in the `maintweekday` parameter is met. Once `maintweekday` is

reached, the log file is rolled back automatically at maintime. During the roll back, data collected after the days parameter reaches its maximum value are removed from the log file.

If the value specified in the size parameter is met first before the days parameter, the roll back will happen based on the size parameter and it overrides the days parameter.

For example, if "size global=20" and "days global=40" is used in the parm file, and if the log files reaches a maximum size 20 MB before 40 days of data are logged, the log file roll back is done based on the size parameter.

Maintweekday

The maintweekday parameter specifies the day of the week when the log file roll back happens if the days parameter is met. The roll back will happen at maintime.

For example, if "maintweekday=Mon" is used in the parm file, the log file roll back is done once the value specified in the days parameter is met on "Monday" at maintime. It is recommended that the value for maintweekday be set to a day in the week when the system utilization is low.

The maintweekday parameter is an optional parameter. If maintweekday parameter is specified in the parm file, it should be used along with the days parameter. This parameter is not considered if it is not used with the days parameter in the parm file. If maintweekday is not specified in the parm file when the days parameter is specified, the default value is "maintweekday=Sun".

For example, if "days global=30", "application=20", "process=30", "device=20", "transaction=10", "maintweekday=Wed" and if the log file reaches the number of days specified in the days parameter, log file collection continues until the day specified in the maintweekday. Once maintweekday is reached, log file roll back takes place removing the exceeded number of days of data from the start of the log file. This maintenance is done at maintime.

Flush

The `flush` parameter specifies the data collection intervals (in seconds) at which all instances of application and device data are logged. The flush intervals must be in the range 300-32700 and be an even multiple of 300. The default value of `flush` interval is 3600 seconds for all instances of application and device data.

You can disable the `flush` parameter by specifying the value as 0 (zero). If the `flush` parameter is set to 0, `scopent` does not log application and the device data that does not meet the thresholds specified in the `parm` file.

gapapp

The `gapapp` parameter in the `parm` file controls the modification of application class data to account for any difference between the global (system-wide) data and the sum of application data. Application data is derived from process-level instrumentation. Typically there is difference between the global metrics and the sum of applications. In systems that have high process creation rates the difference is significant. You can choose from the following options:

- If `gapapp` is blank, an application named `gapapp` is added to the application list.
- If gapapp = UnassignedProcesses, an application named UnassignedProcesses is added to the application list.
- If gapapp = ExistingApplicationName (or) gapapp = other, the difference to the global values is added to the specified application instead of the other application being logged separately and adding a new entry to the application list.

Application Definition Parameters

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

You can group logically related processes together into an application to log the combined effect of those processes on computing resources such as memory and CPU.

Each process on the system belongs to one application only. Processes are based on name, not program path. Therefore, two processes with the same name but different paths (file system location), are considered to be the same process.

An application can be a list of files combined with other files that have a priority range. If `file` and `priority` parameters are both specified for the same application, a process must meet the specifications of both `file` and `priority` to belong to that application.

Application

The application name defines an application or class that groups together multiple processes and reports on their combined activities. It is a string of up to 19 characters used to identify the application. Application names cannot contain embedded blanks.

The `application` parameter must precede any combination of `file, 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` statement.

If `file` and `priority` parameters are specified for the same application, a process must match one of the file name type parameters. A process cannot belong to a particular application if it fails to match either of the two parameters. However, you can use the `OR` parameter to allow more than one application to apply to the same application.

You can define up to 998 applications. Performance Agent predefines an application called `other`. The `other` application collects all processes not defined by `application` statements in the `parm` file.

For example:

```
application Office
file winword* excel* msaccess*
application xyz
file xyz*,startxyz*
```

You can have a maximum of 1000 file specifications for all applications combined. The previous example includes five 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.

By default, no user applications are defined.

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 previous `application` statement issued. An error is generated if no `application` statement is found. The *file name* can be any of the following:

- A single program file such as `winword`.
- A group of 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 for all `file` specifications.

When you define executable files for an application in the `parm` file, no file extensions are required. For example, you can define winword in the `parm` file without its `.exe` extension.

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

You can enter multiple file names on the same parameter line (separated by spaces) or in separate file statements. File names cannot be qualified by a path name. For example:

```
application payroll
file account1,basepay,endreport
application Office
file winword* excel*
file 123* msaccess*
```

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

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

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 System
file nt*
or
priority = 13-31
```

This defines the application (System) that consists of any programs with names beginning with "`nt`" plus programs with any name running at a priority between 13 and 31, inclusive.

User

The `user` parameter specifies which users (login names) belong to the application. The format is as below:

```
application <application_name>
file <file_name>
user [<Domain_Name>]\<User_Name
```

The domain name in the `user` parameter is optional. You must specify the domain name to specify the user names of non-local system.

For example:

```
application test_app
file test
user TestDomain\TestUser
```

If you specify the user name without the domain name in the `user` parameter, the user names will be considered to be the user names of the local system.

For example:

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

You can only use the wild card asterisk (*) to ensure the user names with a similar string of characters prefixed before the asterisk (*) and suffixed after the asterisk (*) belong to the application.

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.

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 REALTIME
priority 16-31c
```

Processes can range in priority from 0 to 31.

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.

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

Application Definition Examples

The following examples show application definitions.

```
application perftools
file scope* alarm* perf*

application editors
file edit* write*

application DB
file ora* infrm*
```

`*` except for real time processes.

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

Program	Application
scopent	perftools
edit	editors
write	editors
oracle	DB

Sample `parm` file applications for use with performance software are available on the following directory:

<*disk drive*>:\Program Files\HP\HP BTO Software\examples\ovpaconfig

Configure Data Logging Intervals

The default collection intervals used by `scopent` are 60 seconds for process data and 300 seconds for global and all other classes of data. You can override this using the `collectioninterval` parameter in the `parm` file. The values must satisfy the following conditions:

- The collection intervals for process data can be configured between 5 to 60 seconds in steps of 5 seconds.
- The collection interval for global data can be configured to one of the following values: 15, 30, 60 and 300 seconds. The global collection interval must be greater than or equal to process interval, and a multiple of the process collection interval. The global collection interval applies to the global metrics and all non-process metric classes such as filesystem and application.

Example:

```
collectioninterval process=15, global=30
```

In this example, the collection interval for process data is set to 15 seconds, and global and all other classes of data to 30 seconds:

The collection interval for process data:

- Is a valid value (this value is a multiple of 5)

- Divides evenly into collection interval for global data, 30 seconds (30%15 = 0)

The collection interval for global data:

- Is a valid value (this value can be 15, 30, 60 or 300)
- Is a multiple of process interval 15 seconds (30%15 = 0)

Therefore, these values for the collection intervals are valid.

Starting and Stopping Data Collection

The `scopent` collector is designed to run continuously. The only time you should stop it is when any of the following occurs:

- You are updating the Performance Agent software to a new release.
- You are resizing a Performance Agent log file.
- You are modifying configuration files such as `parm` or the alarm definition (`alarmdef`) file.
- You are backing up your log files, so that all log files are synchronized.
- You are shutting down the system.
- You are adding the hardware or modifying the configuration changes. Changes made take effect only when `scopent` is started.

Starting and Stopping from the Windows GUI

Performance Agent Services start the collection and logging of resource and performance data on your system. To start the Performance Agent services from the Windows GUI, follow these steps:

1. To open **OV Performance Agent**, click **Start** > **Programs** > **OV Performance Agent**. OR

   Click **Start/Stop** from the Agent menu on the Performance Agent main window. Or, from the Windows Control Panel, double-click **OVPA** to open the **OV Performance Agent Start/Stop** dialog box.
2. To start all Performance Agent services, click **Start Services**.
3. To stop all Performance Agent services, click **Stop Services**.

Starting and Stopping from the Command Prompt

Performance Agent Services start the collection and logging of resource and performance data on your system. You can start and stop Performance Agent services from the Windows command prompt or from a batch file using the `ovpacmd` (`ovpacmd.exe`) command.

- To start all Performance Agent services from the command prompt, type the following:

  ovpacmd start

  or

  ovpacmd start all

- To stop all Performance Agent services from the command prompt, type the following:

  ovpacmd stop

  or

  ovpacmd stop all

The executable `mwacmd.exe` is still provided in this release for backward compatibility. The functionality is same as `ovpacmd.exe`.

Daylight Savings

During daylight savings, the system time is set back by 1 hour in the relevant time zones. At this point, data collection stops for an hour until the system time synchronizes with the timestamp of the last logged record.

When daylight savings is turned off, the system time advances by 1 hour and hence the timestamp of the next logged record advances by an hour. This introduces a one hour gap after the last logged record even though data collection does not stop.

Changing System Time Manually

When the system time is set back manually, data collection stops and `coda` does not respond to commands such as **`ovcodautil`** and **`perfstat`**. These utilities will hang after the system time is set back. To continue logging data and to get responses from `coda`, perform the following steps:

1. Stop `coda`. To stop `coda`, type the following command:

   ovc -stop coda

2. Backup the `coda.*` files in the <*DataDir*>`\datafiles\` directory and remove them.

3 Start `coda`. To start `coda`, type the following command:

```
ovc -start coda
```

Effective Data Collection Management

Efficient analysis of performance depends on how easy it is to access the performance data you collect. This sections 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

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 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 `scopent` 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 `scopent` and by the DSI logging process. For more information on DSI log file maintenance, see the *HP Performance Agent for Windows: Data Source Integration Guide*.

Setting Mainttime

Normally, `scopent` only performs 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. The default for the `mainttime` parameter is 23:30 hours (11:30 pm). If this is a time when your system is often busy, you should change the `mainttime` value in the `parm` file to a time when your system is less busy.

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 PerfView. Some ways to reconfigure the `scopent` log file sizes are discussed below.

The `scopent` collector logs different types of data into their own log files. This enables 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 month 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, and then use the resize log file function in Performance Agent or in the `utility` program to change your log file size. The resize log file function scans the log files and determines how much data is being logged each day. It 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 are automatically resized by `scopent`.

The `scopent` data collector rolls back log files at the `mainttime` specified in the `parm` file. If you edit the `parm` file and restart `scopent`, the log files are not rolled to the new sizes until the `mainttime` occurs. It is important to have `scopent` 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 is 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 data, this can result in a log file significantly exceeding its configured maximum size before it is rolled back.

For every configured interval, `scopent` checks the available disk space on the system where the log files reside. If the available disk space falls below 1 MB, `scopent` 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 at least two days' worth of data in them), they are rolled back.
- If, following the log file maintenance, the available disk space is still not greater than 1 MB, `scopent` 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 that are logged on a short data interval. By default, process data is logged each minute and all other data is logged every 5 minutes. Both the process data and other data is configurable. To make room for new data, older data is removed when the log files reach their

maximum sizes. 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 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 weeks' 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. The following example shows a script that would perform the weekly processing:

```
rem  Extract detailed data into monthly archive files.

   #extract -gapdt -xm
```

Each week during the month the data is appended to the prior week's data. When a new month starts, `extract` creates a new archive log file and splits that week's data into the appropriate monthly archive log file. The log files are 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 it is 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.

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 or you can restore a monthly detailed extracted file, and then extract only one week's global and application summary data for analysis with Performance Manager.

It is imperative that you extract all data (detail) to the monthly log files. You cannot get data from a log file on a subsequent extraction if that data was not included in the original extraction.

3 Using HP Performance Agent

Introduction

To access the Performance Agent graphical interface click the HP Performance Agent Software icon in the following folder:

Start→Programs→HP→Performance Agent→ HP Performance Agent Software

Figure 2 HP Performance Agent Main Window

This chapter describes the following tasks that you perform using the Performance Agent graphical interface:

- Data Types and Classes
- Summarization Levels
- Ranges of Data to Extract or Export
- Extracting Log File Data and Exporting Log File Data

- Archiving Log File Data
- Resizing a Log File
- Scanning a Log File for information
- Analyzing a Log File
- Configuring Export Templates
- Configuring User Options
- Configuring Collection Parameters
- Configuring Alarm Definitions
- Checking Performance Agent Status
- Building Collections of Performance Counters

Before you start using Performance Agent for tasks that involve extracting, exporting, or archiving data, read the following sections. These sections discuss the selection of data types, summarization levels, and ranges of the data to be extracted, exported, or archived.

Data Types and Classes

The following types of `scopent` log file data can be selected for extraction or exporting:

Type of Data	Type of Measurement
`global`	system-wide, or global information
`application`	processes in each user-defined application
`configuration`	system configuration usage
`process`	selected interesting processes
`disk`	disk devices usage
`filesystem`	logical disks usage
`logicalsystem`	logical system usage

Type of Data	Type of Measurement
`cpu`	CPU usage
`netif`	network interface devices usage
`transaction`	transaction tracking data

DSI log file data can be selected for exporting according to class. Each class represents one source of incoming data and consists of a group of related data items (metrics) that are logged together. For more information, see "How Log Files Are Organized" in Chapter 2 of the *HP Performance Agent for Windows: Data Source Integration Guide*.

Summarization Levels

To export data, you must specify the level of summarization you want – detail, summary, or both – when exporting log file data.

- Detail specifies that detail data from a 5-minute period is exported from all data types except process, from which detail data from a 1-minute period is exported.
- Summary specifies that data summarized over a 1-hour period is exported.
- Detail and summary together provide a maximum amount of exported data.

Summarization affects the size of the exported data. For example, hourly summary data is about one-tenth the size of 5-minute detail data.

Ranges of Data to Extract or Export

You can select specific data to extract or export depending on the date and time it was logged. For example, you might want data that was logged every day (Monday through Sunday) from 8:00 am to 8:00 pm during a 30-day period starting at a specific date.

If you do not specify a specific range of data to be exported, data is extracted or exported using the default starting date – the date 30 days before the last date in the log file. Or, if less than 30 days of data is present, the date of the earliest record in the log file.

You can also limit extraction or export to data that was logged during specific hours of the day that correspond to work shifts (such as 8:00 am to 5:00 pm) from Monday through Friday. If no shift is specified, the default is extraction or export of 24-hours' worth of data for every day including weekends.

Extracting Log File Data

The data collector, `scopent`, continuously collects data and logs it into raw log files. You can extract specific data from the default global log file set, `logglob`, into extracted log files that can later be used for archiving or for analyzing by analysis programs such as Performance Manager. You can also extract data from existing extracted log files. You *cannot* extract DSI log file data.

When you specify `logglob`, all other raw log files in the log file set automatically open. For example, it is not necessary to open the `logproc` log file to extract process data; opening `logglob` enables you to access all data types in the raw log file set.

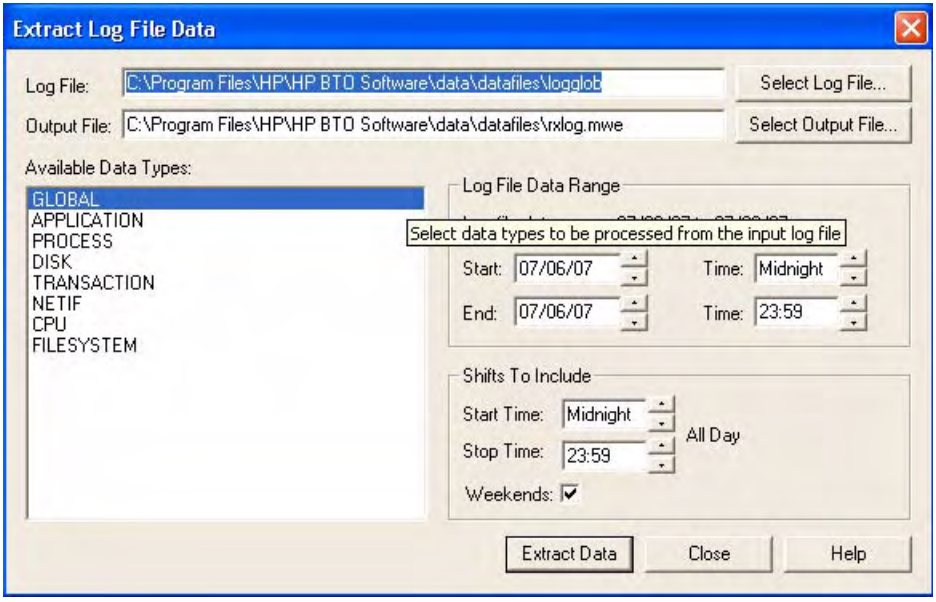


Figure 3 Extract Log File Data Dialog Box

When you display the Extract Log File Data or Export Log File Data dialog boxes after starting Performance Agent, the name of the default global log file, `logglob`, appears in the Log File box to indicate the currently active log file. `Logglob` remains active until you close Performance Agent or select a different log file. When you select a different log file, that file's name appears in the Log File box as the currently active log file.

To Extract Log File Data

To exttact log file data, perform the following steps:

1. Click **Extract** from the **Logfile** menu on the main window. The Extract Log File Data dialog box appears, showing the names of the currently active log file and the currently selected output file. You can easily specify a different log file and output file.
2. After selecting the type of data to be extracted, select the range of log file data to be extracted and shifts to include. Click the **Extract Data** button to start the extract process.

For step-by-step instructions for extracting log file data, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Extract log file data**."

Exporting Log File Data

You can export specific data from raw or extracted `scopent` log files or from DSI log files into formatted export files that can be used by spreadsheets and other reporting tools.

The export function does *not* remove any data from the log file.

The following sections discuss:

- File attributes that you can specify for the exported data
- Export templates you can use to simplify the export task.
- Default export file
- Overview of the export task

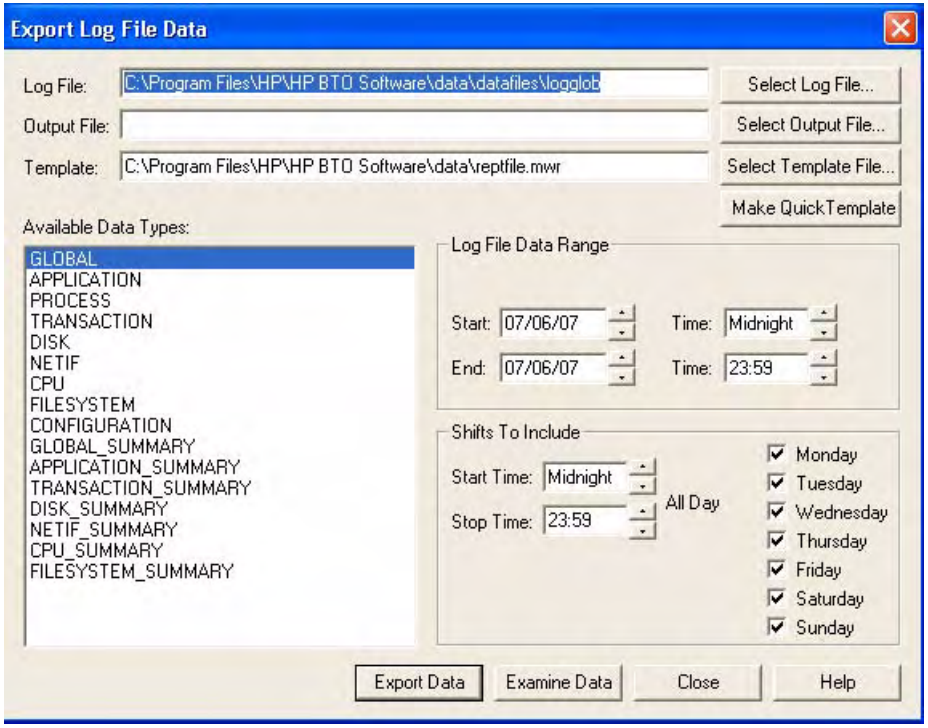


Figure 4 Export Log File Data Dialog Box

File Attributes

You can assign various file attributes to your exported data, including file formats, values that represent missing data, field separators, column headings, number of minutes for summary intervals, layout, data types, and specific metrics to be included in the export file. These attributes can be saved in export template files or specified directly using the Make Quick Template dialog box. (For more information, see Making a Quick Export Template on page 65.)

File Format

The output format for an exported file can be ASCII, datafile, binary, or WK1 (spreadsheet):

- ASCII format is printable character data, suitable for printed reports or post-processing by user-written programs or batch files.
- Datafile format is similar to ASCII except that all non-numeric items are enclosed in double quotes (" "). Datafile format is useful for transferring data to most spreadsheets and graphic packages.
- Binary format is not printable. It is more compact because numeric 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 maintains the highest metric accuracy.
- WK1 (spreadsheet) format is compatible with Microsoft Excel, and other spreadsheet, database, and graphing products.

Missing Value

An exported file can contain the data value that replaces data missing from the source log file. Missing data can occur when a metric is not available for certain versions of the `scopent` collector. In addition, the multiple layout export formats for applications, disks, and transactions reserve space in the output record for every application, disk, or transaction. If no data was logged for a particular entry during an interval, its data will be “missing”.

Field Separators

A field separator character can be inserted between each metric in ASCII and datafile format exported files. Separator characters can be printable or non-printable (such as a tab character).

The default separator character is a blank space, but many programs require a comma as a field separator.

Summary Minutes

The number of minutes for each summary interval can be specified. The number of minutes can range from five to 1440 minutes (one day).

Headings

Column headings can be included in exported files. The first record in the file is exported data (except in WK1 format files). However, if you include headings in the file, ASCII and datafile format files have the export file title (if specified) plus column headings for each column of metrics written *before* the first data records in the file. Headings in binary format files are written before the first record in the file and contain descriptions of the metrics.

WK1 files always contain headings.

Multiple Layout

You can specify multiple layouts (per record output) for multi-instance data types such as application or disk.

Single layout writes one record for every application or disk 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 or disk.

Export File Title

You can specify the title for an export file. The title can contain literal strings as well as substitution keywords. The following items can be substituted in the `export title` string:

`!date`	the date the export file is created
`!time`	the time of day the export file is created
`!logfile`	the name of the log file from which data is obtained
`!collector`	the name and version of the collector program (either `scopent` or `dsilog`)
`!class`	the type of data requested
`!system_id`	the identifier of the system that collects the `scopent` raw or extracted log file data (not valid with DSI log file data)

For example, you could type the following string:

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

The string would generate a title similar to the following:

```
gemini data from logglob on 10/25/99 08:30 AM
```

Export File Templates

The export task uses export templates that define the file attributes for your exported file. The default file attributes are taken from the file `<rpmtools>\data\reptfile.mwr` that specifies:

- ASCII file format
- A 0 (zero) for the missing value
- A blank space as the field separator
- 60-minute summaries
- Headings are included
- A recommended set of metrics for a given data type or class is included in the export

You can either specify a different export template file or make ad hoc file attribute specifications (see Making a Quick Export Template on page 65).

You can also create customized export templates using the Configure Export Templates dialog box (see Configuring Export Templates on page 67).

Default Export Files

If you do not specify an output file to contain your exported data, the export task creates a default output file in your <*disk drive*>:`\Program Files\HP\HP BTO Software\data\datafiles` directory based on the data type and level of summarization you specified.

scopent Data

When you export `scopent` log file data, the following default file names are assigned to the exported files.

`xfrdGLOBAL.ext`	global detail data
`xfrsGLOBAL.ext`	global summary data
`xfrdAPPLICATION.ext`	application detail data
`xfrsAPPLICATION.ext`	application summary data
`xfrdPROCESS.ext`	process detail data
`xfrdDISK.ext`	disk device detail data
`xfrsDISK.ext`	disk device summary data
`xfrdCPU.ext`	CPU detail data
`xfrsCPU.ext`	CPU summary data
`xfrdFILESYSTEM.ext`	filesystem detail data
`xfrsFILESYSTEM.ext`	filesystem summary data
`xfrdNETIF.ext`	`netif` detail data
`xfrsNETIF.ext`	`netif` summary data
`xfrdTRANSACTION.ext`	transaction tracking detail data
`xfrsTRANSACTION.ext`	transaction tracking summary data

xfrdCONFIGURATION.ext	configuration detail data
xfrdLOGICAL.ext	logical system detail data file
xfrsLOGICAL.ext	logical system summary data file

When the export task completes, you can view the contents of the output export file by clicking the **Examine Data** button in the Export Log File Data dialog box.

The default file names are created from the data type name. The prefix is either xfrd or xfrs depending on whether the data is detail or summary data. The extension (.ext) is the file format specified in the export template file: asc (ASCII), bin (binary), dat (datafile), or wk1 (spreadsheet).

For example:

xfrdNETIF.wk1 contains detailed data for the NETIF data type in spreadsheet format.

xfrsAPPLICATION.asc contains summarized data for the application data type in ASCII format.

DSI Data

When you export DSI log file data, the default file names are created from the class name. The prefix is either xfrd or xfrs, depending on whether the data is detail or summary data. The extension is the file format specified in the export template file: asc (ASCII), dat (datafile), or wk1 (spreadsheet).

For example:

xfrdACCTG.wk1 contains detailed data for the ACCTG class in spreadsheet format.

xfrsPERSONL.asc contains summarized data for the PERSONL class in ASCII format.

To Export Log File Data

To export log file data, perform the following steps:

1. Click **Export** from the Logfile menu on the main window. The Export Log File Data dialog box appears, showing the names of the currently active log file and the currently selected export template file. You can specify a different log file and export template file.

If you specify an output file, data from all selected data types or classes are placed in the same file. If you do not specify an output file, a default output file is created based on the data type or class and summarization level you specify. (See Default Export Files on page 62.)

2 After selecting the one or more types or class of data to be exported, select the range of log file data to be exported and shifts to include.

3 To override the file attributes specified in your default export template, or to select specific metrics to be included in your export file, click the **Make Quick Template** button (see To Make a Quick Template on page 65).

4 After you complete the steps for making a quick template or select specific metrics for your export file, return to the Export Log File Data dialog box. Click the **Export Data** button to start the export process.

5 When the export finishes, you can view the contents of your export file by clicking the **Examine Data** button.

For step-by-step instructions for exporting log file data, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Export log file data**."

Making a Quick Export Template

Use the Make Quick Template function in the Export Log File Data dialog box to select specific metrics to be included in your export file and to change any of the file attributes and metrics that are specified in the export template you selected for your export.

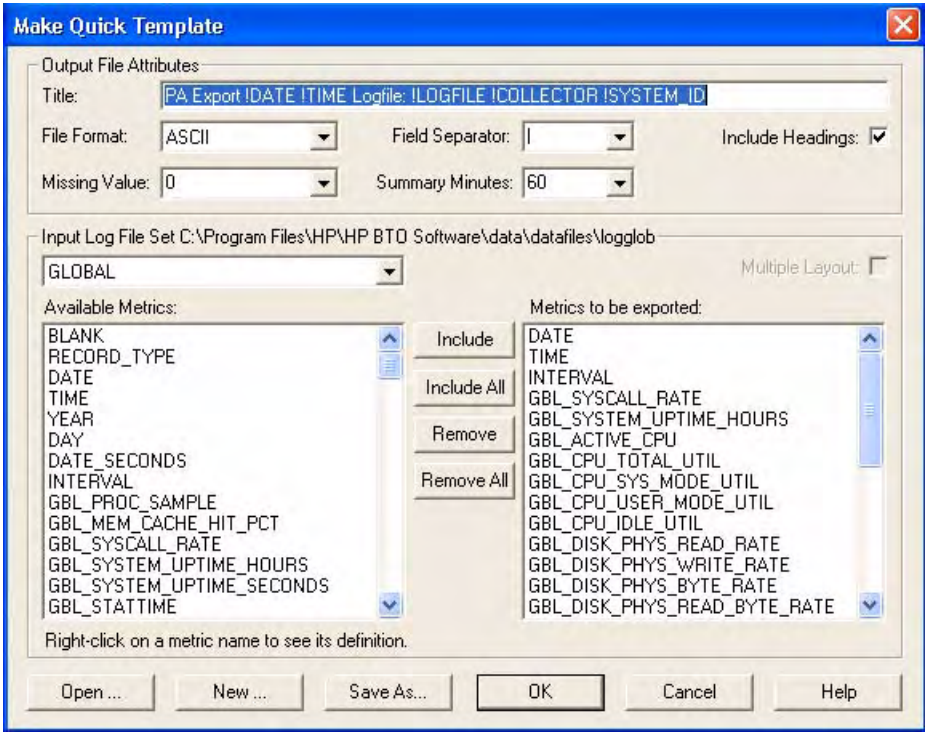


Figure 5 Make Quick Template Dialog Box

To Make a Quick Template

To make a quick template, perform the following steps:

1 Click the **Make Quick Template** button in the Export Log File Data dialog box. The Make Quick Template dialog box appears, showing the title of the export file selected for the export.

2 The boxes below Output File Attributes show the current settings for the export file, based on the file attributes set in the export template selected for the export. You can modify any of these settings.

3 To use a different export template file, click the **Open** button.

4 You also have the option of clearing all existing settings from the Make Quick Template dialog box and creating a totally new export template file by clicking the **New** button.

Selecting Metrics to Export

- After selecting the data type or class of the metrics to be exported, you can select the specific metrics that you want included in the export. Each data type or class has its own set of metrics that are listed under Available Metrics. The metrics listed under "Metrics to be exported" are the metrics specified by the current export template to include in the export. You can either use that list, remove metrics from the list, or select other available metrics to include in the export.
- If you select the `application`, `disk`, `cpu`, `filesystem`, `netif`, or `transaction` data types, select the **Multiple Layout** check box to generate multiple layouts (per record output), or leave it cleared to generate single layout.

Saving Your Selections

After making your selections, you can either:

- Proceed with the export using the selections you made to the file attributes and to the list of metrics to be exported but WITHOUT saving the changes to any template file.
- Save your selections to a new export template file that will be available for use in future exports. The original export template file remains unchanged.

If you save your selections to a new export template file, you must include the `.mwr` file name extension when you specify the new template file name. For example, `mytmplte.mwr`

For step-by-step instructions for making a quick export template, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Make a quick export template**."

Configuring Export Templates

Use the **Export Templates** command from the Configure menu to customize an existing export template file or create a new export template file. Use the Configure Export Template dialog box to select new file attributes and specific metrics to be included in the template.

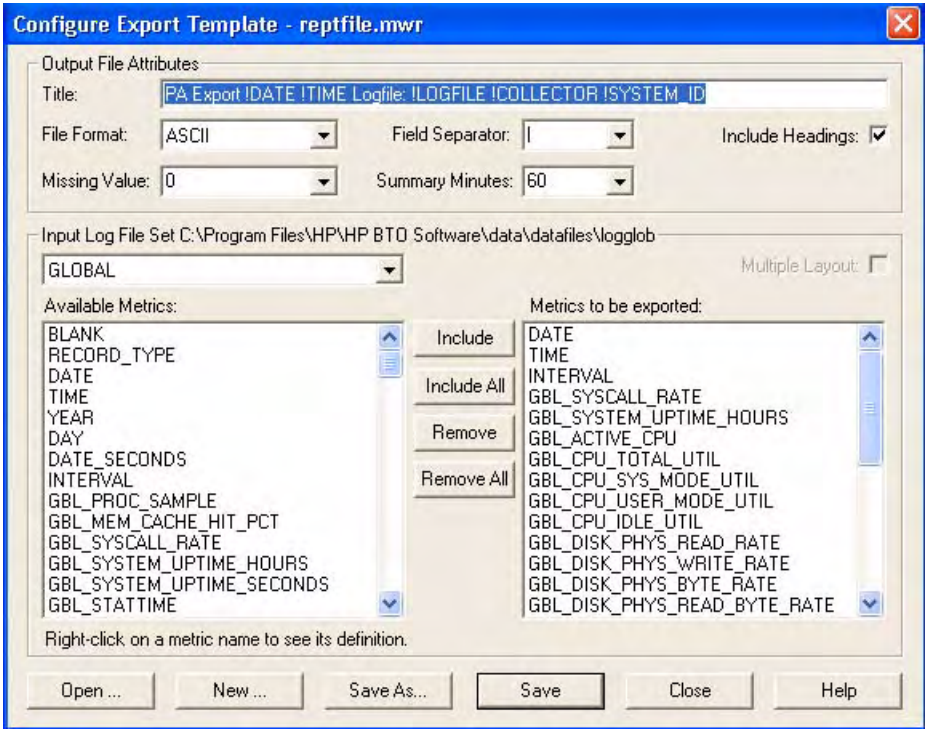


Figure 6 Configure Export Template Dialog Box

To Configure an Export Template

To configure an export template, perform the following steps:

1. Click **Export Templates** from the Configure menu on the main window. The Configure Export Template dialog box appears showing the name of the currently open export template file in the dialog box title. To edit a different export template file, click the **Open** button.

2 The boxes below Output File Attributes show the current settings for the export file, based on the file attributes set in the export template you are configuring. You can modify any of these settings.

3 You can also clear all existing settings from the Configure Export Template dialog box and create a totally new export template file by clicking the **New** button.

Selecting Metrics for Export

- After selecting the data type or class of the metrics to be exported, you can select the specific metrics that you want included in the export. Each data type or class has its own set of metrics that are listed under Available Metrics. The metrics listed under "Metrics to be exported" are the metrics specified by the current export template to include in the export. You can use that list, remove metrics from the list, or select other available metrics to include in the export.
- If you select the `application`, `disk`, `cpu`, `filesystem`, `netif`, or `transaction` data types, select the **Multiple Layout** check box to generate multiple layouts (per record output), or leave it cleared to generate single layout.

Saving Your Selections

You have three choices for saving the template you edited:

- Save the changes to the current template file.
- Save the changes to a new template file, in which case the original template file remains unchanged.
- Cancel the changes to avoid making changes to any template file.

If you save your selections to a new export template file, you must include the .mwr file name extension when you specify the new file name.

If you click the **Close** button *after* you make changes to the template file but *before* you saved them, you are prompted to either cancel or save your changes. Clicking the **Cancel** button returns you to the Configure Export Template dialog box.

For step-by-step instructions for configuring an export template file, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Configure an export template file**."

Archiving Log File Data

Use the **Archive** command from the Logfile menu to extract selected portions of `scopent` log file data for archiving and future data analysis.

For archival purposes, data can only be extracted from the raw log files . The extracted data is automatically placed in an archival output file in the *<disk drive>*`:Program Files\HP\HP BTO Software\data\datafiles` directory whose name reflects the selected archival period. These files can be copied to tape for offline storage and then deleted to release disk space.

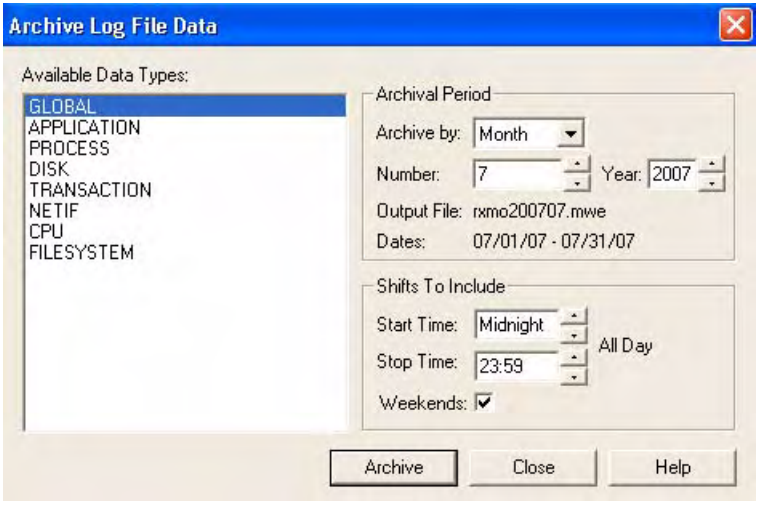


Figure 7 Archive Log File Data Dialog Box

Archival Periods

You can select data to be extracted based on data logged during a specific weekly, monthly, or yearly period.

You can also limit extraction to data that was logged during specific hours of the day that correspond to work shifts and include or exclude weekends (Saturday and Sunday). If no shift is specified, the default is extraction of 24-hours' worth of data for every day. By default, weekends are included.

Appending Archived Data

The archiving function has a special feature. Depending on which archival period you select – weekly, monthly, or yearly – the *previous* output file for that archival period is automatically checked to see if it contains data extracted up to the last day. If not, the data is appended to the file to complete the previous archival period's extraction.

For example, on May 7, 1999, you begin archiving monthly data for May 1999. An output file named `rxmo199905.mwe` is created containing data from May 1 through the current date (May 7).

On June 4, 1999, another monthly archival period is invoked. Before the `rxmo199906.mwe` file is created for June, the `rxmo199905.mwe` file from the previous month is checked. When it is found to be incomplete, data is appended to it to complete the extraction through May 31, 1999. Then, the `rxmo199906.mwe` file is created to hold data from June 1, 1999, to the current date (June 4).

As long as another monthly (weekly, yearly) archival period is invoked at least once a month (week, year), this feature will complete each archival period's file before creating the next archival period's file.

Archiving Tips

Here are some suggestions for archiving your log file data:

- Once a month, specify the monthly archival period and extract all the detail data from your raw log files into a single extracted log file.
- If your system generates more than 64 MB of data each month, you may need to extract data on a weekly basis, or you can eliminate process detail data from the extraction.
- Extract global summary and application summary data on a yearly basis, which should minimize the disk space required. This archive file can then be used for long-term analysis of trends.

To Archive Log File Data

To archive log file data, perform the following steps:

1. Click **Archive** from the Logfile menu on the main window. The Archive Log File Data dialog box appears. The data to be archived is extracted from the raw log file set.
2. After selecting the type of data to be archived from the Available Data Types list, specify the archival period – Week, Month, or Year, and any shifts to include.
3. Click the **Archive** button to start the archiving process.

For step-by-step instructions for archiving log file data, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Archive log file data**."

Analyzing a Log File

Use the **Analyze Log File** command from the Logfile menu to analyze data in the raw log file set against alarm definitions in an alarm definitions file, and report on any resulting alarm activity.

This task enables you to evaluate whether or not your alarm definitions are a good match against the historical data collected on your system. It also enables you to decide if your alarm definitions will generate too many or too few alarms on your analysis system.

The raw log files being analyzed are referenced in Performance Agent's default data source, SCOPE. To analyze a different log file, place a USE statement in your alarm definitions file that specifies the name of the data source that references that log file. (For more information, see USE Statement on page 271.)

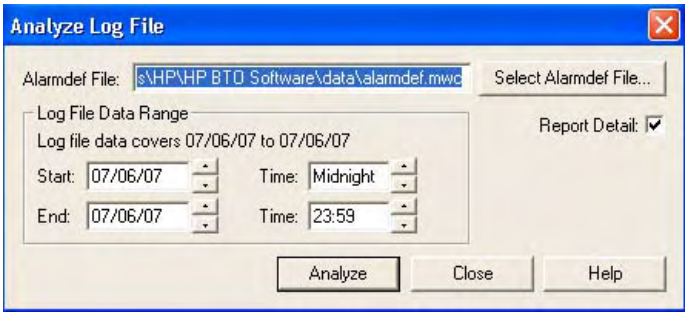


Figure 8 Analyze Log File Dialog Box

Range of Data to be Analyzed

You can analyze log file data that was collected during a specific period of time. If you do not specify a specific range of data to be analyzed, data is analyzed using the default starting date – 30 days before the latest date in the log file or, if fewer than 30 days of data are present, the date of the earliest record in the log file.

Analysis Report

As this task executes, it generates a printable report that lists alarm events and an alarm summary. (Alarm events are listed only if the Report Detail box in the Analyze Log File dialog box is checked.)

- Alarm events include alarm START, END, and REPEAT status plus any text in associated PRINT statements. Also, if any text in PRINT statements are listed as conditions (in IF statements) and become true, the text is included . EXEC statements are not executed but are listed so you can see what would have been executed.
- Alarm summaries show a count of the number of alarms that occurred and the amount of time each alarm was active (on). The count includes alarm starts and repeats, but not alarm ends.

For additional information about analyzing historical data, see Analyzing Historical Data for Alarms on page 249.

To Analyze a Log File

To analyze a log file, perform teh following steps:

1. Click **Analyze** from the Logfile menu in the main window. The **Analyze Log File** dialog box appears showing the name of the currently selected alarm definitions file.
2. To use a different alarm definitions file, click the **Select Alarmdef** File button.
3. Select the range of log file data to be analyzed.
4. Check the **Report Detail** box if you want to include alarm events in the analysis report. Otherwise, only the alarm summary is generated.
5. Click the **Analyze** button to start the analysis. The analysis results are displayed in a MeasureWare Agent Report Viewer window.

For step-by-step instructions for analyzing a log file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Analyze a log file**."

Scanning a Log File

Use the **Scan Log File** command from the Logfile menu to scan a `scopent` log file and create a report on its contents. You can either scan an entire log file or scan portions of a log file for data that was collected during a specific period of time.

The report produced by the scan consists of 12 sections. The following four sections of the report are always printed.

- Process summary report
- Collector coverage summary
- Log file contents summary
- Log file empty space summary

The following eight sections of the report are printed only if you select **Report Detail** in the Scan Log File dialog box.

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

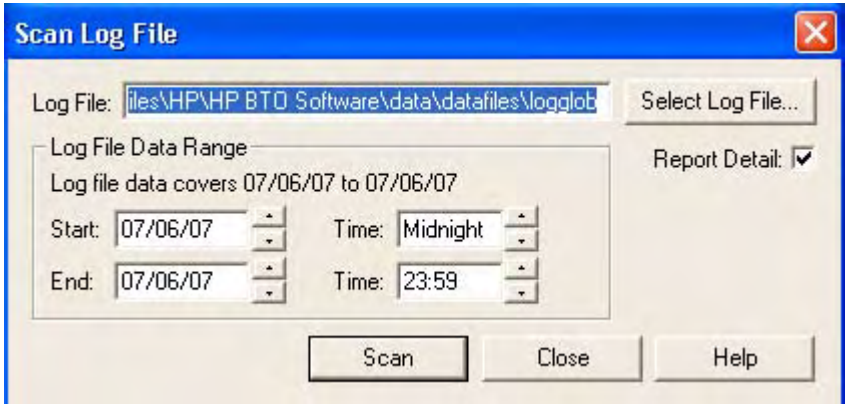


Figure 9 Scan Log File Dialog Box

To Scan a Log File

To scan a log file, perform the following steps:

1. Click **Scan Log File** from the Logfile menu on the main window. The Scan Log File dialog box appears with the name of the currently open log file highlighted.
2. To scan a different log file, click the **Select Log File** button.
3. To scan data that was logged during a specific time period, under Log File Data Range, select or type the dates and times for the beginning and end of that time period.
4. Check the **Report Detail** box if you want a complete scan report. Otherwise, only a subset of the report is generated.
5. To start the scan process, click the **Scan** button. The scan results are shown in a Performance Agent Report Viewer window.

For step-by-step instructions for scanning a log file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Scan a log file**."

Resizing a Log File

Use the **Resize Log File** command from the Logfile menu to change the size of your raw `scopent` log files. You can resize using either the size in megabytes for the given file or the number of days of data the file should hold.

The maximum size of a raw log file is specified in the `size` parameter in the `parm` file. Resizing the log file gives you more control over how often the log file data is rolled back.

You can select any of the following types of data to resize: `global`, `application`, `process`, `device`, or `transaction`, which correspond to the raw log files `logglob`, `logappl`, `logproc`, `logdev`, and `logtran`. You then choose how the resize will be performed – in megabytes or by number of days. Depending on which type of resize you choose, the Log File Settings box in the Resize Log File dialog box shows the following:

- The Maximum Size fields show current file size, the new file size, and the change made by the resize.
- The Empty Space fields show the amount of room in the current file, the amount required in the file after the resizing process is complete, and the change. These values are used to determine if any of the data currently in the log file must be removed in the resizing process.
- The Data Records fields show the amount of data records contained in the current log file and the new amount of data records that will be in the resized log file.

Log file sizes are maintained in megabytes. Often it is more convenient to specify sizes in days rather than megabytes. If you select "Size in Days", all units on the dialog box will change to "days". The conversion from megabytes to days is based on a "megabytes-per-day" value for each type of data. Initially, estimated values are used for this conversion.

A more precise value can be obtained by clicking the **Calibrate** button. The calibrate function actually measures the existing log files for more precise megabytes-per-day values. If you are specifying size in megabytes, no conversion is needed and the calibrate function need not be used.

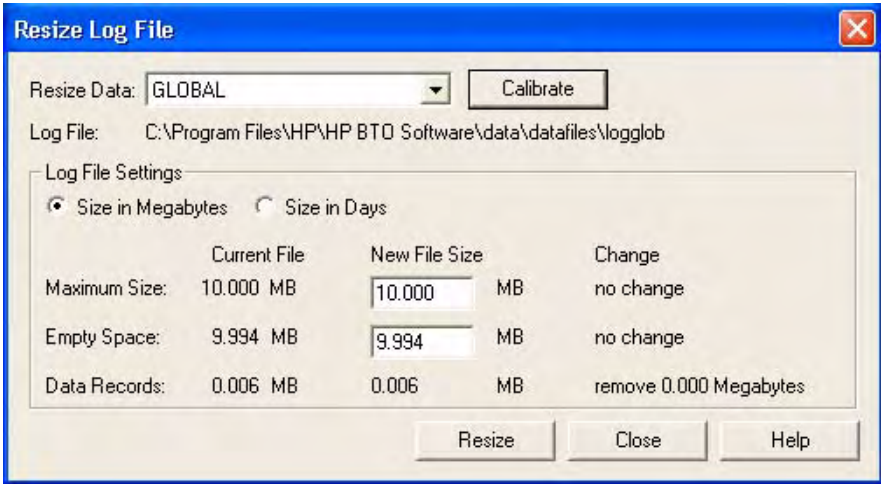


Figure 10 Resize Log File Dialog Box

To Resize a Log File

Before resizing a log file, you *must* stop the `scopent` collector. To stop `scopent`, follow the steps in Starting and Stopping Data Collection on page 43.

Attempting to resize a log file without first stopping `scopent` will not affect the existing log file. To resize a log file, perform the following steps:

1. Once `scopent` is stopped, choose **Resize Log File** from the Logfile menu on the main window to display the Resize Log File dialog box.
2. In the Resize Data box, select the type of data to be resized: **global**, **application**, **process**, **device**, or **transaction**.
3. Select either **Size in Megabytes** or **Size in Days**. Depending on what you selected, the Current File and New File Sizes are shown.
4. To perform the resize based on the New File Sizes shown, click the **Resize** button to start the resize process.

To calibrate your log file

To get a more accurate estimate of how much additional space to add to the log file when sizing in days, perform the following steps:

1. Click the **Calibrate** button. Within moments, the actual number and size of the data records that were logged in the file during the last 30 days are displayed.
2. Click the **Close** button to return to the Resize Log File dialog box. New Current File and New File Size values based on the calibration are shown, if sizing in days.
3. Click the **Resize** button to resize the log file.
4. Before performing another task, start `scopent` using the steps in Starting and Stopping Data Collection on page 43.

For step-by-step instructions for resizing a log file, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Resize a log file**."

Configuring User Options

Use the **Options** command from the Configure menu to control the display of the toolbar, status bar, dialog help tips, and Tip of the Day on your main window and while you are using Performance Agent.

You can also use the **Options** command to configure an editor or word processor for modifying the collection parameters and alarm definitions files and to select the type of status information you want to view when you choose the **Status command** from the Agent menu.

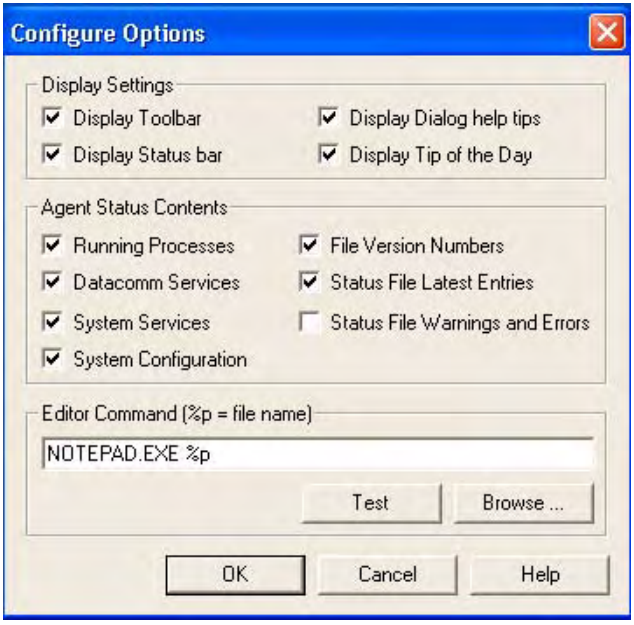


Figure 11 Configure Options Dialog Box

To Configure User Options

To configure user options, perform the following steps:

1 Click the **Options** command from the Configure menu in the main window to display the Configure Options dialog box.

2 Select the **Display Toolbar** check box to display the toolbar in the main window.

3 Select the **Display Status Bar** check box to display current status at the bottom of each dialog box and in the main window.

4 Select the **Display Dialog Help Tips** check box to display help tips within dialog boxes.

5 Select the Display Tip of the Day check box to display the current day's tip when you open the Performance Agent main window.

To configure an editor or word processor

To configure, perform teh following steps:

1 Type the editor's directory path and file name in the Editor Command box, using the `.exe` file name extension (for example, `C:\MSOffice\winword\winword.exe`).

2 Click the **Browse** button to display the Select a Text Editor dialog box from which you can select your editor.

3 Click the **Test** button to make sure that the editor you selected is configured and then click **OK**.

To configure which agent status information you want to view

- Select one or more of the option boxes shown under Agent Status Contents and then click **OK**.

For step-by-step instructions for configuring user options, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Configure user options**."

Configuring Collection Parameters

Use the **Collection Parameters** command from the Configure menu to check the syntax of the `parm` file that is used by `scopent` for data collection. You can examine the `parm` file's settings for syntax errors and warnings and to see how much room is available for defining applications.

If any warnings or errors are found and you want to correct them, or if you want to change or add `parm` file parameters, you can easily modify the `parm` file using the `Edit Parm File` function.

A detailed description of the `parm` file and its parameters can be found in Chapter 2, Managing Data Collection.

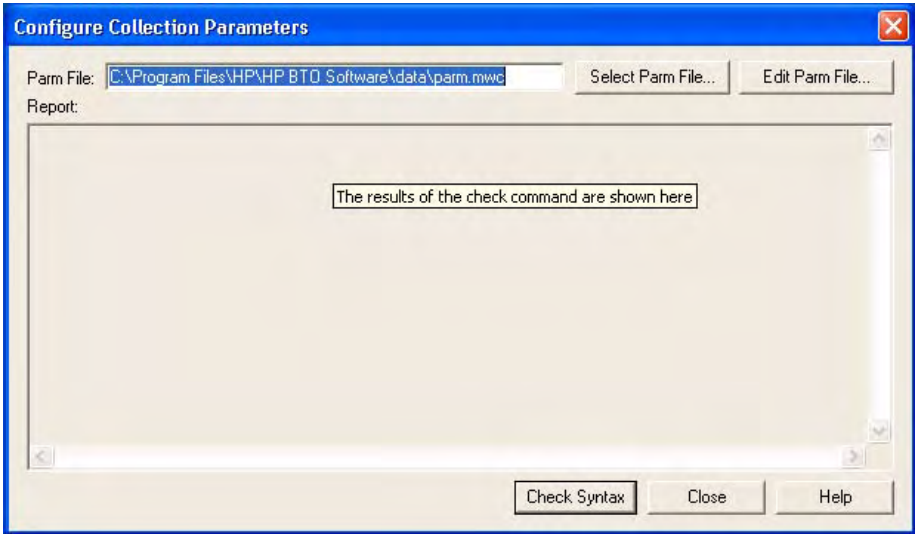


Figure 12 Configure Collection Parameters Dialog Box

To Check the Syntax of a Collection Parameters File

To check the syntax, perform the following steps:

1. Click **Collection Parameters** from the Configuration menu on the Performance Agent main window. The Configure Collection Parameters dialog box appears showing the name of the currently open `parm.mwc` file in the Parm File box.
2. To check a different `parm` file, click the **Select Parm File** button.

3 To check the syntax of the `parm` file, click the **Check Syntax** button. Any resulting warnings or errors are displayed in the Performance Agent Report Viewer window.

4 To modify any portion of the `parm` file, click the **Edit Parm File** button (see the next section, To Modify a Collection Parameters File on page 82). You can position the Edit Parm File and the Configure Collection Parameters dialog boxes on your screen so that you can use both at the same time.

For step-by-step instructions for checking the syntax of the `parm` file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Check the syntax of a collection parameters file.**"

To Modify a Collection Parameters File

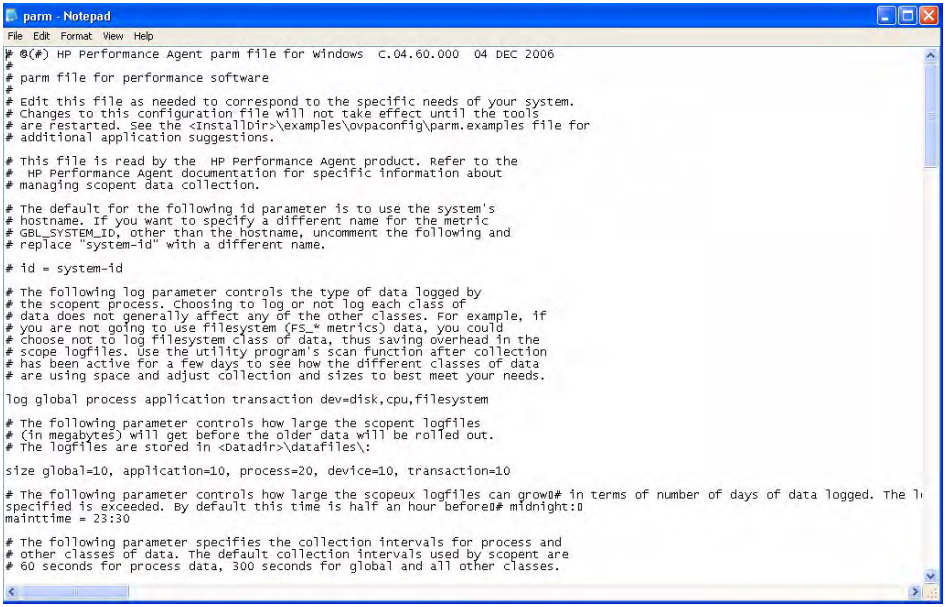


Figure 13 Modify Collection Parameters File Window

To modify, perform the following steps:

1 Click **Collection Parameters** on the Configuration menu on the Performance Agent main window and then click the **Edit Parm File** button in the Configure Collection Parameters dialog box. The contents of the currently

open parm file are displayed in a previously specified editor or word processor. (To specify an editor or word processor, see Configuring User Options on page 79.)

2 Before you make any changes to the file, see Modifying the parm File on page 24, for some rules and conventions to follow.

3 Modify the file as necessary and save the file in text format.

You cannot modify your parm file with the Windows **WordPad** editor while Performance Agent is running. You must stop Performance Agent, use **WordPad**, and then restart Performance Agent. However, you can use the **Notepad** editor to modify your file while Performance Agent is running.

To activate the changes

Before proceeding with another task, you *must* activate any changes you made to the parm file. Perform the following steps:

1 Click **Start/Stop** from the Agent menu on the Performance Agent main window to open the MeasureWare Services window.

2 Select the **Data Collection** check box.

3 Click the **Refresh** button.

4 Click the **Close** button to return to the main window.

For step-by-step instructions for modifying the parm file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Modify a collection parameters file.**"

If you use WordPad, Notepad, or Microsoft Word to modify your parm.mwc file and then use the **Save As** command to save the file, the default .txt extension will automatically be added to the file name. You will then have a file named parm.mwc.txt. To retain the parm.mwc file name, use the **Save As** command to save your file as a text file and enclose the file name in double quotes ("). For example: "parm.mwc".

Configuring Alarm Definitions

You use the **Alarm Definitions** command from the Configure menu to check the syntax of the alarm definitions in an alarm definitions file (`alarmdef.mwc`). When you determine that the alarm definitions syntax is correct, you can analyze a log file against the alarm definitions to check for alarms in historical log file data (see Analyzing a Log File on page 72).

If any warnings or errors are found and you want to correct them, or if you want to add or delete alarm definitions, you can easily modify the alarm definitions file using the **Edit Alarmdef File** button in the Configure Alarm Definitions dialog box.

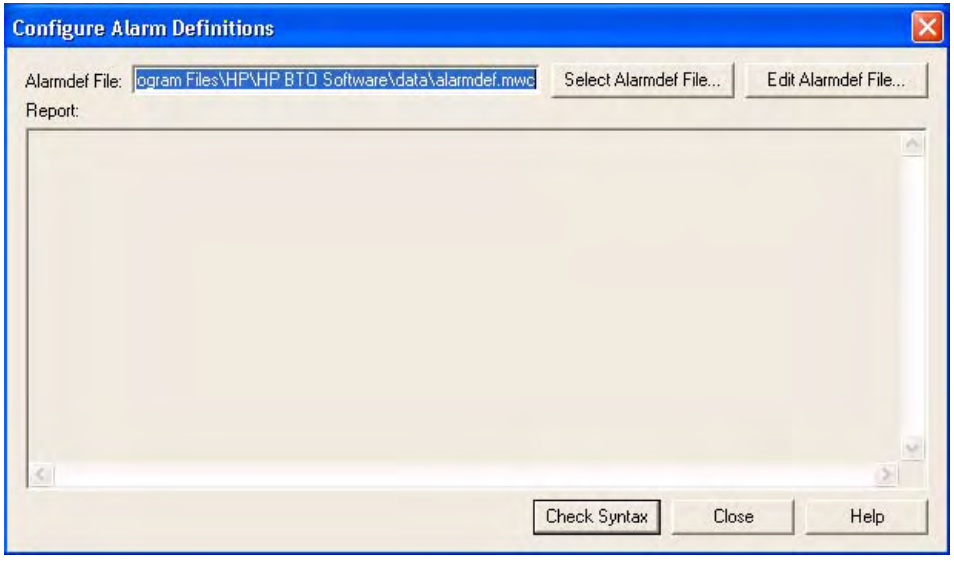


Figure 14 Configure Alarms Definitions Dialog Box

To Check the Syntax of an Alarm Definitions File

To check the syntax, perform the following steps:

1. Click **Alarm Definitions** from the Configure menu on the Performance Agent main window. The Configure Alarm Definitions dialog box appears showing the name of the currently open alarm definitions file.
2. To check a different alarm definitions file, click the **Select Alarmdef File** button.

3 Click the **Check Syntax** button to start the syntax checking process. After a few seconds, the checking results are displayed, including any warnings or errors, in the Performance Agent Report Viewer window.

4 To modify any portion of the alarm definitions file, click the **Edit Alarmdef File** button (see the next section, To Modify an Alarm Definitions File on page 85).

For step-by-step instructions for checking the syntax of an alarm definitions file, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Checking the syntax of an alarm definitions file**."

To Modify an Alarm Definitions File

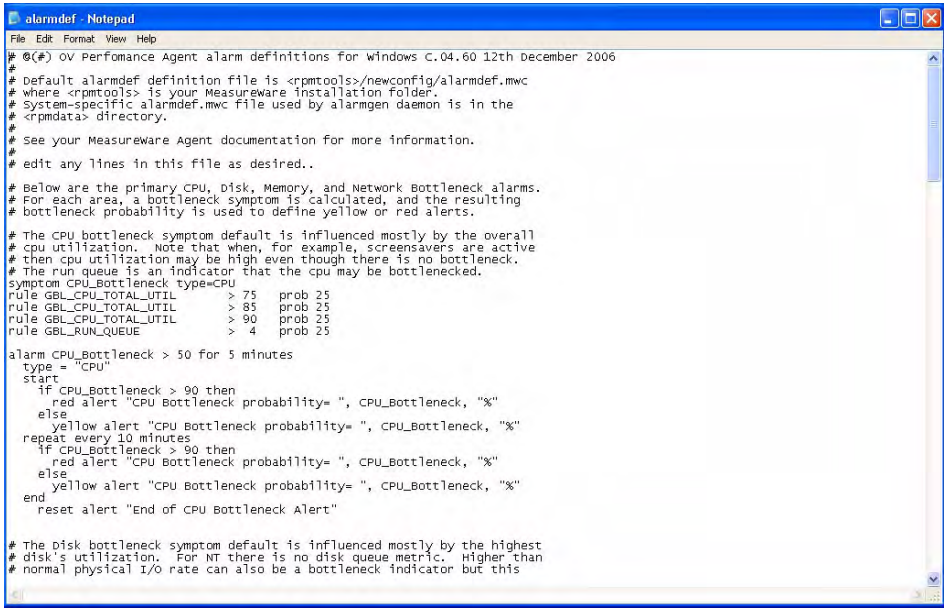


Figure 15 Modify Alarm Definitions File Window

To modify, perform the following steps:

1 Click **Alarm Definitions** from the Configure menu on the Performance Agent main window and then click the **Edit Alarmdef File** button in the Configure Alarm Definitions dialog box. The contents of the currently open alarm

definitions file are displayed in a previously specified editor or word processor. (To configure an editor or word processor, see Configuring User Options on page 79.)

2 Before you make any changes to the file, see the Alarm Syntax Reference on page 252, for detailed information about alarm definitions.

3 Modify the file as necessary and save it in text format.

If you use WordPad, Notepad, or Microsoft Word to modify your alarm definitions file and then use the **Save As** command to save the file, the default `.txt` extension will automatically be added to the file name. You will then have a file named `alarmdef.mwc.txt`. To retain the `alarmdef.mwc` file name, use the **Save As** command to save your file as a text file and enclose the file name in double quotes ("). For example, "`alarmdef.mwc`".

To activate the changes

Before proceeding with another task, you *must* activate any changes you made to the alarm definitions file. Perform the following steps:

1 Click **Start/Stop** from the Agent menu on the Performance Agent main window to open the MeasureWare Services window.

2 Select the **Alarm Definitions** check box.

3 Click the **Refresh** button.

4 Click the **Close** button to return to the main window.

For step-by-step instructions for modifying an alarm definitions file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Modify an alarm definitions file.**"

Configuring Data Sources

Performance Agent uses a set of repository servers that provide previously collected data to the alarm generator and the Performance Manager analysis product. There is a repository server for each specific data source such as `scopent` log files or DSI log files. Each data source consists of a single log file set. The data source is configured in the `datasources` file that resides in the `<DataDir>\conf\perf` directory. When you first start up Performance Agent after installation, a default data source named SCOPE is already configured and provides a `scopent` log file set.

To add repository servers for other data sources, configure them in the `datasources` file. When you restart Performance Agent, the `perflbd` daemon looks for `perflbd.mwc` (the file that resides in `<DataDir>` directory is a link to `datasources` file), reads it, and starts a repository server for each data source it finds. For more information on configuring data sources, see *HP Performance Agent Installation and Configuration Guide*.

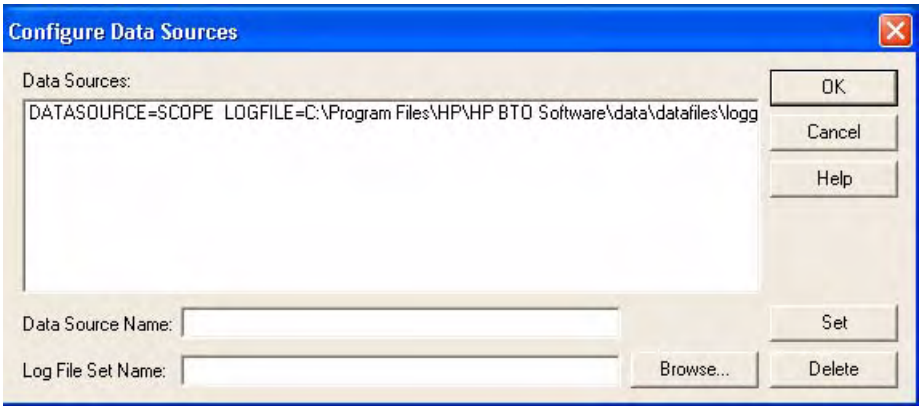


Figure 16 Configure Data Sources Dialog Box

Data Sources File Format

Each entry you place into the `datasources` file (the `perflbd.mwc` file that resides in `<DataDir>` directory is a link to `datasources` file) represents a data source consisting of one log file set. The entry specifies the data source name by which the repository server is to be known and where the data it contains is to be found. Entries are case-insensitive. The syntax is:

datasource=*datasource_name* **logfile=***logfile_set*

- **datasource** is a keyword. *datasource_name* is the name used to identify the data source used in alarm definitions or analysis software. Data source names must be unique. The maximum length for the *datasource_name* is 64 characters.
- **logfile** is a keyword. *logfile_set* is the fully qualified name that identifies the log file set. It can be a raw log file set created by `scopent`, an extracted log file created by the `extract` task, or a DSI log file set. If you specify a log file path name that contains embedded blanks, you must place double quotes (") around the path name.

When specifying a `scopent` log file set, use only the `logglob` file name. You do not need to specify other raw log file names because they are accessed as a single log file set.

The same applies when specifying a DSI log file set. Specify only the name of the DSI root file. You do not need to specify any of the other files in the DSI log file set.

Configuring Data Sources from Remote Locations

The universal naming convention (UNC) is required when you specify a log file set that resides on a network share. At system start-up, the Performance Agent service is started automatically, and drive mappings for remotely connected file systems are not established until the user logs on. Therefore, any data source that uses a drive-mapped name to reference a log file on a remote system causes `Coda` and/or `perflbd` to generate an invalid data source error. If you start the Performance Agent service *after* logging on, the data source is processed because the drive mappings are now established.

The Performance Agent service is set up (by default) to run under the System Account, which can be an issue because of access privileges to data source files. The `perflbd` process, running under the System Account, may not have access to the files that are specified as data sources. There are two ways to fix this problem:

- Give all users access to the log files so they can be accessed by `perflbd` running under the System Account.

- Set up the Performance Agent service to run in an account that has access to needed log files, thereby letting `perflbd` access those files. You can do this in the Startup screen on the Windows NT Services control panel applet (the Services applet is accessible on Windows 2000 and Windows XP from Administrative Tools in the Control Panel).

Here are three examples of data source entries:

Example 1:

The following example shows the default SCOPE data source residing on the default
<*disk drive*>`:\Program Files\HP\HP BTO Software\data\datafiles\`directory.

```
datasource=SCOPE logfile="C:\Program Files\HP\HP BTO
Software\data\datafiles\logglob"
```

Example 2:

In the following example, the universal naming convention (UNC) is used to specify a log file set that resides on a network share.

```
datasource=RXLOG logfile=\\lab_sys\my_share\rxlog
```

Example 3:

The following example shows the SCOPE data source residing in a directory whose path name contains an embedded blank.

```
datasource=SCOPE logfile="C:\Program Files\HP\HP BTO Software\/
data\donna test\logglob"
```

To Configure Data Sources

To configure data sources, perform the following step:

1 Click **Data Sources** from the Configure menu on the Performance Agent main window.

  The Configure Data Sources dialog box appears listing the current data source entries in the `perflbd.mwc` file. Each entry represents a single data source.

To modify the log file set name in a data source

To modify, perform the following steps:

1. Select the data source in the Data Sources list.
2. Click the **Log File Set Name** box, modify the log file set name, and click the **Set** button.

To add a new data source

To add a new data source, perform the following steps:

1. Click the **Data Source Name** box and enter a new name.
2. Click the **Log File Set Name** box, enter a new fully qualified log file set name, and click the **Set** button.

   Or,

3. Click the **Browse** button to select an existing data source.

To delete a data source

To delete, perform the following steps:

1. Select the data source in the Data Sources list.
2. Click the **Delete** button
3. When you finish configuring your data sources file, click **OK**.

To activate the changes:

Before proceeding with another task, you *must* activate any changes you made to the data sources. Perform the following steps:

1. Choose **Start/Stop** from the Agent menu on the Performance Agent main window to open the MeasureWare Services window.
2. Click the **Stop Services** button to stop MeasureWare services.
3. When the **Stop Services** button appears dimmed, click the **Start Services** button.
4. Click the **Close** button to return to the main window.

For step-by-step instructions for modifying the data sources file, choose **Help Topics** from the Help menu, select "**How Do I…?**," and then select "**Modify a data source file**."

Configuring Transactions

You use the transaction configuration file, `ttdconf.mwc`, to customize collection of transaction data for an application. The file defines the transaction name, performance distribution range, and the service level objective you want to meet for each transaction. Optionally, you can define transactions that are specific to an application.

The default `ttdconf.mwc` file contains three entries. Two entries define transactions used by the Performance Agent `scopent` collector, and a third entry, `tran=*` registers all transactions in applications that were instrumented with Application Response Measurement (ARM) API function calls.

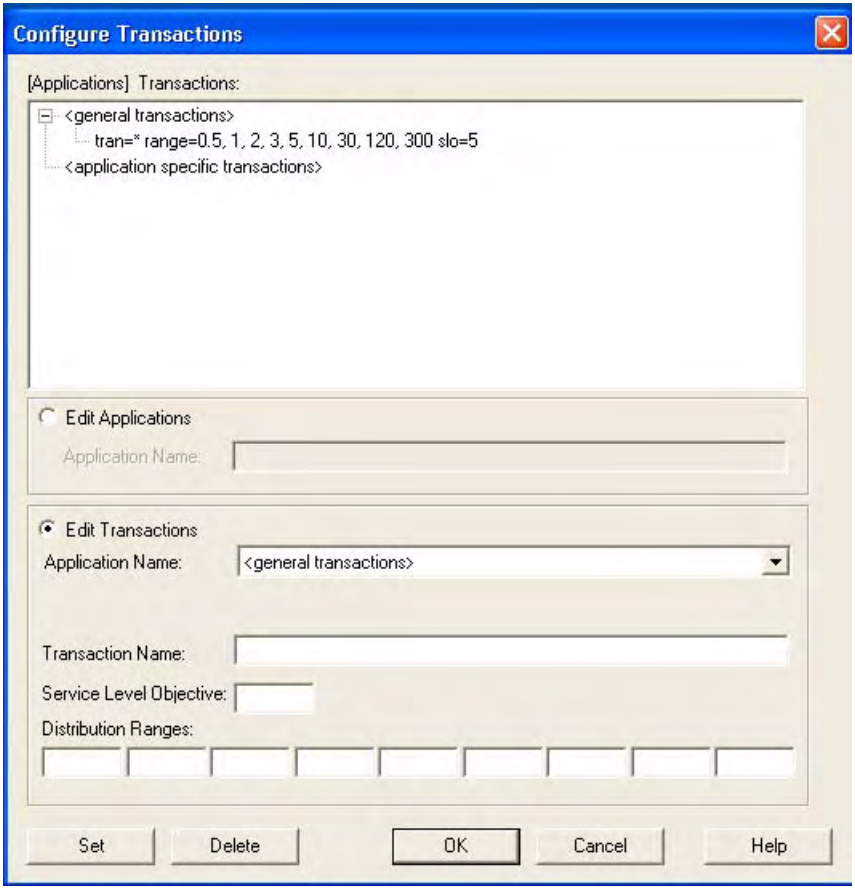


Figure 17 Configure Transactions Dialog Box

If you are adding new applications to your system that use the service level objective and range values from the `tran=*` entry in the default `ttdconf.mwc` file, you do not have to do anything to incorporate the new transactions. All of the default values are applied automatically to them.

However, if you are adding applications to your system that have transactions with their own unique service level objectives and distribution range values, you must add these transactions to the `ttdconf.mwc` file.

The order of the entries in the `ttdconf.mwc` file is not relevant. Exact matches are sought first. If none are found, the longest match with a trailing asterisk (*) is used.

To Configure Transactions

Before you make any changes to the file, see Chapter 2 of your *HP Performance Agent for Windows: Tracking Your Transactions* manual for descriptions of the configuration file format, transaction and application names, performance distribution ranges, and service level objectives. To configure, perform the following steps:

1. Click **Transactions** from the Configure menu on the Performance Agent main window to display the Configure Transactions dialog box. Using this dialog box, you can perform the following tasks:
   a. Add a general transaction
   b. Add an application-specific transaction
   c. Modify a transaction's performance distribution range or service level objective
   d. Delete a transaction

For step-by-step instructions for performing these tasks, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Configure transactions**."

Configuring Persistent DSI Collections

Use the **Persistent DSI Collections** command from the Configure menu to check the syntax or modify the DSI configuration file, `dsiconf.mwc`. The `dsiconf.mwc` file is used to configure continuous logging of data collections that were brought into Performance Agent from outside sources. For more information, see the *HP Performance Agent for Windows: Data Source Integration Guide*.

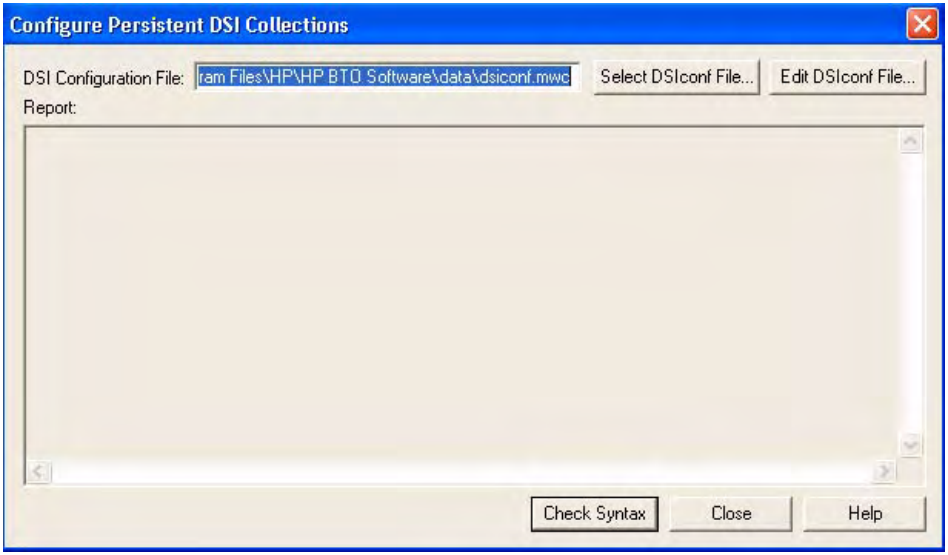


Figure 18 Configure Persistent DSI Collections Dialog Box

To Check the Syntax of the DSI Configuration File

To check, perform the following steps:

1. Click **Persistent DSI Collections** from the Configure menu on the Performance Agent main window. The Configure Persistent DSI Collections dialog box shows the name of the currently open `dsiconf.mwc` file.
2. To check a different `dsiconf.mwc` file, click the **Select DSIconf File** button.
3. To check the syntax of the file, click the **Check Syntax** button. Any resulting warnings or errors are displayed in the Performance Agent Report Viewer window.

4 To modify any portion of the file, click the **Edit DSIconf File** button (see the next section, To Modify a DSI Configuration File). You can position the Edit DSIconf File and the Configure Persistent DSI Collections dialog boxes on your screen so that you can use both at the same time.

For step-by-step instructions for checking the syntax of the DSI configuration file, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Check the syntax of a DSI configuration file**."

To Modify a DSI Configuration File

To modify, perform the following steps:

1 Click **Persistent DSI Collections** from the Configure menu on the Performance Agent main window and then click the **Edit DSIconf File** button in the Configure Persistent DSI Collections dialog box. The contents of the currently open `dsiconf.mwc` file are displayed in a previously specified editor or word processor. (To specify an editor or word processor, see Configuring User Options on page 79.)

2 Before you make any changes to the file, see "Configure the DSI Service" in Chapter 4 of the *HP Performance Agent for Windows: Data Source Integration Guide* for rules and conventions to follow.

3 Modify the file as necessary and save the file in text format.

To activate the changes:

Before proceeding with another task, you *must* activate any changes you made to the `dsiconf.mwc` file. Perform the following steps:

1 Click **Start/Stop** from the Agent menu on the Performance Agent main window to open the MeasureWare Services window.

2 Select the **Persistent DSI Collections** check box.

3 Click the **Refresh** button.

4 Click the **Close** button to return to the main window.

For step-by-step instructions for modifying a DSI configuration file, choose **Help Topics** from the Help menu, select "**How Do I...?,**" and then select "**Modify a DSI configuration file**."

If you use WordPad, Notepad, or Microsoft Word to modify your `dsiconf.mwc` file and then use the **Save As** command to save the file, the `default .txt` extension will automatically be added to the file name. You will then have a file named `dsiconf.mwc.txt`. To retain the `dsiconf.mwc` file name, use the **Save As** command to save your file as a text file and enclose the file name in double quotes ("). For example: "`dsiconf.mwc`"

Checking Performance Agent Status

Use the **Status command** from the Agent menu to review the current status of Performance Agent processes. The information is generated by the `perfstat` program.

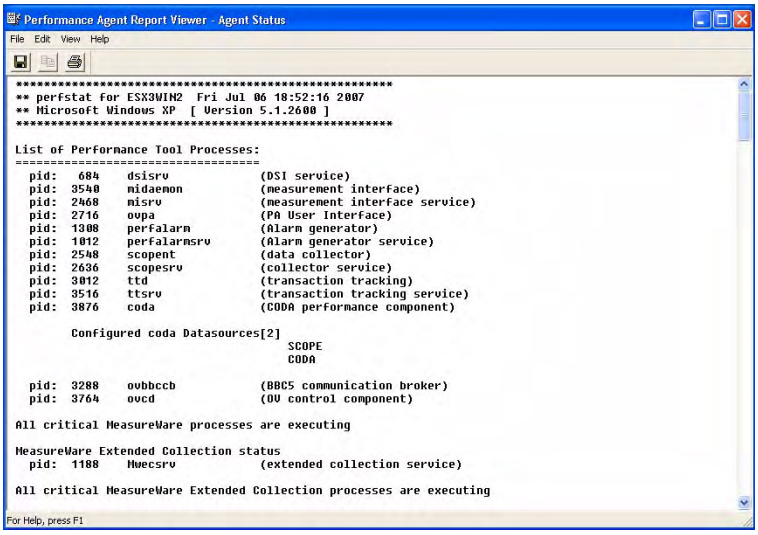


Figure 19 HP Performance Agent Status Window

You can designate which specific information to include in the status report by choosing the **Options** command from the Configure menu display and selecting any of the following options in the Configure Options dialog box.

Running Processes

Background and foreground processes that are currently running for Performance Agent (and if installed, Performance Manager) are listed. Any background processes that should be running but are *not* running are listed.

Datacomm Services

Datacomm services locate and communicate with the Performance Agent datacomm services. They show whether or not the alarm generator data base server (`agdbserver`) process is running and responsive. They list each

repository server, the name of its data source, and the name of the log file set being used. If Performance Manager is installed, the Performance Manager alarm services that are running are listed.

If data communications are not enabled , this information may take more than 30 seconds to generate while it waits for datacomm services to respond.

System Services

The current status of Performance Agent System Services such as the MeasureWare Collector, MeasureWare Agent, Transaction Manager, and Measurement Interface is shown.

System Configuration

System name, operating system version, and processor type.

File Version Numbers

Version numbers of Performance Agent (and if installed, Performance Manager) files. Any critical files that are missing are noted.

Status File Latest Entries

The latest few entries from each performance tool status file.

Status File Warnings and Errors

Any lines from the performance tool status files that contain "Error" or "Warning" are listed. A very large listing can be produced in cases where warnings have been ignored for long periods of time.

To list current status:

To list the current status, perform the following step:

1. Click **Status** from the Agent menu on the Performance Agent main window. The Performance Agent Report Viewer displays the information you selected from the Configure Options dialog box.

To get a complete report of all status information

Perform the following step:

1. Click **Report** from the Agent menu. The Performance Agent Report Viewer displays a complete list of all status information.

For step-by-step instructions for checking Performance Agent status, choose **Help Topics** from the Help menu, select "**How Do I...?**," and then select "**Check status of OV Performance Agent processes**."

You can also run the `perfstat` program from the Windows Command Prompt.

Building Collections of Performance Counters

Performance Agent provides access to Windows performance counters that are used to measure system, application, or device performance on your system. You use the Extended Collection Builder and Manager (ECBM) to select specific performance counters to build data collections.

Building a Performance Counter Collection

To build a collection, choose **Extended Collections** from the Agent menu on the Performance Agent main window. The Extended Collection Builder and Manager window appears, showing a list of Windows objects in the left pane. For instructions on building collections, choose **Help Topics** from the Help menu in the Extended Collection Builder and Manager window.

After you build your collections of Windows performance counters, use the Extended Collection Manager pane at the bottom to register, start, and stop new and existing collections.

Managing a Performance Counter Collection

To manage your data collections, use the Extended Collection Manager pane at the bottom of the Extended Collection Builder and Manager. Initially, no collections appear because you must register a collection before you can start collecting data.

After you register or store the collection you created, the Extended Collection Manager pane shows a list of current collections. The Extended Collection Manager pane also displays the state of every collection and enables you to view information (properties) about the collection itself. For instructions on managing your collections, choose **Help Topics** from the **Help** menu in the Extended Collection Builder and Manager window.

Tips for Using Extended Collection Builder and Manager

- The `<datadir>\paperdocs\mwa\C\monxref.txt` file contains a cross-reference of Performance Agent metrics to Windows performance counters and commands. Logging data through the Extended Collection Builder and Manager for metrics already collected by Performance Agent incurs additional system overhead.
- When you use the Extended Collection Builder to create collections, default metric names are assigned to Windows performance counters for internal use with Performance Manager and Performance Agent. These default names are generally not meaningful or easy to decipher. To make metric names more meaningful or match them to the metric names supplied by their source application, modify metric attributes by right-clicking or double clicking the metric name after you drag it from left to right pane in the Extended Collection Builder and Manager window. (See the Extended Collection Builder and Manager online help for detailed instructions.)
- If you start 61 or more collections, the collections beyond 60 go into error states. This may cause problems with other collections.
- If you collect logical disk metrics from a system configured with **Wolfpack**, you must restart the collection in order to collect data for any new disk instances not present when the collection was registered.
- Successful deletion of collections requires restarting Performance Agent after deleting the collection. If Performance Agent is not restarted, you might get an error during the delete operation. This error typically means that some files were not successfully deleted. You may need to manually delete any files and directories that remain after you restart Performance Agent.
- Extended Collection Builder and Manager may report missing values for some metrics with cache counters. The problem may occur under some circumstances when a metric value gets an overflow. A message is also sent to the ECBM status file. You can resolve the problem by restarting the collection.
- The following are tips for displaying Extended Collection Builder and Manager metrics in Performance Manager:
  - Performance Manager for Windows cannot graph data or metric values greater than 2 million. A graph with a line segment where either end point exceeds 2 hundred million will be inaccurate. For each

value that exceeds 1 billion, you will see a "bad data value" warning and you must select OK on each. To see the accurate data values, use the "show" dropdown menu from the graph, and then "drill down" to "numbers".

The character limit of `system:datasource:class:metric` in an on-screen graph is 80 to 90 characters, depending on font and character width. Printing from this graph displays what you see on the screen.

— PerfView for UNIX (now known as Performance Manager for UNIX) cannot display values greater than 1 billion. A line segment will not be graphed if at least one of the two end points contains a data value that is greater than 1 billion. The raw data values can be viewed using the "drill down" pulldown menu.

The limit of `system:datasource:class:metric` in an on-screen graph is about 110 characters depending on font and character width.

The limit of `system:datasource:class:metric` on a printed line graph is 126 characters. (Stacked and pie graphs do not handle long metric names.)

Explanations of Extended Collection Builder and Manager concepts, and instructions on creating and viewing data collections are available in the Extended Collection Builder and Manager online help. To view online help, from your desktop select **Start** → **Programs** → **HP** → **Performance Agent** → **ECB-ECM Online Help**. You can select **Extended Collections** from the Agent menu in the Performance Agent main window and select **Help Topics** from the Help menu in the Extended Collection Builder and Manager window. Online help is available by selecting the **Help** button in the dialog boxes that appear in the Extended Collection Builder and Manager.

Administering ECBM from the Command line

You can run the ECBM program from the <rpmtools>\bin directory using the Windows Command prompt.

Collections can be managed from the command line using the following command:

```
\rpmtools\bin\mwcmcmd.exe
```

To display various options, type the following command:

```
\rpmtools\bin\mwcmcmd /?
```

To start the stopped collections, type the following command:

```
mwcmcmd start <collection_name(s)>
```

To start a new collection from a variable-instance policy, type the following command:

```
mwcmcmd start <policy_name> <collect_name> <instance(s)>
[options]
```

The following options are available:

```
-i <sampling_interval>  -  change sampling interval (seconds)

-l <logfile_path_name>  - change default log location

-a <alarm_file>         - change the alarm definitions file
```

To stop the active collections, type the following command:

```
mwcmcmd stop <collection_name(s)>
```

The following is the command to register a policy file:

```
mwcmcmd register <policy_file> <collection/policy_name>
[options]
```

The following options are available only when registering a fixed-instance policy file:

```
-i <sampling_interval>   - change sampling interval (seconds)

-l <logfile_path_name>   - change default log location

-a <alarm_file>          - change the alarm definitions file
```

To delete a single collection:

```
mwcmcmd delete <collection/policy_name> [options]
```

The following options are only available when deleting a collection:

```
-p <archive_path>    - archives logfiles to specified path
-r                   - restarts Performance agent
```

To delete multiple collections or policies:

```
mwcmcmd delete { <collection/policy_name(s)> | -c | -all }
   -c      - deletes ALL collections
   -a      - deletes ALL collections and policies
```

When deleting more than one policy/collection at a time, Performance Agent will be automatically restarted, and all associated logfiles will be deleted.

To list all registered collections and policies, type the following command:

```
mwcmcmd list
```

To list the properties of a collection or policy, type the following command:

```
mwcmcmd properties <collection/policy_name>
```

To list the variable-instance objects in a policy, type:

```
mwcmcmd objects <policy_name>
```

For best results, make sure the system is running Performance Agent C.03.30 and above version. The previous version of Performance Agent had problems registering multi-instance collections from the command-line, although the graphical interface was unaffected.

4 Utility Program

Run the Performance Agent `utility` program from the `<rpmtools>\bin\` directory using the Windows Command Prompt.

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

- Scan raw or extracted `scopent` log files and produce a report showing:
  - Dates and times covered
  - Times when the `scopent` collector was not running
  - Changes in `scopent` parameter settings
  - Changes in system configuration
  - Log file disk space
  - Effects of application and process settings in the parm file
- Resize raw log files
- Check the `parm` file for syntax warnings or errors
- Check the `alarmdef` file for syntax warnings or errors
- Check the `dsiconf` file for syntax warnings or errors
- Process log file data against alarm definitions to detect alarm conditions in historical data

This chapter provides instruction for the following:

- Running the Utility Program
- Using Interactive Mode
- Using the Utility Command Prompt Interface
- Utility Scan Report Details and Scan Report Information

For a detailed description of the `utility` commands, see Chapter 5, Utility Commands.

Examples of `utility` tasks can be found in online help for the `utility` program.

Running the Utility Program

There are three ways to run the `utility` program in your Windows Command Prompt window:

- Command Prompt – You control the program using command options and arguments in the Command Prompt.
- Interactive mode – You supply interactive commands and parameters while executing the program with `stdin` set to the Windows Command Prompt window.

  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. 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 commands and parameters.

The syntax for the Command Prompt interface is described in detail in this chapter.

Interactive and batch modes use the same command syntax. 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 parameters (for which there are no defaults) and optional parameters (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 parameter and enables you to 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 Prompt and batch mode. You can supply additional data or correct mistakes in interactive mode, but not in Command Prompt and batch modes.

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

```
checkdef c:\rpmtoools\data\alarmdef.mwc
detail off
start today-1
analyze
```

The checkdef command checks the alarm definitions syntax in the alarmdef.mwc 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.mwc file.

Example of Using Interactive and Batch Modes

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 enables you to 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 is the following:

- Batch users: The batch job terminates because the `logfile` parameter has no default.
- Interactive users: You are 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 is the following:

- Batch users: Resizing continues because `yes` is the default action.
- Interactive users: You are prompted to supply the action before resizing takes place.

Utility Command Prompt Interface

In addition to the interactive and batch mode command syntax, parameters and commands can be passed to the `utility` program through the Windows Command Prompt interface. The Command Prompt interface fits into the environment by enabling the `utility` program to be easily invoked by batch files and enabling its input and output to be redirected to files.

For example, to use the Command Prompt equivalent of the resizing example shown in the previous section, enter:

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

Command Prompt options and arguments are listed in the following table. The referenced command descriptions can be found in Chapter 5, Utility Commands.

Table 2 Command Prompt Arguments

Command Option	Argument		Description
`-b`	date	time	Sets starting date and time (See the command description for start on page 156.)
`-e`	date	time	Sets ending date and time. (See the command description for stop on page 158.)
`-l`	logfile		Specifies input logfile. (See the command description for logfile on page 140.)
`-f`	listfile		Specifies output listing file. (See the command description for list on page 139.)
`-D`			Enables detail for analyze, scan, and parm file checking. (See the command description for detail on page 133.)
`-d`			Disables detail for analyze, scan, and parm file checking. (See the command description for detail on page 133.)

Table 2 Command Prompt Arguments

Command Option	Argument		Description
`-v`			Echoes Command Prompt commands as they are executed.
`-xd`	`dsicheck`		Syntax checks the `dsiconf.mwc` file. (See the command description for dsicheck on page 134.)
`-xp`	`parmfile`		Syntax checks a `parm` file. (See the command description for parmfile on page 143.)
`-xc`	`alarmdef`		Syntax checks `alarmdef.mwc` and sets its file name to use with `-xa` (or analyze command). (See the command description for checkdef on page 132.)
`-xa`			Analyzes the log files using the `alarmdef.mwc` file. (See the command description for analyze on page 130.)
`-xs`	logfile		Scans the log files and produces a report. (See the command description for scan on page 151.)
`-xr`	global application process device transaction EMPTY=*nnn* SPACE=*nnn*	SIZE=*nnn* DAYS=*nnn* YES NO MAYBE	Resizes a log file. (See the command description for resize on page 145.)
`-?` or `?`			Prints Command Prompt syntax.

Example of Using the Command Prompt Interface

The following situation applies while executing command options and arguments entered in the Command Prompt:

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 Prompt.

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

This has the following meaning:

`-xp`	Syntax checks the default `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 `scopent` log file and writes a report on its contents. The report's contents depend on the commands issued before issuing the `scan` command. (For more information, see the description of scan on page 151.)

Table 3 summarizes the information contained in all scan reports and in reports that are produced only when the `detail on` command (the default) is used with the `scan` command.

Table 3 Information Contained in Scan Reports

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.

Table 3 Information Contained in Scan Reports

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 consists of three levels:

- 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 contents 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 following example shows a portion of a report listing the contents of the `parm` file.

```
06/03/00 15:28 System ID="89120HOM"
SCOPENT/ X.10.27.42 SAMPLE INTERVAL = 300,300,60 Seconds, Log
version=D
Configuration: 80686 O/S NT 4.0, SvcPk 3 CPUs=1
Logging Global(NoDisk) Application Process Transaction
Device= Disk records

Thresholds: CPU= 10.00%, Disk=10.0/sec,
    First=5.0 sec, Resp=30.0 sec, Trans=100
    Nonew=FALSE, Nokilled=FALSE, Shortlived=FALSE (<1 sec)
    ProcMem=500
```

```
Memory:  Physical = 64948 KB, Swap = 142096 KB,
    Available to users = 51960 KB

Data collected on 0 NETIF interfaces:

06/03/00 15:28 Data collected on 1 Disk Devices:
```

The date and time listed on the first line correspond to the *first date and time* in the global log file and indicate when `scopent` 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 there is 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:

```
06/01/00 08:39 Application(1) = "other"
 Priority range = 0 - 31
 Comment=all processes not in user-defined applications

06/01/00 08:39 Application(2) = "System"
 Priority range = 15-31
 File=SYSTEM,SMSSS,WINLOGON,LSASS,SPOOLSS,NTVDM,EVENTLOG
 File=RPCSS,PROGMAN,SERVICES,CONTROL

06/01/00 08:39 Application(3) = "DeskTop"
 Priority range = 0 - 31
 File=123*,WINWORD*,EXCEL*,MSACCESS*,WINHLP*,INFOVIEW,CMD
```

During the scan, you are notified of applications that were added or deleted. This decision is based entirely on the application name. (Additions and deletions are determined by comparing the spelling and case of the old names to the new set of logged 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 `scopent` 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
- `scopent` 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 `scopent` started.

The following example shows the change notifications that occur when two new disk drives are added to the system.

```
03/13/99 17:30 The number of disk drives changed from 2 to 4
03/13/99 17:30  New disk device Disk #52 = "3"
03/13/99 17:30  New disk device Disk #54 = "4"
```

parm File Application Addition/Deletion Notifications

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

User-defined applications can be added or deleted each time `scopent` 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 was not changed, it is not printed.

The following example shows that a new application was started.

```
03/13/00 17:30 Application 4 Accounting progs were added
```

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.

scopent Off-Time Notifications

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

If an extracted file contains only summary information, times are rounded to the nearest hour. For example:

```
05/03/00 12:02 - 05/03/00 16:32 collector off (04:30)
```

The first date and time (`05/03/00 12:02`) indicate the last valid data record in the log file before `scopent` was restarted. The second date and time (`05/03/00 16:32`) indicate when `scopent` was restarted.

The last field (in parentheses) shows how long `scopent` was *not* running. The format is *ddd*/*hh:mm:ss*, where *ddd* are days and *hh:mm:ss* are hours, minutes, and seconds. Zeros to the left are deleted.

In this example, `scopent` was off on May 3, 2000, between 12:02 pm and 4:32 pm. The summary information shows that data was not collected for four hours, 30 minutes, and four seconds.

Application-Specific Summary Report

To obtain this report, use the `scan` command with its default `detail on` and there is 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 enable 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 CPU	PERCENT OF DISK	TOTAL TRANS
other	7	0.7%	0.0%	100.0%
System	7	47.5%	71.6%	0.0%
DeskTop	6	51.8%	28.4%	0.0%
ScreenSaver	0	0.0%	0.0%	0.0%
All user applications	65.0%	99.3%	100.0%	0.0%

Summaries

This section describes the following summaries:

- Process log reason summary
- Scan start and stop actual dates and times
- Application overall summary
- `scopent` 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 `scopent`. The report lists every reason why 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: 05/03/00 3:53 PM to 05/03/00 4:38 PM
There were 0.6 hours of process data
Process records were logged for the following reasons:

Log Reason      Records   Percent   Recs/hr
--------------- -------   -------   -------
New Processes        29     58.0%      46.9
Killed Processes     25     50.0%      40.4
CPU Threshold        11     22.0%      17.8
```

A process can be logged for more than one reason at a time. Record counts and percentages do 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 `scopent` 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 Summary

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         03/03/00 12:40 PM
Scan stopped on         03/11/00  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. 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 `scopent` is being used to capture system activity. If the percentage of time `scopent` was off is high, as in the following example, you should review your operational procedures for starting and stopping `scopent`.

```
The total time covered was       :      40:11 out of 44.41
Time lost when collector was off:      04.30  10.07%
The scope collector was started :          2 times
```

This report is more complete if global detail data is included in the scan. If only summary data is available, you determine the time `scopent` 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 is the total time less the covered time.

The formats for the three times mentioned are:

ddd/hh:mm:ss

In this instance, *ddd* are days and *hh:mm:ss* are hours, minutes, and 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 MegaBytes  Records MegaBytes  Start  Finish   Days

Global  4       0,00    314,5     0,072 01.08.02 to 01.08.02   0,0

Application 7   0,00    604,2     0,075 01.08.02 to 01.08.02   0,0

Process  1      0,00   1464,4     0,281 01.08.02 to 01.08.02   0,0
```

```
Disk      5      0,00    432,0     0,020 01.08.02 to 01.08.02   0,0
NETIF  8       0,00    691,2     0,030 01.08.02 to 01.08.02   0,0
CPU    4       0,00    345,6     0,039 01.08.02 to 01.08.02   0,0
Filesystem  16  0,00  1382,4
 0,171 01.08.02 to 01.08.02   0,0
Tran   8       0,00    576,0     0,300 01.08.02 to 01.08.02   0,0
Overhead              0,03
      ------- ---------  ------- ---------
TOTAL  53       0,04   5810,3     0,988
```

The columns consist of the following:

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 `scopent` runs.

Column	Explanation
`Dates`	The first and last valid dates for the data records of each data type scanned.
`Full Days`	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 `scopent` 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 2898 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 on page 123). 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.

5 Utility Commands

Introduction

This chapter contains the following:

- Utility Commands and Summary of `utility` Command Syntax and Parameters
- Command Descriptions where the commands are listed in alphabetical order

Utility Commands

`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 to `log`.

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

Table 4 Summary of `utility` Command Syntax and Parameters

Command	Parameter
`analyze`	
`checkdef`	`alarmdef.mwc` file
`detail`	`on` `off`
`dsicheck`	`dsiconf.mwc` file
`exit` `e`	
`guide`	
`licheck`	
`list`	*filename* or `*`
`logfile`	*logfile*
`menu` `?`	
`parmfile`	*parmfile*
`quit` `q`	

Table 4 Summary of `utility` Command Syntax and Parameters

Command	Parameter

`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`
`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*]

Command Descriptions

Listed below are the descriptions for the commands. The commands are listed in alphabetical order:

analyze

Use the `analyze` command to analyze the data in a log file against alarm definitions in an alarm definitions file and report resulting alarm status and activity. Before issuing the `analyze` command, 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 Prompt mode, (`-xa`), the default alarm definitions file (`alarmdef.mwc`) file is used.

For detailed information about alarm definitions, see Chapter 8, Performance Alarms.

Syntax

```
analyze [\directorypath\logfile]
```

How to Use It

When you issue the `analyze` command, it analyzes the log files in the default SCOPE data source against the alarm definitions in the `alarmdef.mwc` file.

To analyze a specific log file, you can override the default SCOPE data source. Place a USE statement in your alarm definitions that specifies the name of the data source containing that log file. For more information on how to do this, see the description of the USE Statement on page 271.

The `analyze` command enables you to evaluate whether or not your alarm definitions are a good match against the historical data collected on your system. It also enables you to 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 8, Performance Alarms.

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 are on page 156 and page 158 respectively.)

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. SYMPTOMS are not evaluated. 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.

To see the alarm summary report only, issue the `detail off` command.

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 8, Performance Alarms.

Syntax

`checkdef [\directorypath\alarmdef.mwc]`

Parameters

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

How To Use It

When you determine 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.mwc` file is used.

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

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 Prompt 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, `scopeNT` 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 page 130, page 143, and page 151 respectively.

dsicheck

Use the `dsicheck` command to check the syntax of the data source integration (DSI) logging configurations in the `dsiconf.mwc` file and report any warnings or errors that are found. The `dsiconf.mwc` file is used by the Performance Agent DSI service for continuous logging of data collections that have been brought into Performance Agent from an outside source.

Syntax

`dsicheck [\directorypath\dsiconf.mwc]`

Parameters

`dsiconf.mwc`	The name of any DSI configuration file. This can be a user-specified file or the default `<rpmtools>\data\dsiconf.mwc` file. If no directory path is specified, the current directory is searched.

How To Use It

When you issue the `dsicheck` command, it checks the `dsiconf.mwc` file for the entries that are in each DSI data collection to be logged. The datafeed, logfile, and class entries are required; the `dsiparms` entry is optional. All keywords must be in upper case. Use a semi-colon to separate each DSI collection in the `dsiconf.mwc` file.

For descriptions of the DSI logging configuration, see "Configuration Files" in Chapter 4 of the *Performance Agent for Windows: Data Source Integration Guide*.

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`

How to Use It

- To enter guided commands mode from `utility`'s interactive mode, type **`guide`** and press **Enter**.
- To accept the default value for a parameter, press **Enter**.
- 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.

licheck

Use this command to check the validity of the product license. If the license is valid, it can be a trial version or a permanent version.

Syntax

licheck

Example

To check the validity of the license, type the command:

utility -licheck

If the license is valid and permanent, the following message appears:

```
The permanent OVPA software has been installed
```

help

Use the `help` command to access online help.

Syntax

`help` **`[`***keyword***`]`**

How to Use It

This command opens the comprehensive online help topics for the `utility` program as well as other Performance Agent components. Use the help contents and index to locate the information you need.

list

Use the `list` command to specify the list file for all `utility` reports. The contents of the report listed depend 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`***`]`**

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`
- 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 on 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 is 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.

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 Prompt mode and do not provide a log file name, the default `C:\rpmtools\data\datafiles\logglob` file is used. If you are in interactive mode and do not provide a log file name, you are prompted to provide one.

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`.

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
`logls`	logical systems 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 on page 154.

You can open another log file at any time by entering another logfile command. Any currently active 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.

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

If you must have more than one set of raw log files on the same system, create a separate directory for each set of files. Although the log file names cannot be changed, different directories may be used. 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

```
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
DSICHECK [dsiconf]    Check parsing on a dsi configuration 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
```

parmfile

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

Syntax

`parmfile [`*`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 default `parm.mwc` file is used.

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

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 `scopeNT` 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. 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 `scopeNT` collector because resizing can occur any time the file is filled. Using `utility` 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 according to 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. 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 page 150 for an explanation of how to use the `resize` command in this manner.

Default resizing parameters are shown in the following table.

Table 5

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 `scopeNT` using the steps under Starting and Stopping Data Collection on page 43.

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 `C:\temp\scopelog` before deleting the original file. Make sure there is sufficient disk space in the datafiles directory 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 c:\rpmtools\data\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 `scopeNT` to resize them). You could schedule a batch file 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 batch file 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
```

If you use the above batch file, remember to stop `scopeNT` before running it, using the steps under Starting and Stopping Data Collection on page 43. 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 batch file.

scan

Use the `scan` command to read a `scopeNT` log file and write a report on its contents. (For a detailed description of the report, see Utility Scan Report Details on page 114.)

Syntax

```
scan [\directorypath\logfile]
```

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 (`logglob`). 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 report listing 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 see detail, list, start, and stop commands.

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

The following 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

sh

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

Syntax

`sh or !` **`[`***system command*`]`**

Parameters

`sh` *system command*	Executes the *system command* and returns to `utility`. The *system command* is any system command.
`!` *system command*	Same as above.

How to Use It

Following the execution of the single command, you automatically return to `utility`. To issue multiple system commands without returning to `utility` after each one, you can start a new command interpreter by issuing the command:

`sh cmd`

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

The default starting date & time = 05/05/00 11:50 AM (FIRST + 0)
The default stopping date & time = 05/10/00 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM
```

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

Use **`show all`** to produce a more detailed list, as shown below:

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

The default starting date & time = 05/01/00 11:50 AM (FIRST + 0)
The default stopping date & time = 05/10/00 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM

    Global      file: c:\RPMTOOLS\data\datafiles\logglob
    Application file: c:\RPMTOOLS\data\datafiles\logappl
    Process     file: c:\RPMTOOLS\data\datafiles\logproc
    Index       file: c:\RPMTOOLS\data\datafiles\logindx
    System ID:  ros35254Ssch
    System Type 80686 O/S NT 4.0
    Data Collector: NT X.00.06
```

```
    File created:   05/05/00
    Data Covers:    6 days to 05/10/00
    Shift is:       All Day

Data records available are:
   Global Application Process

Maximum file sizes:
   Global=10.0 Application=10.0 Process=20.0
```

start

Use the `start` command to specify the beginning of the subset of a log file that you want to scan or analyze. `Start` enables you to 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. Otherwise, the default is undefined.

Syntax

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

Parameters

`date`	The starting date for the `scan` or `analyze` process. The date is specified in the system short date format.
`time`	The starting time for the scan or analyze process. The time is specified in the system time format. Twenty-four hour time is accepted in all languages. For example, 23:30 is 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, 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 before 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 before 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.

stop

Use the `stop` command to specify the end of a subset of a log file that you want to scan or analyze. `Stop` enables you to 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. Otherwise, the default is undefined.

Syntax

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

Parameters

`date`	The stopping date for the scan or analyze process. The date is specified in the system short data format.
`time`	The stopping time for the scan or analyze process. The time is specified in the system time format. Twenty-four hour time is accepted in all languages. For example, 23:30 is 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, midnight (12 am) is assumed. A stop 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 before 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 before 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.

6 Extract Program

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

The `extract` and `export` functions copy data from the a log file. No data is removed.

Two types of log files are used by Performance Agent:

- `scopeNT` log files, which contain data collected in Performance Agent by the `scopeNT` collector.
- DSI log files, which contain user-defined data collected by external sources such as applications and databases. The data is subsequently logged by Performance Agent's DSI programs.

Use the `extract` program to perform the following tasks:

- Extract subsets of data from raw `scopeNT` log files into an extracted log file format that is suitable for placing in archives, transport between systems, and analysis by 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 log files into ASCII, binary, datafile, or WK1 (spreadsheet) formats suitable for reporting and analysis, or for importing into spreadsheets or similar analysis packages.

The maximum limits for the extract and export functions are as follows:

- Output file from the extract function can not exceed 3.5 GB.
- Output file from the export function can not exceed 4GB.

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 provides instructions for the following:

- Running the Extract Program
- Interactive Mode
- Extract Command Prompt Interface and Summary of Command Options, Arguments and Descriptions
- Export Function

Running the Extract Program

There are three ways to run the `extract` program from your Windows Command Prompt window:

- Command Prompt — You can control `extract` by using command options and arguments in the Command Prompt.
- Interactive mode Ò You supply interactive commands and parameters while executing the program with `stdin` set to an interactive 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 commands and parameters.

The syntax for the Command Prompt interface is similar to typical Windows Command Prompt interfaces on other programs and is described in detail in this chapter.

The syntax is the same for interactive and batch modes: 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* parameters (for which there are no defaults) and *optional* parameters (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 enables you to 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 Prompt and batch mode, because you can supply additional data or correct mistakes in interactive mode, but not in Command Prompt and batch modes.

Interactive Mode

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

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

```
logfile c:\rpmtools\data\datafiles\logglob
application summary
start 5/15/02
export
```

The **`logfile`** command opens `logglob`, the default global log file. The **`application summary`** command specifies that summarized application data is exported. The **`start`** command specifies that only data logged after 5/15/99 will be exported. The **`export`** command starts the exporting of the data.

Extract Command Prompt Interface

In addition to the interactive and batch mode command syntax, command options and arguments can be passed to the `extract` program through the Windows Command Prompt interface. The Command Prompt interface fits into the typical Windows environment by allowing the `extract` program to be easily invoked by batch files, and enabling its input and output to be redirected to files.

For example, to take commands from a file called `extin` and direct reports to a file called `extout` while directing all error messages to a file called `exterr`, type:

```
extract < extin > extout 2> exterr
```

Command Prompt arguments are listed in the following table. The referenced command descriptions can be found in Chapter 7, Extract Commands.

Examples of Command Prompt arguments and options can be found in the online help for the `extract` program in the Windows Command Prompt.

Table 6 Summary of Command Options, Arguments and Descriptions

Command Option	Argument		Description
`-b`	`date`	`time`	Sets starting date and time. (See the command description for start on page 232.)
`-B`		`UNIX start time`	Sets starting time in UNIX format.
`-e`	`date`	`time`	Sets ending date and time. (See the command description for stop on page 234.)
`-E`		UNIX end time	Sets ending time in UNIX format.
`-s`	`time-time`	`noweekends`	Start time, end time, weekends. (See the command description for shift on page 229.)

Table 6 Summary of Command Options, Arguments and Descriptions

Command Option	Argument		Description
`-l`	logfile		Specifies input log file. (See the command description for logfile on page 216.)
`-r`	export template file		Specifies an export template file for export function. (See the command description for report on page 227.)

`-C`	classname	opt	Specifies `scopeNT` data to extract or export, or DSI data to export. (See the command description for class on page 199.)
			`opt -`
			`detail`
			`summary(export only)`
			`both`

`-i`			Specifies `scopent` to extract or export data from logical systems.

Table 6 Summary of Command Options, Arguments and Descriptions

Command Option	Argument		Description
gapkcdnt GADNT			Specifies types of data to extract/ export: g = global detail. (See the command description for global on page 209.) a = application detail. (See the command description for application on page 197.) p = process detail (See the command description for process on page 225.) k = process killed. (See the command description for process on page 225.) d = disk device detail (See the command description for disk on page 202.) c = configuration detail (See the command description for configuration on page 201.) n = netif detail (See the command description for netif on page 222.) t = transaction detail. (See the command description for transaction on page 236.) G = global summary (See the command description for global on page 209.) A = application summary (See the command description for application on page 197.) D = disk device summary (See the command description for disk on page 202.) N = netif summary (See the command description for netif on page 222 .) T = transaction summary (See the command description for transaction on page 236.)

Table 6 Summary of Command Options, Arguments and Descriptions

Command Option	Argument		Description
-v			Generates verbose output report formats.
-f	filename	,new ,append ,purge	Sends extract or export data to a file. If no filename, sends extract data to rxlog.mwc and export data to xfr*logfile.ext. (See the command description for output on page 223.)
-we	1.....7		Sets days to exclude from export; 1=Sunday. (See the command description for weekdays on page 237.)
-xp	xopt		Exports data to external format files. (See the command description for export on page 204.)
-xt	xopt		Extracts data in system internal format. (See the command description for extract on page 207.)
			xopt = *dwmy* (Day Week Month Year) *dwmy*-[offset] *dwmy* [absolute]
-xw	week		Extracts a calendar week's data. (See the command description for weekly on page 238.)
-xm	month		Extracts a calendar month's data. (See the command description for monthly on page 220.)
-xy	year		Extracts a calendar year's data. (See the command description for yearly on page 240.)
-? Or ?			Displays Command Prompt syntax

When you are evaluating arguments and executing command options entered in the Command Prompt, the following rules apply:

- 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 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 are processed.
- If an action is specified (`-xp`, `-xw`, `-xm`, `-xy`, or `-xt`), the program executes those commands 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 Prompt, `extract` does not read from its `stdin` file; instead it terminates following the action:

  `extract -f rxdata.mwc -r rept1.mwr -xp d-1 -G`

  This translates into:

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

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

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 so all extraneous output to `stdout` is eliminated.

Export Function

The `export` command converts Performance Agent raw, extracted, or DSI log file data into exported files. 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, `<rpmtools>\data\datafiles\logglob`, from which you want to export data, the default export template file, `<rpmtools>\data\reptfile.mwr`, that defines the format of the exported data, and then 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 text format is suitable for printing.

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
`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 sumaries
`logicalsystem`	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 a export template file. This file can be created using any text editor or word processor that enables you to save a file in ASCII format.
- You can also use the default template file, `<rpmtools>\data\reptfile.mwr`, which 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 included
  - A recommended set of metrics for a given data type included in the export

If you are extracting or exporting data from log files that are created from a specific platform, it is recommended that you use the `reptall` file from the same platform. This is because the metric classes supported are different on different platforms.

Sample Export Tasks

Two sample export template files, `repthist.mwr` and `reptall.mwr`, are furnished with Performance Agent. These files are located in the `<rpmtools>\data\` directory. You can use `repthist.mwr` and `reptall.mwr` 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.mwr` export template file contains the specifications for generating 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 commands to generate a graph of the last seven days and produce approximately two pages (34 pages of data should be produced if 5-minute detail is specified instead of hourly summaries).

```
logfile c:\program files\HP\HP BTO Software\data\datafiles\
report c:\program files\HP\HP BTO Software\data\repthist.mwr
global summary
start today-7
export
```

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

`c:\>print` *filename*

Producing a Customized Export Template File.

To create a totally new export template file, make a copy of the export template file `<rpmtools>\data\reptall.mwr` and customize it. The `reptall.mwr` file contains every possible metric for each type of `scopeNT` data so you need only uncomment those metrics that are of interest to you. This is easier than retyping the entire export template file.

You can use the `Guide` command to create a new export template file. In guided mode, you are prompted to copy the `reptall.mwr` file and then specify a `scopeNT` or DSI log file to dynamically create a list of data types and classes and metric names. You can also use the Performance Agent graphical interface to create a new export template file. For more information, see Making a Quick Export Template on page 65.

Export Data Files

If you used the `output` command to specify the name of an output file before issuing the `export` command, all exported `scopeNT` data is appended 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` program's `output stdout` command before issuing the `export` command.

If the output file is set to the default, the exported data is separated into as many as 20 different default output files depending on the type of data being exported. The default export `scopeNT` 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

`xfrdLOGICAL.ext`	Logical system detail data file
`xfrsLOGICAL.ext`	Logical system summary data file

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

If you are exporting DSI log file data, the default file names are created from the class name. The prefix is either `xfrd` or `xfrs`, depending on whether the data is detail or summary data. The extension is either `asc` (ASCII), `bin` (binary), `dat` (data), or `wk1` (spreadsheet).

No output file is created unless you specify the type of data and associated items that match the data in the export template file before 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 section Export File Title on page 179.

`format` Specifies the format for the exported data .

ASCII

ASCII (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 maintains the highest metric accuracy. It is not suitable for direct printing.

WK1 (spreadsheet)

The WK1 (spreadsheet) format is compatible with Lotus 1-2-3, 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 formats 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 to differentiate missing from zero data.

A data item may be missing if it was:

- Not logged by a particular version of the `scopeNT` collector
- Not logged by `scopeNT` because the instance (`application`, `disk`, `transaction`, or `netif`) it belongs to was not active during the interval
- 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`, or `netif`. Single layout writes one record for every application (`disk` , `transaction`, or `netif`) 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 (*disk*, `transaction`, or `netif`)
`output`	Specifies where exported data is to be output. It can be specified for each class or data type it is exporting 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 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`, `netif`, `configuration`, or `class name`. This starts a section of the export template file that lists the data items (metrics) 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 a export template file. For example:

```
data type global
   output=myglobal.mwe
   gbl_cpu_total_util
```

```
data type application
   output=myapp.mwe
   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.

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 class or data type requested.
`!collector`	The name and version of the collector program.
`!system_id`	The identifier of the system that collected the raw or extracted log file data (not valid with DSI log file data).

The following string is an example:

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

This example generates an export file title similar to the following:

```
gemini data from \rpmtools\data\logglob on 02/02/00 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.
2. For this example, you want an ASCII file without headings and with fields separated by commas.

   Create and save the following export template file in the `<rpmtools>\data\` directory. Call it `report1.mwr`.

   ```
   REPORT "sample export template file (report1.mwr)"
   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 <disk\drive>\program
   files\hp\openview\data\report1.mwc
   ```

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 tell the program from where it should get the performance data, you are prompted to do so. In this example, because 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 01/01/00 to
01/31/00

                            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 created — `xfrsGLOBAL.asc` and `xfrsAPPLICATION.asc` — contain the global and application summary data in the specified format.

Output of Exported Files

Exported files are created with the following characteristics:

- Maximum number of bytes in each record: 32000.
- Maximum disks supported in the disk data type fields: 256.
- Maximum LANs supported: 4.
- Maximum applications supported: 32.

The contents of each file are:

export title line	If `export title` and `headings on` were specified.
names (`application`, `netif`, 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	

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 according to their range and internal accuracy. Since all fields will not be the same length, you must 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.

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 endien" 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 "endiens".

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, 1980 (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)
 -9   Netif 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)
```

```
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)

ClassID            Class Data Record  ( 5 minute detail record)
ClassID +1,000,000 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.

`Title Record`	This record (Record ID -1) is written whenever you specify headings on, regardless of whether you specify. 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 that correspond to the exported items.
`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) is only present in transaction tracking data files that use 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.
`Disk Device Name Record`	This record (Record ID -7) is only present in disk device data files that use 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.
`Netif Name Record`	This record (Record ID -9) is only present in netif (LAN) data files that use 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.

Binary Title Record

The Title Record for Binary files contains information designed to assist programmatic interpretation of the exported file's contents. This record is 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. 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 than 4 bytes of space, then its corresponding item ID field is 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 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 that 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 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 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 is repeated with the new transaction name appended to the end of the original list. Transactions that are deleted after the start of the data extraction are not removed from the Multiple Layout data record.

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 is 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 are not 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 is 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 are not removed from the Multiple Layout data record.

7 Extract Commands

This chapter contains the following information:

- Extract Commands and Extract Commands and Parameters
- Commands for Extracting and Exporting Data and Their Log Files
- Command Description

Extract Commands

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 the online help for the `extract` program in the Windows Command Prompt.

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

Table 7 Extract Commands and Parameters

Command	Parameter
application	on detail summary(export only) both off (default)
class	detail (default) summary(export only) both off
configuration	on detail off (default)
disk	on detail summary(export only) both off (default)
exit e	

Table 7 Extract Commands and Parameters

Command	Parameter
export	day[ddd] [-*days*] [ddd][yyddd] week [ww] [-*weeks*] [ww][yyww] month[mm] [-*months*][mm][yymm] year [yy] [-*years*] [yy][yyyy]
extract	day[ddd] [-*days*] [ddd][yyddd] week [ww] [-*weeks*] [ww][yyww] month[mm] [-*months*][mm][yymm] year [yy] [-*years*] [yy][yyyy]
global	on detail (default) summary both off
logical	on detail summary both off (default)
guide	
help	
licheck	
list	filename *
logfile	logfile
menu	
monthly	yyymm mm

Table 7 Extract Commands and Parameters

Command	Parameter
netif	on detail summary both off (default)
output	outputfile ,new ,purgeboth ,append
process	on detail [app#[-#],..] off killed
quit q	
report	[export template file] ,show
shift	starttime - stoptime all day noweekends
sh !	system 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*]

Table 7 Extract Commands and Parameters

Command	Parameter
transaction	on detail summary both off (default)
weekdays	1.....7
weekly	yyww ww
yearly	yyyy yy

The following table lists the commands that are used to extract and export data and the types of log files used (scopeNT log files or DSI log files).

Table 8 Commands for Extracting and Exporting Data and Their Log Files

Command	Extract Data	Export Data	ScopeNT Log Files	DSI Log Files
application	x	x	x	
class	x	x	x	x
configuration		x	x	
disk	x	x	x	
export		x	x	x
extract	x		x	
global	x	x	x	
logfile	x	x	x	x
monthly	x		x	

Table 8 Commands for Extracting and Exporting Data and Their Log Files

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

Command Description

Listed below are the descriptions for the commands. The commands are listed in alphabetical order:

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[summary]
 both]
   [off]
```

Parameters

`on` or `detail`	Specifies that raw, 5-minute detail data should be extracted or exported. When using Performance Manager, detail data must be included in an extracted file before drawing application graphs with points every 5 minutes.
`summary`	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 1hour (`export`) Summarization can significantly reduce the size of the resulting 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 exported.
`off`	Specifies that no application data is to be extracted or exported.

class

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

The default is `class off`.

Syntax

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

Parameters

`classname`	Name of a group of similarly classified metrics.
`detail`	For DSI log files, specifies how much detail data is exported according to the time set in the DSI log file. (For more information, see "The Logging Process" in Chapter 4 of the *HP Performance Agent for Windows NT/2000: Data Source Integration Guide*.) For `scopeNT` log files, specifies that raw, 5-minute detail data should be extracted or exported.

`summary`	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 1 hour (`export`) Summarization can significantly reduce the size of the resulting 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`	Specifies that both detail data and summary data are to be exported.
`off`	Specifies that no DSI data is to be exported.

Examples

To get detail data on a DSI log file that contains a class named `acctg_info`, execute the following command:

```
class acctg_info detail
```

Once the log file is specified by the user and opened by `extract`, 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 detail
cla acctg_info det
```

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

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 `shift`, `start`, `stop`, or `weekdays` commands that are used with the configuration command are ignored.

The configuration command affects only the export function. It does not affect the `extract` function because the `extract` function always extracts system configuration information.

disk

Use the `disk` command to select 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`	Specifies that raw, 5-minute detail data should be extracted or exported.
`summary`	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 1 hour (`export`) Summarization can significantly reduce the size of the resulting exported data, depending on the summarization interval used. For example, hourly summary data is about one-tenth the size of 5-minute detail data. Summarization reduces the size of the disk device data to about one-tenth the size of the detail data.
`both`	Specifies that detail data and summary data are to be exported.
`off`	Specifies that no disk device data is to be extracted or exported.

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 an 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 before 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 has five digits (`99002` means January 2, 1999) and all other parameters would have four digits (`month 9904` means April of 1999).

If you did not previously specify a log file or an export template file, the logfile command uses the default `logglob` file, and the report command uses the default `reptfile.mwr` file.

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

The `export` command creates up to 12 different default output files for `scopeNT` log file data 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
xfrdLOGICAL.ext	Logical system detail data file
xfrsLOGICAL.ext	Logical system summary data file

In this instance, 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 on whether 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:

xfrdNETIF.wk1 contains detailed data for the data type name NETIF in spreadsheet format.

xfrsDISK.asc contains summarized data for the data type name DISK in ASCII format.

When you export DSI log file data, the default files names are created from the class name. The prefix and extensions are the same as for scopeNT data.

For example:

xfrdACCTG_INFO.wk1 contains detailed spreadsheet data for the class name ACCT_INFO

xfrsACCTG_INFO.asc contains summarized ASCII data for the class name ACCT_INFO

For more information about exporting data, see Export Function on page 171.

extract

Use the `extract` command to start the process of copying data into an extracted file format. Extracted files can be used for archiving or for analysis by analyzer programs.

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 before 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 be composed of the last two digits of the year and the absolute interval number in that year. In this format, the absolute day has five digits (`99002` means January 2, 1999) and all other parameters have four digits (`month 9904` 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 = logglob
output file = rxlog.mwe,new
```

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

The size of an extracted log file cannot exceed 3.5GB.

You can extract detail data from a detail extract file and summary data from a summary extract file. However, we do not recommend that you extract detail data from a summary extract file. Doing so results in gaps in the data because the data remains in 1-hour increments.

global

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

The default is `global detail`.

Syntax

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

Parameters

`detail` or `on`	Specifies that raw detail collected at 5-minute intervals is to be extracted or exported. Detail data must be extracted if you want to draw PerfView global graphs with points every 5 minutes.
`summary`	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 1 hour (`export`) Summarization can significantly reduce the size of the resulting 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`	Specifies that detail data and summary data are to be extracted or exported.
`off`	Specifies that no global data is to be extracted or exported.

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

logical

Use the `logical` command to specify the logical system data to be extracted or exported.

The default is `logical off`.

Syntax

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

Parameters

`on` or `detail`	Specifies that raw, 5-minute detail data should be exported.
`summary`	The summary parameter 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` only)
`both`	Specifies that detail data and summary data are to be extracted or exported.
`off`	Specifies that no `logicalsystem` data is to be extracted or exported.

guide

Use the `guide` command to enter guided commands mode. The guided command interface leads you through the various `extract` commands and prompts you to perform 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 **Enter**.
- 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.

If you are exporting DSI log file data, we recommended 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`

How to Use It

This command opens the online help topics for the `extract` program as well as other Performance Agent components. Use the help contents and index to locate the information you need.

licheck

Use this command to check the validity of the product license. If the license is valid, it can be a trial version or a permanent version.

Syntax

licheck

Example

To check the validity of the license, type the command:

extract -licheck

If the license is valid and permanent, the following message appears:

```
The permanent OVPA software has been installed
```

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.

logfile

Use the `logfile` command to open a `scopeNT` 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 name of the default global log file, `<rpmtools>\data\datafiles\logglob`. You can either accept the default or specify a different log file.

Syntax

`logfile [\directorypath\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:

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

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

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.

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
NETIF         [DETAIL|SUMMARY|BOTH|OFF] Extract Logical NETIF
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, the two-digit number of the month being extracted, and the `.mwe` file name extension. For example, data extracted in March 1999 would be output to a file named `rxmo199903.mwe`.

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 the 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.mwe` containing data from May 1 through the current date (May 7).

On June 4, 1999, another `monthly` command is executed. Before the `rxmo199906.mwe` file is created for the current month, the `rxmo199905.mwe` 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.mwe` 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.mwe` (May) and `rxmo199906.mwe` (June) — you can assume that the first file is complete for that month and it can be archived and purged.

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.

netif

Use the netif command to specify the type of netif (logical network device) data to export. Netif data is logged in the logdev file. The default is netif off.

Netif data can be extracted only by using extract Command Prompt arguments in the Windows NT/2000 Command Prompt. For an example of the arguments used, see the online help topic for the extract program in the Windows NT/2000 Command Prompt.

Syntax

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

Parameters

on or detail	Specifies that raw, 5-minute detail data should be exported.
summary	The summary parameter 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 1 hour (export only)
both	Specifies that detail data and summary data are to be extracted or exported.
off	Specifies that no netif data is to be extracted or exported.

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

You must add the `.mwe` file name extension to your output file name. For example, `appout.mwe`.

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.mwe`

For export:

```
xfrdGLOBAL.ext
xfrsGLOBAL.ext
xfrdAPPLICATION.ext
xfrsAPPLICATION.ext
xfrdPROCESS.ext
xfrdDISK.ext
xfrsDISK.ext
xfrdNETIF.ext
xfrsNETIF.ext
xfrdTRANSACTION.ext
xfrsTRANSACTION.ext
xfrdCONFIGURATION.ext
```

In this instance, `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 PC, although this can be redirected using a batch file).

`output stdout`

or

`output *`

To return the output to its default settings, type:

`output default`

or

`output -`

You can override the default output file names for exported files using the output parameter in the export template file.

You cannot 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.

process

Use the `process` command to specify whether or not to extract or export process data. The default is `process on`.

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 specific 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.

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.

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, `C:\rpmtools\data\reptfile.mwr`, 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 enables 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.

The `,show` parameter of the `report` command discussed here is different from the `show` command discussed on page 230.

Export templates can be configured and modified for use only with data logged by Performance Agent for Windows NT/2000. They cannot be used on any other platforms.

sh

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

Syntax

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

Parameters

`sh` *system command*	Executes the *system command* and returns to `extract`. The *system command* is any system command.
`!` *system command*	Same as above.

How to Use It

Following the execution of the single command, you automatically return to `extract`. To issue multiple system commands without returning to `extract` after each one, you can start a new command interpreter by issuing the command:

```
! command
```

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

```
shift [starttime-stoptime]
       [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 that 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.

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]`

The `show` command discussed here is different from the , `show` parameter of the `report` command discussed on page 227.

Examples

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

```
Logfile:      Default

Output:       Default
Report:       Default
List:         "stdout"

The default starting date & time = undefined     (LAST -30)
The default stopping date & time = undefined     (LAST -  0)

GLOBAL           DETAIL           records will be processed
```

Use show all to produce a more detailed list, similar to the example on the next page:

```
Logfile:      Default

Output:       Default
Report:       Default
List:         "stdout"

The default starting date & time = undefined     (LAST -30)
The default stopping date & time = undefined     (LAST -  0)

GLOBAL           DETAIL           records will be processed

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 that is 30 full days before the last date in the log file, or if fewer 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 starting date for the `extract` or `export` function. The date is specified in the system short date format.
`time`	The starting time for the `extract` or `export` function. The time is specified in the system time format. 24-hour time is accepted in all languages. For example, 23:30 is 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 *before* 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 *before* 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)

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

Parameters

`date`	The stopping date for the `extract` or `export` function. The date is specified in the system short date format.
`time`	The stopping time for the `extract` or `export` function. (The time is specified in the system time format. If no stop time is given, 1 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 before 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 before 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)

transaction

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

Syntax

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

Parameters

`on` or `detail`	Specifies that raw, 5-minute detail data is to be extracted or exported.
`summary`	Specifies that raw data is to be summarized into one data point per hour before being exported.
`both`	Specifies that both 5-minute detail data and hourly summary data are to be extracted or exported.
`off`	Specifies that no transaction data is to be extracted or exported.

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

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, to export data that was logged only on Monday through Thursday, exclude data from Friday, Saturday, and Sunday from your 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` **`[`***yywww***`]`**
`[`*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 `C:\rpmtools\data\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 four digits of the year, the two-digit week number for the week being extracted, and the `.mwe` file name extension. For example, the 12th week of 1999 (from Monday, March 22 to Sunday, March 28) would be output to a file named `rxwe199912.mwe`.

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 specified 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.mwe` 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.mwe` file is created for the current week, the `rxwe199920.mwe` 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.mwe` 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 completes each week's file before creating the next week's file. When you see two adjacent weekly files (for example, `rxwe199920.mwe` and `rxwe199921.mwe`), you can assume that the first file is complete for that week, and it can be archived and purged.

The weeks are numbered according to 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 output file is named `rxwe` followed by the current year (*yyyy*) and week of the year (*ww*) and the `.mwe` file extension.

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.

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 `<rpmtools>\data\datafiles\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 and the `.mwe` file name extension. Thus, data from 1999 would be output to a file named `rxyr1999.mwe`.

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 specified 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, 1997. This created a log file named `rxyr1997.mwe` that contained data from January 1, 1997, to the current date (December 15).

On January 5, 1999, another `yearly` command was executed. Before the `rxyr1999.mwe` file was created for that year, the `rxyr1998.mwe` file from the previous year was opened and checked. When it was found to be incomplete, data was appended to it to complete its extraction until December 31, 1998. Then, the `rxyr1999.mwe` file was 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 completes each year's file before creating the next year's file. When you see two adjacent yearly files (for example, `rxyr1998.mwe` and `rxyr1999.mwe`), you can assume safely 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).

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, uses the settings of the `output` command. It cannot append data to the previous year's extracted file because it does not know its name.

8 Performance Alarms

This chapter provides an Introduction to Performance Alarms and instructions to perform the following tasks to monitor performance using alarms:

- Processing Alarms
- Alarm Definition Components and Alarm Syntax Reference
- Customizing Alarm Definitions

Performance Agent can send alarms only to Performance Manager 3.x and Operations Manager. Performance Agent cannot send alarms to Performance Manager 4.x and above versions.

Introduction to Performance Alarms

You can use Performance Agent to define alarms. These alarms notify you when `scopeNT`, DSI, or Extended Collection Builder metrics meet conditions that you have defined.

To define alarms, you specify conditions on each Performance Agent system that, when met or exceeded, trigger an alert or action. You define alarms in the Performance Agent alarm definitions text file, `alarmdef.mwc`.

As data is logged by `scopeNT` or user-created collectors, it is compared to the alarm definitions to determine if a condition is met. When this occurs, an alert or action can be triggered.

With the real time alarm generator you can configure where you want alert notifications sent and whether you want local actions performed. Alert notifications can be sent to your central PerfView analysis system where you can see graphs of metrics that characterize the performance of your systems or applications. SNMP traps can be sent to HP Network Node Manager. Alert notifications can also be sent to Operations Manager. Local actions can be performed on your Performance Agent system.

You can analyze historical log file data against the alarm definitions, and report the results using the **Analyze** command from the Logfiles menu in the main window or the `utility` program's `analyze` command.

Processing Alarms

As performance data is collected by Performance Agent, it is compared to the alarm conditions defined in the `alarmdef.mwc` file to determine whether the conditions were met. When a condition is met, an alarm is generated and the actions defined for alarms (ALERTs, PRINTs, and EXECs) are performed. You can set up how you want the alarm information communicated once an alarm is triggered. For example, you can:

- Send information to PerfView analysis software and create an alarm symbol in the HP Network Node Manager map associated with PerfView
- Send SNMP traps to Network Node Manger
- Send messages to Operations Manager
- Execute a system command on the local system to send yourself a message

How Alarms Are Processed

When you first start up Performance Agent, the `coda` daemon or the repository servers look for each data source configured in the datasources configuration file and start the alarm generator. Every data source mentioned in your alarm definitions must have a corresponding entry in datasoures. For more information about datasources, and starting and stopping the alarm generator, see Chapter 2 of the *HP Performance Agent for Windows: Installation & Configuration* guide.

As data is collected in the log files, it is compared to the alarm definitions in the `alarmdef.mwc` file. When an alarm condition is met, the actions defined in the alarm definition are carried out. Actions can include:

- Local actions performed via system commands
- Messages sent to PerfView, Network Node Manger, and Operations Manager

Alarm Generator

The Performance Agent alarm generator handles the communication of alarm notifications. The alarm generator consists of the alarm generator server (`Perfalarm`), the alarm generator database (`agdb`), and the utility program `agsysdb`.

The `agdb` contains a list of SNMP nodes. The `agsysdb` program is used for displaying and changing the actions taken by alarm events.

When you start Performance Agent, `perfalarm` starts and reads the `agdb` to determine where and whether to send alarm notifications. It also checks to see if an Operations Manager agent is on the system.

Use the following Windows Command Prompt option to view 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, add your system name to the `agdb` using the command:

```
agsysdb -add systemname
```

Every ALERT generated causes an SNMP trap to be sent to the system you defined. The trap text contains the same message as the ALERT.

To stop sending SNMP traps to a system, delete the system name from the `agdb` using the command:

```
agsysdb -delete systemname
```

Performance Agent uses the `ovtrap` program to send SNMP traps to Network Node Manager. If you want `ovtrap` to use a fully qualified domain name when sending those traps, you must set a system-wide environment variable in the Windows Control Panel.

To set the variable in the Control Panel:

1. Click **System**
2. Click **Environment**
3. Under variable name, type **`MWA_FQDN`**

4 Under value, type the fully qualified domain name.

5 Click **Apply**.

6 Click **OK**.

If this variable is not set, `ovtrap` uses the unqualified host name that is returned by Windows sockets.

Sending Messages to Operations Manager

You can have alert notifications sent to Operations Manager if there was an Operations Manager agent on the same system as Performance Agent when Performance Agent was installed. The Operations Manager agent communicates with the central Operations Manager system. If the Operations Manager agent was *not* installed on the Performance Agent system when Performance Agent was installed, the alarm generator does not try to send alert notifications to Operations Manager, and only local actions are executed.

To stop sending information to Operations Manager when an Operations Manager agent is running on the Performance Agent system, enter the command:

```
agsysdb -ovo OFF
```

If you install the Operations Manager agent after Performance Agent and the alarm generator is already running, you *must* restart your Windows system. This ensures that the system path is updated so that the alarm generator can find the Operations Manager agent.

Executing Local Actions

Without an Operations Manager agent running on the Performance Agent system, local actions in EXEC statements are executed.

You can change the default to turn off local actions using the command:

```
agsysdb -actions off
```

If you want local actions to *always* execute even if the Operations Manager agent is running, type:

```
agsysdb  -actions always
```

The following table lists the settings for sending information to Operations Manager and for executing local actions:

Flags	Operations Manager Agent Running	Operations Manager Agent Not Running
Operations Manager Flag		
off	No alert notifications sent to Operations Manager.	No alert notifications sent to Operations Manager.
on	Alert notifications sent to Operations Manager.	No alert notifications sent to Operations Manager.
Local Actions Flag		
off	No local actions executed.	No local actions executed.
always	Local actions executed even if Operations Manager agent is running.	Local actions executed.
on	Local actions sent to Operations Manager.	Local actions executed.

Errors in Processing Alarms

The last error that occurred when sending an alarm is logged in the agdb. To read this database, type:

```
agsysdb -l
```

The following information is displayed:

```
PA alarming status:


OVO messages : on      Last Error : none
Exec Actions : on
Analysis system: <hostname>, Key=<ip address>
PerfView : no Last Error : <error number>
SNMP : yes Last Error : <error number>
```

Analyzing Historical Data for Alarms

You can use the **Analyze Log File** command from the Logfile menu in the Performance Agent main window to find alarm conditions in log file data (see Analyzing a Log File on page 72). This is different from the processing of real time alarms explained previously because you are comparing historical data in the log file to the alarm definitions in the `alarmdef.mwc` file to determine what alarm conditions would have been triggered.

You also can use the `utility` program's `analyze` command to find alarm conditions in a log file (see Chapter 2, Managing Data Collection).

Examples of Alarm Information in Historical Data

The following examples show what is reported when you analyze alarm conditions in historical data.

In the first example, START, END, and REPEAT statements are defined in the alarm definition. An alarm-start event is listed every time an alarm meets 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.

```
09/10/99 11:15  ALARM [1] START
  CRITICAL: CPU test 99.97%
```

```
09/10/99 11:20  ALARM [1] REPEAT
  WARNING: CPU test 99.997%

09/10/99 11:25  ALARM [1] END
  RESET: CPU test  22.86%
  EXEC: net send mypc help!
```

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.mwc":

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 PerfView or HP IT/Operations, or an SNMP trap to HP Network Node Manager
- An EXEC, which performs a system 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 application or process, and variables. (For more information, see Alarm Syntax Reference on page 252.)

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 shows the statements that are available in the alarm syntax. You may want to look at the `alarmdef.mwc` 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 "system command"

PRINT message

IF condition
   THEN action
   [ELSE action]

{APPLICATION, PROCESS, DISK, TRANSACTION, NETIF} 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 or vertical lines 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:

- 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 enables 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 following example, `highest_cpu` defines a variable. 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 = highest_cpu
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]]
```

In this instance, "==" 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 the following:

```
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. In Performance Agent, metric definitions can be found by choosing **Metrics Dictionary** from the Help menu in the main window. If you are using PerfView, choose **On Metrics** from the PerfView 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`, and `transaction`, 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`.

When specifying fully qualified multi-instance metrics and using aliases within aliases, and one of the aliases has a class identifier, 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, 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 274.

If you have 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
```

Any metric names containing special characters (such as asterisks) must be aliased before they are specified. (See ALIAS Statement on page 274.)

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'
```

If alarms are being sent to Operations Manager, be aware about possible problems with using special characters in a message. Some special characters, such as greater than (>) and less than (<) characters, have special meaning to the Windows command interpreter and cause unexpected occurrences, such as failure to send the alarm to Operations Manager. If these special characters *must* be used in a message, they must be embedded in escape sequences within the message. For example, do not you the following:

```
RED ALERT "cpu > 50"
```

Instead, use the following:

```
RED ALERT "cpu \">\" 50"
```

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

- When the memory paging rate has been too high
- 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 PerfView, you can use **TYPE** to categorize alarms and to specify the name of a graph template to use. PerfView can only accept eight characters, so only eight characters are shown.
- **SERVICE** is a quoted string of up to 200 characters. If you are using ServiceNavigator, which is being released with Operations Manager 5.0, you can link your Performance Agent alarms with the services you defined in ServiceNavigator. (See the Operations Manager documentation.)

  SERVICE="Service_id"
- **SEVERITY** is an integer from 0 to 32767. If you are sending alarms to PerfView, you can use this to categorize alarms.
- **START**, **REPEAT**, and **END** are keywords that specify what action to take when alarm conditions are met, met again, or stopped. 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 action, is the ALERT statement. 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 a comparison between two items. For example:

  ```
  item1 {>, <, >=, <=, ==, !=}item2
  ```

  ```
  [AND, OR[item3 {>, <, >=, <=, ==, !=}item4]]
  ```

  In this instance, "==" 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 sub-conditions. 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
  ```

  This causes 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 1 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 `scopeNT` data, the duration is 5 minutes; however, the duration for process data is 1 minute. For DSI data, the duration is 5 seconds or longer.

- **REPEAT EVERY** *duration* **SECONDS, MINUTES** specifies the time period before the alarm is repeated.

Do not use embedded double quotes (") in alarm text; `alarmgen` will fail to send the alarm to Operations Manager. Use embedded single quotes (') instead. For example:

```
red alert "\'message with embedded single quotes'\"
```

How It Is Used

The alarm cycle begins on the first interval that all of the ANDed, or one of ORed alarm conditions was 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.

For PerfView 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 automatically sends one to PerfView and Operations Manager, and sends 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.mwc` file. Do not add this example to your `alarmdef.mwc` file without removing the default alarm condition, or your subsequent alert messages might 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 PerfView, to Network Node Manager via an SNMP trap, or as a message to Operations Manager. For more

information, see Alarm Generator on page 246. If you have PerfView configured correctly, the RED ALERT statement places the `"swap utilization still very high"` message in the PerfView 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 sends 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 how to cause a message to be sent to a pager when an event occurs.

```
ALARM gbl_cpu_total_util > 90 FOR 5 MINUTES
  START  {
    RED ALERT "Your cpu is busy."
    EXEC "net send ADMINPC Server Busy"
      }
  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 Operations Manager. The EXEC can trigger a mail message to be sent as a local action on your Performance Agent system, depending on how you have configured your alarm generator. For more information, see Alarm Generator on page 246. If you set up PerfView to receive alarms from this system, the RED ALERT statement places the `"Your CPU is busy"` message in the PerfView Alarms window and causes a message to be sent.

By default, if the Operations Manager agent is running, the local action will not execute. Instead, it is sent as a message to Operations Manager.

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 ALARM example below 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"
```

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 the following:

```
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 PerfView, Network Node Manager, or Operations Manager. It also allows the creation and deletion of the alarm symbols in the Network Node Manager map associated with PerfView and controls the color of the alarm symbols, depending on the severity of the alarm. (For more information, see the PerfView 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 is 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 PerfView. They also send the message with other information to the PerfView Alarms window. CYAN is the default. However, if you are using version C.00.08 or earlier of PerfView, YELLOW is the default.
- **RESET** records the *message* in the PerfView Alarms window and deletes the alarm symbol in the Network Node Manager map associated with PerfView. 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.

  For information on the use of special characters in a message, such as the greater than (>) or less than (<) characters, see Messages on page 257.

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 Operations Manager, depending on how you configured your alarm generator. For more information, see Alarm Generator on page 246. If you configured PerfView to receive alarms from this system, the ALERT sends a message to the PerfView Alarms window.

If an ALERT statement is used outside of an ALARM statement, the alert message will show up in the Alarms window but no symbol will be created in the Network Node Manager map.

For alert messages sent to Operations Manager, the WARNINGS are displayed in cyan in the message browser.

Example

An example ALERT statement is:

```
RED ALERT "CPU utilization = ", gbl_cpu_total_util
```

If you have PerfView and Network Node Manager, this statement creates a red alarm symbol in the Network Node Manager map associated with PerfView and sends a message to the PerfView Alarms window that reads:

```
CPU utilization = 85.6
```

EXEC Statement

The EXEC statement enables you to specify a system command to be performed on the local system. For example, you can use the EXEC statement to run a batch file that calls an IT administrator's pager each time a certain condition is met.

EXEC should be used within an ALARM or IF statement so the system command is performed only when specified conditions are met. If an EXEC statement is used outside an ALARM or IF statement, the action is performed at unpredictable intervals.

Syntax

EXEC "*system command***"**

- *system command* — a command to be performed on the local system, such as a command that could be run from the Windows Command Prompt.

If you specify a directory path in the EXEC statement, you *must* insert double back slashes in the path name. For example:

```
EXEC "mkdir c:\\directory\\filename"
```

The syntax of the EXEC statement requires the path name of the file to be enclosed within double quotes. However, if a path name in the EXEC statement contains spaces, the path name must be enclosed within single quotes, which must be again enclosed within double quotes. For example:

```
EXEC "'C:\\Program Files\\Mail Program\\SendMail.exe'"
```

If any arguments to the system command of the EXEC statement contains single quotes, the program must be enclosed within single quotes irrespective of whether the command contains spaces or no spaces. For example:

```
EXEC "'echo' 'test execution'"
```

In the above example, echo is the program enclosed within single quotes as it contains an argument (in this case, test execution) within single quotes. The entire string of the command must be enclosed in double quotes.

Do not use embedded double quotes (") in EXEC statements; `alarmgen` will fail to send the alarm to the HP Operations Manager. Use embedded single quotes (') instead. For example:

```
EXEC "'echo' 'dialog performance problem'"
```

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 Operations Manager, local actions will not usually be performed. For more information, see Alarm Generator on page 246.

The path statement must be set for either binaries or batch files used in local actions; otherwise, they must be fully qualified. If an output file is created by the EXEC, it will end up in the `system32` directory unless its path statement is set or it is fully qualified.

Example

In the following example, the EXEC statement runs a batch file that calls a pager when the `gbl_disk_util_peak` metric exceeds 20.

```
IF gbl_disk_util_peak > 20 THEN
EXEC "callpage.bat 555-1234"
```

The next example shows the EXEC statement sending a message to the system administrator's PC when the network packet rate exceeds 1000 per second average for 15 minutes.

```
ALARM gbl_net_packet_rate > 1000 for 15 minutes
  TYPE = "network busy"
  SEVERITY = 5
  START
```

```
  {
  RED ALERT "network is busy"
  EXEC "net send ADMINPC Network Busy"
  }
END
RESET ALERT "NETWORK OK"
```

Be careful when using the EXEC statement with system commands or batch files that have high overhead if it will be performed often.
The alarm generator executes the command and waits until it completes before continuing. Be careful not to specify commands that take a long time to complete.

PRINT Statement

The PRINT statement enables 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. Numeric values are right-justified. The `-` (minus sign) specifies left-justification. Alphanumeric strings are always left-justified.

  For information on the use of special characters within a message, such as the greater than (>) or less than (<) characters, see Messages on page 257.

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. It can be used by itself or any place in the `alarmdef.mwc` 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]]
  ```

  In this instance "==" 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.

- *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 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 was 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 configured PerfView correctly, the message `"End of CPU Bottleneck Alert"` is displayed in the PerfView 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 Operations Manager, 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"
```

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, DISK, CPU, FILESYSTEM, TRANSACTION, NETIF, LOGICAL}LOOP *action*

- **APPLICATION, PROCESS, DISK, CPU, FILESYSTEM, TRANSACTION, NETIF, LOGICAL** — 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.

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 utility at each interval. When the statement `"highest_cpu_util = highest_cpu_util"` is executed during the first interval, `highest_cpu_util` is initialized to 0. During subsequent intervals, `highest_cpu_util` is initialized to the value from the previous interval.

```
highest_cpu_util = highest_cpu_util
APPLICATION loop
  IF app_cpu_total_util > highest_cpu_util THEN
   {
     highest_cpu_util = app_cpu_total_util
     big_app = app_name
     }
```

```
  ALARM highest_cpu_util > 50 for 15 minutes
    START
      RED ALERT  big_app, " is the highest CPU user at ",
highest_cpu_util, "%"
    REPEAT EVERY 15 minutes
      CYAN ALERT big_app, " is the highest CPU user at ",
highest_cpu_util, "%"
  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.mwc` file.

Syntax

INCLUDE "*filename***"**

In this instance, *filename* is the name of another alarm definitions file. The file name must always be fully qualified. When you specify the directory path in the INCLUDE statement, you must insert double back slashes (`\\`) in the path name. For example:

```
INCLUDE "c:\\rpmtools\\alarms\\alarmdef3.mwc"
```

How It Is Used

The INCLUDE statement should be used to separate logically distinct sets of alarm definitions into separate files.

Example

You have some alarm definitions for transaction metrics in a separate file that is named `trans_alarmdef1.mwc`. You can include it by adding the following line to the alarmdefinitions in your default `alarmdef.mwc` file:

```
  INCLUDE "c:\\program
files\\hp\\openview\\trans\\trans_alarmdef1.mwc"
```

USE Statement

You can add the USE statement to simplify the use of metric names in the `alarmdef.mwc` file when data sources other than the default SCOPE data source are referenced. This enables you to specify a metric name without having to include the data source name.

The data source name must be defined in the `perflbd.mwc` file. The `alarmdef.mwc` file will fail its syntax check if an invalid or unavailable data source name is encountered.

The appearance of a USE statement in the `alarmdef.mwc` file does not imply that all metric names that follow are from the specified data source.

Syntax

USE "*datasourcename***"**

How It Is Used

As the alarm generator server (`alarmgen`) checks the `alarmdef.mwc` file for valid syntax, it builds an ordered search list of all data sources that are referenced in the file. `Alarmgen` 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 `alarmgen'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 255.

`Alarmgen`'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 `alarmgen` encounters the first metric name in the `alarmdef.mwc` 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.

```
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 `alarmgen` checks the syntax of the `alarmdef.mwc` file containing the above statements, it encounters the metric "`gbl_cpu_total_util`" and then tries to find its data source. `Alarmgen` 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 `Alarmgen` continues checking the rest of the `alarmdef.mwc` file.

When `Alarmgen` 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, `Alarmgen` 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 `Alarmgen` does not find a match in any data source for a metric name, an error message is printed and `Alarmgen` terminates.

To change the default search behavior, a USE statement can be added to the beginning of the `alarmdef.mwc` file before any references to metric names. This causes 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.mwc` 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 `Alarmgen` encounters the first metric name "`gbl_cpu_total_util`." Because the "`gbl_cpu_total_util`" metric name is

not fully qualified, Alarmgen 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.

Alarmgen continues checking the rest of the alarmdef.mwc file. When Alarmgen encounters the "ActiveTransactions" metric, it sequentially searches the list of data sources starting with ORACLE7. A match is found and Alarmgen continues searching the rest of the alarmdef.mwc file. If Alarmgen does not find a match in any data source for a metric name (that is not fully qualified), an error message is printed and Alarmgen terminates.

Be careful how you use the USE statement when multiple data sources contain the same metric names. Alarmgen 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 enables you to define a variable and assign a value to it.

Syntax

[VAR] *name* **=** *value*

- *name* — Variables names must begin with a letter and can include digits and the underscore character. Variables 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

Define a variable by assigning something to it. For example:

```
highest_CPU_value = 0
```

This example defines the numeric variable `highest_CPU_value` by assigning it a value of zero.

```
my_name = ""
```

This example defines the alphanumeric variable `my_name` by assigning it an empty string value.

ALIAS Statement

The ALIAS statement enables 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.mwc` 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 must create an alias. You must create an alias for any name that includes special characters. For example, if you have applications called "BIG" and "big," you must create "big" as an alias to ensure that they are viewed as different applications. You must define the alias somewhere in the `alarmdef.mwc` file before the *first* instance of the name you want substituted.

Because you cannot use special characters or uppercase and lowercase in the syntax, using the application name "AppA" and "appa" could cause errors because the processing is unable to distinguish between the two. You must alias "AppA" to give it a uniquely recognizable name as shown in the following example.

```
ALIAS appa_uc = "AppA"
  ALERT "CPU alert for AppA.util is", appa_uc:app_cpu_total_util
```

If you are using aliases within aliases, and one of the aliases has a class identifier, use the following alias syntax :

```
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 within the SYMPTOM statement as you need. 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]
  ```

  In this instance, "==" 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 > 50  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 are typical uses of alarm definitions.

Example of a CPU Problem

```
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 too high at ", gbl_cpu_total_util, "%"
  EXEC "callpage.bat 555-3456"}
END ALERT "CPU at ", gbl_cpu_total_util, "% - RELAX"
```

The ALERT could trigger an SNMP trap to be sent to Network Node Manager or a message to be sent to Operations Manager, 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 a message is sent to the PerfView 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 PerfView Alarms window showing the global CPU utilization, the time the alert ended, and a note to RELAX.

Example of Swap Utilization

If you configured PerfView correctly, the following 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 PerfView 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 Operations Manager, 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 8:00 am to 5:00 pm.

```
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 ALET "cpu still too high, but at the end of shift"
       ELSE
         RESET ALERT  "cpu back to normal"
     }
```

Examples of Disk Instance Alarms

The following three examples of alarm syntax generate alarms for a specific disk instance. Alarms can be generated for a particular disk by identifying the specific disk instance name and corresponding metric name.

The first example uses a user-defined data source name SYSTEM1 for which there is a disk named Disk3.

```
USE "SYSTEM1"
 ALARM Disk3:bydisk_phys_read > 1000 for 5 minutes
   START
      RED ALERT "Disk 3 red alert"
   REPEAT EVERY 10 MINUTES
      YELLOW ALERT "Disk 3 yellow alert"
   END
      RESET ALERT "Disk 3 reset alert"
```

In the next example, the use of a fully qualified name generates alarms for a specific disk instance. See the PerfView `utility repinfo` for additional syntax information. The `repinfo utility` is a program that lists the Performance Agent fully qualified metrics from all data sources on a given system. This utility helps you identify metric information that you can use in the PerfView Command Prompt or in the Performance Agent `alarmdef.mwc` file.

```
ALARM SYSTEM1:Disk3(DISK):bydsk_phys_read > 1000 for 5 minutes
   START
      RED ALERT "Disk 3 red alert"
   REPEAT EVERY 10 MINUTES
      YELLOW ALERT "Disk 3 yellow alert"
   END
      RESET ALERT "Disk 3 reset alert"
```

In the next example, aliasing is required when special characters are used in the disk instance.

```
ALIAS diskname="Disk#/+3/"
 ALARM diskname:bydsk_phys_read > 1000 for 5 minutes
 TYPE="Disk"
   START
      RED ALERT "Disk 3 red alert"
   REPEAT EVERY 10 MINUTES
      YELLOW ALERT "Disk 3 yellow alert"
   END
      RESET ALERT "Disk 3 reset alert"
```

Customizing Alarm Definitions

You specify the conditions that generate alarms in the alarm definitions file `alarmdef.mwc`. When Performance Agent is first installed, the `alarmdef.mwc` file contains a set of default alarm definitions. You can use these default alarm definitions or customize them to suit your needs.

To customize the `alarmdef.mwc` file, see Configuring Alarm Definitions on page 84.

You can use a unique set of alarm definitions for each Performance Agent system, or you can standardize monitoring of a group of systems by using the same set of alarm definitions across the group.

If the `alarmdef` file is very large, the Performance Agent alarm generator and `utility` programs may not work as expected. This problem may or may not occur based on the availability of system resources.

The best way to learn about performance alarms is to experiment with adding new alarm definitions or changing the default alarm definitions.

A Viewing MPE Log Files

MPE log file data collected by the `scopeXL` collector can be viewed with PerfView. Before viewing the data, you must first extract it and then load the log files as a local data source on your PerfView system.

To view your MPE log file data using PerfView, follow these steps:

1. Log on 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 MeasureWare Agent and PerfView are running.
5. Log on to your PerfView 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 PerfView.
7. Add the extracted MPE log file data as a local data source. (For more information, see "Add a Local Data Source" in PerfView's online help.)
8. View the data.

Glossary

alarm

An indication of a period of time in which performance meets user-specified alarm criteria. Alarm information can be sent to a PerfView analysis system and to Network Node Manager and Operations Manager. Alarms can also be identified in historical log file data.

alarm generator

The process that handles the communication of alarm information. It consists of the alarm generator server (`Perfalarm`) and the alarm generator database (`agdb`). The `agsysdb` program uses a Windows Command Prompt interface for displaying and changing the actions taken by alarm events.

alarmdef file

A Performance Agent ASCII text file (`alarmdef.mwc`) 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 `scopent` log files or DSI log files. `coda` reads the data from the data sources listed in the `datasources` configuration file.

collections

A defined set of counters/metrics that has been registered and assigned a unique name and is based on a policy established by the Extended Collection Builder.

counters

In Windows, units pertaining to an object (threads and processes, sections of shared memory, and physical devices) that can be measured (or counted).

data source

A data source consists of one or more classes of data in a single `scopeNT` or DSI log file set. For example, the default Performance Agent data source, is a `scopeNT` log file set consisting of global data. *See also* **repository server** and **perflbd.mwc**. The data source is configured in the datasources file that resides in the `<DataDir>\conf\perf directory`.

data source integration (DSI)

The technology that enables 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 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.

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 Performance Agent process that logs user-defined data received from `stdin`.

DSI log files

Log files, created by the dsilog process, that contain user-defined data collected outside of Performance Agent. *See also* **dsilog**.

empty space

The difference between the maximum size of a log file and its current size.

extract

The Performance Agent program that helps you manage your data. In `extract` mode, raw or previously extracted log files can be read in, reorganized or filtered as desired, and the results combined into a single, easy-to-manage extracted log file. In `export` mode, raw or previously extracted log files can be read in, reorganized or filtered as desired, and the results are written to class-specific exported data files for use in spreadsheets and analysis programs.**extracted log file**

A 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.

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**.

logls

The raw log file that contains information about logical systems, when Performance Agent is monitoring a virtualized environment.

logtran

The raw log file that contains measurements of transaction data.

parm file

A Performance Agent file (`parm.mwc`) containing the collection parameters used by `scopeNT` to customize data collection.

Performance Manager

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.

perflbd

The location broker daemon that reads the `perflbd.mwc` file when Performance Agent is started and starts a repository server for each data source that has been configured. *See also* **perflbd.mwc** and **repository server**.

perflbd.mwc

A configuration file that resides in the `<DataDir>` directory and is linked to `datasources` file. Each entry in the file represents a `scopeNT` or DSI data source consisting of a single log file set. *See also* **data source** and **repository server**.

performance alarms

See **alarms**.

PerfView

PerfView 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.

policy

An Extended Collection Builder performance measurement configuration file that contains information about the Windows NT/2000 counter set, instance selection, log file locations, and data collection rates and calculations. This file can be reused to define uniquely named collections of performance counters/ metrics on multiple PCs using Windows NT/2000.

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**.

raw log file

A file containing summarized measurements of system data. The `scopeNT` program collects 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.

repository server

A server that provides data to the alarm generator and the PerfView analysis product. There is one repository server for each data source configured in the `perflbd.mwc` configuration file consisting of a `scopeNT` or DSI log file set. A default repository server is provided at start up that contains a single data source consisting of a `scopeNT` log file set.

resize

Changing the overall size of a raw log file using the either the resize task in the Performance Agent graphical interface or 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 `scopeNT` collector to indicate that the collection process is running. Removing the `RUN` file causes `scopeNT` to terminate.

rxlog

The default output file created when data is extracted from raw log files. The size of the rxlog file cannot exceed 3.5 GB.

rxlog.mwe

The default output file created when data is extracted from raw log files.

scopeNT

The Performance Agent collector program that collects performance data and writes (logs) this raw measurement data to raw log files for later analysis or archiving.

scopeNT log files

The six log files that are created by the `scopeNT` collector: `logglob`, `logappl`, `logproc`, `logdev`, `logtran`, and `logindx`.

status.scope

The file created by the `scopeNT` collector to record status, data inconsistencies, or errors.

transaction tracking

The 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, Performance Agent must have a process running that is instrumented with the Application Response Measurement (ARM) API.

utility

A Performance Agent program that enables you to open, resize, scan, and generate reports on raw and extracted `scopeNT` log files. You can also use it to check `parm` file syntax, check the `alarmdef` file syntax, and obtain alarm information from historical log file data.

Index

A

B

C

D

E

F

G

H

N

O

P

Q

R

S

T

U

V

W

Y

We appreciate your feedback!

If an email client is configured on this system, by default an email window opens when you click on the bookmark “Comments”.

In case you do not have the email client configured, copy the information below to a web mail client, and send this email to **docfeedback@hp.com**

Product name:

Document title:

Version number:

Feedback: