HP Operations Agent

适用于 Windows®、 Linux、 HP-UX、 Solaris 和 AIX 操作系统

软件版本：11.00

用户指南

文档发行日期：2010 年 10 月
软件发行日期：2010 年 10 月

法律声明

担保

HP 产品和服务的唯一担保已在此类产品和服务随附的明示担保声明中提出。此处的任何内容均不构成额外担保。HP 不会为此处出现的技术或编辑错误或遗漏承担任何责任。

此处所含信息如有更改，恕不另行通知。

限制权利图例

机密计算机软件。拥有、使用或复制操作需要 HP 的有效许可证。根据 FAR 12.211 和 12.212，商业计算机软件、计算机软件文档和商业项目的技术数据已按照供应商的标准商业许可条款授权给美国政府。

版权声明

商标声明

Intel® 和 Itanium® 是 Intel Corporation 在美国和其他国家 / 地区的商标。

Microsoft®、 Windows®、 Windows® XP 和 Windows Vista® 是 Microsoft Corporation 在美国的注册商标。

UNIX® 是 The Open Group 的注册商标。

致谢

本产品包含由 Eric Young (eay@cryptsoft.com) 编写的加密软件。

本产品包含由 OpenSSL Project (http://www.openssl.org/) 开发用于 OpenSSL 工具包的软件。

本产品包含由 Tim Hudson (tjh@cryptsoft.com) 编写的软件。

本产品包含由 Apache Software Foundation (http://www.apache.org/) 开发的软件。

本产品包含 “zlib” 通用压缩库， Copyright © 1995-2002 Jean-loup Gailly and Mark Adler。

文档更新

此文档的标题页包含以下标识信息：

- 软件版本号，表示软件版本。
- 文档发行日期，在每次更新文档时更改。
- 软件发行日期，表示此版本软件的发行日期。

要检查是否有最新更新或验证您所使用的文档是否为最新版，请转到：

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

此站点要求您注册 HP Passport 才能登录。要注册 HP Passport ID，请转到：

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

或单击 HP Passport 登录页上的 **New users - please register** 链接。

如果订阅相应的产品支持服务，还将收到更新的版本或新版本。有关详细信息，请联系您的 HP 销售代表。

支持

访问 HP Software 在线支持网站：

www.hp.com/go/hpsoftwaresupport

此网站提供了联系信息以及有关 HP Software 提供的产品、服务和支持的详细信息。

HP Software 在线支持为客户提供了自解决功能。您可以通过它快速有效地访问管理业务所需的交互技术支持工具。作为重要的支持客户，您可以享受使用支持网站所带来的以下好处：

- 搜索感兴趣的知识文档
- 提交并跟踪支持案例和改进请求
- 下载软件补丁
- 管理支持合同
- 查找 HP Support 联系人
- 检查有关可用服务的信息
- 加入与其他软件客户的讨论中
- 研究并注册软件培训

大多数支持区域要求您以 HP Passport 用户身份注册才能登录。许多区域还需要支持合同。要注册 HP Passport 用户 ID，请转到：

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

要查找有关访问级别的详细信息，请转到：

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

目录

1 简介

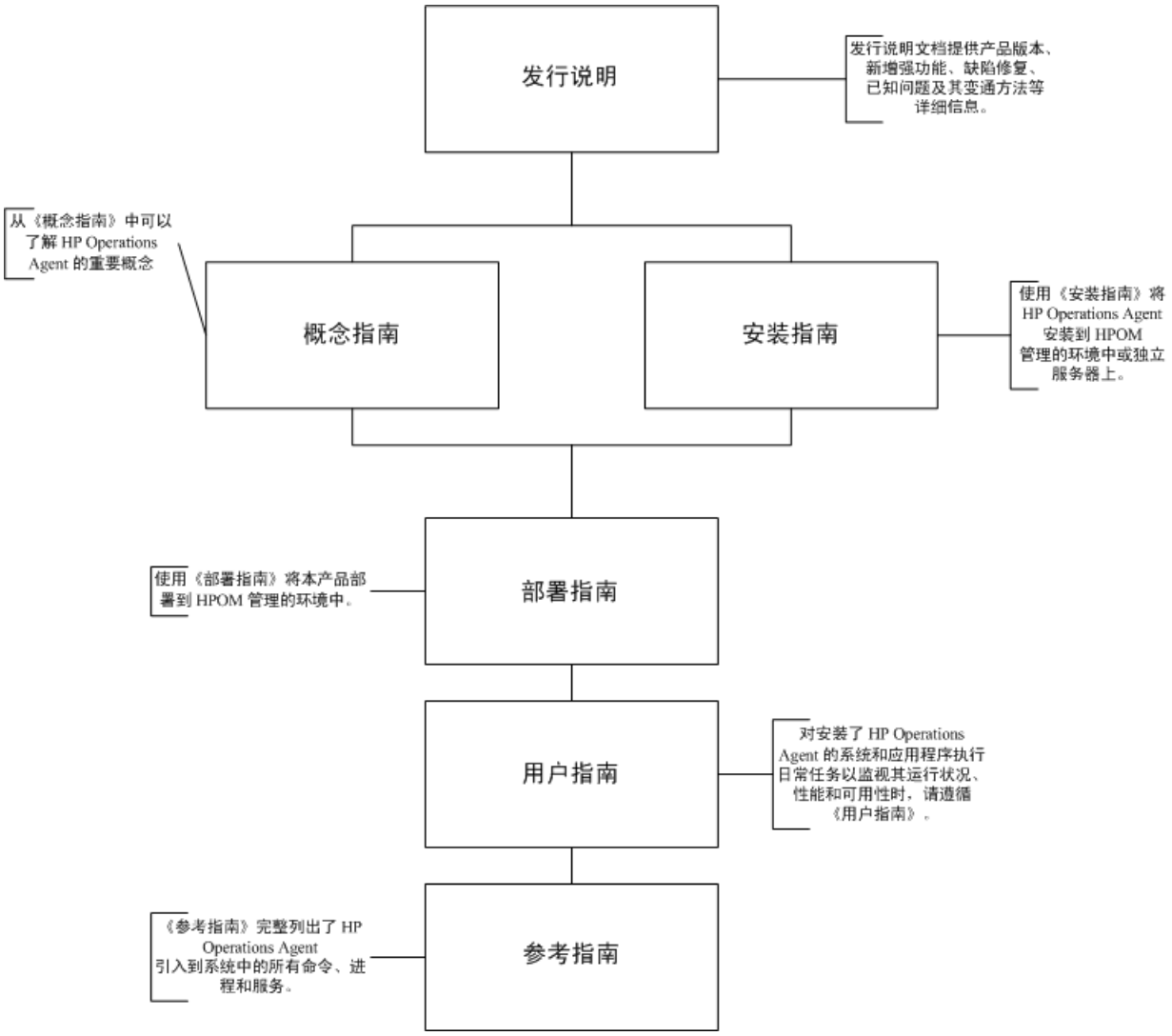
HP Operations Agent 通过收集指示系统运行状况、性能和必需元素可用性的各种度量来帮助您监视系统。 HP Operations Agent 的嵌入式数据收集器允许您收集并记录环境中系统的性能度量。

HP Operations Agent 使用日志文件存储收集的度量，并为您提供机制在收集的度量与预设的阈值不匹配时生成警报消息。可以使用 HP Reporter 和 HP Performance Manager 等数据分析工具查看收集的度量数据。如果安装了代理程序并将其配置为用于 HP Operations Manager (HPOM) 管理服务器，则可以从 HPOM 控制台集中监视代理程序节点的运行状况和性能。

文档图

文档图显示了 HP Operations Agent 所有主要文档的列表。需要帮助时，可以通过该图来判断所需文档。

图 1　　HP Operations Agent 的文档图



相关文档

可以在产品媒体的 `paperdocs` 目录内找到 HP Operations Agent 的所有用户文档。要检查是否有最新更新或验证您所使用的文档是否为最新版，请转到：

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

此站点要求您注册 HP Passport 才能登录。要注册 HP Passport ID，请转到：

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

或单击 HP Passport 登录页上的 **New users - please register** 链接。

表 1　　HP Operations Agent 的用户文档

文档	使用	关键主题
发行说明	有关产品版本、新功能和已知问题的信息，请参考此文档。	• 新功能 • 增强功能 • 修复 • 已知问题和限制
概念指南	《概念指南》帮助您了解不同环境中 HP Operations Agent 的工作机制。	• HP Operations Agent 简介 • HP Operations Agent 的主要组件
安装指南	可以使用《安装指南》将 HP Operations Agent 安装到以下环境中： • HPOM 管理服务器（用于 HPOM 管理的分布式管理环境中） • 独立服务器（收集本地服务器的系统性能度量，供 HP Performance Manager 等外部数据分析工具使用）	• 从 HPOM 控制台安装 HP Operations Agent • 手动安装 HP Operations Agent • 许可
部署指南	使用此指南可以从 HPOM 中央管理服务器将 HP Operations Agent 部署到监视环境中。	• 在 HPOM 管理服务器和 HP Operations Agent 之间建立安全通信通道。 • 将 HP Operations Agent 配置为在高可用性群集环境中工作。 • 从 HPOM 控制台远程管理 HP Operations Agent 配置。
参考指南	《参考指南》完整地列出了 HP Operations Agent 节点上可用的所有命令、进程和服务。	• 命令行实用程序 • 配置变量

2 管理数据收集

HP Operations Agent 提供了数据收集器，用于收集和记录监视的系统的系统性能数据。数据收集器程序 **scope** 可用于存储系统上收集的数据。可以使用 HP Performance Manager 或 HP Reporter 查看和分析存储的数据。

scope 收集器使您可对系统执行以下任务：

- 收集指示监视的系统运行状况和性能的度量数据
- 将收集的度量数据记录到不同日志文件中

scope 不记录 NFS 数据，但您可以通过 HP GlancePlus 在本地文件系统上查看 NFS 数据。

配置参数文件 parm 可用于配置 scope 收集器的默认数据记录机制。通过修改 parm 文件中的参数，可以控制 scope 收集器的以下属性：

- 数据记录间隔
- 数据类型
- 日志文件大小

在节点上安装 HP Operations Agent 之后，必须修改 parm 文件以配置 scope 的数据收集机制。

收集日志文件

scope 数据收集器（在 UNIX 和 Linux 节点上为 scopeux，在 Windows 节点上为 scopent）收集、汇总系统资源利用率的性能测量值，并根据 parm 文件 log line 中指定的数据类将数据记录到以下日志文件中：

- logglob
- logappl
- logproc
- logpcmd
- logdev
- logtran
- logls
- logindx

日志文件中的记录的时间戳指示数据收集的开始时间。感兴趣的进程的概念是帮助最小化数据记录量的过滤器，从 parm 文件进行控制。

scope 不记录 NFS 数据，但可以通过 GlancePlus 在本地文件系统上查看 NFS 数据。

logglob

logglob 文件包含系统范围 （全局）资源利用率信息的测量值。 scope 收集器对全局数据进行汇总，并按 parm 文件中指定的间隔定期记录这些数据。

logappl

logappl 文件包含在 parm 文件内定义的应用程序中运行的进程的合计测量值。scope 收集器对应用程序数据进行汇总，并按 parm 文件中指定的间隔定期记录这些数据。

logproc

scope 收集器识别您可能感兴趣的进程，然后在 logproc 文件中记录所识别进程的合计测量值。scope 基于以下条件识别进程：

- 进程开始
- 进程结束
- parm 文件中指定的配置详细信息

logpcmd

logpcmd 文件包含对 logproc 文件中记录的进程执行的命令行活动的详细信息。

您无法控制 logpcmd 文件的大小、滚动更新和记录间隔。

logpcmd 文件存储在节点的以下目录中：

- 在 Windows 上：*%ovdatadir%*\datafiles
- 在 UNIX （和 Linux）上：/var/opt/perf/datafiles

文件最多可以存储 25 MB 的数据。数据收集机制启动时，scope 收集器创建 logpcmd 文件的第一个实例，扩展名为 0。logpcmd0 文件达到 25 MB 的限制时， scope 创建 logpcmd 文件的第二个实例 - logpcmd1 文件。

如果 logpcmd1 文件超过 25 MB 的限制，数据开始从 logpcmd0 文件滚动更新。

logdev

logdev 文件包含单个设备性能的测量值。scope 收集器对设备数据进行汇总，并按 parm 文件中指定的间隔定期记录这些数据。

logtran

logtran 文件包含 ARM 事务数据的测量值。scope 收集器对事务数据进行汇总，并按 parm 文件中指定的间隔定期记录这些数据。有关收集事务数据的详细信息，请参见第 339 页的什么是事务跟踪？。

logls

logls 文件包含有关逻辑系统的信息。scope 收集器对逻辑系统数据进行汇总，并按 parm 文件中指定的间隔定期记录这些数据。

logls 文件仅在适用于 HPVM、Hyper-V 主机、vSphere Management Assistant (vMA)、Solaris 全局区域和 AIX-LPAR 的 HP Operations Agent 上可用。

logindx

logindx 文件包含访问其他日志文件中的数据所需的信息。

scope 状态

除了日志文件以外，启动 scope 时还创建其他两个文件。它们是驻留在 /var/opt/perf/datafiles/ 目录中的 RUN 文件和驻留在 /var/opt/perf/ 目录中的 status.scope 文件。

创建 RUN 文件是为了指示 scope 进程正在运行。删除此文件会终止 scope。

/var/opt/perf/status.scope 文件充当 scope 进程的状态/错误日志。每次启动、停止 scope 收集器或遇到警告或错误时，会在此文件中追加新信息。要查看 scope 的最新状态和错误信息，请使用 perfstat -t 命令。

parm 文件

parm 文件是一种文本文件，它包含告诉 scope 记录特定性能测量值的指示。

在全新安装期间， HP Operations Agent 将默认 parm 文件放在两个不同的目录中：

- 在 Windows 上：

  在 Windows 上， parm 文件的扩展名是 .mwc (parm.mwc)。

  — *%ovinstalldir%*\newconfig

  — *%ovdatadir%*

- 在 HP-UX、 Solaris 和 Linux 上：

  — /opt/perf/newconfig

  — /var/opt/perf

- 在 AIX 上：

  — /usr/lpp/perf/newconfig

  — /var/opt/perf

scope 的数据收集机制由 *%ovdatadir%* (Windows) 或 /var/opt/perf（UNIX 或 Linux）目录下的 parm 文件中的设置控制。

如果要修改默认收集机制，必须修改 *%ovdatadir%* (Windows) 或 /var/opt/perf（UNIX 或 Linux）目录下的 parm 文件中的设置。

升级节点上的 HP Operations Agent（从旧版本 HP Performance Agent 升级）时，升级过程会更新 newconfig 目录下的 parm 文件的副本。驻留在其他目录下的 parm 文件不受影响，继续控制节点上的数据收集机制。此方法实际上使您在升级本产品后也能保留配置的数据收集机制。您可以在本产品升级后随时比较 parm 文件的现有配置设置和 newconfig 目录下的 parm 文件新版本，然后进行必需的更改。

parm 文件是为了收集平均量的日志文件数据设置的。最大量取决于您的系统。请参见第 22 页的参数描述中的 size 参数描述。

修改 parm 文件

可以使用任何可将文件保存为 ASCII 格式的字处理器或编辑器修改 parm 文件。

修改 parm 文件或新建 parm 文件时，以下规则和约定适用：

- 任何指定的参数都覆盖默认值。有关默认值，请参见 newconfig 目录下的 parm 文件。
- 将参数指定到 parm 文件中的顺序并不重要。
- 如果指定一个参数多次，则参数的最后一个实例有效。
- file、 user、 group、 cmd、 argv1 和 or 参数必须遵循它们定义的 application 语句。
- 应用程序参数必须按顺序列出，以便进程可以聚合到第一个与其匹配的应用程序中。
- 在所有命令和参数语句中都可以使用大写字母和 / 或小写字母。

- 在每个语句中都可以使用空格或逗号分隔关键字。
- 可以在 parm 文件中对参数进行注释。将忽略任何以注释代码 (/*) 或英镑标记 (#) 开头的行。

修改 parm 文件之后，必须重新启动性能收集组件以使更改生效。要重新启动性能收集组件，请运行以下命令：

在 Windows 上

%ovinstalldir%bin\ovpacmd REFRESH COL

在 HP-UX、Linux 或 Solaris 上

/opt/perf/bin/ovpa -restart scope

在 AIX 上

/usr/lpp/perf/bin/ovpa -restart scope

如果要使用实时度量访问 (RTMA) 组件，还必须重新启动 perfd 进程：

在 Windows 上

%ovinstalldir%bin\ovpacmd REFRESH RTMA

在 HP-UX、Linux 或 Solaris 上

/opt/perf/bin/pctl restart

在 AIX 上

/usr/lpp/perf/bin/pctl restart

parm 文件参数

scope 由收集参数 (parm) 文件中执行以下任务的特定参数控制：

- 设置原始 scope 日志文件的最大磁盘空间量。
- 指定要记录的数据类型。
- 指定记录数据的间隔。
- 指定要记录的进程和度量的属性。
- 定义要收集和记录的性能数据的类型。
- 指定应监视的用户可定义的应用程序集。应用程序可以是一个或多个按组监视的程序。
- 指定 scope 何时执行日常日志文件维护活动才不会影响系统可用性。

您可以修改这些参数，以将 scope 配置为记录符合监视的系统要求的性能数据（请参见第 18 页的修改 parm 文件）。

第 20 页的表 2 中列出了 scope 使用的 parm 文件参数。如表中所示，某些参数只用于特定系统。有关这些参数的详细描述，请参见第 22 页的参数描述和第 31 页的应用程序定义参数。

表 2　　scope 使用的 parm 文件参数

参数	值或选项
id	系统 ID
log	• global • application [=prm] [=all] （[=prm] 仅限 HP-UX） • process • device=disk,lvm,cpu,filesystem,all （lvm 仅限 HP-UX） • transaction=correlator,resource （resource 仅限 HP-UX） • logicalsystem （对于 Solaris，仅 Solaris 10 操作环境或更高版本支持逻辑系统） 对于 AIX，仅 LPAR on AIX 5L V5.3 ML3 及更高版本和 WPAR on AIX 6.1 TL2 全局环境支持逻辑系统。 启用 lpar 记录： logicalsystems=lpar logicalsystems 启用 wpar 记录： logicalsystems=wpar 同时启用 lpar 和 wpar 记录： logicalsystems=lpar,wpar logicalsystems=wpar,lpar logicalsystems=all
mainttime	hh:mm（24 小时时间格式）
scopetransactions	on off
subprocinterval	以秒为单位的值（除 HP-UX 外）
javaarg **注：**仅限 UNIX/Linux。	true false
procthreshold （与 threshold 相同）	cpu= *百分比* disk= *速率*（除 Linux 和 Windows 外） memory=nn（以 MB 为单位的值） nonew nokilled shortlived
appthreshold	cpu= *百分比*
diskthreshold	util= *速率*
bynetifthreshold	iorate= *速率*

表 2　　scope 使用的 parm 文件参数 （续）

参数	值或选项
fsthreshold	util= *速率*
lvthreshold	iorate= *速率*
bycputhreshold	cpu= *百分比*
wait	cpu= *百分比* （仅限 HP-UX） disk= *百分比* （仅限 HP-UX） mem= *百分比* （仅限 HP-UX） sem= *百分比* （仅限 HP-UX） lan= *百分比* （仅限 HP-UX）
application	*应用程序名称*
file	*文件名[, ...]*
argv1	第一个命令参数 [,]
cmd	命令行正则表达式
user	*用户登录名称[,]*
group	*组名[,]*
or	
priority	从低到高 （范围因平台而异）
size	（值以 MB 为单位） process=*nn （最大值是 4096）* 下面所有类的最大值都是 2048。 global=*nn* application=*nn* device=*nn* transaction=*nn* logicalsystem=*nn*
days	global=*nn* （值以天为单位） application=*nn* process=*nn* device=*nn* transaction=*nn* logicalsystem=*nn*
maintweekday	Sun|Mon|Tue|Wed|Thu|Fri|Sat
collectioninterval	process=ss （值以秒为单位） global=ss
gapapp	空白 unassignedprocesses existingapplicationname other

表 2　　scope 使用的 parm 文件参数 （续）

参数	值或选项
Flush	ss （值以秒为单位） 0 （禁用数据刷新）
zone_app	true false （仅限 Solaris 10 及更高版本）
proccmd 注：仅限 UNIX/Linux。	0 （禁用进程命令记录） nnnn （进程命令的长度数字值。最大值是 1024）
ignore_mt	true（对照系统中的活动核心数规范化的全局类的 CPU 度量报告值） false（对照系统中的活动 CPU 线程数规范化的全局类的 CPU 度量报告值） 无效 （关闭多线程处理）

参数描述

以下是各个 parm 文件参数的描述。

- ID
- Log
- Thresholds
- scopetransactions
- subprocinterval
- gapapp
- wait
- Size
- Mainttime
- Days
- Maintweekday
- javaarg
- Flush
- zone_app
- proccmd
- ignore_mt

ID

系统 ID 值是用于识别系统的字符串。分配的默认 ID 是系统的主机名。如果要修改分配的默认 ID，请确保所有系统都具有唯一的 ID 字符串。此标识符包括在日志文件中以识别收集其数据的系统。最多可以指定 39 个字符。

log

log 参数指定 scope 要收集的数据类型。

- log global 使 scope 将全局记录记录到 logglob 文件中。必须有全局数据记录才可查看和分析系统的性能数据。全局度量不受应用程序或进程数据的记录选项或值的影响。
- log application 使 scope 将活动应用程序记录记录到 logappl 文件中。默认情况下，scope 只记录间隔期间有活动进程的应用程序。
  - parm 文件中的 log application=all 使 scope 在每个间隔将所有应用程序 （包括活动和不活动的）都记录到 logappl 文件中。

    在需要使用应用程序警报的特定环境中，可能需要 application=all 选项。例如，可以在应用程序变为不活动时生成警报 (APP_ALIVE_PROC)。

    启用此选项后，由于每个间隔都要记录所有应用程序，日志文件 logappl 的增长速度会更快。可以使用 utility 程序的 scan 函数监视 scope 日志文件的利用率。
  - 仅在 HP-UX 上可以指定参数 log application=prm，以使 scope 将活动的 Process Resource Manager (PRM) 组记录到 logappl 文件中。如果指定此参数，scope 将不记录 parm 文件中列出的用户定义的应用程序集。此外，所有收集的应用程序度量都将反映 PRM 上下文，并按 APP_NAME_PRM_GROUPNAME 度量分组。

    应用程序记录选项不影响全局或进程数据。
- log process 使 scope 将有关感兴趣的进程的信息记录到 logproc 文件中。第一次创建进程、进程结束、进程超过 parm 文件中为应用程序指定的阈值时，该进程可能会成为感兴趣的进程。进程阈值记录选项不影响全局或应用程序数据。
- log device=disk,lvm, cpu, filesystem 使 scope 将有关各个磁盘、逻辑卷（仅限 HP-UX）、CPU 和文件系统的信息记录到 logdev 文件中。

  如果监视的系统不运行 HP-UX 操作系统，请勿使用 lvm。

  默认情况下，仅记录间隔期间通过磁盘、卷和接口生成 I/O 的相应磁盘、卷和接口。不管所选的日志设备选项如何，始终记录 netif （逻辑 LAN 设备）记录和磁盘记录 (HP-UX)。

  例如，要请求记录各个磁盘、逻辑卷、CPU 和网络接口的记录，但*不*记录各个文件系统的记录，请使用以下设置：

  ```
  log device=disk,lvm,cpu。
  ```

  - 指定 filesystem 时，不管文件系统是否活动，都在每个间隔记录所有装入的本地文件系统。

— parm 文件中的 log device=all 使 scope 在每个间隔将所有磁盘、逻辑卷、CPU 和网络接口设备（包括活动和不活动的）都记录到 logdev 文件中。

启用此选项后，由于每个间隔都要记录所有设备，logdev 文件的增长速度会更快。可以使用 utility 程序的 scan 函数监视日志文件利用率和大小变化。

- log transaction 使 scope 将 ARM 事务记录记录到 logtran 文件中。要使 scope 收集数据，您的系统上必须在运行已通过应用程序响应测量 (ARM) API 检测的进程。（有关详细信息，请参见第 339 页的什么是事务跟踪？。）

  log transaction 参数的默认值是 no resource 和 no correlator。

  要启用资源数据收集（仅限 HP-UX）或相关器数据收集，请指定 **log transaction=resource** 或 **log transaction=correlator**。指定 **log transaction=resource, correlator** 可以同时记录资源数据和相关器数据。

- log logicalsystems 使 scope 将有关逻辑系统的信息记录到 logls 文件中。逻辑系统数据按 parm 文件中指定的间隔定期汇总。

  在 AIX 6.1 TL2 上，LPAR 和 WPAR 的 BYLS 记录可以使用 parm 文件中的 logicalsystems 参数配置。请参见第 20 页的表 2。

日志文件都是自动创建的，而不管日志选项如何。如果禁用特定记录类型，对应的日志文件不会自动从监视的系统删除。

如果指定不带任何选项的 log， scope 仅记录全局和进程数据。

Thresholds

threshold 参数使 scope 仅将严重信息记录到日志文件中，过滤掉不必要不严重的系统详细信息。

以下参数指定各种度量类的阈值。某个数据类的特定实例超过指定阈值时， scope 将记录该实例的记录。

可以指定较低阈值使 scope 记录更多的数据，也可以指定较高阈值使 scope 记录更少的数据，以减少记录的平均记录数。下面是可用的 threshold 参数：

- Procthreshold
- appthreshold
- diskthreshold
- bynetifthreshold
- fsthreshold
- lvthreshold
- bycputhreshold

Procthreshold

procthreshold 参数用于设置活动级别以指定感兴趣的进程的条件。要使用此参数，必须启用进程记录。 procthreshold 只影响记录的进程，不影响其他数据类。

必须在同一参数行指定 threshold 选项 （用逗号隔开）。

进程数据的 procthreshold 选项

cpu
设置 CPU 利用率百分比，进程必须超过该值才能成为 “感兴趣的”进程并被记录。
百分比值是表示 CPU 总体使用情况的实数。例如，cpu=7.5 表示在 1 分钟示例中超过 7.5 个百分比的 CPU 利用率时记录进程。

disk
（在 Linux 和 Windows 上不可用。）设置每秒物理磁盘 I/O 速率，进程必须超过该值才能成为 “感兴趣的”进程并被记录。
值为实数。例如， disk=8.0 表示平均物理磁盘 I/O 速率超过每秒 8 KB 时记录进程。

memory
设置内存阈值，进程必须超过该值才能成为 “感兴趣的”进程并被记录。
值以 MB 为单位，精确到 100 KB。如果设置此选项，会比较内存阈值与 PROC_MEM_VIRT 度量值。将记录超过内存阈值的每个进程，这与磁盘和 CPU 进程记录阈值类似。

nonew
此选项指定如果尚未超过任何阈值，则禁用记录新进程。如果不指定此选项，将记录所有新进程。在 HP-UX 上，如果不指定 shortlived，则仅记录持续时间超过一秒的新记录。

nokilled
此选项指定如果未超过任何阈值，则禁用记录退出的进程。如果不指定此选项，将记录所有终止（退出）的进程。在 HP-UX 上，如果不指定 shortlived，则仅记录超过一秒的终止进程。

shortlived
启用在间隔中记录运行时间少于一秒的进程。（这通常会大幅增加记录的进程数。）如果 scope 在 parm 文件中找到 threshold shortlived，不管 cpu 或 disk threshold 如何，只要删除了 nonew 和 nokilled 选项，就记录 shortlived 进程。默认值是不记录 shortlived 进程。（如果不想记录 shortlived 进程，请不要在 threshold 参数中指定 shortlived。）

process
procthreshold 指定 PROCESS 类的阈值。默认值如下：

- 上一间隔中占用了处理器的 10% 以上 cpu 的进程
- 进程的虚拟内存集大小在 900 MB 以上
- 进程的平均物理磁盘 I/O 速率大于每秒 5 KB

appthreshold

appthreshold 参数用于指定 APPLICATION 数据类的阈值（APP_CPU_TOTAL_UTIL 度量）。阈值条件基于应用程序必须超过才能在日志文件中记录该应用程序的 CPU 利用率百分比。

parm 文件中的默认设置使 scope 记录 CPU 利用率超过 0% 的应用程序。

diskthreshold

diskthreshold 参数用于指定 DISK 类的阈值。DISK 类的阈值条件基于磁盘执行 I/O 的持续时间百分比 （BYDSK_UTIL 度量）。

parm 文件中的默认设置使 scope 记录忙于执行 I/O 超过 10% 持续时间的磁盘的详细信息。

bynetifthreshold

bynetifthreshold 参数指定 NETIF 类的阈值。 Netif 数据类的阈值条件基于网络接口每秒传输的数据包数 （BYNETIF_PACKET_RATE 度量）。

parm 文件中的默认设置使 scope 记录每秒传输超过 60 个数据包的网络接口的详细信息。如果不指定此参数值或注释掉此参数， scope 将记录所有非空闲的网络接口的详细信息。

fsthreshold

fsthreshold 参数指定 FILESYSTEM 类的阈值。file system 数据类的阈值条件基于文件系统占用的磁盘空间的百分比 （FS_SPACE_UTIL 度量）。

parm 文件中的默认设置使 scope 记录占用磁盘空间 70% 以上的文件系统的详细信息。

lvthreshold

lvthreshold 指定 LOGICALVOLUME 类的阈值。逻辑卷数据类的阈值基于每秒 I/O (LV_READ_RATE + LV_WRITE_RATE)。

parm 文件中的默认设置使 scope 记录每秒 I/O 超过 35 的逻辑卷的详细信息。

bycputhreshold

bycputhreshold 参数指定 CPU 类的阈值。CPU 数据类的阈值条件基于 cpu 繁忙的时间百分比 (BYCPU_CPU_TOTAL_UTIL)。

parm 文件中的默认设置使 scope 记录忙碌时间超过 90% 的 CPU 的详细信息。

scopetransactions

scope 收集器本身已通过 ARM（应用程序响应测量）API 调用检测来记录自己的事务。scopetransactions 标记确定是否记录 scope 事务。默认值是 scopetransactions=on；scope 将记录两个事务：Scope_Get_Process_Metrics 和 Scope_Get_Global_Metrics。如果不想记录

这两个 scope 事务，请指定 scopetransactions=off。第三事务是 Scope_Log_Headers，将始终记录该事务，不受 scopetransactions=off 影响。

有关 ARM 的详细信息，请参见第 339 页的什么是事务跟踪？。

subprocinterval

subprocinterval 参数如有指定，则覆盖 scope 用于采集进程数据的默认设置。进程数据和全局数据按 parm 文件中指定的间隔定期记录。但是，scope 每隔几秒钟就探测一次其检测以捕获短期活动。该检测采样间隔默认为 5 秒。进程数据记录间隔必须是 subprocinterval 的偶数倍。有关详细信息，请参见第 35 页的配置数据记录间隔。

在有数千个活动线程或进程的一些系统上，应该将 subprocinterval 设置得更长以减少 scope 总开销。在希望记录很多短期进程的其他系统上，可以考虑将 subprocinterval 设置得更低，即使在这种情况下应密切监视对 scope 开销的影响也是如此。此设置的值必须是 parm 文件中指定的进程记录间隔的乘数。

低 subprocinterval 值将缩小全局度量和除 HP-UX 以外的所有其他操作系统上的应用程序总和之间的差异。

gapapp

parm 文件中的 gapapp 参数控制应用程序数据类的修改，以解释全局 （系统范围）数据和应用程序数据总和之间的任何差异。

应用程序数据从进程级别的检测中获取。全局度量和应用程序总和之间通常存在差异。在进程创建速率很快的系统中，这种差异将很明显。您可以选择以下选项：

- 如果 gapapp 为空，将在应用程序列表中添加名为 gapapp 的应用程序。
- 如果 gapapp = UnassignedProcesses，将在应用程序列表中添加名为 UnassignedProcesses 的应用程序。
- 如果 gapapp = ExistingApplicationName （或） gapapp = other，将向指定应用程序添加与全局值的差异，而不是单独记录并将新条目添加到应用程序列表。

wait

可以使用 wait 参数 （仅限 HP-UX）捕获等待系统资源的进程的详细信息。wait 参数的值可以用百分比指定。当进程等待系统资源 cpu、disk、mem、sem 和 lan 的间隔百分比超过 wait 参数中指定的值时，将该进程的详细信息记录到 logproc 文件中。

有关值和选项，请参见 scope 使用的 parm 文件参数。

例如，如果进程记录间隔定义为 60 秒， CPU 的 wait 参数设置为 50%，则在 logproc 文件中捕获任何等待 CPU 时间超过或等于 30 秒的进程。

Size

size 参数用于设置任何原始日志文件的最大大小（以 MB 为单位）。不能设置小于 1 MB 的大小。

scope 收集器在初始化时读取这些规范。如果收集期间有任何日志文件达到其最大大小，这些文件将继续增长直到达到 mainttime，这时它们将自动回滚。在回滚期间，从日志文件删除最旧的 25% 数据。原始日志文件设计为在不受 size 参数限制的情况下最长保留一年的数据。请参见第 3 章 utility scan 报告详细信息部分的第 69 页的日志文件内容摘要和第 70 页的日志文件空余空间摘要。

如果更改了 parm 文件中的 size 规范， scope 会在启动时检测到更改。如果最大日志文件大小减少到不能容纳现有数据的值，将在 scope 启动时自动调整大小。如果指定的新最大值能够容纳现有数据，则不采取任何操作。

resize 命令在 TMPDIR 环境变量设置的目录中创建新文件 scopelog，然后再删除原始日志文件。请参见如何使用。

对日志文件最大大小所做的更改均在 mainttime 参数中指定的时间生效。

不管 size 参数如何设置，scope 日志文件的最大大小还受一年期间存储的数据量的限制。原始 scope 日志文件不能包含一年以上的数据，因此当日志增长到该时间长度时，会覆盖超过一年的数据。有关如何对所需的超过一年的数据创建存档日志文件的信息，请参见第 121 页的 extract。

Mainttime

scope 在每天的特定时间回滚日志文件（如果需要）。默认时间可以使用 mainttime 参数更改。例如，设置 **mainttime=8:30** 即在每天 8:30 am 执行日志文件维护。

建议将 mainttime 设置为系统利用率最低的时间。

日志文件维护仅滚出超过一天的数据，因此如果日志文件（如 logproc）增长很快，在 24 小时内就达到限制，其大小可能超过配置的大小限制。

Days

days 参数指定任何原始数据日志文件在既定时间点可以存储数据的最长天数。此参数值必须在 1 到 365 之间。此参数使 scope 数据收集器维护日志文件。

在数据收集期间，如果日志文件数据的天数达到 days 参数中指定的天数，则继续收集数据，直到达到 maintweekday 参数中指定的日期。一旦达到 maintweekday，将在 maintime 自动回滚日志文件。在回滚期间，从日志文件删除 days 参数达到最大值之后收集的数据。

数据收集期间回滚日志文件时，如果在 days 参数之前的某一天达到 size 参数中指定的值， size 参数将覆盖 days 参数。

例如，如果 parm 文件中设置为 “size global=20” 和 “days global=40”，且在日志文件中记录 40 天的数据之前日志文件就达到了 20 MB 的最大大小，这时将根据 size 参数执行日志文件回滚。

Maintweekday

maintweekday 参数指定如果满足 days 参数则在周几执行日志文件回滚。回滚将从 maintime 开始。

例如，如果 parm 文件中设置为 “maintweekday=Mon”，一旦满足 days 参数中指定的值，即在 “星期一” maintime 执行日志文件回滚。建议将 maintweekday 值设置为一周中系统利用率低的那一天。

maintweekday 参数是可选参数。如果在 parm 文件中指定了 maintweekday 参数，应该将其与 days 参数一起使用。如果在 parm 文件中没有与 days 参数结合使用，将不会考虑此参数。如果在 parm 文件中指定了 days 参数，但未指定 maintweekday，则默认值是 “maintweekday=Sun”。

例如，如果 “daysglobal=30” 、“application=20” 、“process=30” 、“device=20” 、“transaction=10”、“maintweekday=Wed”，并且日志文件达到了 days 参数中指定的天数，将继续收集数据，直到达到 maintweekday 中指定的日期。一旦达到 maintweekday，将执行日志文件回滚，从日志文件开头删除超过天数的数据。此维护在 maintime 执行。

javaarg

此参数仅在 UNIX/Linux 上有效。

javaarg 参数是可以设置为 true 或 false 的标记。它 “只” 影响 proc_proc_argv1 度量的值。

javaarg 设置为 false 或 parm 文件中未定义该参数时，进程的 proc_proc_argv1 度量始终设置为命令字符串中第一个参数的值。

javaarg 设置为 true 时，用命令字符串中可以找到的 class 或 jar 规范覆盖 proc_proc_argv1 度量（仅限 java 进程）。换句话说，对于文件名为 java 或 jre 的进程，如果在操作系统提供的参数列表中可以找到数据，则用无前导短划线的第一个参数覆盖 proc_proc_argv1 度量，不使用 -classpath 或 -cp。

这听起来很复杂，但是当您的系统上正在运行 java 进程时就很清楚了：设置 **javaarg=true**，用 class 或 jar 名称记录 proc_proc_argv1 度量。如果要定义特定于 java 的应用程序，此设置会很有用。类名在 proc_proc_argv1 中时，可以使用 argv1= application 限定符按类名定义应用程序。

Flush

flush 参数指定要记录应用程序和设备数据的所有实例的数据记录间隔（以秒为单位）。flush 间隔必须在 300 到 32700 之间，并且是 300 的偶数倍。

对于应用程序和设备数据的所有实例， flush 间隔的默认值是 3600 秒。

可以将值指定为 0 （零）以禁用 flush 参数。如果 flush 参数设置为 0， scope 不会记录不符合 parm 文件中指定的阈值的应用程序和设备数据。

zone_app

zone_app 标记指示性能收集组件收集特定于 Solaris 区域的数据。zone_app 标记设置为 true 时，会影响所有应用程序类 (APP_*) 度量的收集。将忽略 parm 文件中列出的所有用户定义的应用程序集，并根据安装了性能收集组件的计算机上正在运行的区域收集应用程序度量。

例如，假定一台 Solaris 计算机以两个非全局区域“zone1”和“zone2”运行。性能收集组件将忽略 parm 文件应用程序集，而创建“global”、“zone1”和“zone2”三个应用程序。每个应用程序的性能测量都将基于从各个区域下运行的进程中获取的度量值。例如，区域“zone1”的 APP_CPU_TOTAL_UTIL 度量将根据 zone1 中运行的所有进程的 cpu 利用率值进行计算。

未启用 zone_app 标记时，或者当性能收集组件在 Solaris 9 或更低版本上运行时， APP_LS_ID 度量将报告不可用 (na) 值。将根据 parm 文件中列出的用户定义的应用程序集执行应用程序分组。

仅 Solaris 10 和更高版本支持区域功能。

proccmd

此参数仅在 UNIX/Linux 上有效。

使用 proccmd 参数可将进程命令记录到 HP Operations Agent 数据存储。可以在此参数中指定数字值的进程命令长度。最大数字值是 1024。默认情况下，此参数值设置为 0，表示禁止记录进程命令。

ignore_mt

如果将此参数设置为 true，性能收集组件在对照监视的系统上的活动内核数规范化度量值后，记录全局类的所有 CPU 相关的度量。

当此参数设置为 false 时，性能收集组件在对照监视的系统上的线程数规范化度量值后，记录全局类的所有 CPU 相关的度量。

如果系统上禁用多线程处理属性，性能收集组件会忽略此参数。最后，GBL_IGNORE_MT 度量的值记录为 true。

如果在 Windows、Linux 或 Solaris 系统上启用或禁用多线程并行处理 (SMT)，则必须重新启动系统。

应用程序定义参数

以下参数属于应用程序定义参数：application、 file、 user、 group、 cmd、 argv1 和 or。

性能收集组件将逻辑上相关的进程分组到一个应用程序，以记录进程对内存和 CPU 等计算资源的综合影响。

在 PRM 模式 （仅限 HP-UX）中，记录活动的 PRM 组，忽略 parm 文件中列出的用户定义的应用程序集。

应用程序可以是同时还受用户名、组名或参数选择限定的一组文件 （基本程序名称）、一组命令或文件和命令的组合。所有应用程序限定符都可以单独使用，也可以组合使用。例如，如果同时使用 cmd 和 user 限定符，进程必须同时满足命令字符串规范和用户名规范才能归类到该应用程序。下面详细讨论了每个限定符。

系统上的任何进程都只属于一个应用程序。一个进程不会计入两个或两个以上的应用程序中。

Application

应用程序名称定义组合多个进程并报告它们的组合活动的应用程序或类。

- 应用程序名称用于识别应用程序，最多由 19 个字符组成。
- 应用程序名称可以小写或大写，可以包含字母、数字、下划线和嵌入空格。不要在 parm 文件中使用同一应用程序名称多次。
- 等号 (=) 是应用程序关键字和应用程序名称之间的可选符号。
- application 参数必须用在引用它的 file、user、group、cmd、argv1 和 or 参数的任何组合之前，并且所有这些参数都对照最后一个应用程序工作负载定义应用。
- 每个参数最多可包含 170 个字符，可包括回车符，不允许使用继续符。如果文件列表长于 170 个字符，请在下一行另一个 file、 user、 group、 cmd 或 argv1 语句之后继续列表。
- 最多可以定义 998 个应用程序。性能收集组件预定义名为 other 的应用程序。other 应用程序收集 parm 文件中的 application 语句未捕获的所有进程。

  例如：

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

  application xyz
  file xyz*,startxyz
  ```

  所有应用程序合在一起最多可以有 4096 条 file、user、group、argv1 和 cmd 规范。上面的示例包括九条文件规范。xyz* 只算作一条规范，即使它可以与多个程序文件匹配。）

  如果一个程序文件包括在多个应用程序中，它将记录在第一个包含它的应用程序中。

默认 parm 文件包含一些可以修改的示例应用程序。 examples 目录还包含其他可以复制到 parm 文件并根据需要进行修改的示例 （在名为 parm_apps 的文件中）。

File

file 参数指定属于应用程序的程序文件。包括这些程序的所有交互或后台执行。它应用于发出的最后一个 application 语句。如果未找到 application 语句，则生成错误。

文件名可以是以下任意一种：

- 单个 UNIX 程序文件，比如 vi。
- 一组 UNIX 程序文件 （用通配符表示），比如 xyz*。在这种情况下，包括任何以字母 xyz 开头的程序名称。带有通配符的文件规范在允许的最大值中只算作一条规范。

file 参数的名称长度限制为 15 个字符。等号 (=) 是 file 参数和文件名之间的可选符号。

可以在同一个参数行输入多个文件名 （用逗号隔开），或在单独的文件语句中输入多个文件名。文件名不能用路径名称限定。文件规范与设置为进程的 argv[0] 值 （通常为基本名称）的特定度量 PROC_PROC_NAME 进行比较。例如：

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

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

如果不指定 file 参数，所有满足其他参数的程序都符合要求。

星号 (*) 是除 cmd 限定符 （请参见下面）以外 parm 文件应用程序限定符唯一支持的通配符。

argv1

argv1 参数指定根据 PROC_PROC_ARGV1 度量的值为应用程序选择的进程。它通常是命令行的第一个参数，但 javaarg=true 时除外，这种情况下它是 java 进程的 class 或 jar 名称。此参数使用的模式匹配语法与 parm 参数使用的相同，就像 file= 和 user=。每个选择条件都可以用星号作为通配符匹配，一行可以有多个用逗号分隔的选择。

例如，以下应用程序定义获取命令行中第一个参数是 -title、 -fn 或 -display 的所有进程：

```
application = xapps
argv1 = -title,-fn,-display
```

以下应用程序定义获取特定 java 应用程序 （当 javaarg=true 时）：

```
application = JavaCollector
argv1 = com.*Collector
```

下面显示了 argv1 参数与 file 参数的可用组合方式：

```
application = sun-java
file = java
argv1 = com.sun*
```

cmd

cmd 参数指定命令字符串在应用程序中包括的进程，由所执行的程序及其参数组成。与其他选择参数不同，此参数除了使用星号字符以外还允许扩展使用通配符。

与正则表达式类似，允许使用扩展模式匹配。有关模式条件的完整描述，请参见 UNIX 手册页的 fnmatch 部分。与其他参数不同，此参数每行只能有一个选择，但是可以有多行。

下面显示了 cmd 参数的用法：

```
application = newbie
cmd = *java *[Hh]ello[Ww]orld*
```

User

user 参数指定哪些用户 （登录名称）属于应用程序。格式如下：

application <应用程序名称>

file <文件名>

user [<域名>]\<用户名>

user 参数中的域名是可选的。要指定非本地系统的用户名，必须指定域名。

例如：

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

如果在 user 参数中指定用户名而未指定域名，则将用户名视为本地系统的用户名。

例如：

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

只能使用星号通配符 (*) 确保星号 (*) 之前和星号 (*) 之后具有类似字符串的用户名属于应用程序。如果不指定 user 参数，所有满足其他参数的程序都符合要求。

user 参数的名称长度限制为 15 个字符。

Group

group 参数指定哪些用户组名称属于应用程序。

例如：

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

如果不指定 group 参数，所有满足其他参数的程序都符合要求。

group 参数的名称长度限制为 15 个字符。

Or

使用 or 参数可对同一应用程序应用多个应用程序定义。在单个应用程序定义中，进程必须至少与参数的一个类别匹配。用 or 参数分隔的参数视为独立定义。如果进程与任意定义的条件匹配，则它属于应用程序。

例如：

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

上面的示例定义应用程序 (Prog_Dev_Group2) 由用户 **julie** 运行的所有程序以及用户 **mark** 执行的其他程序 （vi、 store 和 dmp）组成。

Priority

可通过在 priority 参数中指定值，将应用程序的进程限制在指定范围内。

例如：

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

进程的优先级可以在 **-511** 到 **255** 之间，具体取决于运行性能收集组件的平台。优先级可以随进程的生命周期而变化。此计划程序调整时间共享进程的优先级。也可以通过编程或在执行时更改优先级。

按输入的顺序处理 parm 文件，限定符的第一个匹配项定义特定进程所属的应用程序。因此，更具体的应用程序定义优先于更常规的定义是正常的。

应用程序定义示例

以下示例显示了应用程序定义。

```
application firstthreesvrs
cmd = *appserver* *-option[123]*
application oursvrs
cmd = *appserver*
user = xyz,abc

application othersvrs
```

```
cmd = *appserver*
cmd = *appsvr*
or
argv1 = -xyz
```

下面是如何使用前面的 parm 文件记录多个程序的示例。

命令字符串	用户登录	应用程序
/opt/local/bin/appserver -xyz -option1	**xyz**	firstthreesvrs
./appserver -option5	**root**	othersvrs
./appserver -xyz -option2 -abc	**root**	firstthreesvrs
./appsvr -xyz -option2 -abc	**xyz**	othersvrs
./appclient -abc	**root**	other
./appserver -mno -option4	**xyz**	oursvrs
appserver -option3 -jkl	**xyz**	firstthreesvrs
/tmp/bleh -xyz -option1	**xyz**	othersvrs

配置数据记录间隔

scope 对进程数据所用的默认收集间隔是 60 秒，对全局数据和其他所有数据类所用的默认收集间隔是 300 秒。可在 parm 文件中使用 collectioninterval 参数覆盖此值。值必须满足以下条件：

- 进程数据的收集间隔可以配置为 5 到 60 秒之间的值，步长为 5 秒。进程数据的收集间隔必须是 subproc 间隔的倍数 （请参见 subprocinterval），且必须是全局收集间隔的整除数。
- 全局数据的收集间隔可以配置为以下某个值：15、30、60 和 300 秒。全局收集间隔必须大于或等于进程间隔，且是进程收集间隔的倍数。全局收集间隔应用于全局度量和所有非进程度量类（比如文件系统和应用程序）。

示例 1：

```
collectioninterval process=15, global=30
subprocinterval = 5
```

在此示例中，进程数据的收集间隔设置为 15 秒，全局数据和所有其他类数据的收集间隔设置为 30 秒， subprocinterval 设置为 5 秒：

进程数据的收集间隔：

- 是有效值 （此值是 5 的倍数）
- 是 subprocinterval 5 秒的倍数 (15%5 = 0)
- 是全局数据收集间隔 30 秒的整除数 (30%15 = 0)

全局数据的收集间隔：

- 是有效值 （此值可以是 15、 30、 60 或 300）
- 是进程间隔 15 秒的倍数 (30%15 = 0)

因此收集间隔的这些值有效。

示例 2:

```
collectioninterval process=15, global=30
subprocinterval = 10
```

在此示例中，进程数据的收集间隔设置为 15 秒，全局数据和所有其他类数据的收集间隔设置为 30 秒， `subprocinterval` 设置为 10 秒：

进程数据的收集间隔

- 是有效值 （此值是 5 的倍数）
- *不*是 `subprocinterval` 10 秒的倍数 (15%10 != 0)
- 是全局收集间隔 30 秒的整除数 (30%15 = 0)

全局数据的收集间隔为 30 秒

- 是有效值 （此值可以是 15、 30、 60 或 300）
- 是进程间隔 15 秒的倍数 (30%15 = 0)

因此收集间隔的这些值无效。

对于 VMware ESX Server 上的性能收集组件，全局数据和所有其他非进程数据的数据类的记录间隔可以配置为 60 或 300 秒。

在启用了超线程 / 多线程并行处理的系统上规范化 CPU 度量

在启用了超线程 / 多线程并行处理 (HT/SMT) 的系统上，物理 CPU 支持两个或更多硬件线程。因此，可以在硬件线程上同时运行多个软件进程或线程。在配备有多核处理器的系统上，可以在单独的内核上同时运行多个线程。

性能收集组件提供了若干个 CPU 相关的度量，帮助您分析和了解监视的系统的 CPU 利用率。默认情况下，在所有启用了 HT/SMT 的系统上，性能收集组件通过对照监视的系统上可用的线程数规范化收集的数据来计算所有 CPU 相关的度量值。当单个线程完全占用整个 CPU 内核时，使用基于线程的规范化算出的值并不总是体现 CPU 利用率的真实情况。

本版本的 HP Operations Agent 引入了新配置参数 `ignore_mt`，使您能够将性能收集组件配置为记录已使用基于内核的规范化算出的 CPU 相关数据。用基于内核的规范化算出的度量值能更准确地反映 CPU 利用率的状况，从而帮助您在分析系统性能时作出更有效的决策。

记录使用基于内核的规范化算出的度量

在 HP-UX 上，可以将性能收集组件配置为记录用基于内核的规范化算出的所有 CPU 相关度量。在其他平台上，可以将性能收集组件配置为在记录前使用基于内核的规范化计算 GLOBAL 类的 CPU 相关度量。

要将性能收集组件配置为使用基于内核的规范化计算 CPU 相关的度量，请执行以下步骤：

在 HP-UX 上

1. 以具有根特权的身份登录到系统。
2. 根据要求配置 parm 文件。不要在 parm 文件中设置 `ignore_mt` 标记。

   在 HP-UX 上， parm 文件中的 `ignore_mt` 标记的值不影响性能收集组件的操作。

3. 根据需要定义警报规则。
4. 运行以下命令：

   midaemon -ignore_mt
5. 使用以下命令启动 HP Operations Agent：

   opcagt -start

   性能收集组件开始使用基于内核的规范化记录所有 CPU 相关的度量 （所有类）。

在其他平台上

1. 以具有根特权或管理特权的身份登录到系统。
2. 根据要求配置 parm 文件。将 parm 文件中的 `ignore_mt` 标记设置为 true。
3. 根据需要定义警报规则。
4. 使用以下命令启动 HP Operations Agent：

   在 Windows 上

   %ovinstalldir%bin\opcagt -start

   在 HP-UX、Linux 或 Solaris 上

   /opt/OV/bin/opcagt -start

   在 AIX 上

   /usr/lpp/OV/bin/opcagt -start

   性能收集组件开始使用基于内核的规范化记录 GLOBAL 类的 CPU 相关度量。

停止和重新启动数据收集

`scope` 收集器和其他关联的进程设计为连续运行。只有在以下某种情况下才应停止它们：

- 正在将性能收集组件软件更新到新版本。
- 正在添加或删除事务配置文件 ttd.conf 中的事务。（有关详细信息，请参见第 339 页的什么是事务跟踪？。）
- 正在修改事务配置文件 ttd.conf 中的分发范围或服务级别目标 (SLO)。（有关详细信息，请参见第 339 页的什么是事务跟踪？。）
- 正在更改 parm 文件并要使更改生效。对 parm 文件所做的更改仅在启动 scope 时生效。
- 正在使用 utility 程序的 resize 命令调整性能收集组件日志文件的大小。
- 正在关闭系统。
- 正在添加硬件或修改配置更改。所做的更改仅在启动 scope 时生效。

停止数据收集

ovpa 和 mwa 脚本的 stop 选项可确保停止 scope 和其他性能收集组件进程时不丢失数据。

要手动停止数据收集，请输入以下命令：

- *在 Windows 上：*

  *%ovinstalldir%*bin**\ovpacmd -stop**

- *在 HP-UX、Linux 和 Solaris 上：*

  /opt/perf/bin/ovpa -stop

- *在 AIX 上：*

  /usr/lpp/perf/bin/ovpa -stop

scope 不记录 NFS 数据，但可以通过 GlancePlus 在本地文件系统上查看 NFS 数据。

重新启动数据收集

在停止性能收集组件进程或更改了配置文件后要使更改生效时，有多种选项可用于重新启动数据收集。

要在系统关闭后或停止系统后启动 scope 和其他性能收集组件进程，请使用 **< 安装目录 >/ovpa start**（如果使用的是 coda）。这里的安装目录指的是安装性能收集组件的目录。

要在 scope 和其他进程正在运行时重新启动它们，请使用 **< 安装目录 >/ovpa restart**（如果使用的是 coda）。这里的安装目录指的是安装性能收集组件的目录。此命令停止当前正在运行的进程并再次启动它们。

重新启动 scope 时，性能收集组件继续使用停止程序之前使用的相同日志文件（logglob、logappl、logproc、logdev、logtran、logls 和 logindx）。新记录追加到现有文件的末

尾。如果要将数据收集到新文件集中，并且不保留任何历史信息，则应重命名 scope 日志文件或将其存档，并在重新启动前一并删除*所有* scope 日志文件，因为文件之间会同步数据。

ttd.conf 配置文件中的 SEM_KEY_PATH 条目用于在 UNIX 平台上为 ttd 和 midaemon 进程中使用的信标生成 IPC 键。使用的默认值是 /var/opt/perf/datafiles。

如果 midaemon 或 ttd 由于 sem id 冲突没有响应，则可以更改 SEM_KEY_PATH 的值。

夏令时

夏令时期间，系统时间在相关时区中回拨一小时。此时，数据收集停止一小时，直到系统时间与上一条记录的时间戳同步。

关闭夏令时时，系统时间前进一小时，因此下一条记录的时间戳也前进一小时。这样即使不停止数据收集，上一条记录之后也会有一小时的时间缺口。

手动更改系统时间

将系统时间手动往回拨时，数据收集停止，命令（比如 **perfstat**）不工作。系统时间回拨时，这些实用程序挂起。要继续记录数据和收到所有命令的响应，请执行以下步骤：

1 运行以下命令：

```
ovc -stop coda
```

2 备份 <*数据目录*>\datafiles\ 目录下的 coda.* 文件，并删除它们。

3 运行以下命令：

```
ovc -start coda
```

有效数据收集管理

有效的性能分析取决于访问收集的性能数据的难易程度。此部分讨论活动（比如管理日志文件、数据存档和系统分析）的有效策略，以使数据收集进程更容易、更有效和更有用。

控制日志文件所用的磁盘空间

性能收集组件具有自动管理自己创建的日志文件的功能。您可以针对特殊用途配置此自动进程或使用备用手动进程。自动日志文件管理进程的工作方式如下：

- 每个日志文件都有配置的最大大小。首次安装性能收集组件时，会提供默认最大大小。但是，您可以重新配置这些值。
- 每个日志文件达到其最大大小时，scope 数据收集器都会在 mainttime 执行“回滚”。此回滚期间会删除日志文件中最旧的 25% 数据，为要添加的新数据留出空间。

对 scope 收集的数据和 DSI 记录进程收集的数据而言，自动日志文件维护很相似但又不完全相同。有关 DSI 日志文件维护的详细信息，请参见第 255 页的数据源集成概述。

设置 mainttime

通常 scope 只在每天的特定时间执行日志文件回滚。这是为了确保在非高峰时段执行回滚操作，不影响系统的正常使用。对日志文件执行回滚检查的时间通过 parm 文件中的 mainttime 参数设置。

设置日志文件最大大小

选择日志文件最大大小应是已用磁盘空间与可供即时分析的历史数据量之间的一种平衡。小的日志文件可节省磁盘空间，但限制了 Performance Manager 等工具可以绘制的时间量。下面讨论了重新配置 scope 日志文件大小的一些方式。

scope 将不同类型的数据记录到各自的日志文件中。这是为了让您能够选择专用于每种类型的磁盘空间。例如，全局数据不大占用空间，但您往往每个月要访问一次以绘制数据图形，以便为趋势分析和容量规划练习提供很好的统计基础。

进程数据可能会比全局数据占用更多的磁盘空间，因为每分钟都可能有很多感兴趣的进程。而且，进程数据的时间值没有全局数据的那么大。了解有关今天和昨天都运行过的进程的详细信息可能非常重要。有时可能需要了解上周运行了哪些进程。但是，不可能需要确切地了解上个月运行了哪些进程。

典型用户可以决定保持以下数据联机：

- 用于趋势分析的三个月的全局数据
- 用于排除故障的一个月的进程数据
- 用于趋势分析和负载平衡的三个月的应用程序数据
- 用于磁盘负载平衡的两个月的设备数据

可以编辑 parm 文件为各个日志文件设置 size 参数。 size 以 MB 为单位指定。例如：

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

保留特定天数的数据所需的 MB 值可能随数据类型、系统配置和系统活动而变化。确定系统上需要多大空间来保存日志文件的最佳方式是收集一周左右的数据，然后使用 utility 程序的 resize 命令更改日志文件大小。resize 命令扫描日志文件，从而确定每天记录的数据量。然后为您从天换算到 MB。此函数还会更新 parm 文件。

管理进程大小调整

设置了自动日志文件维护后，就不需要其他活动。当日志文件达到配置的最大大小时， scope 将自动调整它们的大小。

scope 在 parm 文件中指定的 mainttime 回滚日志文件。如果编辑 parm 文件并重新启动 scope，日志文件在 mainttime 到来时才会回滚到新大小。在指定的 mainttime 时间必须正在运行 scope，否则日志文件可能永远不会回滚。

在两次维护之间的时间段内，日志文件可以超过配置的最大大小，这不会导致立即回滚。

不要将日志文件大小调整到不能保留一整天的数据。这意味着，日志文件可以增长到至少保留一天的数据再回滚。这通常不成问题，但如果将 parm 文件参数设置为收集大量的进程数据或应用程序数据，或者将大小值设置得太小，就会导致日志文件在回滚前远远超过配置的最大大小。

scope 会按 parm 文件中指定的全局数据收集间隔定期检查日志文件所在的文件系统上可用的磁盘空间。如果可用的磁盘空间低于 1 MB， scope 会执行以下操作以确保它不占用更多的可用空间：

- 立即执行日志文件维护，而不等到定期日志文件维护时间。如果有日志文件超过最大大小（且内含超过一天的数据），则进行日志文件回滚。
- 如果在执行日志文件维护后可用磁盘空间仍然不超过 1 MB，scope 则将相应的错误消息写入其 status.scope 文件并停止收集数据。

数据存档

自动日志文件管理使分析时有最新的日志文件数据可用。将来自原始日志文件的数据存档。进程数据和全局数据按 parm 文件中指定的间隔定期记录。有关详细信息，请参见第 35 页的配置数据记录间隔。当日志文件达到最大大小时，删除旧数据以为新数据留出空间。如果要使日志文件数据保留更长的时间，应启动数据存档进程。选择哪个进程取决于您的需要。下面是一些可行的做法：

- 将原始日志文件调整到很大，剩下的交给自动日志文件维护去完成。这是最轻松的存档方法，但几个月后它可能消耗大量磁盘空间。
- 将原始日志文件中的数据提取到提取的存档文件，然后再将其从原始日志文件中删除。制定一个过程，将存档文件复制到磁带等长期存储设备直到需要时再取文件。
- 仅将原始日志文件的一部分提取到提取的存档文件。例如，由于进程数据容量高、时间值小，您可能不想将进程数据存档。
- 可以使用上述方法的某些组合。

建议使用以下过程进行数据存档：

- 将原始日志文件大小调整到能容纳要保持联机的详细信息数据。
- 每周将详细的原始数据复制到要移到脱机存储的文件一次。

管理进程存档

按上文所述调整原始日志文件大小。选择至少能容纳两周的数据的日志文件大小（假定每周仅进行一次存档处理）。

每周安排运行 extract 程序的进程一次。下例显示了每周执行处理的脚本：

```
#extract -gapdt -xm
```

当月的每一周都会将数据追加到前一周的数据之后。新的一个月开始时，extract 新建存档日志文件，并将该周的数据拆分成合适的月度存档日志文件。这些日志文件命名为 rxmo，后跟四位数的年份和两位数的月份。（例如，1999 年 12 月的数据会存储在名为 rxmo199912 的文件中。）

在每个月的月初，上个月的日志文件完成，并开始新建日志文件。因此，只要存在多个 rxmo 日志文件，就可以安全地将最新日志文件以外的其他所有日志文件复制到脱机存储，直到需要时再取文件。需要访问存档数据时，恢复所需的存档文件，然后使用 extract 或 utility 程序进行访问。

根据系统配置和活动级别，一个月累积的磁盘空间量可能会很大。如果是这样，可以将上例中的 -xm 替换成 weekly 命令 -xw，从而将详细信息存档文件分为较小的文件。

另一个方法是选择不存档详细的进程数据。

在前面示例中讨论的详细提取方法保留了所有收集的性能数据。如果需要深入调查某一情况，可以将这些文件恢复到磁盘进行分析。

可以使用 extract 程序合并多个提取的文件中的数据，或划分数据子集更方便传输和分析。

例如，可以合并若干年度提取的文件的数据以执行多个年份的趋势分析。

不支持将 HP Operations Agent 节点上创建的日志文件移到使用旧版本 HP Performance Agent 的系统。

3 使用 HP Operations Agent

配置数据收集机制后，如果要同时使用代理程序和 HPOM，可通过在节点上部署 HPOM 策略使用操作监视组件的不同组件。例如，如果部署测量阈值策略，则监视代理程序组件开始工作。尽管可以在 HPOM 策略中提供大多数监视详细信息，但某些组件可能仍然需要额外配置才能在节点上执行。

配置监视代理程序

可以启动和配置监视代理程序来监视不同的源。在节点上部署测量阈值策略后，监视代理程序组件开始工作。代理程序根据策略中的规范开始监视以下几类源的对象：

- **外部**：可将数字值发送到代理程序的外部程序。
- **嵌入式性能组件**：代理程序的数据存储中可用的数据。
- **MIB**：管理信息库 (MIB) 中的条目。
- **实时性能管理**：Windows 性能日志和警报。
- **程序**：由 HPOM 启动的将数字值发送到代理程序的外部程序。
- **WMI**：WMI 数据库。

要使用 HPOM 策略监视来自以上源的对象，请参见以下主题：

- *HPOM for Windows：HPOM for Windows 联机帮助*中的*事件策略编辑器*部分。
- *HPOM on UNIX/Linux：*《HPOM for UNIX 9.00 概念指南》（*HPOM for UNIX 9.10 Concepts Guide*）中的“实现消息策略（Implementing Message Policies）”部分。

配置代理程序以监视 MIB 对象

在节点上部署测量阈值策略 （源类型设置为 MIB）之后，监视代理程序就开始查询可以用 `public` 公共字符串访问的 MIB 对象。如果要将监视代理程序配置为使用非默认的公共字符串，请执行以下步骤：

1 以具有根特权或管理特权的身份登录到节点。

2 转到命令提示符 (shell) 处。

3 转到以下目录：

Windows：

%ovinstalldir%`bin`

HP-UX、Solaris 或 Linux：

`/opt/OV/bin`

AIX：

`/usr/lpp/OV/bin`

4 运行以下命令：

- 要使用非默认的公共字符串：

  `ovconfchg -ns eaagt SNMP_COMMUNITY` < *公共字符串* >

  在此实例中， < *公共字符串* > 是您选择的非默认公共字符串。

- 要使用不同的公共字符串：

  `ovconfchg -ns eaagt SNMP_COMMUNITY_LIST`
  < *公共字符串列表* >

  在此实例中，<*公共字符串列表*> 是您选择的公共字符串的逗号分隔列表。HP Operations Agent 按您用上面的命令指定的顺序处理公共字符串列表。

例如：

ovconfchg -ns eaagt SNMP_COMMUNITY_LIST "C1,C2,C3"

HP Operations Agent 先尝试与节点建立 SNMP 会话，并尝试使用公共字符串 C1 对 OID 执行 SNMP Get 操作。如果操作不成功， HP Operations Agent 将用公共字符串 C2 执行同一操作，依此类推。

如果 HP Operations Agent 无法使用通过 SNMP_COMMUNITY_LIST 指定的所有公共字符串进行访问，它将尝试使用通过 SNMP_COMMUNITY 指定的公共字符串。如果代理程序用所有指定的公共字符串都未能获取数据，它将开始使用默认公共字符串 public。

监视子 OID

创建 MIB 测量阈值策略时，必须将 MIB ID 字段设置为要监视的 MIB 对象的 OID。如果要监视对象下一层的子 OID，请按以下格式在 MIB ID 字段中输入 OID：

<*父 OID*>**.***

例如，如果要监视对象 .1.3.6.1.2.1.2.2.1 的所有子 OID，必须在 MIB ID 字段中指定 **.1.3.6.1.2.1.2.2.1.***。

还可以按以下格式指定子实例的名称或描述的位置：

<*父 OID*>**.*:**<*n*>

在此实例中， <*n*> 是子实例的名称或描述的位置。

例如，如果要监视对象 .1.3.6.1.2.1.2.2.1 的由 .1.3.6.1.2.1.2.2.1 的 MIB 子树中第二个元素描述的所有子 OID，必须在 MIB ID 字段中指定 **.1.3.6.1.2.1.2.2.1.*:2**。

监视的对象的持久性

此版本的 HP Operations Agent 会定期存储监视的对象和会话变量的值，以便在发生中断或故障的情况下还保留着这些值。但是，您可以将代理程序配置为不保留监视的对象和会话变量的值。

要将代理程序配置为不保留监视的值，请执行以下步骤：

1 以具有必要特权的身份登录到节点。

2 在命令提示符处运行以下命令：

```
ovconfchg -ns eaagt set OPC_MON_SAVE_STATE FALSE
```

从现在开始，代理程序将不再保留监视的对象和会话变量的值。

配置事件拦截器

默认情况下，事件拦截器可收集来自于远程管理站或启用了 SNMP 的设备的 SNMP 陷阱，然后根据配置生成相应事件。可通过配置以下属性修改事件拦截器的默认行为：

- SNMP_TRAP_PORT：默认端口是 **162**。可以将此值修改为 HP Operations Agent 节点上的任何可用端口。
- SNMP_TRAP_FORWARD_DEST_LIST：可使用此属性设置要转发所有可用 SNMP 陷阱的远程管理站的地址。可以指定由逗号分隔的多个系统名称（及端口详细信息）。
- SNMP_TRAP_FORWARD_ENABLE：默认情况下，此属性设置为 **FALSE**。将此属性设置为 **TRUE** 后，可允许事件拦截器将 HP Operations Agent 节点上可用的 SNMP 陷阱转发到远程计算机或管理站。

- SNMP_TRAP_FORWARD_COMMUNITY：可使用此属性指定传入陷阱的源计算机的公共字符串和要接收被转发 SNMP 陷阱的目标计算机。源计算机的公共字符串必须与目标计算机的公共字符串匹配。
- SNMP_TRAP_FORWARD_FILTER：可使用此属性按可用 SNMP 陷阱的 OID 过滤这些陷阱，并只将选择的陷阱转发到远程计算机。过滤机制中可使用通配符 (*)。例如，如果将此属性设置为 **1.2.3.*.*.***，则事件拦截器将转发 OID 以 1.2.3 开头的所有 SNMP 陷阱。默认情况下，允许事件拦截器转发陷阱时，会转发所有可用陷阱。

  如果源计算机的公共字符串与目标计算机的公共字符串不匹配，则陷阱转发功能失效。

要修改事件拦截器的默认行为，请执行以下步骤：

1 以具有必要特权的身份登录到节点。

2 在命令提示符处运行以下命令：

- 要修改端口号，请运行以下命令：

  ovconfchg -ns eaagt set SNMP_TRAP_PORT < *端口号* >

  必须为 < *端口号* > 指定整数值。确保指定的 < *端口号* > 可用。

- 要允许事件拦截器将 SNMP 陷阱转发到远程计算机，请运行以下命令：

  ovconfchg -ns eaagt set SNMP_TRAP_FORWARD_ENABLE TRUE

- 如果允许事件拦截器将 SNMP 陷阱转发到远程计算机，请运行以下命令指定目标计算机的详细信息：

  ovconfchg -ns eaagt set SNMP_TRAP_FORWARD_DEST_LIST "< *计算机名称* >:< *端口* >**"**

  < *计算机名称* > 是要接收 SNMP 陷阱的计算机的完全限定域名，< *端口* > 是计算机的 HTTPS 端口。如果要指定多个目标，请用逗号分隔计算机详细信息。

- 如果要只将节点上您选择的 SNMP 陷阱转发到远程计算机，请运行以下命令：

  ovconfchg -ns eaagt set SNMP_TRAP_FORWARD_FILTER "<*OID 过滤器*>"

  <*OID 过滤器*> 是使用通配符的 OID。事件拦截器过滤可用陷阱中与指定的 OID （使用通配符）匹配的陷阱，然后将它们转发到目标计算机。

配置 RTMA 组件

实时度量访问 (RTMA) 组件使您可以本地或远程实时访问系统性能度量。您安装 HP Operations Agent 之后，作为 RTMA 组件一部分的 perfd 进程开始以默认配置在节点上运行。您可以从 `perfd.ini` 文件修改 `perfd` 进程的配置设置，此文件可在节点上的以下目录中找到：

- 在 Windows 上：*%ovdatadir%*
- 在 UNIX （和 Linux）上：`/var/opt/perf`

参数	描述	默认值
interval	以秒为单位的数据收集频率。该值必须是 60 的倍数或因子。	10
port	perfd 使用的端口。	5227
depth	全局度量值保留在 perfd 缓存中的持续时间。此数据用于数据汇总。	30

参数	描述	默认值
maxrps	perfd 每秒接受的最多会话请求数。如果请求数超过这个限制， perfd 将暂停一秒，然后在日志文件中记录此事件的详细信息。日志文件 status-perfd.< 端口 > 位于节点上的以下目录中： • *在 Windows 上：%ovdatadir%* • *在 UNIX（和 Linux）上：* `/var/opt/perf`	20
maxtpc	对于每个客户端系统， perfd 接受的最多会话数。可用会话数达到此限制之后，如果有其他请求到达， perfd 会拒绝该请求。	30
maxcps	perfd 在给定时刻接受的最多同时会话请求数。如果请求数超过限制，服务器将暂停 3 秒，才能建立会话。	2

参数	描述	默认值
lightweight	如果设置为 true，perfd 将停止收集进程、应用程序、NFS 操作、逻辑系统和 ARM 的数据。此外，在 HP-UX 上也不会收集 HBA 和 LVM 数据。	false
localonly	如果设置为 true，perfd 将只能在本地计算机上配置。 如果设置为 true，除来自主机系统 (localhost) 并通过回环接口的连接请求， perfd 均将拒绝。拒绝的连接请求的详细信息将记在状态文件中。	false
ipv4	此选项只允许 perfd 接受 IPv4 连接。默认情况下，如果 perfd 无法创建 IPv6 套接字，它将自动切换到仅允许 IPv4 套接字。如果要显式禁用任何 IPv6 通信，请使用此选项。	false

要更改默认设置，请执行以下步骤：

1 在节点上，用文本编辑器打开 `perfd.ini` 文件。

2 修改设置。

3 保存文件。

4 重新启动 HP Operations Agent 使更改生效。

配置代理程序用户

操作监视组件可以非根或非特权用户身份运行，但性能收集组件必须始终以根用户或管理用户身份运行。如果将 HP Operations Agent 配置为以没有根特权或管理特权的用户身份运行，您将无法使用性能收集组件，最重要的是，您将无法收集任何系统性能数据。

安装 HP Operations Agent 之后，它在 Windows 节点上以本地系统帐户运行，在 UNIX/Linux 节点上以根帐户运行。但是，您可以将代理程序配置为以非默认帐户运行。

代理程序以当前正在运行它的用户帐户启动自动命令或操作员触发的命令。但是，您可将代理程序配置为以其他用户帐户启动命令。

在 Windows 上更改默认用户

要在 Windows 上使用的代理程序用户必须满足以下要求：

- 运行代理程序的用户权限：
  - 作为服务登录
  - 管理审核和安全日志
- 管理节点的用户权限：
  - 关闭系统

    这就允许代理程序关闭系统 （例如，用户在控制台中启动关机工具时）。
  - 调试程序

    这就允许代理程序收集有关进程的信息和终止进程 （例如，用户在控制台中启动列表进程或终止进程工具时）。

- 允许代理程序作为代理程序用户以外的用户启动命令和工具的用户权限：
  - 充当操作系统的一部分。
  - 调整进程的内存配额 （在 Windows 的某些版本中也叫 “增加配额”）
  - 更换进程级别的令牌。
- 注册表项的权限：
  - `HKEY_LOCAL_MACHINE/Software/Hewlett-Packard/OpenView`

    用户必须对此注册表项及其全部子对象具有完全控制权。
  - `HKEY_LOCAL_MACHINE/Software/Microsoft/WindowsNT/CurrentVersion/Perflib`

    用户必须有读取此注册表项的权限，代理程序才能访问性能数据。

您可能需要为需要执行的管理任务分配额外权限。例如：

- 如果需要能使用策略监视日志文件，代理程序用户必须有读取该日志文件的权限。
- 如果需要能使用自动命令、操作员触发的命令、工具或计划任务启动程序，代理程序用户必须有启动该程序的权限。

  此外，必须在 `eaagt` 命名空间中设置参数 `OPC_PROC_ALWAYS_INTERACTIVE=NEVER`。可以在代理程序安装默认设置中配置此参数，也可以在命令提示符处使用 `ovconfchg` 或 `ovconfpar` 来配置。您设置此参数后，代理程序启动的进程无权访问默认桌面。此设置应用于日志文件封装器预处理和监视代理程序调用的脚本。
- 代理程序以没有管理权限的用户帐户运行时，某些智能插件可能需要额外的配置或用户权限。有关更多详细信息，请参见各智能插件的文档。

要在 Windows 上更改默认的代理程序用户，请执行以下步骤：

1 *可选*。创建运行代理程序的新用户。

2 *可选*。创建新组，并将用户添加为该组的成员。

3 在节点上打开命令提示符，并输入以下命令：

cscript "%OvInstallDir%bin\ovswitchuser.vbs"-existinguser < *域* \ *用户* > **-existinggroup** < *组* > **-passwd** < *密码* >

在此实例中：

< *域* \ *用户* > 是域和用户名。

< *组* > 是用户所属的组的名称，例如 AgentGroup。

< *密码* > 是用户的密码。

该命令在组级别分配基本代理程序功能需要的用户权限，而不是分配到单个用户。因此，选择要使用的组时要小心。建议创建专用于代理程序用户的新组，并将代理程序用户添加为成员。

4 要重新启动代理程序，请运行以下命令：

a **ovc -kill**

b **ovc -start**

控制服务和代理程序进程现在以您指定的用户身份运行。

配置代理程序用户启动或停止服务和进程

在其操作过程中，代理程序可能需要启动或停止服务和进程（包括代理程序自己的进程）。您必须配置已配置为运行该代理程序的用户，使其获得启动或停止服务和进程的特权。

如果您希望代理程序用户在 Windows 节点上启动或停止服务，请执行以下步骤：

1 以具有管理特权的身份登录到节点。

2 转到目录 *%ovinstalldir%*lbin\xpl。

3 运行以下命令：

ovsetscmpermissions.vbs -user < *代理程序用户* > **-f**

4 运行以下命令，重新启动代理程序：

- **ovc -kill**
- **ovc -start**

在 UNIX/Linux 上更改默认用户

默认情况下，在安装 UNIX 或 Linux 操作系统的节点上，代理程序以根帐户运行。但是，您可将代理程序配置为以其他用户帐户运行。例如，您可能希望代理程序以权限少于根帐户的帐户运行。或者，您可能希望代理程序以有权限通过网络访问远程系统的帐户运行。

必须测试用户帐户是否有正确运行代理程序和管理节点的合适权限。您可能需要为需要执行的管理任务分配额外权限。例如：

- 如果需要能使用策略监视日志文件，代理程序用户必须有读取该日志文件的权限。
- 如果需要能使用自动命令、操作员触发的命令、工具或计划任务启动程序，代理程序用户必须有启动该程序的权限。
- 代理程序以其他用户身份运行时，某些智能插件可能需要额外的配置或用户权限。有关更多详细信息，请参见各智能插件的文档。

要在 UNIX/Linux 上更改默认用户，请执行以下步骤：

1 *可选。*创建运行代理程序的新用户。

2 *可选。*创建新组，然后将用户添加为该组的成员。

3 以根身份登录节点，然后打开 shell 提示符。

4 转到以下目录：

在 HP-UX、Linux 或 Solaris 上

`/opt/OV/bin`

在 AIX 上

/usr/lpp/OV/bin

5 运行以下命令，停止代理程序：

ovc -kill

6 运行以下命令，更改代理程序用户：

ovswitchuser.sh -existinguser < *用户* > **-existinggroup** < *组* >

在此实例中：

< *用户* > 是运行代理程序的用户的名称。

< *组* > 是用户所属的组的名称，例如 AgentGroup。该命令赋予该组对代理程序数据目录中的所有文件以及所有已安装包的完全控制权。如果您之前启动过该命令并指定了不同的组，则该命令将取消对以前组的文件的控制。

在代理程序的数据目录中设置组 ID 标记。此标记意味着您指定的组也将拥有代理程序基本目录中的任何新文件和子目录。

该命令在组级别分配基本代理程序功能需要的用户权限，而不是分配到单个用户。因此，选择要使用的组时要小心。建议创建专用于代理程序用户的新组，并将代理程序用户添加为成员。

7 默认情况下，HP Operations Agent 包括在端口 383 上侦听来自管理服务器的入站连接的通信中介器。但是，在 UNIX 和 Linux 节点上，非根用户不能打开 0 到 1023 范围内的端口。因此，必须在节点上配置通信中介器才能在不同端口（高于 1023）上侦听。还必须配置连接到节点的管理服务器，使其出站连接发往正确端口。

必须通过在 bbc.cb.ports 命名空间中设置 PORTS 参数来配置通信中介器端口。可使用以下方式配置此参数：

- 在代理程序安装默认设置中配置这些值。如果需要配置通信中介器端口的节点很多，建议使用此方式。在创建或迁移节点之前，必须计划并配置安装默认设置。

- 在命令提示符处使用 ovconfchg 或 ovconfpar。

  该值必须包含一个或多个主机名或 IP 地址，并采用以下格式：

  < *主机* **>:<** *端口* **>[,<** *主机* **>:<** *端口* **>] ...**

  例如，要在主机名为 node1.emea.example.com 的节点上将通信中介器端口配置为 5000，则在该节点上以及其他任何打开到该节点的连接的管理服务器上使用以下命令：

  ```
  ovconfchg -ns bbc.cb.ports -set PORTS
  node1.domain.example.com:5000
  ```

8 要启动代理程序，请运行以下命令：

  a **su <** *用户* **>**

  b **ovc -start**

  控制服务和代理程序进程现在以您指定的用户身份运行。

更改命令的默认用户

默认情况下，代理程序以当前正在运行它的用户帐户启动自动命令或操作员触发的命令。但是，您可将 HP Operations Agent 配置为以其他用户帐户启动命令。您可通过设置节点上 eaagt 名称空间中的 OVO_STD_USER 参数执行此操作。可使用以下方式配置此参数：

- 在 HP Operations Agent 安装默认设置中配置这些值。如果需要配置用户的节点很多，建议使用此方式。在创建或迁移节点之前，必须计划并配置安装默认设置。
- 在命令提示符处使用 ovconfchg 或 ovconfpar。

以 <*用户*>/|<*加密的密码*> 格式指定 OVO_STD_USER 的值

- 用用户名称替换 <*用户*>。对域用户，指定域和用户名，如 EXAMPLE\AgentUser。对本地用户，只指定名称，如 AgentUser。

- 将 < *加密的密码* > 替换成命令 **opcpwcrpt** < *密码* > 的输出。可在管理服务器上在命令提示符处启动该命令。

配置或启动工具时也可使用 OVO_STD_USER。指定用户名 $OVO_STD_USER，保留密码为空。

必须测试用户帐户是否有正确运行命令和工具的合适权限。

如果代理程序未能以 OVO_STD_USER 启动命令或工具，代理程序可以当前运行该代理程序的同一用户帐户启动该命令或工具。例如，如果您指定了错误的用户或密码，可能发生这种情况。

4 使用 utility 程序

utility 程序是一款用于管理和报告日志文件、收集参数 (parm) 文件和警报定义 (alarmdef) 文件相关信息的工具。可以交互或批处理模式使用 utility 程序来执行以下任务。

- 扫描原始或提取的日志文件，并生成显示以下信息的报告：
  - 覆盖的日期和时间
  - scope 收集器未运行的时间
  - scope 参数设置的更改
  - 系统配置的更改
  - 日志文件磁盘空间
  - 收集参数 (parm) 文件中应用程序和进程设置的影响
- 调整原始日志文件大小
- 检查 parm 文件中是否有语法警告或错误
- 检查 alarmdef 文件中是否有语法警告或错误
- 对照警报定义处理日志文件数据以检测历史数据中的警报条件

本章包含以下主题：

有关 utility 程序命令的详细描述，请参见第 5 章 utility 命令。

运行 utility 程序

可采用三种方式运行 utility 程序：

- 命令行模式 - 在命令行中使用命令选项和参数控制 utility 程序。
- 交互模式 - 执行程序时提供交互命令和参数，将 stdin 设置为交互终端或工作站。
  如果您是有经验的用户，可以只快速指定给定任务所需的那些命令。如果您是新用户，可能希望使用 utility 程序的 guide 命令获取一些使用命令方面的帮助。在引导模式下，系统会要

求您从选项列表选择以执行任务。在引导模式下，每个完成任务的交互命令执行后都会被列出，因此您可以看到是如何使用它们的。您可以随时退出和重新进入引导模式。

- 批处理模式 - 可以运行程序并将 stdin 重定向到包含交互命令和参数的文件。

命令行界面的语法与其他程序上的典型 UNIX 命令行界面类似，本章将进行详细描述。

交互模式和批处理模式的命令语法相同。命令可以按任意顺序输入；如果命令有关联的参数，则必须在输入对应命令之后立即输入该参数。

有两种类型的参数：必需（无默认值）和可选（提供默认值）。utility 处理这些参数的方式取决于它的运行模式。

- 在交互模式下，如果缺少可选参数，程序将显示默认参数，您可以确认或覆盖该参数。如果缺少必需参数，程序将提示您输入参数。
- 在批处理模式下，如果缺少可选参数，程序将使用默认值。如果缺少必需参数，程序将终止。

交互模式下处理错误和缺少数据的方式与命令行和批处理模式下的处理方式不同。在交互模式下您可以提供其他数据或更正错误，在命令行和批处理模式下则不行。

使用交互模式

使用 utility 程序的交互模式需要您发出一系列命令以执行特定任务。

例如，如果要检查日志文件以确认当天记录的数据中是否存在警报条件，则可以在调用 utility 程序之后发出以下命令：

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

checkdef 命令检查 alarmdef 文件中的警报定义语法，然后设置并保存文件名以供 analyze 命令使用。 detail off 命令导致 analyze 命令仅显示警报摘要。 start today-1 命令指定仅分析昨天记录的数据。 analyze 命令对照 alarmdef 文件分析默认 SCOPE 数据源中的原始日志文件。

使用交互模式和批处理模式的示例

以下示例显示了 utility 程序的 resize 命令在批处理模式下和交互模式下的不同工作方式。

resize 命令用于设置以下函数的参数：

- 要调整大小的日志文件类型。

- 新文件的大小。
- 文件中要留下的空余空间量。
- 指定是否执行大小调整的操作。

此 resize 命令示例将全局日志文件大小调整到最多可以包含 120 天的数据，空余空间等于 45 天。命令及其参数为：

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

不管是以交互模式还是从批处理作业输入此命令，结果都相同。

第一个参数 global 指示要调整大小的日志文件。如果不提供此参数，对交互用户和批处理用户而言随之发生的操作如下：

- 批处理用户 - 因为 logfile 参数无默认值，批处理作业会终止。
- 交互用户 - 将提示您选择要调整大小的日志文件类型以完成命令。

最后一个参数 yes 指示无条件地执行调整大小操作。

如果不提供 yes 参数，对交互用户和批处理用户而言随之发生的操作如下：

- 批处理用户 - 因为 yes 是默认操作，将继续调整大小。
- 交互用户 - 调整大小前提示您提供此操作。

在批处理模式或交互模式下使用 resize 命令之前，必须首先停止数据收集。有关详细信息，请参见第 2 章第 37 页的停止和重新启动数据收集。

utility 命令行界面

除了交互模式和批处理模式命令语法以外，还可将命令选项及其关联的参数通过命令行界面传递到 utility 程序。命令行界面允许 shell 脚本轻松调用 utility 程序，并允许输入和输出重定向到 UNIX 管道，因而完全适用于典型的 UNIX 环境。

例如，要使用与上一部分 “使用交互模式” 中所示的示例等效的命令行，请输入：

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

下表中列出了命令行选项和参数。所引用命令的描述可在第 5 章 utility 命令中找到。

表 3　　命令行参数

命令选项	参数		描述
-b	date	time	指定 analyze 或 scan 函数的开始日期和时间。（请参见第 4 章的 start 命令。）
-e	date	time	指定 analyze 或 scan 函数的结束日期和时间。（请参见第 4 章的 stop 命令。）
-l	logfile		指定要打开的日志文件。请参见第 4 章的 logfile 命令。）
-f	listfile		指定输出列表文件。（请参见第 4 章的 list 命令。）
-D			启用 analyze、scan 和 parm 文件检查的详细信息。请参见第 4 章的 detail 命令。）
-d			禁用 analyze 和 parm 文件检查的详细信息。（请参见第 4 章的 detail 命令。）
-v			执行命令行命令后回显它们。
-xp	parmfile		检查 parm 文件的语法。（请参见第 4 章的 parmfile 命令。）
-xc	alarmdef		检查 alarmdef 文件的语法并设置其文件名以供 -xa （或 analyze 命令）使用。（请参见第 4 章的 checkdef 命令。）
-xa			对照 alarmdef 文件分析日志文件。（请参见第 4 章的 analyze 命令。）
-xs	logfile		扫描日志文件并生成报告。（请参见第 4 章的 scan 命令。）
-xr	global application process device transaction ls EMPTY=nnn SPACE=nnn	SIZE=nnn DAYS=nnn YES NO MAYBE	调整日志文件大小。（请参见第 4 章的 resize 命令。）
-? 或 ?			显示命令行语法。

使用命令行界面的示例

在命令行输入命令选项和参数时，以下情况适用：

错误和缺少数据的处理方式与对应批处理模式命令中的处理方式完全相同。即，如果可能，将对缺少的数据使用默认值，而所有错误都将导致立即终止程序。

禁用回显命令和命令结果。 utility 不从其 stdin 文件读取信息。它在执行命令行中的操作后终止。

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

解释为：

-xp	检查默认 parm 文件的语法。
-d	在 scan 报告中禁用详细信息。
-xs	执行 scan 操作。未指定日志文件，因此扫描默认日志文件。

utility scan 报告详细信息

utility 程序的 scan 命令读取日志文件并编写有关其内容的报告。报告的内容取决于在发出 scan 命令之前发出的命令。（有关详细信息，请参见第 5 章 utility 命令中的 scan 命令描述。）

下表汇总了所有 scan 报告中包含的信息以及仅当 scan 命令与 detail on 命令一起使用（默认）时才生成的报告中所包含的信息。

表 4　　scan 报告中包含的信息

初始值	
初始 parm 文件全局信息和系统配置信息	仅在指定 detail on 时打印。
初始 parm 文件应用程序定义	仅在指定 detail on 时打印。
按时间顺序排列的详细信息	
parm 文件全局更改	仅在指定 detail on 时打印。
parm 文件应用程序更改	仅在指定 detail on 时打印。
收集器关闭时间通知	仅在指定 detail on 时打印。
特定于应用程序的摘要报告	仅在指定 detail on 时打印。

表 4　　scan 报告中包含的信息

摘要	
进程摘要报告	只要扫描进程数据就打印。
收集器覆盖时间摘要	总是打印。
日志文件内容摘要	总是打印。包括覆盖的空间和日期。
日志文件空余空间摘要	总是打印。

Scan 报告信息

utility scan 报告中的信息划分为三个类型：

- 初始值
- 按时间顺序排列的详细信息
- 摘要

初始值

此部分介绍以下初始值：

- 初始 parm 文件全局信息
- 初始 parm 文件应用程序定义

初始 parm 文件全局信息

要获取此报告，请使用带默认 detail on 的 scan 命令。

此报告列出在日志文件中最早记录全局信息时的 parm 文件配置设置。随后的全局信息更改通知都基于此报告中的值。如果特定参数没有更改通知，表明该参数在扫描期间保持其原始设置。

以下示例显示了列出 parm 文件内容的报告部分。

```
06/03/99 15:28 System ID="Homer"
scopeux/UX A.10.00 SAMPLE INTERVAL = 300,300,60 Seconds, Log version=D
Configuration: 9000/855, O/S A.10.00 CPUs=1
Logging Global Process records
        Device= Disk FileSys 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)
HP-UX Parms: Buffer Cache Size = 16384KB, NPROC = 532
Wait Thresholds: CPU=100.00%, Memory=100.00%
Impede=100.00%
Memory: Physical = 84.0 MB, Swap = 124304.0 MB, Available to users = 66.5
MB. There are 2 LAN interfaces: 0, 1.
```

```
06/03/99 15:28 There are 2 disk devices:
   Disk #1976            = "/dev/hdisk0"
   Disk #1987            = "/dev/hdisk1"
```

第一行所列的日期和时间对应于全局日志文件中的*最早日期和时间*，它表示启动 scope 的日期和时间。数据记录可能已滚出全局日志文件，因此此报告中的日期和时间不一定表示日志文件中的*第一条全局记录*。

初始 parm 文件应用程序定义

要获取此报告， 请使用带默认 `detail on` 的 `scan` 命令， 并且日志文件中要有应用程序数据。

此报告列出在日志文件中列出第一条应用程序记录时的每个应用程序的名称和定义。 您收到的任何应用程序添加或删除通知都基于此初始应用程序列表。 例如：

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

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

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

在扫描期间，系统会通知您添加或删除的应用程序。添加和删除通过比较旧应用程序名称与新记录的应用程序名称集的拼写和大小写来确定。不尝试检测应用程序定义的更改。如果检测到新名称的应用程序，则列出新名称及其新定义。

此记录中的日期和时间是当前在日志文件中记录第一条应用程序记录之前，最后一次启动 scope 的日期和时间。

按时间顺序排列的详细信息

此部分介绍以下按时间顺序排列的详细信息：

- parm 文件全局更改通知
- parm 文件应用程序添加和删除通知
- scope 关闭时间通知
- 特定于应用程序的摘要报告

parm 文件全局更改通知

要获取此报告，请使用带默认 `detail on` 的 `scan` 命令。

只要发现有关 scope 启动的记录，就将生成此报告。

以下示例显示了系统中添加了两个新的磁盘驱动器时生成的更改通知。

```
03/13/99 17:30 The number of disk drives changed from 9 to 11
03/13/99 17:30  New disk device scsi-4 = "c4d0s*"
03/13/99 17:30  New disk device scsi-3 = "c3d0s*"
```

parm 文件应用程序添加 / 删除通知

要获取此报告，请使用带默认 `detail on` 的 `scan` 命令，并且日志文件中要有应用程序数据。

每次启动 scope 时，都可以添加或删除用户定义的应用程序。如果发现应用程序名称与上一个应用程序名称集不匹配，则打印应用程序添加、删除或更改通知。如果应用程序的名称未更改，则不打印。

以下示例显示启动了新应用程序。

```
03/13/99 17:30 Application 4 "Accounting_Users_1" was added
User=ted,rebecca,test*,mark,gene
```

不检查应用程序定义是否有更改。当应用程序名称有更改时列出应用程序定义，但是，不会检测现有应用程序定义更改而名称未更改的情况。

scope 关闭时间通知

要获取此报告，请使用带默认 `detail on` 的 `scan` 命令。

如果提取的文件仅包含摘要信息，时间四舍五入到最近的小时。例如：

```
06/03/99 11:00 - 06/03/99 12:34 collector off (01:34:04)
```

第一个日期和时间 (`06/03/99 11:00`) 表示重新启动 scope 前日志文件中最后一条有效的数据记录的日期和时间。第二个日期和时间 (`06/03/99 12:34`) 表示重新启动 scope 的日期和时间。

最后一个字段（圆括号中）显示*未*运行 scope 的时间长度。格式是 *ddd*/*hh:mm:ss*，其中 *ddd* 是天数， *hh:mm:ss* 是小时、分钟和秒。删除了左侧的零。

在此示例中，scope 关闭时间为 1999 年 6 月 3 日 11:00 am 到 12:34 pm。摘要信息显示 1 小时 34 分钟 4 秒未收集数据。

特定于应用程序的摘要报告

要获取此报告，请使用带默认 `detail on` 的 `scan` 命令，并且日志文件中要有应用程序数据。

此报告可以帮助您定义应用程序。使用此报告可以识别系统资源积累过多或过少的应用程序，以及可以与其他应用程序合并的应用程序。如果应用程序积累过多的系统资源，将其拆分成多个应用程序可能更好。

应以有利于制定系统性能调整决策的方式定义应用程序。系统资源不可能在各个应用程序中均匀积累。

一旦应用程序定义更改，即生成特定于应用程序的摘要报告，以便您访问更改前后的应用程序定义数据。

最终报告是针对所有应用程序生成的。此报告只覆盖自上一个报告以来的时间，而不是日志文件所覆盖的整个时间。例如：

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

摘要

此部分介绍以下摘要：

- 进程日志原因摘要
- 扫描开始和停止的实际日期和时间
- 应用程序总体摘要
- scope 覆盖时间摘要
- 日志文件内容摘要
- 日志文件空余空间摘要

进程日志原因摘要

要获取此报告，日志文件中必须有进程数据。

此报告帮助您为 scope 设置感兴趣的进程阈值。报告中列出进程被视为感兴趣进程从而记录在日志中的每一个原因，以及记录的满足每个条件的进程总数。

以下示例显示了进程日志原因摘要报告：

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

Log Reason       Records   Percent   Recs/hr
---------------  -------   -------   -------
New Processes      17619     53.9%      44.7
```

```
Killed Processes    16047     49.1%      40.7
CPU Threshold        3169      9.7%       8.0
Disk Threshold       1093      3.3%       2.8

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

如果发出 detail on 命令，每次阈值更改时都将生成此报告，以便您可以评估该更改的影响。每个报告都覆盖自上一个报告以来的时间。扫描完成时生成最终报告，它覆盖自上一个报告以来的时间。

如果发出 detail off 命令，则只生成一个覆盖整个扫描期间的报告。

可以通过修改 parm 文件的 threshold 参数，并提高生成最多进程日志记录的感兴趣原因的阈值来减少 scope 记录的进程数据量。要增加记录的数据量，则降低感兴趣区域的阈值。

在上面的示例中，提高 CPU 阈值或设置 nonew 阈值都可以减少进程数据所用的磁盘空间量 （相对应的是记录的信息较少）。

扫描开始和停止

只要扫描到有效数据，就打印此摘要报告。它提供扫描开始和停止的实际日期和时间。例如：

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

应用程序总体摘要

要获取此报告，日志文件中必须有应用程序数据。

此报告从总体上显示用户定义的应用程序中而不是其他应用程序中积累的系统活动数。如果用户应用程序未捕获大量的关键资源，可以考虑为用户应用程序中可包括的进程扫描进程数据。

例如：

```
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%)
```

收集器覆盖时间摘要

只要扫描到有效的全局数据或应用程序数据，就打印此报告。它显示使用 scope 收集系统活动的情况。如果 scope 关闭的时间百分比很高 （如下面的示例所示），应检查启动和停止 scope 的操作过程。

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

如果扫描中包括全局详细信息数据，此报告将更完整。如果只提供摘要数据，可以仅将 scope 的停止和开始时间四舍五入到最近的小时。（如果是这样，报告中将打印相应的警告消息。）

“The total time covered”是数据记录的所有起止时间的总和。“out of”时间值是将结束日期和时间减去开始日期和时间计算出来的。它应表示已记录的总时间。“Time lost when collector was off”值等于总时间减去覆盖的时间。

上述三个时间的格式是：

ddd/hh:mm:ss

其中 *ddd* 是天数， *hh:mm:ss* 是小时、分钟和秒。

在上面的示例中，收集的总时间是 108 天 16 小时 14 分 51 秒。

日志文件内容摘要

*只要*扫描到有效数据，就打印日志文件内容摘要。它包括覆盖的日志文件空间和日期。当用 resize 命令调整日志文件大小时，此摘要报告很有用。

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

这些列描述如下：

列	说明
Type	记录的数据的常规类型。有一个特殊类型 Overhead： Overhead 是日志文件占用 （或保留）的磁盘空间量*与*扫描的数据记录实际使用的磁盘空间量的比值。 如果未扫描整个日志文件， Overhead 包括未扫描的数据记录。如果扫描了整个文件，Overhead 会说明任何使数据无法记入文件的磁盘空间量不足的情况*以及*任何文件访问支持结构。提取的日志文件的开销高于原始日志文件的开销是很正常的，因为后者还具有其他支持更快速定位的结构。
Total	针对每种数据类型，扫描的总记录数和磁盘空间。
Each Full Day	scope 每运行 24 小时扫描的记录数和所用的磁盘空间量。
Dates	扫描每种数据类型的数据记录的第一个和最后一个有效日期。
Full Day	扫描此数据类型数据的整天 （24 小时）数。 如果 scope 覆盖时间不等于 100% 的扫描时间， Full Days 可能不等于开始日期与停止日期之差。

TOTAL 行 （在所列数据底部）供您了解每天要使用的磁盘空间量以及预计可以累积的数据量。

日志文件空余空间摘要

为每个扫描的日志文件打印此摘要。例如：

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

可用于存储更多数据的空间量是基于以下因素算出的：

- 文件中未使用的空间量
- 每个 24 小时都要记录的数据的兆字节数的扫描值

如果每天扫描的兆字节值低得不切实际，则此计算后的值将替换为默认值。

如果扫描提取的文件，由于所有数据类型共享同一个提取的文件，您收到的将是单行报告。

5 utility 命令

本章介绍 utility 程序的命令。它包括按字母顺序列出命令的语法摘要和命令参考部分。

可以按大小写字母的任意组合输入 utility 命令和参数。仅命令名称的前三个字母是必需的。例如， logfile 命令可以输入为 **logfile**，也可以缩写为 **log** 或 **LOG**。

有关如何使用这些命令的示例，请参见 utility 程序的联机帮助。

下面几页的表包含 utility 命令语法和参数的摘要。

表 5　　utility 命令：语法和参数

命令	参数
analyze	
checkdef	alarmdef 文件
detail	on off
exit e	
guide	
list	*文件名*或 *
logfile	logfile
menu ?	
parmfile	parmfile
quit q	

表 5　　utility 命令：语法和参数 （续）

命令	参数
resize	global application process device transaction days=maxdays size=max MB empty=days space=MB yes no maybe
scan	*日志文件* （操作还受 list、 start、 stop 和 detail 命令影响。）
show	all
sh !	系统命令
start	*日期* [*时间*] today [- *天数*] [*时间*] last [- *天数*] [*时间*] first [+ *天数*] [*时间*]
stop	*日期* [*时间*] today [- *天数*] [*时间*] last [- *天数*] [*时间*] first [+ *天数*] [*时间*]

analyze

使用 analyze 命令对照警报定义 (alarmdef) 文件中的警报定义分析日志文件数据，并报告生成的警报状态和活动。在发出 analyze 命令之前，应运行 checkdef 命令检查警报定义语法。checkdef 还设置并保存警报定义文件名以供 analyze 使用。如果在 analyze 之前未运行 checkdef，系统会提示您输入警报定义文件名。

如果使用的是命令行模式，则使用默认警报定义文件 /var/opt/perf/alarmdef。

有关警报定义的详细信息，请参见第 151 页的性能警报。

语法

analyze

如何使用

发出 analyze 命令时，它对照 alarmdef 文件中的警报定义分析数据源配置文件 datasources 中指定的日志文件。

可使用 analyze 命令评估警报定义是否与系统上收集的历史数据很好地匹配。它还可用于确定警报定义是否将在分析工作站上生成过多或过少的警报。

此外，您还可以用警报定义文件中设置的定义（IF 语句）执行数据分析，因为满足条件时您会收到 PRINT 语句的信息输出。有关如何使用警报定义中的 IF 和 PRINT 语句的说明，请参见第 9 章性能警报。

可以选择与 analyze 一起运行 start、 stop 和 detail 命令以自定义分析进程。这些命令按以下顺序指定：

```
checkdef
start
stop
detail
analyze
```

如果要分析特定时间段收集的日志文件数据，请使用 start 和 stop 命令。（本章稍后将会介绍 start 和 stop 命令。）

在执行 analyze 命令时，它会列出 alarm start、end 和 repeat status 等警报事件，以及关联的 PRINT 语句中的任何文本。当 IF 语句中的条件为 true 时，也会列出 PRINT 语句中的任何文本。不执行 EXEC 语句，但列出 EXEC 语句，以便您可以看到已执行的操作。警报摘要报告显示警报数计数和每个警报处于活动状态 (on) 的时间。计数包括 alarm starts 和 repeats，但不包括 alarm ends。

如果只要查看警报摘要报告，请发出 detail off 命令。但是，如果使用的是命令行模式，detail off 为默认值，因此需要指定 -D 才能查看警报事件和警报摘要。

示例

checkdef 命令检查 alarmdef 文件中的警报定义语法，并保存 alarmdef 文件名，稍后供 analyze 命令使用。 start today 命令指定仅分析今天记录的数据。最后， analyze 命令对照 alarmdef 文件中的警报定义分析 datasources 文件中指定的默认 SCOPE 数据源的日志文件。

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

要使用命令行参数执行上面的任务，请输入：

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

checkdef

使用 checkdef 命令检查警报定义文件中的警报定义语法，并报告发现的任何警告或错误。此命令还设置并保存警报定义文件名以供 analyze 命令使用。

有关警报定义语法及如何指定警报定义的描述，请参见第 9 章性能警报。

语法

checkdef [/ 目录路径 /alarmdef]

参数

alarmdef　任何警报定义文件的名称。它可以是用户指定的文件或默认 alarmdef 文件。如果不指定目录路径，将搜索当前目录。

如何使用

确定警报定义正确后，可以使用 analyze 命令对照日志文件中的数据处理警报定义。

在批处理模式下，如果不指定警报定义文件，将使用默认 alarmdef 文件。

在交互模式下，如果不指定警报定义文件，系统会提示您指定一个。

示例

checkdef 命令检查 alarmdef 文件中的警报定义语法，然后保存 alarmdef 文件名，稍后供 analyze 命令使用。

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

要使用命令行参数执行上面的任务，请输入：

utility -xc

detail

使用 detail 命令控制 analyze、 parmfile 和 scan 报告中打印的详细信息级别。

在交互模式和批处理模式下默认值是 detail on，在命令行模式下默认值是 detail off。

语法

detail　[on]

[off]

参数

on　　打印 parm 文件的有效内容及 parm 文件错误。打印完整的 analyze 和 scan 报告。

off　　在 parm 文件报告中，*不*打印应用程序定义。在 scan 报告中，*不*打印 scope 收集时间、初始 parm 文件全局信息和应用程序定义。在 analyze 报告中，*不*打印警报事件和警报操作。

如何使用

有关如何与 analyze、 scan 和 parmfile 命令一起使用 detail 命令的说明，请参见本章的 analyze、 parmfile 和 scan 命令描述。

示例

有关使用 detail 命令的示例，请参见本章的 analyze、 parmfile 和 scan 命令描述。

exit

使用 exit 命令终止 utility 程序。 exit 命令与 utility 程序的 quit 命令等效。

语法

```
exit
e
```

guide

使用 guide 命令进入引导命令模式。引导命令界面指引您完成各种 utility 命令，并提示您执行可用的最常见任务。

语法

```
guide
```

如何使用

- 要从 utility 的交互模式进入引导命令模式，请输入 **guide** 并按 **Return**。
- 要接受参数默认值，请按 **Return**。
- 要终止引导命令模式并返回交互模式，请在 guide> 提示符处输入 **q**。

此命令不提供所有可能的参数设置组合。它选择应为大多数用户生成有用结果的设置。

help

使用 help 命令访问 utility 程序的联机帮助。

语法

help [*关键字*]

如何使用

可以输入参数获取有关 utility 命令和任务或 help 本身的信息。通过输入关键字导航到不同主题。如果有多页信息，显示会暂停，等待您按 **Return** 再继续。输入 **q** 或 **quit** 可以退出帮助系统并返回到 utility 程序。

还可以请求特定主题的帮助信息。例如，

help tasks

或

help resize parmfile

使用这种格式的 help 命令时，您会收到指定主题的帮助文本，并且依然留在 utility 命令输入上下文中。因为不是以交互模式进入帮助子系统，输入下一个 utility 命令之前不必输入 **quit**。

list

使用 list 命令指定所有 utility 报告的输出文件。报告的内容取决于在 list 命令之后发出的其他命令。例如，在 logfile、detail on 和 scan 命令之前使用 list 命令会生成日志文件详细摘要报告的列表文件。

语法

list [*文件名* **]|***

其中 * 将输出设置回 stdout。

如何使用

可采用两种方式指定报告的列表文件：

- 调用 utility 程序时重定向 stdout，输入：

  utility > utilrept

- 或者，在 utility 运行时使用 list 命令，输入：

  list utilrept

这两种情况下，用户交互和错误都打印到 stderr，报告转到指定的文件。

list 命令中的*文件名*参数必须是您具有写访问权限的有效文件名。现有文件的新输出追加到现有内容的末尾。如果文件不存在，将创建一个文件。

要确定当前输出文件，请发出不带参数的 list 命令。

如果输出文件不是 stdout，大多数命令将回显到输入它们时的输出文件。

示例

list 命令生成提取的日志文件 rxlog 的摘要报告。list utilrept 命令将 scan 报告列表定向到磁盘文件。detail off 指定报告不包含完整详细信息。scan 命令读取 rxlog 并生成报告。

list * 命令将列表设备设置回默认 stdout。 !lp utilrept 将磁盘文件发送到系统打印机。

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

要使用命令行参数执行上面的任务，请输入：

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

logfile

使用 logfile 命令打开日志文件。许多 utility 程序函数都必须打开日志文件。可以通过发出 logfile 命令显式打开日志文件，也可以通过发出某个其他命令隐式打开日志文件。如果在批处理模式或命令行模式下，且未指定日志文件名称，将使用默认 /var/opt/perf/datafiles/logglob 文件。如果在交互模式下，且未指定日志文件名称，系统会提示您提供一个文件或接受默认 /var/opt/perf/datafiles/logglob 文件。

语法

logfile [*日志文件* **]**

如何使用

可以指定原始或提取的日志文件的名称。如果指定提取的日志文件名称，将从该单个文件获取所有信息。除全局日志文件 logglob 外，不需要指定任何原始日志文件。打开 logglob 后您可以访问其他日志文件中的所有数据。

原始日志文件有以下名称：

logglob	全局日志文件
logappl	应用程序日志文件
logproc	进程日志文件
logdev	设备日志文件
logtran	事务日志文件
logls	逻辑系统日志文件
logindx	索引日志文件

成功打开日志文件后，即打印或显示包含一个或多个日志文件常规内容的报告，如下一页中的示例所示。

```
Global      file: /var/opt/perf/datafiles/logglob version D
Application file: /var/opt/perf/datafiles/logappl
Process     file: /var/opt/perf/datafiles/logproc
Device      file: /var/opt/perf/datafiles/logdev
Transaction file: /var/opt/perf/datafiles/logtran
Index       file: /var/opt/perf/datafiles/logindx
System ID:  homer
System Type 9000/715 S/N 6667778899 O/S HP-UX B.10.20. A
Data Collector: SCOPE/UX C.02.30
File Created:   06/14/99
Data Covers:    27 days to 7/10/99
Shift is:       All Day

Data records available are:
  Global Application Process Disk Volume Transaction

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

The first GLOBAL      record is on 06/14/99 at 12:00 AM
The first APPLICATION record is on 06/25/99 at 12:00 AM
The first PROCESS     record is on 07/06/99 at 12:01 AM
The first DEVICE      record is on 05/01/99 at 11:50 AM
The first TRANSACTION record is on 05/01/99 at 11:55 AM
The default starting date & time = 05/01/99 11:50 AM   (FIRST +   0)
The default stopping date & time = 07/10/99 11:59 PM   (LAST -    0)
```

可以用 show 命令验证打开的文件，该命令稍后再作介绍。

您随时可以再输入一个 logfile 命令打开另一个日志文件。在打开新日志文件之前，关闭当前打开的任何日志文件。

resize 和 scan 命令需要打开日志文件。如果当前未打开日志文件，将执行隐式 logfile 命令。

不要重命名原始日志文件。访问这些文件则假定标准日志文件名称有效。

如果同一系统上必须有多个原始日志文件集，请为每个文件集创建一个单独目录。虽然不能更改日志文件名称，但可以使用不同目录。如果无论如何都要调整日志文件大小，必须具有所有日志文件的读 / 写访问权限。

menu

使用 menu 命令打印可用 utility 命令的列表。

语法

menu

示例

```
utility> menu
Command  Parameters       Function
```

```
HELP     topic]           Get information on commands and options
GUIDE                     Enter guided commands mode for novice users
LOGFILE  [logname]        Specify a log file to be processed
LIST     [filename|*]     Specify the listing file
START    [startdate time] Set starting date & time for SCAN or ANALYZE
STOP     [stopdate  time] Set ending   date & time for SCAN or ANALYZE
DETAIL   [ON|OFF]         Set report detail for SCAN, PARMFILE, or ANALYZE
SHOW     [ALL]            Show the current program settings
PARMFILE [parmfile]       Check parsing of a parameter file
SCAN     [logname]        Read the log file and produce a summary report
RESIZE   [GLOB|APPL|PROC|DEV|TRAN]
         [DAYS=][EMPTY=]  Resize raw log files
CHECKDEF [alarmdef]       Check parsing and set the alarmdef file
ANALYZE                   Analyze the log file using the alarmdef file
! or Sh  [command]        Execute a system command
MENU or ?                 List the commands menu (This listing)
EXIT or Q                 Terminate the program
utility>
```

parmfile

使用 parmfile 命令查看用于数据收集的性能收集组件 parm 文件的设置并检查 **parm** 文件的语法。

语法

parmfile [/ *目录路径/parmfile*]

如何使用

可以使用 parmfile 命令执行以下任何操作：

- 检查 parm 文件中是否有语法警告并查看生成的设置。检查所有参数的语法是否正确，并报告错误。完成语法检查之后，仅报告适用的设置。
- 查明为定义应用程序预留的空间。
- 如果指定 detail on，则打印 parm 文件的有效内容、任何未覆盖的默认设置及应用程序定义。

在批处理模式下，如果不指定 parm 文件名，将使用默认 parm 文件。

在交互模式下，如果不提供 parm 文件名，系统会提示您提供一个。

示例

parmfile 命令检查当前 parm 文件的语法，并报告任何警告或错误。 Detail on 列出记录参数设置。

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

要使用命令行参数执行上面的任务，请输入：

utility -xp -D

quit

使用 quit 命令终止 utility 程序。 quit 命令与 utility 程序的 exit 命令等效。

语法

```
quit
q
```

resize

使用 resize 命令管理原始日志文件集的空间。它是用于调整原始日志文件大小以保持文件与其内部控制结构之间协调的*唯*一程序。如果使用其他工具，可能会删除这些控制结构或破坏其有效性。

utility 程序*不能*用于调整提取文件的大小。如果要调整提取文件的大小，请使用 extract 程序创建新的提取日志文件。

语法

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

参数

log file type	指定要调整其大小的原始数据类型：global、application、process、device 或 transaction，它们分别对应原始日志文件 logglob、logappl、logproc、logdev 和 logtran。如果不指定数据类型，并且以批处理模式运行 utility，批处理作业将终止。如果以交互模式运行 utility，系统会提示您根据现有的日志文件提供数据类型。
days 和 size	指定日志文件的最大大小。实际大小取决于文件中的数据量。
empty 和 space	指定完成调整大小操作后文件中所需的最小空间量。此值用于确定在调整大小进程中是否必须删除当前日志文件的一些数据。

您可能希望日志文件在调整大小操作后的指定天数才填满。可以使用 resize 命令的这一功能最小化 scope 收集器必须调整日志文件大小的次数，因为只要文件已填满即可调整大小。使用 resize 强制日志文件中有一定的空余空间可在您希望其执行调整大小操作时调整日志文件大小。

days 和 empty 值以天为单位输入， size 和 space 值以 MB 为单位输入。通过对日志文件使用平均每天兆字节值将天数换算为 MB。此换算系数根据记录的数据类型和系统的特定特征而变化。

如果在发出 resize 命令之前对现有日志文件发出 scan 命令，可以获得更准确的平均每天兆字节换算系数。 scan 测量系统的累积速率。如果不执行扫描，或测量的换算系数似乎不合理， resize 命令将对每个数据类型使用默认换算系数。

yes　　指定应无条件地执行大小调整。如果不以交互模式运行 utility，这是默认操作。如果以交互模式运行 utility 时未指定操作，系统会提示您提供操作。

no　　指定不应执行大小调整。如果要查看大小调整报告，但不希望当时就执行大小调整，则可以将此参数指定为一项操作。

maybe　　指定应由 utility 决定是否调整文件大小。此参数强制 utility 根据日志文件中当前的空余空间量 （在调整大小之前）和 resize 命令中指定的空间量作出决策。如果当前日志文件的空间量超过指定的空余空间量，将*不*调整文件大小。如果当前日志文件的空间量少于指定的空余空间量，则调整文件大小。

maybe （*续*）　　如果不从日志文件删除数据就可以调整大小 （例如，增加最大日志文件大小或减少最大日志文件大小，而不用删除现有数据），则调整大小。
maybe 参数主要用于定期批量执行。有关如何以此方式使用 resize 命令的说明，请参见下面的 “示例”小节。

下表显示了默认调整大小参数。

表 6　　默认调整大小参数

参数	如果交互执行	如果批量执行
log file type	提示您输入每个可用的日志文件类型。	无默认值。这是必需参数。
days size	当前文件大小。	当前文件大小。
empty space	当前的空余空间量或足以保留当前文件中的所有数据的空余空间量，两者中的较小者。	当前的空余空间量或足以保留当前文件中的所有数据的空余空间量，两者中的较小者。

表 6　　默认调整大小参数 （续）

参数	如果交互执行	如果批量执行
yes no maybe	提示您按报告的磁盘空间结果执行。	yes。将调整大小。

如何使用

在调整日志文件大小之前，*必须*使用第 2 章第 37 页的停止和重新启动数据收集下面的步骤停止性能收集组件。

原始日志文件必须打开才能执行大小调整。在发出 resize 命令之前，用 logfile 命令打开原始日志文件。不能用任何其他进程打开文件。

resize 命令在 TMPDIR 环境变量设置的目录中创建新文件 scopelog，然后再删除原始日志文件。如果不设置环境变量 TMPDIR，将使用 /var/tmp 目录（在 IBM AIX 4.1 及更高版本上为 /tmp）作为临时位置。确保 TMPDIR 指定的目录或 /var/tmp 目录（在 IBM AIX 4.1 及更高版本上为 /tmp）中有足够的磁盘空间可以容纳原始日志文件，再执行大小调整过程。

调整大小之后，日志文件由数据和空余空间组成。保留的数据是从最大文件大小减去所需空余空间算出的。调整大小期间删除的任何数据都会丢失。要将日志文件数据保留更长的时间，请在执行调整大小操作*前*使用 extract 将此数据复制到提取的文件。

resize 命令报告

调整原始日志文件大小时会生成一个标准报告。它显示调整大小前后最大文件大小、数据记录和空余空间这三种相互关联的磁盘空间类别。例如：

```
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
```

每天兆字节值用于换算天数和 MB 值。它是执行 scan 函数期间获得的值或要调整大小的数据类型的默认值。

最右边的列是每种类别的日志文件空间的净更改摘要。最大大小和空余空间可以增加、减少或保持不变。数据记录可以在调整大小期间未删除任何数据，也可以删除指定数据量。

如果以交互模式执行大小调整，并且有一个或多个参数是默认值，您可以获得初步大小调整报告。此报告汇总当前日志文件内容和提供的任何参数。提供此报告是为了帮助回答有关未指定参数的问题。例如：

```
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)     ??
```

在此示例中，提示您提供文件的空余空间量，才能给出最终大小调整报告。如果不为交互式大小调整提供操作参数，会询问您是否在生成最终大小调整报告后立即调整日志文件大小。

示例

以下命令用于调整原始进程日志文件的大小。在调整大小之前执行扫描，以提高天数计算的准确性。

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

`days=60` 指定最多保留 60 天的数据。`empty=30` 指定此文件当前空余空间为 30 天。也就是说，文件中只有 30 天的数据时就调整文件大小，以便为总共 60 天空间的另外 30 天数据留出空间。 `yes` 指定不管当前空余空间为多少都应执行调整大小操作。

下一个示例显示了如何在批处理模式下使用 `resize` 命令，以确保日志文件在接下来的一周中不会填满（强制 `scope` 调整文件大小）。可以使用 `at` 命令安排 `cron` 脚本，该命令指定最小空间量（如 7 天，为了安全起见也可能是 10 天）。

以下 shell 脚本完成此任务：

```
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
```

如果任何日志文件中当前有 10 天或更多的空余空间，指定 maybe 而不是 yes 可避免任何调整大小操作。请注意，最大文件大小默认为每个文件当前最大的文件大小。这允许在不影响此脚本的情况下将文件调整到新的最大大小。

scan

使用 scan 命令读取日志文件并编写有关其内容的报告。（有关报告的详细描述，请参见第 3 章第 63 页的 utility scan 报告详细信息。）

语法

scan

如何使用

scan 命令需要打开日志文件。扫描的日志文件是以下项的第一个文件：

- scan 命令中指定的日志文件。
- 由任何前面的命令打开的最后一个日志文件。
- 默认日志文件。

  在这种情况下，系统会提示交互用户在必要时覆盖默认日志文件名称。

以下命令会影响 scan 函数的运行：

detail	指定报告的详细程度。默认值是 detail on，指定完整详细信息。
list	将输出重定向到另一个文件。默认值是列出到标准列表设备。
start	指定要扫描的第一条日志文件记录的日期和时间。默认值是日志文件的开头。
stop	指定要扫描的最后一条日志文件记录的日期和时间。默认值是日志文件的结尾。

有关 detail、 list、 start 和 stop 命令的详细信息，请参见本章相应部分的描述。

scan 命令报告由 12 个部分组成。可以在发出 scan 命令之前发出 detail 命令来控制报告中包括的部分。

即使指定 detail off，也总是打印以下四个部分：

- 扫描开始和停止的实际日期和时间
- 收集器覆盖时间摘要
- 日志文件内容摘要
- 日志文件空余空间摘要

如果指定 detail on（默认值），则打印以下部分：

- 初始 parm 文件全局信息和系统配置信息
- 初始 parm 文件应用程序定义
- parm 文件全局更改
- parm 文件应用程序添加 / 删除通知
- 收集器关闭时间通知
- 特定于应用程序的摘要报告

如果扫描了应用程序数据，即使指定 detail off 也总是打印以下部分：

- 应用程序总体摘要

如果扫描了进程数据，即使指定 detail off 也总是打印以下部分：

- 进程日志原因摘要

示例

当前默认全局日志文件的扫描从 1999 年 6 月 1 日 7:00 AM 记录的记录开始，到当前日期和时间结束。

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

要使用命令行参数执行上面的任务，请输入：

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

sh

使用 sh 可以不用退出 utility 而输入 shell 命令，方法是输入 **sh** 或感叹号 (**!**) 后跟 shell 命令。

语法

sh 或 **! [shell 命令]**

参数

`sh ls`　　执行 `ls` 命令并返回到 `utility`。

`!ls`　　同上。

如何使用

在执行单个命令后，将自动返回到 `utility`。如果要发出多条 shell 命令，在每条命令之后不返回到 `utility`，则可以启动新 shell。例如，

```
sh ksh
```

或

```
!ksh
```

show

使用 `show` 命令列出打开的文件的名称和可设置的 `utility` 参数的状态。

语法

```
show [all]
```

示例

使用 `show` 生成的列表可能如下所示：

```
Logfile:  /var/opt/perf/datafiles/logglob
List:     "stdout"
Detail:   ON for ANALYZE, PARMFILE and SCAN functions
The default starting date & time = 10/08/99 08:17 AM (FIRST + 0)
The default stopping date & time = 11/20/99 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM
```

默认的轮换时间仅供参考。在 utility 中不能更改轮换时间。

使用 `show all` 生成的更详细列表可能如下所示：

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

```
Maximum file sizes:
  Global=10.0 Application=10.0 Process=20.0 Device=10.0 Transaction 10.0
MB
List    "stdout"
Detail  ON for ANALYZE, PARMFILE and SCAN functions
The default starting date & time = 10/08/99 11:50 AM (FIRST + 0)
The default stopping date & time = 11/20/99 11:59 PM (LAST  - 0)
The default shift = 12:00 AM - 12:00 AM
```

start

使用 start 命令可以指定要扫描或分析的日志文件一部分的开头。start 可用于在特定日期和时间记录的数据处启动 scan 或 analyze 进程。

默认开始日期和时间设置为当前用 logfile 命令打开的日志文件中任何类型的第一条记录的日期和时间。

语法

	[日期	*[时间]]*	
start	**[today**	*[- 天数]*	*[时间]]*
	[last	*[- 天数]*	*[时间]]*
	[first	*[+ 天数]*	*[时间]]*

参数

日期　　日期格式取决于所用系统上配置的本机语言。如果不使用本机语言或将默认语言设置为 C，日期格式则为 *mm/dd/yy*（月 / 日 / 年），比如 06/30/99 表示 1999 年 6 月 30 日。

时间　　时间格式也取决于所用的本机语言。 C 语言的时间格式是 *hh:mm* am 或 *hh:mm* pm（12 小时格式的“小时 : 分钟”加 am/pm 后缀），比如 07:00 am 表示上午 7 点钟。所有语言都接受 24 小时制时间。例如， 23:30 会认为是 11:30 pm。
如果以不可接受格式输入日期或时间，将会显示正确格式的示例。
如果不指定开始时间，则假定是午夜 (12 am)。一天的午夜开始时间始于当天的*开始*（24 小时制的 00:00）。

today　　指定当天。参数 today-*天数*指定当天日期之前的天数。例如， today-1 表示昨天的日期， today-2 表示前天的日期。

last　　可用于表示日志文件中包含的最晚日期。参数 last-*天数*指定日志文件中最晚日期之前的天数。

first　　可用于表示日志文件中包含的最早日期。参数 first+ *天数*指定日志文件中最早日期之后的天数。

如何使用

如果您有一个很大的日志文件，又不想扫描或分析整个文件，这时 start 命令会非常有用。您还可以用它指定扫描数据的特定时间段。例如，要扫描日志文件中当前日期之前的 14 天开始记录的数据，可以指定 **today-14**。

使用 stop 命令进一步限制要扫描的日志文件记录。

如果不确定系统上是否安装了本机语言支持，可以在运行 utility 之前发出以下语句来强制 utility 使用 C 日期和时间格式：

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

或在 C Shell 中

```
setenv LANG C
```

示例

默认全局日志文件的扫描从 1999 年 8 月 5 日 8:00 AM 记录的记录开始，到当前日期和时间结束。

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

要使用命令行参数执行上面的任务，请输入：

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

stop

使用 stop 命令指定要扫描或分析的日志文件一部分的结尾。stop 可用于在特定日期和时间记录的数据处终止 scan 或 analyze 进程。

默认停止日期和时间设置为当前用 logfile 命令打开的日志文件中任何类型的最后一条记录的日期和时间。

语法

	[日期	*[时间]]*	
stop	**[today**	*[- 天数]*	*[时间]]*
	[last	*[- 天数]*	*[时间]]*
	[first	*[+ 天数]*	*[时间]]*

参数

日期　　日期格式取决于所用系统上配置的本机语言。如果不使用本机语言或将默认语言设置为 C，日期格式则为 *mm*/*dd*/*yy*（月 / 日 / 年），比如 06/30/99 表示 1999 年 6 月 30 日。

时间　　时间格式也取决于所用的本机语言。 C 语言的时间格式是 *hh:mm* am 或 *hh:mm* pm（12 小时格式的“小时 : 分钟”加 am/pm 后缀），比如 07:00 am 表示上午 7 点钟。所有语言都接受 24 小时制时间。例如， 23:30 会认为是 11:30 pm。
如果以不可接受格式输入日期或时间，将会显示正确格式的示例。
如果不指定停止时间，则假定是午夜前一分钟 (11:59 pm)。一天的午夜 (12 am) 停止时间止于当天的*结束*（24 小时制的 23:59）。

today　　指定当天。参数 today-*天数*指定当天日期之前的天数。例如， today-1 表示昨天的日期， today-2 表示前天的日期。

last　　可用于表示日志文件中包含的最晚日期。参数 last-*天数*指定日志文件中最晚日期之前的天数。

first　　可用于表示日志文件中包含的最早日期。参数 first+ *天数*指定日志文件中最早日期之后的天数。

如何使用

如果您有一个很大的日志文件，又不想扫描整个文件，这时 stop 命令会非常有用。您还可以用它指定扫描数据的特定时间段。例如，可以扫描日志文件中当前日期之前一个月开始记录的七天数据。

如果不确定系统上是否安装了本机语言支持，可以在运行 utility 之前发出以下语句来强制 utility 使用 C 日期和时间格式：

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

或在 C Shell 中

```
setenv LANG C
```

示例

扫描从 1999 年 7 月 5 日 8:00 am 记录的记录开始，到 1999 年 7 月 18 日 11:59 pm 记录的最后一条记录结束这 14 天的数据。

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

要使用命令行参数执行上面的任务，请输入：

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

6 使用 extract 程序

extract 程序有两个主要功能：从原始日志文件提取数据，并将数据写入提取的日志文件中。extract 还用于导出日志文件数据，供电子表格等分析产品使用。

初始安装性能收集组件后，必须启动服务以完成文件安装，之后 extract 才能起作用。

extract 和 export 函数从日志文件*复制*数据；*不*删除数据。

性能收集组件使用三种类型的日志文件：

- scope 日志文件，包含 scope 收集器在性能收集组件中收集的数据。
- 提取的日志文件，包含从原始 scope 日志文件提取的数据。
- DSI（数据源集成）日志文件，包含由外部源（如应用程序和数据库）收集的用户定义的数据。该数据随后由性能收集组件的 DSI 程序记录。

使用 extract 程序可执行以下任务：

- 将原始 scope 日志文件中的数据子集提取为适合存档、系统间的传输以及适合 Performance Manager 分析的提取日志文件格式。*不能*从 DSI 日志文件提取数据。
- 通过从提取的格式文件提取或导出数据、将数据追加到现有的提取的日志文件以及按类型、日期和轮换（一天中的几点钟）组织数据来管理存档的日志文件数据。
- 将原始或提取的 scope 日志文件和 DSI 日志文件中的数据提取为适合报表和分析或适合导入到电子表格或类似分析包的 ASCII、二进制、数据文件或 WK1（电子表格）格式。

- extract 函数不能生成摘要数据。摘要数据只能由 export 函数生成。
- extract 和 export 函数的最大限制如下：
  - extract 函数的输出文件不能超过 3.5 GB
  - export 函数的输出文件不能超过 4 GB

如何执行各种任务及如何使用 extract 命令的示例可以在 extract 程序的联机帮助中找到。

本章包含以下主题：

运行 extract 程序

可采用三种方式运行 extract 程序：

命令行模式

在命令行中使用命令选项和参数控制 extract 程序。

交互模式

执行程序时提供交互命令和参数，将 stdin 设置为交互终端或工作站。
如果您是有经验的用户，可以只快速指定给定任务所需的那些命令。如果您是新用户，可能希望指定引导模式以获取有关使用 extract 的更多帮助。在引导模式下，系统会要求您从选项列表选择以执行任务。在引导模式下，每个完成任务的交互命令执行后都会被列出，因此您可以看到是如何使用它们的。您可以随时退出或重新进入引导模式。

批处理模式

可以运行程序并将 stdin 重定向到包含交互命令和参数的文件。

语法

命令行界面的语法与其他程序上的标准 UNIX 命令行界面类似，本章将进行详细描述。

交互模式和批处理模式的命令语法相同：命令后跟一个或多个参数。命令可以按任意顺序输入；如果命令有关联的参数，则必须在输入对应命令之后立即输入该参数。

有两种类型的参数：*必需*（无默认值）和 *可选*（提供默认值）。 extract 程序处理这些参数的方式取决于它的运行模式。

- 在交互模式下，如果缺少 *可选* 参数，程序将显示默认参数，您可以确认或覆盖该参数。如果缺少 *必需* 参数，程序将提示您输入参数。
- 在批处理模式下，如果缺少 *可选* 参数，程序将使用默认值。如果缺少 *必需* 参数，程序将终止。

交互模式下处理错误和缺少数据的方式与命令行和批处理模式下的处理方式不同，因为在交互模式下您可以提供其他数据或更正错误，在命令行和批处理模式下则不行。

使用交互模式

使用 extract 程序的交互模式需要您发出一系列命令以执行特定任务。

例如，如果要从默认全局日志文件中导出自 2003 年 5 月 15 日开始收集的应用程序数据，在调用 extract 程序之后需发出以下命令

```
logfile /var/opt/perf/datafiles/logglob
application detail
start 5/15/2003
export
```

logfile 命令打开默认全局日志文件 /var/opt/perf/datafiles/logglob。start 命令指定只导出 2003 年 5 月 15 日之后记录的数据。 export 命令开始导出数据。

extract 命令行接口

除了交互模式和批处理模式命令语法外，还可将命令选项和参数通过命令行界面传递到 extract 程序。命令行界面允许 shell 脚本轻松调用 extract 程序，并允许输入和输出重定向到 UNIX 管道，因而完全适用于典型的 UNIX 环境。

例如，与上一部分第 92 页的使用交互模式中所示的示例等效的命令行是：

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

在命令行模式下，全局日志文件 /var/opt/perf/datafiles/logglob 是默认的；您不必指定。

下表中列出了命令行选项和参数。所引用命令的描述可在第 7 章 extract 命令中找到。

表 7　　命令行参数

命令选项	参数		描述
-b	date	time	指定 extract 或 export 函数的开始日期和时间。（请参见第 6 章的 start 命令。）
-B		UNIX *开始时间*	指定 extract 或 export 函数的 UNIX 格式的开始时间。
-e	date	time	指定 extract 或 export 函数的结束日期和时间。（请参见第 6 章的 stop 命令。）
-E		UNIX *停止时间*	指定 extract 或 export 函数的 UNIX 格式的停止时间。
-s	time-time	noweekends	指定特定时段的开始和结束时间，排除周末。（请参见第 6 章的 shift 命令。）
-l	logfile		指定输入日志文件。（请参见第 6 章的 logfile 命令。） /var/opt/perf/datafiles/logglob 是默认值。

表 7　　命令行参数 （续）

命令选项	参数		描述
-r	export template file		指定 export 函数的导出模板文件。（请参见第 6 章的 report 命令。）
-C	classname	opt	指定要提取或导出的 scope 数据，或要导出的自描述 (DSI) 数据。（请参见第 6 章的 class 命令。） opt = detail （默认值） summary both off
-i			指定要从逻辑系统提取或导出的 scope 数据。
-k			仅导出已终止的进程。如果使用此选项，则在 reptfile 中包括 PROC_INTEREST 度量。
gapkcdzntuy iGADZNTUYI			指定要提取 / 导出的数据类型： g = 全局详细信息。（请参见第 6 章的 global 命令。）全局详细信息在默认情况下是关闭的。 a = 应用程序详细信息。（请参见第 6 章的 application 命令。） p = 进程详细信息 （请参见第 6 章的 process 命令。） k = 终止的进程。（请参见第 6 章的 process 命令。） c = 配置详细信息 （请参见第 6 章的 configuration 命令。） d = 磁盘设备详细信息 （请参见第 6 章的 disk 命令。） z = lvolume 详细信息（请参见第 6 章的 lvolume 命令。） n = netif 详细信息 （请参见第 6 章的 netif 命令。）

表 7　　命令行参数 （续）

命令选项	参数		描述
gapkcdzntuy iGADZNTUYI （续）			t = 事务详细信息 u = CPU 详细信息 y = 文件系统详细信息 i = 逻辑系统详细信息 **注：**以下摘要选项仅适用于 export；extract 函数不支持数据汇总。 G = 全局摘要 全局摘要在默认情况下是关闭的。 A = 应用程序摘要 D = 磁盘设备摘要 （请参见第 6 章的 disk 命令。） Z = lvolume 摘要 （请参见第 6 章的 lvolume 命令。） N = netif 摘要 （请参见第 6 章的 netif 命令。） I = 逻辑系统摘要
gapkcdzntuy GADZNTUY （续）			T = 事务摘要 U = CPU 摘要 （请参见第 6 章的 cpu 命令。） Y = 文件系统摘要 （请参见第 6 章的 filesystem 命令。）
-v			生成详细输出报告格式。
-f	filename	,new ,append ,purge	将提取或导出数据发送到文件。如果没有文件名，则将数据发送到默认输出文件。（请参见第 6 章的 output 命令。）
-ut			显示导出的 DSI 日志文件数据中 UNIX 格式的日期和时间。
-we	1.....7		指定要从 export 排除的日子； 1= 星期日。（请参见 weekdays 命令描述。）
-xp	xopt		将数据导出到外部格式文件。（请参见第 6 章的 export 命令。）

表 7　　命令行参数 （续）

命令选项	参数		描述
-xt	xopt		以系统内部格式提取数据。（请参见第 6 章的 extract 命令。） xopt = *dwmy* (日 周 月 年) *dwmy*-[offset] *dwmy* [absolute]
-xw	week		提取日历周的数据。（请参见第 6 章的 weekly 命令。）
-xm	month		提取日历月的数据。（请参见第 6 章的 monthly 命令。）
-xy	year		提取日历年的数据。（请参见第 6 章的 monthly 命令。）
-? 或 ?			显示命令行语法。

对参数求值并在命令行输入命令选项时，以下规则适用：

- 错误和缺少数据的处理方式与对应批处理模式命令中的处理方式完全相同。即，如果可能，将对缺少的数据使用默认值，而所有错误都将导致立即终止程序。
- 除非使用 -v 参数启用详细模式，否则将禁用命令和命令结果的回显。
- 如果未指定有效操作（-xp、-xw、-xm、-xy 或 -xt），extract 将在处理完所有参数后开始从其 stdin 文件读取命令。
- 如果指定了操作 （-xp、 -xw、 -xm、 -xy 或 -xt），程序将在对所有其他参数求值之后执行这些命令选项，而不管它们在参数列表中的位置。
- 如果在命令行中指定了操作， extract 程序不会从其 stdin 文件读取信息；它将终止执行操作：

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

解释为：

-f rxdata　　输出到当前目录中名为 rxdata 的文件

-r rept1　　文件 /var/opt/perf/rept1 包含所需的导出格式

-xp d-1　　导出今天减 1 （昨天）的数据

-G　　导出全局摘要数据。

请注意，结束之前不会执行实际导出，因此 -G 参数是在导出前处理的。

还请注意，由于未指定日志文件，因此它使用默认的 logglob 文件。

因为指定了操作 (-xp)，一旦完成导出， extract 程序即终止，而不从其 stdin 文件读取。此外，未用 -v 命令选项设置详细模式，因此 stdout 的所有无关输出都将删除。

export 函数的概述

extract 程序的 export 命令将性能收集组件的原始、 DSI 或提取的日志文件数据转换成导出文件。export 命令以四种可能格式之一写入文件：ASCII、数据文件、二进制和 WK1（电子表格）。导出文件可以多种方式使用，如报告、自定义图形包、数据库和用户编写的分析程序。

如何导出数据

您可以通过以下最简单的方式导出数据：

- 指定要从中导出数据的默认全局日志文件 /var/opt/perf/datafiles/logglob
- 指定定义导出数据的格式的默认导出模板文件 /var/opt/perf/reptfile
- 启动 export 函数。

导出的数据放在当前目录中名为 xfrdGLOBAL.asc 的默认输出文件中。输出文件的 ASCII 格式适于打印。

如果要导出除此默认数据集以外的对象，可将其他命令和文件与 export 命令结合使用。

可导出以下数据类型：

global	5 分钟和每小时摘要
application	5 分钟和每小时摘要
process	1 分钟详细信息
disk device	5 分钟和每小时摘要
lvolume	5 分钟和每小时摘要
transaction	5 分钟和每小时摘要
configuration	每次启动数据收集器时都显示一条包含 parm 文件信息和系统配置信息的记录。
任何 DSI 类	DSI 日志文件的间隔和摘要
netif	5 分钟和每小时摘要
cpu	5 分钟和每小时摘要
filesystem	5 分钟和每小时摘要

- 您可以指定每种数据类型所需的数据项 （度量）。
- 可以指定收集数据时段的开始和结束日期，以及轮换和周末排除过滤器。

- 可以在导出模板文件中指定导出数据所需的格式。此文件可使用任何可将文件保存为 ASCII（文本）格式的文本编辑器或字处理器创建。
- 还可使用默认导出模板文件 /var/opt/perf/reptfile。此文件指定以下输出格式设置：
  - ASCII 文件格式
  - 0（零）表示缺少值
  - 字段分隔符是空格
  - 60 分钟摘要
  - 包括列标题
  - 导出中包括给定数据类型的建议度量集

如果从特定平台创建的日志文件提取或导出数据，建议使用同一平台的 reptall 文件。这是因为不同平台上支持的度量类列表不同。

示例导出任务

性能收集组件提供两个示例导出模板文件：repthist 和 reptall。这两个文件位于 /var/opt/perf/ 目录中。可使用 repthist 和 reptall 执行常见的导出任务，或将其作为自定义任务（如接下来描述的任务）的起点。

生成可打印的 CPU 报告

repthist 导出模板文件包含用于生成系统随时间变化的 CPU 和磁盘使用情况字符图的规范。此图由可在能进行 132 列打印的任何设备上打印的可打印字符组成。例如，可使用以下 extract 程序命令生成最近七天的图，大约生成两页（如果指定 5 分钟详细信息而非每小时摘要，则有 34 页）。

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

导出的数据在名为 xfrsGLOBAL.asc 的导出文件中。要打印该文件，请输入：

```
lp xfrsGLOBAL.asc
```

生成自定义的导出文件

如果要创建全新的导出模板文件，请使用 extract 程序的 guide 命令复制导出模板文件并自定义。在引导模式下，从 /var/opt/perf/ 目录复制 reptall 文件，并读取为动态创建数据类型和度量名称列表而指定的 scope 或 DSI 日志文件。

reptall 文件包含每种类型的 scope 日志文件数据的每种可能度量，因此只需要取消注释您感兴趣的那些度量。这比重新输入整个导出模板文件容易。

导出数据文件

如果在发出 export 命令之前已使用 output 命令指定输出文件的名称，所有导出的数据都将写入这个文件。如果以交互方式运行 extract 程序，并要直接将数据导出到工作站（标准输出文件），请在发出 export 命令之前指定 extract 命令 output stdout。

如果输出文件设置为默认值，则导出的数据根据导出数据的类型最多可分成 14 个不同的默认输出文件。

默认的导出日志文件名称有：

xfrdGLOBAL.ext	全局详细信息数据文件
xfrsGLOBAL.ext	全局每小时摘要数据文件
xfrdAPPLICATION.ext	应用程序详细信息数据文件
xfrsAPPLICATION.ext	应用程序每小时摘要数据文件
xfrdPROCESS.ext	进程详细信息数据文件
xfrdDISK.ext	磁盘设备详细信息数据文件
xfrsDISK.ext	磁盘设备每小时摘要数据文件
xfrdVOLUME.ext	逻辑卷详细信息数据文件
xfrsVOLUME.ext	逻辑卷摘要数据文件
xfrdNETIF.ext	Netif 详细信息数据文件
xfrsNETIF.ext	Netif 摘要数据文件
xfrdCPU.ext	CPU 详细信息数据文件
xfrsCPU.ext	CPU 摘要数据文件
xfrdFILESYSTEM.ext	文件系统详细信息数据文件
xfrsFILESYSTEM.ext	文件系统摘要数据文件
xfrdTRANSACTION.ext	事务详细信息数据文件
xfrsTRANSACTION.ext	事务摘要数据文件
xfrdCONFIGURATION.ext	配置数据文件

其中 ext= asc (ASCII)、bin（二进制）、dat（数据文件）或 wk1（电子表格）。

*只有*在发出 export 命令之前指定与导出模板文件中的数据匹配的类型和关联项后，才会创建输出文件。

导出模板文件语法

导出模板文件可包含以下全部或部分信息，具体取决于您希望以何种格式导出数据，以及希望导出文件包含什么：

```
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
```

参数

`report` 指定导出文件的标题。有关详细信息，请参见下一部分第 102 页的导出文件标题。

`format` 指定导出数据的格式。

ASCII

`ASCII`（或文本）格式最适合将文件复制到打印机或终端。它不在字段两边加双引号 (")。

Datafile

`datafile` 格式与 ASCII 格式类似，但非数字字段两边加了双引号。使用双引号后难以确保严格的列对齐，所以建议不要直接打印 `datafile` 格式的文件。`datafile` 格式是最容易导入到大多数电子表格和图形包中的格式。

Binary

`Binary` 格式占用的空间更小，因为数字值表示为二进制整数。它是最适合输入到用户编写的分析程序的格式，因为它需要的转换最少，度量准确性最高。它不适合直接打印。

WK1 (spreadsheet)

`WK1` （spreadsheet）格式与 Microsoft Excel 及其他电子表格和图形程序兼容。

headings　指定是否在导出文件中包含列出的度量的列标题。如果指定了 headings off，列标题将不写入文件。文件中的第一条记录是导出的数据。如果指定了 headings on， ASCII 和 datafile 格式将写入的每列度量的导出标题和列标题放在第一条数据记录*之前*。binary 格式文件中的列标题包含文件中度量的描述。WK1 格式始终包含列标题。

separator　指定 ASCII 或 datafile 格式的数据中每个字段之间打印的字符。默认的分隔符是空格。很多程序都首选逗号作为字段分隔符。可以将分隔符指定为任何打印或非打印字符。

summary　指定每个摘要间隔的分钟数。该值确定摘要记录的每条记录中包括的时间。默认间隔是 60 分钟。摘要值可设置为 5 到 1440 分钟 （1 天）之间。

missing　指定要用来代替缺少的数据的数据值。用于缺少的数据的默认值是零。您可以指定其他值，以便区分缺少的数据与 0 数据。以下情况下可能缺少数据项：

- 特定版本的 scope 收集器未记录该数据项
- scope 未记录改数据项，因为它所属的实例 （application、 disk、 transaction、 netif）在该间隔期间未处于活动状态，或者
- 在使用 DSI 日志文件的情况下，间隔期间未向 dsilog 程序提供数据，导致“缺少记录”。

默认情况下，导出的数据中不包括缺少的记录。

layout　为 application、disk、transaction、lvolume 或 netif 等数据类型指定 single 或 multiple 布局 （每个记录输出）。

- 单布局为时间间隔期间的每个活动应用程序的每条记录写入一个实例。示例：将一张磁盘导出到一条记录中。
- 多布局针对每个时间间隔在一条记录中写入多个实例，该记录的一部分为每个配置的应用程序保留。示例：将所有磁盘导出到一条记录中。

output　指定导出的数据将写入的输出文件的名称。可通过将 output *文件名*放到指示导出的数据项列表开始的数据类型的那一行之后，为每一类或导出的 data type 指定 output。可以为 output 指定任何有效的文件名。

还可以使用输出命令以交互方式指定名称。您指定的任何名称都将覆盖默认的输出文件名。

data type　指定一个可导出的数据类型：global、 application、 process、 disk、 transaction、 lvolume、 netif、 configuration 或 DSI 类名。该操作将启动列出导出此类型数据时要复制的数据项的导出模板文件部分。

items　指定导出文件中要包含的度量。度量名称按您希望它们在结果文件中列出的顺序列出，每行一个名称。在列出项之前必须指定正确的数据类型。同一导出模板文件可包含任意多数据类型的项列表。每种数据类型都*只*在您选择导出该类型数据时才引用。

output 和 layout 参数可以在一个导出模板文件中使用多次。例如：

```
data type global
   output=myglobal
   gbl_cpu_total_util

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

系统上可以有多个导出模板文件。每个文件都可以定义一组导出的文件格式来适应特定需要。使用 report 命令指定 export 函数要使用的导出模板文件。

不能在一个数据类型中指定不同的布局。例如，在同一个导出文件中，不能以 **layout = multiple** 指定 **data type disk** 一次，然后再以 **layout = single** 指定一次。

导出文件标题

导出文件标题字符串中的以下项可以替换：

!date	执行 export 函数的日期。
!time	执行 export 函数的时间。
!logfile	源日志文件的完全限定名。
!class	所请求数据的类型。
!collector	收集器程序的名称和版本。（对 DSI 日志文件无效。）
!system_id	收集数据的系统的标识符。（对 DSI 日志文件无效。）

例如，字符串

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

生成类似下面的导出文件标题

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

创建自定义图形或报告

假定您要创建包含导出的全局和应用程序数据的自定义图形或报告。您将执行以下操作：

1 确定每个数据类型需要哪些数据项 （度量），以及将以什么格式访问它们。

对于此示例，您希望得到无标题的 ASCII 文件，字段以逗号分隔。

2 创建以下 ASCII 导出模板文件并保存到 /var/opt/perf/ 目录中。将文件命名为 report1。

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

DATA TYPE GLOBAL
   GBL_CPU_TOTAL_UTIL
   GBL_DISK_PHYS_IO_RATE

DATA TYPE APPLICATION
   APP_CPU_TOTAL_UTIL
   APP_DISK_PHYS_IO_RATE
   APP_ALIVE_PROCESSES
```

3 运行 extract 程序。

4 发出 report 命令指定您创建的导出模板文件。

```
report /var/opt/perf/report1
```

5 使用 global 和 application 命令指定全局摘要数据和应用程序摘要数据。

```
global summary
application summary
```

6 发出 export 命令开始导出。

```
export
```

7 由于您未指定程序应从何处获取性能数据，系统将提示您指定。在此示例中，因为默认的日志文件是正确的，只需按 **Enter**。

8 输出与下面类似：

```
exporting global data ........50%......100%
exporting application data .....50%......100%
The exported file contains 31 days of data from 01/01/99 to 01/31/99
                           examined  exported
data type                   records   records     space
----------------------  ---------- ---------- ---------
global      summaries                     736   0.20 Mb
application summaries                    2560   0.71 Mb
                                                ---------
                                                  0.91 Mb
```

您刚创建的两个文件（xfrsGLOBAL.asc 和 xfrsAPPLICATION.asc）包含指定格式的全局和应用程序摘要数据。

导出文件的输出

每个导出文件的内容包括：

导出标题行	如果指定了导出标题和 headings on。
名称（application、netif、lvolume 或 transaction）	如果指定了 headings on 及多布局文件。
标题行 1	如果指定了 headings on。
标题行 2	如果指定了 headings on。
第一条数据记录	

第二条数据记录

...

最后一条数据记录

文件中不会重复报告标题和标题行。

ASCII 和数据文件格式备注

这些格式文件中的数据是可打印的 ASCII 格式。ASCII 和 datafile 格式基本相同，除了数据文件格式中所有非数字字段两边加了双引号外。甚至 datafile 标头信息也包含在双引号中。

ASCII 文件格式不在字段两边加双引号。因此，打印时， ASCII 文件中的数据将正确对齐。

数字值根据其范围和内部准确度确定格式。由于所有字段的长度不一定相同，请确保指定要用于开始每个字段的分隔符。

用户指定的分隔符 （或默认的空格）分隔 ASCII 和 datafile 格式中的各个字段。如果要打印报告，使用空格作为分隔符可能更美观。如果打算用其他程序读取导出模板文件，则使用其他字符作为分隔符可能更有用。

很多应用程序都接受使用逗号作为分隔符，但某些数据项可能包含*并非分隔符*的逗号。这些逗号可能使分析程序混淆。根据执行 extract 程序时指定的本机语言，日期和时间格式可能包含不同的特殊字符。

要使用非打印的特殊字符作为分隔符，请将它输入到导出模板文件中紧跟 separator 参数中的第一个双引号。

- 大多数电子表格接受使用 separator="," 的数据文件格式的文件。
- 很多电子表格包接受单个表中最多有 256 列。导出多布局类型的数据时要小心，因为容易生成总项数超过 256 的表。您可以使用 “report 报告文件 , show” 命令的输出判断您是否可能遇到这个问题。
- 如果有支持下划线的打印机，可通过指定 ASCII 格式和竖线字符 (separator=|) 然后打开下划线打印文件，创建更美观的打印输出。

二进制格式备注

在 binary 格式文件中，数字值写为 32 位整数。这减小了总的文件大小，可节省空间，但程序必须能够读取 binary 文件。建议您不要将 binary 格式文件复制到打印机或终端。

binary 格式中写入非数字数据的方式与 ASCII 格式中的写入方式相同，但它不使用分隔符。要正确使用 binary 格式文件，应使用指定 **report** *报告文件* **,show** 时由 extract 打印的记录布局报告。此报告提供指定的每项的起始字节。

为保持最大精度、避免非标准的 binary 浮点表示，所有数字值都写为成比例的 32 位整数。某些项在截断为整数格式之前，可能要乘以一个常量。

例如，CPU 使用的秒数在截断之前要乘以 1000。要将导出文件中的值转换回实际秒数，请用 1000 除。为方便转换，请指定 **headings on** 将缩放比例写入导出文件。报告标题和特殊标头记录写入 binary 格式文件，以帮助进行程序解译。

二进制整数以运行 extract 程序的系统的本机格式写入。

二进制标头记录布局

二进制格式导出的文件中的每条记录的任何用户指定的数据前面都包含特殊的 16 字节记录标头。report *报告文件*,show 命令包括组成此记录标头的以下四个字段：

二进制记录标头度量

Record Length	记录中的字节数，包括 16 字节记录标头。
Record ID	标识记录类型的数字 （见下）。
Date_Seconds	从 1970 年 1 月 1 日开始经过的时间 （以秒为单位）。
Number_of_vars	此记录中的重复条目数。

Record ID 度量唯一标识记录中所含的数据类型。当前 Record ID 值为：

-1	标题记录
-2	第一条标头记录 （包含项号）
-3	第二条标头记录 （包含项缩放比例）
-4	应用程序名称记录 （适用于多实例应用程序文件）
-5	事务名称记录 （适用于多实例事务文件）
-7	磁盘设备名称记录 （适用于多实例磁盘设备文件）
-8	逻辑卷名称记录 （适用于多实例 Lvolume 文件）
-9	Netif 名称记录 （适用于多实例 Netif 文件）
-10	文件系统名称记录 （适用于多实例 Netif 文件）
-11	CPU 名称记录 （适用于多实例 Netif 文件）

1	全局数据记录	（5 分钟详细信息记录）
101	全局数据记录	（60 分钟摘要记录）
2	应用程序数据记录	（5 分钟详细信息记录）
102	应用程序数据记录	（60 分钟摘要记录）
3	进程数据记录	（1 分钟详细信息记录）
4	配置数据记录	

```
  7   磁盘设备数据记录 （5 分钟详细信息记录）
107   磁盘设备数据记录 （60 分钟摘要记录）
  8   逻辑卷数据记录 （5 分钟详细信息记录）
108   逻辑卷数据记录 （60 分钟摘要记录）
  9   文件系统数据记录 （5 分钟详细信息记录）
109   文件系统数据记录 （60 分钟摘要记录）
 11   Netif 数据记录 （5 分钟详细信息记录）
111   Netif 数据记录 （60 分钟摘要记录）
 12   事务数据记录 （5 分钟详细信息记录）
112   事务数据记录 （60 分钟摘要记录）
 13   CPU 数据记录 （5 分钟详细信息记录）
113   CPU 数据记录 （60 分钟摘要记录）
ClassID +1,000,000 类数据记录 （5 分钟详细信息记录）
ClassID +1,000,000+100 类数据记录 （60 分钟摘要记录）
```

Date_Seconds 度量与用户可选择的 Date_Seconds 度量相同，该度量确保了可以轻松扫描所需日期和时间的记录。

Number_of_vars 度量指示记录中包含的重复字段组数。对于单实例数据类型，此值为零。

对于多实例应用程序记录，Number_of_vars 度量是配置的应用程序数。对于多实例磁盘设备记录，Number_of_vars 度量是配置的磁盘设备数。对于所有标头记录，此度量是允许的最大重复组数。

二进制格式文件的标题和标头记录有特殊格式。这些记录包含描述文件内容以便程序可以正确解译文件所需的信息。如果指定了 headings off，文件中将只有数据记录。如果指定了 headings on，以下记录将出现在所有数据记录之前。

二进制标头记录

标题记录	只要指定了 headings on，就写入此记录 （记录 ID -1），而不管用户是否指定了报告标题。它包含有关日志文件及其源的信息。
第一条标头记录	第一条标头记录 （记录 ID -2）包含对应于日志文件中所含项的唯一项标识号列表。项 ID 号的位置可用于确定文件中每个导出项的位置和大小。
第二条标头记录	第二条标头记录 （记录 ID -3）包含对应于导出项的缩放比例的列表。有关更多详细信息，请参见本部分中稍后论述的 “缩放比例”。
应用程序名称记录	此记录 （记录 ID -4）只存在于使用多布局格式的应用程序数据文件中。它列出对应于文件其余部分中应用程序度量组的应用程序的名称。
事务名称记录	此记录 （记录 ID -5）只存在于使用多布局格式的事务跟踪数据文件中。它列出对应于文件其余部分中事务度量组的事务的名称。
磁盘设备名称记录	此记录 （记录 ID -7）只存在于使用多布局格式的磁盘设备数据文件中。它列出对应于文件其余部分中磁盘设备度量组的磁盘设备的名称。
逻辑卷名称记录	此记录 （记录 ID -8）只存在于使用多布局格式的逻辑卷数据文件中。它列出对应于文件其余部分中逻辑卷度量组的逻辑卷的名称。
Netif 名称记录	此记录 （记录 ID -9）只存在于使用多布局格式的 netif (LAN) 数据文件中。它列出对应于文件其余部分中 `netif` 设备度量组的 `netif` 设备的名称。
文件系统名称记录	此记录 （记录 ID -12）只存在于使用多布局格式的文件系统数据文件中。它列出对应于文件其余部分中文件系统度量组的文件系统的名称。
CPU 名称记录	此记录（记录 ID -13）只存在于使用多布局格式的 CPU 数据文件中。它列出对应于文件其余部分中 CPU 度量组的 CPU 的名称。

二进制标题记录

二进制文件的标题记录包含设计用来帮助对导出文件内容进行程序解译的信息。只要指定了 `headings on`，此记录就会写入导出文件。

二进制标题记录的内容包括：

Record Length	4 字节整数	标题记录的长度
Record ID	4 字节整数	-1
Date_Seconds	4 字节整数	创建导出文件的日期
Number_of_vars	4 字节整数	最大重复变量数
Size of Fixed Area	4 字节整数	记录的非变量部分的字节数
Size of Variable Entry	4 字节整数	每个变量条目的字节数
GMT Time Offset	4 字节整数	偏离格林威治标准时间的秒数
Daylight Savings Time	4 字节整数	=1 夏令时
System ID	40 字符	系统标识
Collector Version	16 字符	数据收集器的名称和版本
Log File Name	72 字符	源日志文件的名称
Report Title	100 字符	用户指定的报告标题

二进制标题记录中的 `Date_Seconds`、`GMT Time Offset` 和 `Daylight Savings Time` 度量在创建 `export` 文件时应用于系统和时间。如果这不是记录数据的同一系统，这些字段将不能正确反映文件中的数据。

二进制项标识记录

二进制文件中的第一条标头记录（记录 ID -2）包含每个导出项的唯一项号。每个项 ID 都是可以用此产品附带的 `rxitemid` 文件交叉引用的 4 字节整数。项 ID 字段与文件其余部分中它们代表的数据字段一致。所有二进制导出的数据项都在导出文件中占用 4 字节的倍数，并且每个都在 4 字节的边界开始。如果某数据项需要多于 4 字节的空间，则其对应的项 ID 字段将从左边以零填充。

例如，进程度量 Program 需要 16 字节。其数据和项 ID 记录将是：

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

二进制缩放比例记录

二进制文件中的第二条标头记录（记录 ID -3）包含每个导出项的缩放比例。数字项以 32 位（4 字节）整数导出到二进制文件，以便尽可能减少不同计算机体系结构以不同方式实现浮点所造成的问题。大多数项在截断以适合整数格式之前，要乘以固定的缩放比例。这样就可将浮点数表示为分数，用缩放比例作分母。

为匹配大多数数据项，每个缩放比例都是 32 位 （4 字节）整数。缩放比例的特殊值用于指示非数字和其他特殊值度量。

特殊缩放比例

非数字度量 （如 ASCII 字段）的缩放比例是零。不应出现 -1 缩放比例，但如果确实出现了，说明 extract 程序中存在内部错误，应该报告。

DATE 格式是字段最低有效 16 位 （最右边的 16 位）中的 MPE CALENDAR 格式。日期的缩放比例是 512。将它缩放为 32 位整数 （用 512 除）将把年份隔离出来作为日期的整数部分，把年份中的日期 （除以 512）隔离为分数部分。

TIME 是 4 字节二进制字段 （时、分、秒、几十分之一秒）。时间的缩放比例是 65536。用 65536 除它形成的数字的整数部分是 “（小时 * 256） + 分钟”。

在二进制文件中处理 Date_Seconds 值更容易。

应用程序名称记录

以多布局格式导出应用程序数据时，将写入特殊的应用程序名称记录以标识应用程序组。对于二进制格式文件，此记录的记录 ID 是 -4。它由二进制记录 16 字节标头和在导出数据的开始日期定义的每个应用程序的 20 字节应用程序名称组成。

如果在数据提取涵盖的时间范围内添加或删除应用程序，则以新应用程序名称重复应用程序名称记录。

事务名称记录

以多布局格式导出事务数据时，将写入特殊的事务名称记录以标识应用程序事务名称。对于二进制格式文件，此记录的记录 ID 是 -5。它由二进制记录 16 字节标头和在导出数据的开始日期配置的每个事务的 60 字节截断的应用程序事务名称组成。如果在数据提取涵盖的时间范围内添加事务，将以追加到原始列表末尾的新应用程序事务名称重复事务名称记录。在数据提取开始之后删除的事务不会从多布局数据记录中删除。

磁盘设备名称记录

以多布局格式导出磁盘设备数据时，将写入特殊的磁盘设备名称记录以标识磁盘设备名称。对于二进制格式文件，此记录的记录 ID 是 -7。它由二进制记录 16 字节标头和在导出数据的开始日期配置的每个磁盘设备的 20 字节磁盘设备名称组成。

如果在数据提取涵盖的时间范围内添加磁盘设备，将以追加到原始列表末尾的新磁盘设备名称重复磁盘设备名称记录。在数据提取开始之后删除的磁盘设备不会从多布局数据记录中删除。

逻辑卷名称记录

以多布局格式导出逻辑卷数据时，将写入特殊的逻辑卷名称记录以标识逻辑卷名称。对于二进制格式文件，此记录的记录 ID 是 -8。它由二进制记录 16 字节标头和在导出数据的开始日期配置的每个逻辑卷的 20 字节磁盘设备名称组成。

如果在数据提取涵盖的时间范围内添加逻辑卷，将以追加到原始列表末尾的新逻辑卷名称重复逻辑卷名称记录。在数据提取开始之后删除的逻辑卷不会从多布局数据记录中删除。

Netif 名称记录

以多布局格式导出 `netif` 数据时，将写入特殊的 Netif 名称记录以标识 netif 设备名称。对于二进制格式文件，此记录的记录 ID 是 -11。它由二进制记录 16 字节标头和在导出数据的开始日期配置的每个 `netif` 设备的 20 字节 `netif` 设备名称组成。

如果在数据提取涵盖的时间范围内添加 `netif` 设备，将以追加到原始列表末尾的新设备名称重复 Netif 名称记录。在数据提取开始之后删除的 netif 设备不会从多布局数据记录中删除。

7 extract 命令

本章介绍 extract 程序的命令。它包括命令语法表、用于提取和导出数据的命令表以及按字母顺序描述命令的命令参考部分。

可以按大小写字母的任意组合输入 extract 的命令和参数。*除了* weekdays 和 weekly 命令需要输入完整名称外，其他命令名称只需要输入前三个字符。例如，命令 application detail 可以缩写为 app det。

有关如何使用这些命令的示例，请参见 extract 程序的联机帮助。

下表汇总了 extract 命令及参数的语法。

extract 函数不能生成摘要数据。摘要数据只能由 export 函数生成。

表 8　　extract 命令：语法和参数

命令	参数
application	on detail summary （仅 export） both （仅 export） off （默认值）
class	detail （默认值） summary （仅 export） both （仅 export） off
cpu	detail summary （仅 export） both （仅 export） off （默认值）
configuration	on detail off （默认值）
disk	on detail summary （仅 export） both （仅 export） off （默认值）

表 8　　extract 命令：语法和参数 （续）

命令	参数
exit e	
export	day[*ddd*] [- *天数*] week [*ww*] [- *周数*] month[*mm*] [- *月数*] year [*yy*] [- *年数*]
extract	day[*ddd*] [- *天数*] week [*ww*] [- *周数*] month[*mm*] [- *月数*] year [*yy*] [- *年数*]
filesystem	detail summary （仅 export） both （仅 export） off （默认值）
global	on detail （默认值） summary （仅 export） both （仅 export） off
guide	
help	
list	filename *
logfile	logfile
lvolume	on detail summary （仅 export） both （仅 export） off （默认值）
menu	
monthl	
y	*yyymm* *mm*
netif	on detail summary （仅 export） both （仅 export） off （默认值）
output	*输出文件* ,new ,purgeboth ,append

表 8　　extract 命令：语法和参数 （续）

命令	参数
process	on detail [app#[-#],...] off （默认值） killed
quit q	
report	[*导出模板文件*], show
shift	*开始时间* - *停止时间* all day noweekends
sh !	shell 命令
show	all
start	*日期* [*时间*] today [- *天数*][*时间*] last [- *天数*][*时间*] first [+ *天数*][*时间*]
stop	*日期* [*时间*] today [- *天数*][*时间*] last [- *天数*][*时间*] first [+ *天数*][*时间*]
transaction	on detail summary （仅 export） both （仅 export） off （默认值）
weekdays	1.....7
weekly	*yyww* *ww*
yearly	*yyyy* *yy*

下表列出了用于提取和导出数据的命令及所用的日志文件类型（scope 日志文件或 DSI 日志文件）。

表 9　　extract 命令：用于提取和导出数据

命令	提取数据	导出数据	scope 日志文件	DSI 日志文件
application	x	x	x	
class	x	x	x	x
configuration		x	x	
cpu	x	x	x	
disk	x	x	x	
export		x	x	x
extract	x		x	
filesystem	x	x	x	
global	x	x	x	
logfile	x	x	x	x
lvolume	x	x	x	
monthly	x		x	
netif		x	x	
output	x	x	x	x
process	x	x	x	
report		x	x	x
shift	x		x	x
start	x	x	x	x
stop	x	x	x	x
transaction	x	x	x	x
weekdays		x	x	x
weekly	x		x	
yearly	x		x	
logicalsystems	x	x	x	

application

使用 application 命令指定要提取或导出的应用程序数据的类型。

默认值是 application off

语法

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

参数

on 或 detail	指定应提取或导出原始的 5 分钟详细信息数据。
summary （仅 export）	指定数据的汇总方式： • 用指定的导出模板文件中的 summary 参数指定的分钟数（仅 export） • 一小时的默认摘要间隔 （export 或 extract） 汇总可以大大减少所生成的提取数据或导出数据的大小，具体取决于使用的汇总间隔。例如，每小时摘要数据大概是 5 分钟详细信息数据大小的十分之一。
both （仅 export）	指定提取或导出详细信息数据和摘要数据。
off	指定不提取或导出此类数据。

如果在使用 Performance Manager，必须在提取的文件中包括详细信息数据，才能用每 5 分钟的点绘制应用程序图形。

示例

在此示例中，application 命令导致导出详细的应用程序日志文件数据：output export 文件包含 myrept 导出模板文件中指定的应用程序度量。

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

要使用命令行参数执行上面的任务，请输入：

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

class

使用 class 命令指定要导出的 DSI 数据类，也可以指定要提取或导出的 scope 数据类。

默认值是 class detail。

语法

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

参数

classname	按相似性分类的度量组的名称。
detail	对于 DSI 日志文件，指定按 DSI 日志文件中设置的时间导出对应详细程度的数据。有关详细信息，请参见第 255 页的数据源集成概述。 对于 scope 日志文件，指定应提取或导出原始的 5 分钟详细信息。
summary bothoff	请参见第 115 页的application 命令描述中的“参数”。只能导出 summary 和 both。

示例

要导出 DSI 日志文件中包含 acctg_info 类的摘要数据，请发出以下命令：

```
class acctg_info summary
```

用户指定日志文件并由 extract 程序打开后，即验证该日志文件中是否存在 acctg_info 类，之后再导出该类数据。

此命令的其他变式有：

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

命令可以大写，也可以小写。类名总是先改为大写再比较。

在以下示例中，导出 fin_info 类的摘要数据。

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

要使用命令行参数执行上面的任务，请输入：

```
extract -l dsi.log -C fin_info summary -xp
```

configuration

使用 configuration 命令指定是否导出系统配置信息。

默认值是 configuration off。

语法

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

参数

on 或 detail	指定应导出所有配置记录。
off	指定不导出配置数据。

导出日志文件中可用的所有配置信息。忽略与 configuration 命令结合使用的任何 begin、end、shift、 start、 stop 或 noweekends 命令。

configuration 命令只影响 export 函数。不影响 extract 函数，因为 extract 总是提取系统配置信息。

示例

在此示例中，configuration 命令导致导出系统配置信息。输出导出文件包含 myrept 导出模板文件中指定的配置度量。

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

要使用命令行参数执行上面的任务，请输入：

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

cpu

使用 cpu 命令指定 CPU 的汇总级别。

默认值是 cpu off。

语法

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

参数

detail	提取或导出 5 分钟的详细信息记录。
summary	导出摘要记录。
both	导出详细信息记录和摘要记录。
off	不提取或导出 CPU 数据。

示例

在此示例中， cpu 命令导致导出从 2001 年 7 月 26 日开始收集的 CPU 详细信息数据。因为没有指定导出模板文件，将使用默认导出模板文件 reptfile。将在输出文件中包括所有 cpu 度量 （如 reptfile 所指定）。

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

要使用命令行参数执行上面的任务，请输入：

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

disk

使用 disk 命令指定要提取或导出的磁盘设备数据的类型。

默认值是 disk off。

语法

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

参数

on 或 detail summary bothoff	请参见本章开头 application 命令描述中的 “参数”。只能导出 summary 和 both。

示例

在此示例中，disk 命令导致导出从 1999 年 7 月 5 日开始收集的 disk 详细信息数据。因为没有指定导出模板文件，将使用默认导出模板文件 reptfile。将在输出文件中包括所有磁盘度量 （如 reptfile 所指定）。

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

要使用命令行参数执行上面的任务，请输入：

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

exit

使用 exit 命令终止 extract 程序。 exit 命令与 extract 程序的 quit 命令等效。

语法

```
exit
e
```

export

使用 export 命令启动将数据复制到导出的文件格式的进程。

语法

export **[day** *[ddd]* **[***yyddd***]** **[-** *天数* **]]**
[week *[ww]* **[***yyww***]** **[-** *周数* **]]**
[month *[mm]* **[***yymm***]** **[-** *月数* **]]**
[year *[yy]* **[***yyyy***]** **[-** *年数* **]]**

参数

使用以下某个参数导出特定间隔的数据。

day	表示一天
week	表示一周，从星期一到星期日
month	表示一个月，从第一个日历日到最后一个日历日
year	表示一年，从第一个日历日到最后一个日历日

如果不在 export 命令中使用任何参数，导出数据所用的间隔将由 start 和 stop 命令设置。

如何使用

可采用四种方式指定特定间隔 （day、 week、 month 和 year）。

- 当前间隔 - 仅指定参数。例如， month 指当月。
- 前一个间隔 - 指定参数、减号和所需当日、当周、当月或当年之前的间隔数。例如， day-1 是昨天， week-2 是当周之前的两周。

- 绝对间隔 - 指定参数和正数。数字表示当年中所需的绝对间隔。例如， day 2 是当年的 1 月 2 日。
- 绝对间隔加上年份 - 指定参数和大的正数。数字应由年份的最后两位数和该年中的绝对间隔数字组成。在此格式中，绝对 day 参数有 5 位数 （99002 指的是 1999 年 1 月 2 日），所有其他参数有 4 位数 （month 9904 指的是 1999 年 4 月）。

如果之前未指定日志文件或导出模板文件， logfile 命令将使用默认全局日志文件 logglob，report 命令将使用默认导出模式文件 reptfile。

使用所有其他参数的设置或默认值。有关其对应操作的详细信息，请参见 application、configuration、global、process、disk、lvolume、netif、cpu、filesystem、transaction、output、shift、start 和 stop 命令的描述。

export 命令根据指定的数据类型和汇总级别创建最多 16 个不同的默认输出文件。

xfrdGLOBAL.ext	全局详细信息数据文件
xfrsGLOBAL.ext	全局每小时摘要数据文件
xfrdAPPLICATION.ext	应用程序详细信息数据文件
xfrsAPPLICATION.ext	应用程序每小时摘要数据文件
xfrdPROCESS.ext	进程详细信息数据文件
xfrdDISK.ext	磁盘设备详细信息数据文件
xfrsDISK.ext	磁盘设备摘要数据文件
xfrdVOLUME.ext	逻辑卷详细信息数据文件
xfrsVOLUME.ext	逻辑卷摘要数据文件
xfrdNETIF.ext	Netif 详细信息数据文件
xfrsNETIF.ext	Netif 摘要数据文件
xfrdCPU.ext	CPU 详细信息数据文件
xfrsCPU.ext	CPU 摘要数据文件
xfrdFILESYSTEM.ext	文件系统详细信息数据文件
xfrsFILESYSTEM.ext	文件系统摘要数据文件
xfrdTRANSACTION.ext	事务详细信息数据文件
xfrsTRANSACTION.ext	事务摘要数据文件
xfrdCONFIGURATION.ext	配置详细信息数据文件

其中 ext = asc、 dat、 bin 或 wk1

默认文件名根据数据类型名称创建。前缀是 xfrd 或 xfrs 则取决于数据是详细数据还是摘要数据。扩展名是指定的 asc（ASCII）、bin（二进制）、dat（数据文件）或 wk1（电子表格）数据格式。

例如， classname = ACCTG_INFO 的 export 文件名有：

xfrdACCTG_INFO.wk1 ACCT_INFO 的详细电子表格数据

xfrsACCTG_INFO.asc ACCT_INFO 的摘要 ASCII 数据

有关导出数据的详细信息，请参见第 5 章第 97 页的 export 函数的概述。

示例

在此示例中， export 命令导致导出昨天 8:00 am 到 5 pm 收集的日志文件数据。因为没有指定导出模板文件，将使用默认导出模板文件 reptfile。将在输出文件中包括所有全局度量（如 reptfile 所指定）。

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

要使用命令行参数执行上面的任务，请输入：

```
extract -gG -b today-1 8:00 am -e today-1 5:00 pm -xp
```

extract

使用 extract 命令启动将数据从原始日志文件复制到提取的文件格式的进程。提取的文件可以用来存档，或供 Performance Manager 等分析程序分析。可以从原始日志文件和提取的文件中提取数据。

extract 命令无法处理 DSI 日志文件的数据。

语法

extract **[day** *[ddd]* **[***yyddd***]** **[-** *天数* **]]**
[week *[ww]* **[***yyww***]** **[-** *周数* **]]**
[month *[mm]* **[***yymm***]** **[-** *月数* **]]**
[year *[yy]* **[***yyyy***]** **[-** *年数* **]]**

参数

使用以下某个参数提取特定间隔的数据：

day	表示一天
week	表示一周，从星期一到星期日
month	表示一个月，从第一个日历日到最后一个日历日
year	表示一年，从第一个日历日到最后一个日历日

如果不在 extract 命令中使用任何参数，提取数据所用的间隔将由 start 和 stop 命令设置。

如何使用

可采用四种方式指定特定间隔 （day、 week、 month 和 year）。

- 当前间隔 - 仅指定参数。例如， month 指当月。

- 前一个间隔 - 指定参数、减号和所需当日、当周、当月或当年之前的间隔数。例如， day-1 是昨天， week-2 是当周之前的两周。
- 绝对间隔 - 指定参数和正数。数字表示当年中所需的绝对间隔。例如， day 2 是当年的 1 月 2 日。
- 绝对间隔加上年份 - 指定参数和大的正数。数字应由年份的最后两位数和该年中的绝对间隔数字组成。在此格式中，绝对 day 参数有 5 位数 （99002 指的是 1999 年 1 月 2 日），所有其他参数有 4 位数 （month 99904 指的是 1999 年 4 月）。

extract 命令启动数据提取。如果之前未指定，在执行 extract 命令时 logfile 和 output 命令采用以下默认设置：

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

使用所有其他参数的设置或默认值。有关其对应操作的详细信息，请参见 application、global、process、disk、lvolume、netif、cpu、filesystem、transaction、shift、start 和 stop 命令的描述。

提取的日志文件的大小不能超过 3.5 GB。

示例

在第一个示例中，提取 2000 年 3 月 1 日到 2000 年 6 月 30 日期间每个工作日 8:00 am 到 5:00 pm 收集的数据。只提取全局和应用程序详细信息数据。

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

要使用命令行参数执行上面的任务，请输入：

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

在第二个示例中，新建名为 rxjan00 的提取日志文件。清除具有该名称的所有现有文件。提取 2000 年 1 月 1 日到 2000 年 1 月 31 日期间收集的所有原始日志文件数据：

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

要使用命令行参数执行上面的任务，请输入：

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

filesystem

使用此命令指定要提取或导出的文件系统数据的汇总级别。

默认值是 filesystem off。

语法

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

参数

detail	提取或导出 5 分钟的详细信息记录。
summary	导出摘要记录。
both	导出详细信息记录和摘要记录。
off	不提取或导出文件系统数据。

示例

在此示例中，filesystem 命令导致导出从 2001 年 7 月 26 日开始收集的文件系统详细信息数据。因为没有指定导出模板文件，将使用默认导出模板文件 reptfile。将在输出文件中包括所有文件系统度量（如 reptfile 所指定）。

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

要使用命令行参数执行上面的任务，请输入：

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

global

使用 global 命令指定要提取或导出的全局数据量。

默认值是 global detail。（在命令行模式下，默认值是 global off。）

语法

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

参数

`detail` 或 `on`	请参见本章开头 application 命令描述中的 “参数”。只能导出 `summary` 和 `both`。
`summary`	
`both`	
`off`	

如何使用

如果要用每 5 分钟的点绘制 Performance Manager 全局数据图，必须提取详细信息数据。

Performance Manager 绘制汇总数据的速度较快，因为生成图形所需的数据记录较少。如果只提取全局摘要数据，将无法用每 5 分钟的数据点绘制 Performance Manager 全局数据图。

`both` 选项允许您用每 5 分钟的点绘制 Performance Manager 全局数据图，同时还通过每小时摘要记录保持了访问速度。

如果您在使用 Performance Manager，建议不要指定 `off` 参数，因为必须有全局数据才能正确了解总体系统行为。除非提取的文件包含至少一种全局数据，否则无法绘制 Performance Manager 全局数据图。

示例

此处的 `global` 命令用于指定*不*导出全局数据 （默认值是 `global detail`） 。只导出详细事务数据。输出导出文件包含 `myrept` 导出模板文件中指定的事务度量。

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

要使用命令行参数执行上面的任务，请输入：

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

guide

使用 `guide` 命令进入引导命令模式。引导命令界面指引您完成各种 `extract` 命令，并提示您执行一些可用的最常见任务。

语法

```
guide
```

如何使用

- 要从 extract 程序的交互模式进入引导命令模式，请输入 **guide**。
- 要接受参数默认值，请按 **Return**。
- 要终止引导命令模式并返回交互模式，请在 guide> 提示符处输入 **q**。

此命令不提供所有可能的参数设置组合。它选择应为大多数用户生成有用结果的设置。可以通过 extract 交互命令模式完全控制 extract 函数。

如果要导出 DSI 日志文件数据，建议使用引导命令模式创建自定义导出模板文件并导出数据。

help

使用 help 命令访问联机帮助。

语法

help [*关键字* **]**

如何使用

可以输入参数获取有关 extract 命令和任务或 help 本身的信息。通过输入关键字导航到不同主题。如果有多页信息，显示会暂停，等待您按 **Return** 再继续。输入 **q** 或 **quit** 可以退出帮助系统并返回到 extract 程序。

还可以请求特定主题的帮助信息。例如，

help tasks

或

help resize parms

使用这种格式的 help 命令时，您会收到指定主题的帮助文本，并且依然留在 extract 命令输入上下文中。因为不是以交互模式进入帮助子系统，输入下一个 extract 命令之前不必输入 **quit**。

list

使用 list 命令指定所有 extract 程序报告的列表文件。

语法

list [*文件* **]**
[*]

如何使用

使用 extract 时随时都可以使用 list 指定列表设备。它使用文件名或列表设备名称输出用户指定的设置。如果列表文件已存在，则在现有文件中追加输出。

发送给列表设备的数据也显示在您的屏幕上。

extract 正在运行时，输入：

list outfilename

要将列表设备返回到用户终端，请输入：

list stdout

或

list *

要确定当前列表设备，请输入不带参数的 list 命令，如下所示：

list

如果列表文件不是 stdout，大多数命令将回显到输入它们时的列表文件。

示例

在以下示例中，列表设备设置为 mylist。后续命令的结果打印到 mylist，并显示在您的屏幕上。

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

logfile

使用 logfile 命令打开日志文件。所有 extract 程序函数都必须打开日志文件。可以通过发出 logfile 命令显式打开日志文件，也可以通过发出 extract 命令或 export 命令隐式打开日志文件。如果不指定日志文件名称， extract 程序会提示您输入日志文件名称，而且还会显示默认全局日志文件 /var/opt/perf/datafiles/logglob。可以接受默认文件，也可以指定其他日志文件。

语法

logfile [*日志文件* **]**

如何使用

要打开日志文件，可以指定原始或提取的日志文件名称。不能指定由 export 命令创建的文件的名称。如果指定提取的日志文件名称，将从该单个文件获取所有信息。如果指定原始日志文件名称，必须指定全局日志文件名称才能访问原始日志文件。这是唯一应指定的原始日志文件名称。

如果日志文件不在当前工作目录中，则必须提供其路径。

全局日志文件和其他原始日志文件必须在相同目录中。它们可以为以下名称：

```
logglob       全局日志文件
logappl       应用程序日志文件
logproc       进程日志文件
logdev        设备日志文件
logtran       事务日志文件
logindx       索引日志文件
```

打开日志文件时，将显示日志文件的常规内容。

不要重命名原始日志文件！访问这些文件时，程序假定标准日志文件名称有效。如果必须重命名日志文件以将来自多个系统的日志文件放在相同系统上进行分析，应先提取数据再重命名提取的日志文件。

示例

这是典型的默认全局日志文件列表。

```
Global      file: /var/opt/perf/datafiles/logglob, version D
Application file: /var/opt/perf/datafiles/logappl
Process     file: /var/opt/perf/datafiles/logproc
Device      file: /var/opt/perf/datafiles/logdev
Transaction file: /var/opt/perf/datafiles/logdev
Index       file: /var/opt/perf/datafiles/logindx
System ID:  homer
System Type 9000/715/ S/N 2223334442 O/S HP-UX B.10.20 A
Data collector: SCOPE/UX C.02.30
File Created:  10/08/99
Data Covers:   44 days to 11/20/99
Shift is:      All Day
Data records available are:
Global Application Process Disk Volume Transaction
Maximum file sizes:
Global=10.0 Application=10.0 Process=20.0 Device=10.0 Transaction=10.0 MB
The first GLOBAL      record is on 10/08/99 at 08:17 AM
The first NETIF       record is on 10/08/99 at 08:17 AM
The first APPLICATION record is on 11/17/99 at 12:15 PM
The first PROCESS     record is on 10/08/99 at 08:17 AM
The first DEVICE      record is on 10/31/99 at 10:45 AM
The Transaction data file is empty
The default starting date & time = 10/08/99 11:50 AM   (LAST -30)
The default stopping date & time  = 11/20/99 11:59 PM   (LAST -0)
```

lvolume

使用 `lvolume` 命令指定要提取或导出的逻辑卷数据的类型。（此命令仅用于 HP-UX 系统。）

默认值是 `lvolume off`。

语法

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

参数

on 或 detail summary both off	请参见本章开头 application 命令描述中的 “参数”。只能导出 summary 和 both。

示例

在此示例中，新建名为 rx899 的提取日志文件，清除具有该名称的所有现有文件。提取 8 月 1 日到 8 月 31 日期间收集的所有逻辑卷数据。

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

要使用命令行参数执行上面的任务，请输入：

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

menu

使用 menu 命令打印可用 extract 命令的列表。

语法

```
menu
```

示例

```
Command     Parameters    Function

HELP        [topic]       Get information on commands and options

GUIDE       Enter guided commands mode for novice users

LOGFILE     [logname]     Specify a log file to be processed

LIST        [filename|*] Specify the listing file

OUTPUT      [filename]
            [,NEW/PURGE/APPEND] Specify a destination file

REPORT      [filename][,SHOW]   Specify an Export Format file for EXPORT"

GLOBAL      [DETAIL/SUMMARY/BOTH/OFF] Extract GLOBAL records

APPLICATION [DETAIL/SUMMARY/BOTH/OFF] Extract APPLICATION records

PROCESS     [DETAIL/OFF/KILLED][APP=] Extract PROCESS records
```

```
DISK         [DETAIL/SUMMARY/BOTH/OFF] Extract DISK DEVICE records
LVOLUME      [DETAIL/SUMMARY/BOTH/OFF] Extract Logical VOLUME records
NETIF        [DETAIL/SUMMARY/BOTH/OFF] Extract Logical NETIF records
CPU          [DETAIL/SUMMARY/BOTH/OFF] Extract CPU records
FILESYSTEM   [DETAIL/SUMMARY/BOTH/OFF] Extract FILESYSTEM records
CONFIG       [DETAIL/OFF]    Export CONFIGURATION records
CLASS       classname[DETAIL/SUMMARY/BOTH/OFF]  Export classname records
TRANSACTION [DETAIL/SUMMARY/BOTH/OFF] Extract TRANSACTION records
START        [startdate time]  Specify a starting date and time for SCAN
STOP         [stopdate  time]  Specify an ending  date and time for SCAN
SHIFT        [starttime - stoptime] [NOWEEKENDS] Specify daily shift times
SHOW         [ALL]  Show the current program settings
EXPORT       [d/w/m/y][-offset] Copy log file records to HOST format files
EXTRACT      [d/w/m/y][-offset] Copy selected records to output (or append)
file
WEEKLY       [ww/yyww]  Extract one calendar  week's data with auto file names
MONTHLY      [mm/yymm]  Extract one calendar month's data with auto file names
YEARLY       [yy/yyyy]  Extract one calendar  year's data with auto file names
WEEKDAYS     [1...7]  Set days to exclude from export 1=Sunday ! or SH
             [command] Execute a system command
MENU or ? List the command menu (this listing)
EXIT or Q Terminate the program
```

monthly

使用 monthly 命令指定按日历月进行数据提取。在执行期间，此命令根据提取数据的月份和年份将开始和停止日期设置为正确的日期。

输出文件的名称由字母 rxmo 后跟提取的四位数年份和两位数月份构成。例如，1999 年 3 月提取的数据输出到名为 rxmo199903 的文件。

语法

monthly **[***yymm***]**
[*mm***]**

参数

monthly	提取当月（默认）的数据。
monthly *mm*	从当年的数据中提取特定月份的数据（其中 *mm* 是 01 到 12 之间的数字）。
monthly *yymm*	提取特定年和月的数据（其中 *yymm* 是由年份的最后两位数和两位数月份构成的单个数字）。例如，要提取 1999 年 2 月的数据，请指定 **monthly 9902**。

如果在执行 monthly 命令之前未指定日志文件，将使用默认 logglob 文件。

如何使用

如果要按月提取数据进行存档，请使用 monthly 命令。

提取和汇总的数据类型遵循 extract 命令的正常规则，可以在执行 monthly 命令之前设置。除非月度输出文件已存在，否则使用这些设置。如果文件已存在，将根据原始指定的数据类型将数据追加到该文件末尾。

monthly 命令具有一个功能，它打开上个月的提取文件并检查其是否填满（是否包含截止该月最后一天提取的数据）。如果不是， monthly 命令将数据追加到此文件以完成上个月的提取。

例如， 1999 年 5 月 7 日执行了 monthly 命令，创建了名为 rxmo199905 的日志文件，该文件包含 5 月 1 日到当天（5 月 7 日）的数据。

1999 年 6 月 4 日再执行一次 monthly 命令。此命令在为当月创建 rxmo199906 文件之前，先打开并检查上个月的 rxmo199905 文件。当发现文件未完成时，将数据追加到该文件以完成到 1999 年 5 月 31 日为止的数据提取。然后，创建 rxmo199906 文件保存 1999 年 6 月 1 日到当天（6 月 4 日）的数据。

只要每月至少执行 monthly 命令一次，此功能就会先完成每个月的文件，再创建下个月的文件。当您看到两个相邻的月度文件时，例如 rxmo199905（5 月）和 rxmo199906（6 月），可以认为第一个文件是 5 月的完整文件，可以存档和清除。

monthly 和 extract month 命令有一点非常相似，它们都提取一个日历月的数据。monthly 命令忽略 output 命令的设置，而使用预定义的输出文件名。如果系统上仍存在上个月的提取日志文件，它还会尝试将缺少的数据追加到该文件中。extract month 命令则使用 output 命令的设置。它不会将数据追加到上个月的提取文件中，因为它不知道提取文件的名称。

示例

在此示例中，提取 1999 年 5 月记录的应用程序详细信息数据。

```
logfile /var/opt/perf/datafiles/logglob
global off
application detail
monthly 9905
```

要使用命令行参数执行上面的任务，请输入：

```
extract -a -xm 9905
```

netif

使用 netif 命令指定要提取或导出的逻辑网络接口 (LAN) 数据的类型。Netif 数据记录到 logdev 文件中。

默认值是 netif off。

语法

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

参数

on 或 detail summary both off	请参见本章开头 application 命令描述中的“参数”。只能导出 summary 和 both。

示例

在此示例中，提取 2000 年 3 月 1 日到 2000 年 6 月 30 日期间每个工作日 8:00 am 到 5:00 pm 收集的 netif 详细信息数据。

```
logfile /var/opt/perf/datafiles/logglob
start 03/01/00
stop 06/30/00
shift 8:00 AM - 5:00 PM noweekends
netif detail
extract
```

要使用命令行参数执行上面的任务，请输入：

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

output

使用 output 命令指定 extract 或 export 函数的输出文件的名称。

可选的第二个参数指定存在同名输出文件时要采取的操作。

语法

```
                        [,new]
output    [ 文件名 ]    [,purge]
                        [,append]
```

参数

,new　　指定输出文件必须是新文件。这是批处理模式下的默认操作。如果存在同名文件，批处理作业将终止。

,purge　　指定应清除所有现有文件以为新的输出文件留出空间。

,append　　指定应在现有提取文件中追加数据。如果不存在指定输出文件名的文件，则创建新文件。

如何使用

如果在批处理模式下未指定操作，将使用默认操作 ,new。在交互模式下，如果发现重复文件，将提示您输入操作。

如果未指定输出文件，将创建默认输出文件。默认输出文件名有：

```
extract: rxlog

export:

   xfrdGLOBAL.ext
   xfrsGLOBAL.ext
   xfrdAPPLICATION.ext
   xfrsAPPLICATION.ext
   xfrdPROCESS.ext
   xfrdDISK.ext
   xfrsDISK.ext
   xfrdLVOLUME.ext
   xfrsLVOLUME.ext
   xfrdNETIF.ext
   xfrsNETIF.ext
   xfrdCPU.ext
   xfrsCPU.ext
   xfrdFILESYSTEM.ext
   xfrsFILESYSTEM.ext
   xfrdTRANSACTION.ext
   xfrsTRANSACTION.ext
   xfrdCONFIGURATION.ext
```

其中 ext=asc （ASCII）、 dat （数据文件）、 bin （二进制）或 wk1 （电子表格）。

在导出操作中可以使用特殊文件名 stdout （或 *），以将输出定向到 stdout 文件 （通常是您的终端或工作站，虽然可以使用 shell 命令重定向）。

```
output stdout
```

或

```
output *
```

要将输出返回到其默认设置，请输入：

```
output default
```

或

```
output -
```

可以在导出模板文件中使用 output 参数覆盖导出文件的默认输出文件名。

不应将提取操作的文件输出到 stdout，因为它们与 ASCII 设备不兼容。也不应将二进制或 WK1 格式的导出操作文件输出到 stdout 文件，原因同上。

注意不要将*提取的*数据追加到现有*导出的*数据文件，不要将导出的数据追加到现有提取的文件。尝试追加错误的数据类型会导致出错。

示例

在此示例中未指定输出文件，因此对提取的应用程序摘要数据使用默认输出文件 rxlog。,purge 选项指定应清除任何现有输出文件。

```
extract>
logfile /var/opt/perf/datafiles/logglob
output rxlog,purge
global off
application detail
extract month 9905
```

要使用命令行参数执行上面的任务，请输入：

```
extract -f rxlog,purge -a -xm 9905
```

process

使用 process 命令指定是否提取或导出进程数据。

默认值是 process off。

语法

```
            [on]
process     [detail]        [application=#[-#] ,...]
            [off]
            [killed]
```

参数

on	指定*应*提取或导出进程数据。
detail	指定 process detail 等同于指定 process on。

off	指定*不应*提取或导出进程数据。
killed	仅指定感兴趣原因包括 killed 的进程。（即测量间隔中终止的进程。）
application	仅指定属于所选应用程序的进程。应用程序可以单个编号或应用程序编号范围来输入（7-9 是指应用程序 7、8 和 9）。应用程序编号由收集数据时 parm 文件中的应用程序定义顺序确定。如果要指定多个应用程序，中间请用逗号分隔。

进程数据可以大幅增加提取的日志文件的大小。如果计划将日志文件复制到工作站进行分析，可能要限制提取的进程数据量。

示例

在此示例中，process 命令指定间隔期间终止且属于应用程序 1、4、6、7、8 或 10 的进程。使用 utility 程序的 scan 命令可以找到特定应用程序的应用程序编号。

```
process killed applications=1,4,6-8,10
```

quit

使用 quit 命令终止 extract 程序。 quit 命令与 extract 程序的 exit 命令等效。

语法

```
quit
q
```

report

使用 report 命令指定 export 函数要使用的导出模板文件。如果不指定导出模板文件，将使用默认导出模板文件 reptfile。导出模板文件用于指定 export 函数中使用的各种输出格式属性。它还指定要导出哪些度量。

如果在交互模式下，且未指定导出模板文件，将以 ASCII 格式导出请求的数据类型的所有度量。

语法

report [*导出模板文件* **] [,show]**

参数

,show　指定应为导出模板文件中指定的所有度量列出字段位置和起始列。当其他程序处理导出文件时可以使用此列表。

如何使用

发出此命令时，会显示一条消息，询问您是否要用之前指定的日志文件验证导出模板中的度量。此验证可确保日志文件中存在导出模板文件中指定的度量。这样您就可以检查导出模板文件中是否有错误。如果不执行验证，此操作会推迟到执行导出时进行。

此处讨论的 report 命令的 ,show 参数与后面要讨论的 show 命令不同。

sh

使用 sh 可以不用退出 extract 而输入 shell 命令，方法是输入 **sh** 或感叹号（**!**）后跟 UNIX shell 命令。

语法

sh 或 **! [shell 命令]**

参数

sh ls　执行 ls 命令并返回到 extract。 shell 命令是指任何系统命令。

!ls　同上。

!ksh　启动 Korn shell。不立即返回到 extract。输入 **exit** 或 **CTRL-d Return** 可以返回到 extract 程序。

如何使用

在执行单个命令后，将自动返回到 extract。如果要发出多条 shell 命令，在每条命令之后不返回到 extract，则可以启动新 shell。

如果发出不带 shell 命令名称的 sh 命令，会提示您提供名称。例如，

```
sh

enter SYSTEM command: ls
```

shift

使用 shift 命令将数据提取限制为符合工作轮换、不包括周末 （星期六和星期日）的特定时段。

默认值是 shift all day，提取包括周末在内的每一天的全天数据。

语法

shift **[** *开始时间 - 停止时间* **]**
[all day]
[noweekends]

参数

开始时间和停止时间参数的输入格式与 start 命令中的时间格式相同。允许跨午夜的轮换。如果将开始时间安排在停止时间*之后*，轮换将始于开始时间，跨过午夜，止于第二天的*停止时间*。

all day	指定默认轮换 12:00 am 到 12:00 am （或 24 小时制的 00:00 到 00:00）。
noweekends	指定不包括星期六和星期日记录的数据。如果在跨午夜的轮换中输入 noweekends，周末将包括从星期六或星期日*开始*的轮换。

示例

在此示例中，提取 1999 年 6 月 15 日开始的每天 10:00 am 到 4:00 pm 收集的磁盘详细信息数据。

```
extract>
logfile /var/opt/perf/datafiles/logglob
global off
disk detail
shift 10:00 am - 4:00 PM
start 6/15/99
extract
```

要使用命令行参数执行上面的任务，请输入：

```
extract -d -b 6/15/99 -s 10:00 AM-4:00 PM -xt
```

show

使用 show 命令列出打开的文件的名称和可设置的 extract 参数的状态。

语法

```
show [all]
```

此处讨论的 show 命令与前面讨论的 report 命令的 ,show 参数不同。

示例

单独使用 show 生成的列表可能如下所示：

```
Logfile:  /var/opt/perf/datafiles/logglob
Output:   Default
Report:   Default
List:     "stdout"
```

```
The default starting date & time = 10/08/99 12:00 AM (LAST -30)
The default stopping date & time = 11/20/99 11:59 PM (LAST -0)
The default shift = 12:00 AM - 12:00 PM
GLOBAL          DETAIL          records will be processed
APPLICATION. . . . . . . . . . NO records will be processed
PROCESS . . . . . . . . . . . .  NO records will be processed
DISK DEVICE. . . . . . . . . . NO records will be processed
LVOLUME. . . . . . . . . . . . NO records will be processed
TRANSACTION. . . . . . . . . . NO records will be processed
NETIF . . . . . . . . . . . . .NO records will be processed
CPU . . . . . . . . . . . . . .NO records will be processed
FILESYSTEM. . . . . . . . . . .NO records will be processed
Configuration . . . . . . . . .NO records will be processed
Use show all to produce a more detailed list that may look like this:
Logfile:          /var/opt/perf/datafiles/logglob
Global      file: /var/opt/perf/datafiles/logglob,version D
Application file: /var/opt/perf/datafiles/logappl
Process     file: /var/opt/perf/datafiles/logproc
Device      file: /var/opt/perf/datafiles/logdev
Transaction file: /var/opt/perf/datafiles/logdev
Index       file: /var/opt/perf/datafiles/logindx
System ID:  homer
System Type 9000/715/ S/N 2223334442 O/S HP-UX B.10.20 A
Data collector: SCOPE/UX C.02.30
File Created:   10/08/99
Data Covers:    44 days to 11/20/99
Shift is:       All Day
Data records available are:
  Global Application Process Disk Volume Transaction
Maximum file sizes:
  Global=10.0 Application=10.0 Process=20.0 Device=10.0
  Transaction=10.0 MB
Output:  Default
Report:  Default
List:    "stdout"
The default starting date & time = 10/08/99 11:50 AM (LAST -30)
The default stopping date & time = 11/20/99 11:59 PM(LAST - 0)
The default shift = 12:00 AM - 12:00 PM
GLOBAL...........DETAIL...........records will be processed
APPLICATION.....................NO records will be processed
PROCESS.........................NO records will be processed
DISK DEVICE.....................NO records will be processed
LVOLUME.........................NO records will be processed
TRANSACTION.....................NO records will be processed
NETIF...........................NO records will be exported
CPU.............................NO records will be processed
FILESYSTEM......................NO records will be processed
Configuration ..................NO records will be exported
Export Report Specifications:
   Interval = 3600, Separator = " "
   Missing data will not be displayed
   Headings will be displayed
   Date/time will be formatted
   Days to exclude: None
```

start

使用 start 命令设置 extract 和 export 函数的开始日期和时间。默认开始日期是日志文件中最晚日期之前满 30 天的日期，如果不到 30 天，*则为*日志文件中最早记录的日期。

语法

start **[日期 [*时间*]]**
[today [- *天数*] [*时间*]]
[last [- *天数*] [*时间*]]
[first [+ *天数*] [*时间*]]

参数

日期　日期格式取决于系统上配置的本机语言。如果不使用本机语言或将默认语言设置为 C，对于 extract 或 export 函数，日期格式则为 *mm*/*dd*/*yy*（月 / 日 / 年），比如 09/30/99 表示 1999 年 9 月 30 日。

时间　时间格式也取决于使用的本机语言。C 语言的时间格式是 hh:mm am 或 hh:mm pm（12 小时格式的 “小时 : 分钟” 加 am 或 pm 后缀）。例如， 07:00 am 表示上午 7 点钟。

所有语言都接受 24 小时制时间。例如， 23:30 会认为是 11:30 pm。

如果日期或时间格式不可接受，将会显示正确格式的示例。

如果不指定开始时间，则假定是午夜 (12:00 am)。一天的午夜开始时间始于当天的*开始*（24 小时制的 00:00）。

today　指定当天。参数的限定 （比如 today- 天数）指定当天日期*之前*的天数。例如，today-1 表示昨天的日期， today-2 表示前天的日期。

last　可用于表示日志文件中包含的最晚日期。参数 last-天数指定日志文件中最晚日期*之前*的天数。

first　可用于表示日志文件中包含的最早日期。参数 first+ 天数指定日志文件中最早日期*之后*的天数。

如何使用

以下命令覆盖 start 命令设置的开始日期。

- weekly
- monthly
- yearly
- extract （如果使用 day、 week、 month 或 year 参数）
- export （如果使用 day、 week、 month 或 year 参数）

示例

在此示例中， start 命令指定要提取的第一个间隔的开始时间是 1999 年 6 月 5 日 8:00 am。output 命令指定输出文件名为 myout。

```
logfile /var/opt/perf/datafiles/logglob
```

```
start 6/5/99 8:00 am
output myout
global detail
extract
```

要使用命令行参数执行上面的任务，请输入：

```
extract -g -b 06/05/99 8:00 AM -f myout -xt
```

stop

使用 stop 命令在指定日期和时间终止 extract 或 export 函数。

默认停止日期和时间是日志文件中记录的*最晚*日期和时间。

语法

stop

[日期 [*时间*]]
[today [- *天数*][*时间*]]
[last [- *天数*][*时间*]]
[first [+ *天数*][*时间*]]

参数

日期　日期格式取决于系统上配置的本机语言。如果不使用本机语言或将默认语言设置为 C，对于 extract 或 export 函数，日期格式则为 *mm*/*dd*/*yy*（月 / 日 / 年），比如 09/30/99 表示 1999 年 9 月 30 日。

时间　时间格式也取决于使用的本机语言。C 语言的时间格式是 *hh:mm* am 或 *hh:mm* pm（12 小时格式的 “小时 : 分钟” 加 am 或 pm 后缀）。例如， 07:00 am 表示上午 7 点钟。

所有语言都接受 24 小时制时间。例如， 23:30 会认为是 11:30 pm。

如果日期或时间格式不可接受，将会显示正确格式的示例。

如果不指定停止时间，则假定是午夜前一分钟 (11:59 pm)。一天的午夜 (12 am) 停止时间止于当天的*结束*（24 小时制的 23:59）。

today　指定当天。参数的限定 （比如 today- 天数）指定当天日期之前的天数。例如，today-1 表示昨天的日期， today-2 表示前天的日期。

last　可用于表示日志文件中包含的最晚日期。参数 last-天数指定日志文件中最晚日期*之前*的天数。

first　可用于表示日志文件中包含的最早日期。参数 first+ 天数指定日志文件中最早日期*之后*的天数。

如何使用

以下命令覆盖 *stop* 命令设置的停止日期。

- weekly

- monthly
- yearly
- extract（如果使用 day、 week、 month 或 year 参数）
- export（如果使用 day、 week、 month 或 year 参数）

示例

在此示例中， stop 命令指定要提取的最后一个间隔的停止时间是 1999 年 6 月 5 日 5:00 pm。 output 命令指定输出文件名为 myout。

```
extract>
logfile /var/opt/perf/datafiles/logglob
start 6/5/99 8:00 AM
stop 6/5/99 5:00 PM
output myout
global detail
extract
```

要使用命令行参数执行上面的任务，请输入：

```
extract -g -b 6/5/99 8:00 AM -e 6/5/99 5:00 PM -f myout -xt
```

transaction

使用 transaction 命令指定要提取或导出的事务数据的类型。

语法

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

参数

on 或 detail summary both off	请参见本章开头 application 命令描述中的 “参数” 。只能导出 summary 和 both。

示例

1999 年 6 月 1 日新建名为 rxmay99 的提取日志文件。清除任何同名的现有文件。提取 1999 年 5 月 1 日到 1999 年 5 月 31 日期间收集的所有原始事务日志文件数据。

```
extract>
logfile /var/opt/perf/datafiles/logglob
output rxmay99,purge
global detail
transaction detail
month 9905
```

要使用命令行参数执行上面的任务，请输入：

```
extract -gt -f rxmay99,purge -xm 9905
```

weekdays

使用 weekdays 命令排除导出特定日的数据 （1 = 星期日）。

语法

weekdays [1|2.....7]

如何使用

如果只要导出一周中某些天的数据，请使用此命令排除*不想*导出数据的那些天。它们有以下值：

```
Sunday     =1
Monday     =2
Tuesday    =3
Wednesday  =4
Thursday   =5
Friday     =6
Saturday   =7
```

例如，如果只要导出星期一到星期四记录的数据，请从导出中*排除*星期五、星期六和星期日的数据。

示例

在此示例中，从导出中排除星期二和星期四记录的所有详细全局数据。输出导出文件包含 myrept 导出模板文件中指定的全局度量。

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
report myrept
weekdays 35
export
```

weekly

使用 weekly 命令指定按日历周进行数据提取。一周定义为七天，从星期一到星期日。

在执行期间，此命令根据提取数据的周和年份将开始和停止日期设置为正确的日期。

语法

weekly [*yyww*]
[*ww*]

参数

weekly	提取当周的数据 （默认值）。
weekly *ww*	从当年的数据中提取特定周的数据 （其中 *ww* 是 01 到 53 之间的任何数字）。
weekly *yyww*	提取特定周*和*年的数据（其中 *yyww* 是由年份的最后两位数和两位数周号构成的单个数字）。例如， 1999 年的第 20 周是 weekly 9920。

如果在执行 weekly 命令之前未指定日志文件，将使用 datafiles 目录中的默认 logglob 文件。

如何使用

如果要按周提取数据进行存档，请使用 weekly 命令。

输出文件的名称由字母 rxwe 后跟提取的年份的最后两位数和两位数周号构成。例如，1999 年的第 12 周 （从 3 月 22 日星期一到 3 月 29 日星期日）会输出到名为 rxwe9912 的文件中。

提取和汇总的数据类型遵循 extract 命令的正常规则，可以在执行 weekly 命令之前设置。除非按周的输出文件已存在，否则使用这些设置。如果文件已存在，将根据原始选择的数据类型将数据追加到该文件末尾。

weekly 命令具有一个功能，它打开*上一周*的提取文件并检查其是否填满 （是否包含截止该周最后一天提取的数据）。如果不是， weekly 命令将数据追加到此文件以完成上一周的提取。

例如， 1999 年 5 月 20 日星期四执行了 weekly 命令，创建了名为 rxwe199920 的日志文件，该文件包含 5 月 17 日星期一到当天 （5 月 20 日）的数据。

1999 年 5 月 26 日星期三再执行一次 weekly 命令。此命令在为当周创建 rxwe199921 文件之前，先打开并检查上一周的 rxwe199920 文件。当发现文件未完成时，将数据追加到该文件以完成到 1999 年 5 月 23 日星期日为止的数据提取。然后，创建 rxwe199921 文件保存 1999 年 5 月 24 日星期一到当天 （5 月 26 日）的数据。

只要每周至少执行 weekly 命令一次，此功能就会先完成每周的文件，再创建下一周的文件。当您看到两个相邻的按周的文件时 （例如 rxwe199920 和 rxwe199921），可以认为第一个文件是该周的完整文件，可以存档和清除。

周按起始日编号。因此，一年中的第一周 （01 周）是从这一年的*第一个*星期一开始的。第一个星期一之前的任何天都属于上一年的最后一周。 weekly 和 extract week 命令有一点非常相似，它们都提取一个日历周的数据。weekly 命令忽略 output 命令的设置，而使用预定义的输出文件名。如果系统上仍存在上一周的提取日志文件，它还会尝试将缺少的数据追加到该文件中。 extract week 命令则使用 output 命令的设置。它不会将数据追加到上一周的提取文件中，因为它不知道提取文件的名称。输出文件名为 rxwe 后跟当年 (*yyyy*) 和周 (*ww*) 的数字。

示例

在此示例中， weekly 命令导致提取当周的数据，并完成上一周的提取文件（如果该文件已存在）。

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
application detail
process detail
weekly
```

要使用命令行参数执行上面的任务，请输入：

```
extract -gap -xw
```

yearly

使用 yearly 命令指定按日历年进行数据提取。

在执行期间，此命令根据提取数据的年份将开始和停止日期设置为正确的日期。

语法

```
yearly    [yyyy]
          [yy]
```

参数

yearly	提取当年的数据 （默认值）。
yearly *yy*	提取特定年的数据 （其中 yy 是 00 到 99 之间的数字）。 指定 00 到 27 假定是 2000 年到 2027 年，而 71 到 99 假定是 1971 年到 1999 年。
yearly *yyyy*	提取特定年的数据 （其中 *yyyy* 是 1971 年到 2027 年的完整年份）。

如果在执行 yearly 命令之前未指定日志文件，将使用默认 logglob 文件。

如何使用

如果要按年提取数据进行存档，请使用 yearly 命令。

输出文件的名称由字母 rxyr 后跟提取的四位数年份构成。因此， 1999 年的数据会输出到名为 rxyr1999 的文件中。

提取和汇总的数据类型遵循 extract 命令的正常规则，可以在执行 yearly 命令之前设置。除非年度输出文件已存在，否则使用这些设置。如果文件已存在，将根据原始选择的数据类型将数据追加到该文件末尾。

yearly 命令具有一个功能，它打开*上一年*的提取文件并检查其是否填满 （是否包含截止该年最后一天提取的数据）。如果不是， yearly 命令将数据追加到此文件以完成上一年的提取。

例如， 1998 年 12 月 15 日执行了 yearly 命令，创建了名为 rxyr1998 的日志文件，该文件包含 1998 年 1 月 1 日到当天 （12 月 15 日）的数据。

1999 年 1 月 5 日再执行一次 yearly 命令。此命令在为当年创建 rxyr1999 文件之前，先打开并检查上一年的 rxyr1998 文件。当发现文件未完成时，将数据追加到该文件以完成到 1998 年 12 月 31 日为止的数据提取。然后，创建 rxyr1999 文件保存 1999 年 1 月 1 日到当天 （1 月 5 日）的数据。

只要每年至少执行 yearly 命令一次，此功能就会先完成每年的文件，再创建下一年的文件。当您看到两个相邻的年度文件时（例如 rxyr1998 和 rxyr1999），可以认为第一个文件是该年的完整文件，可以存档和清除。

*只有*原始日志文件的大小调整到足以容纳*一整年*的数据，上述情况才能实现。更常见的做法是减少原始日志文件的大小，更频繁地执行 yearly 命令 （比如，每月一次）。

yearly 和 extract year 命令有一点非常相似，它们都提取一个日历年的数据。 yearly 命令忽略 output 命令的设置，而使用预定义的输出文件名。如果系统上仍存在上一年的提取日志文件，它还会尝试将缺少的数据追加到该文件中。 extract year 命令则使用 output 命令的设置。它不会将数据追加到上一年的提取文件中，因为它不知道提取文件的名称。

示例

在此示例中，将应用程序和全局详细信息数据追加到现有年度摘要文件（如有必要，则创建文件）。输出文件是 rxyr**yyyy** （其中 *yyyy* 表示当年）。

```
extract>
logfile /var/opt/perf/datafiles/logglob
global detail
application detail
process off
yearly
```

要使用命令行参数执行上面的任务，请输入：

```
extract -ga -xy
```

8 使用 cpsh 程序

只有启用 HP Ops OS Inst to Realtime Inst LTU 或 Glance Pak Software LTU 后，才能使用 cpsh 程序。

cpsh 程序提供新命令行提示符，可用于查看从监视的系统中收集的实时度量数据。

使用交互模式

可以在交互模式下使用 cpsh 程序。如果运行不带任何选项的 `cpsh` 命令， cpsh 程序将打开新命令提示符。在提示符处，可以执行各种查看实时度量详细信息任务。

要打开 cpsh 提示符，请执行以下步骤：

1. 登录到安装了 HP Operations Agent 的系统（用具有根特权 / 管理特权的身份）。
2. 运行以下命令打开本地系统的 cpsh 提示符：

   cpsh

   运行以下命令打开远程系统的 cpsh 提示符：

   cpsh -n < *系统名称* >

   < *系统名称* > 是远程系统的完全限定域名。

   或

   cpsh -n <*IP 地址* >

   <*IP 地址* > 是远程系统的 IP 地址。

   打开远程系统的 `cpsh` 提示符时，请确保远程系统上正在运行 `perfd` 进程。

   将打开 cpsh 提示符。

   要以结构清楚的表格式查看度量数据，请运行带 `-t` 选项的 `cpsh` 命令。

   例如：

   cpsh -t

   或

   cpsh -n < *系统名称* > **-t**

3. 要查看 cpsh 提示符中可用的命令的详细信息，请输入 **help**。

查看实时度量

可以从 cpsh 提示符查看可用度量的实时值。在 cpsh 提示符处执行任何操作之前，必须设置度量上下文。perfd 守护进程和关联的实用程序根据度量类处理可用数据。因此，使用 cpsh 实用程序查看实时数据时，必须总是先设置度量类再执行任何查看可用数据的操作。

要查看某个度量类的度量的实时值，请执行以下步骤：

1 在 cpsh 提示符处，输入 **class** < *度量类* >。

2 要列出特定类的当前设置度量，请输入 **list**。将显示指定度量类的所有默认度量的列表。

3 要查看属于指定类的度量的值，请在 cpsh 提示符处输入 **push**。 cpsh 程序将以表格格式显示度量实时值。

4 要返回到 cpsh 提示符，请按 **Ctrl+C**。

修改度量类

可以将其他可用度量添加到某个度量类的默认度量列表。要在 cpsh 提示符处将度量添加到度量类或从度量类删除度量，请执行以下步骤：

1 打开 cpsh 提示符。

2 在 cpsh 提示符处，输入 **class** < *度量类* >。

3 输入 **list**。将显示指定度量类的所有默认度量的列表。

4 要删除度量，请执行以下步骤：

a 在 cpsh 提示符处，输入 **delete** <*metric_name*>。

b 输入 **list**。指定度量类的度量列表不包括删除的度量。

5 要将度量添加到度量类，请执行以下步骤：

a 在 cpsh 提示符处，输入 **add** <*metric_name*>。

b 输入 **list**。指定度量类的度量列表包括新添加的度量。

查看所有可用度量

要查看属于某个度量类的所有可用度量，请执行以下步骤：

1 打开 cpsh 提示符。

2 在 cpsh 提示符处，输入 **class** < *度量类* >。

3 输入 **list all**。将显示属于指定度量类的所有可用度量的列表。

组织度量类

可以不用对度量类执行连续的添加和删除操作而重新组织度量类。要重新组织度量类以包括所选的度量，请执行以下步骤：

1 打开 cpsh 提示符。

2 在 cpsh 提示符处，输入 **class** < *度量类* >。

3 输入 **init** <*metric_name*> <*metric_name*> <*metric_name*> ...。

指定度量类只合并了用 init 命令指定的度量。

查看度量帮助

可以从 cpsh 提示符处查看每个实时度量的描述。要查看度量描述，请执行以下步骤：

1 打开 cpsh 提示符。

2 输入 class <*metric_class*>。

3 在 cpsh 提示符处，输入 **help** <*metric_name*>。将显示度量的详细信息描述。

查看汇总的度量数据

对于 GLOBAL 和 TABLE 类度量，可以从 cpsh 提示符处查看汇总数据。要查看汇总数据，请执行以下步骤：

1 打开 cpsh 提示符。

2 在 cpsh 提示符处，输入 **class gbl** 或 **class tbl**。

3 输入 **summ** < *间隔* >。在此实例中，< *间隔* > 是指定的以秒为单位的汇总间隔。< *间隔* > 必须是 cpsh 所连接的 perfd 服务器的收集间隔的倍数。

cpsh 实用程序显示属于所选度量类的度量值的以下测量：

- 最大值
- 最小值
- 平均值
- 标准偏差

9 性能警报

可以使用性能收集组件定义警报。当 scope 或 DSI 度量满足或超过您定义的条件时，这些警报会通知您。

要定义警报，必须指定每个监视的系统的条件，满足条件时即触发警报或操作。可以在警报定义文本文件 alarmdef 中定义警报。

scope 或其他收集器记录数据时，会将数据与警报定义进行比较以确定是否满足条件。如果是，将触发警报或操作。

使用实时警报生成器，可以执行以下任务：

- 将警报通知发送到 HPOM 控制台
- 生成警报通知时，创建 SNMP 陷阱
- 将 SNMP 陷阱转发到 SNMP 陷阱侦听器
- 对监视的系统执行本地操作

可以使用 utility 程序的 analyze 命令对照警报定义分析历史日志文件数据，并报告结果。

处理警报

性能收集组件收集性能数据时，会将收集的数据与 alarmdef 文件中定义的警报条件进行比较以确定是否满足条件。满足条件时，生成警报，执行定义的警报操作 （ALERT、 PRINT 和 EXEC）。

当数据收集到日志文件时，会将数据与 alarmdef 文件中的警报定义进行比较。满足警报条件时，执行警报定义中定义的操作。但是，如果数据不记录到日志文件（例如，阈值参数设置为高值时），即使满足 alarmdef 文件中的警报条件也不会生成警报。有关不同度量类的阈值，请参见第 24 页的 Thresholds。

警报定义中定义的操作可以包括：

- 使用 UNIX 命令执行的本地操作
- 发送到 Network Node Manager 和 Operations Manager 的消息

警报生成器

性能收集组件警报生成器处理警报通知的通信。警报生成器由警报生成器服务器 (perfalarm)、警报生成器数据库 (agdb) 和实用程序 agsysdb 组成。

agdb 包含 SNMP 节点的列表。agsysdb 程序用于显示和更改警报事件采取的操作。

当您启动性能收集组件时， perfalarm 也启动，并在启动时读取 agdb 以确定是否发送警报通知以及将警报通知发送到哪里。它还检查系统上是否安装了 Operations Manager 代理程序。

使用以下命令行选项可以查看接收警报通知的目标列表：

```
agsysdb -l
```

将 SNMP 陷阱发送到 Network Node Manager

要将 SNMP 陷阱发送到 Network Node Manager，必须使用以下命令将您的系统名称添加到性能收集组件的 agdb：

agsysdb -add *系统名称*

生成的每个 ALERT 都将导致 SNMP 陷阱发送到定义的系统。陷阱文本包含与 ALERT 相同的消息。

要停止将 SNMP 陷阱发送到系统，必须使用以下命令从 agdb 删除系统名称：

agsysdb -delete *系统名称*

将消息发送到 Operations Manager

如果性能收集组件所在的系统上安装了 Operations Manager 代理程序，您可以将警报通知发送到 Operations Manager。Operations Manager 代理程序与 Operations Manager 中央系统通信。

默认情况下，如果性能收集组件系统上正在运行 Operations Manager 代理程序，警报生成器*不*执行 EXEC 语句中的任何警报中定义的本地操作。而只是将消息发送到 Operations Manager 的事件浏览器。如果性能收集组件系统上*未*在运行 Operations Manager 代理程序，警报生成器不会将警报通知发送到 Operations Manager，也不执行本地操作。

可以使用以下命令更改默认值，即使性能收集组件系统上正在运行 Operations Manager 代理程序也不将信息发送到 Operations Manager：

```
agsysdb -ovo OFF
```

要将性能收集组件陷阱发送到另一个节点，请将以下条目添加到 /etc/services 文件。

```
snmp-trap    162/tcp   # SNMPTRAP
snmp-trap    162/udp   # SNMPTRAP
```

在此实例中， 162 指定端口号。如果希望性能收集组件将陷阱发送到另一个节点，它会检查 /etc/services 文件中是否有 snmp-trap 字符串。如果不提供此条目，陷阱不会发送到另一个节点。

执行本地操作

如果性能收集组件系统上未在运行操作监视组件，则将执行 EXEC 语句中的本地操作。

可以按如下方式更改默认值关闭本地操作：

```
agsysdb -actions off
```

如果希望即使操作监视组件代理程序正在运行也*总是*执行本地操作，请输入：

```
agsysdb -actions always
```

下表列出了将信息发送到 HP Operations Manager (HPOM) 和执行本地操作的设置：

表 10　　将信息发送到 HPOM 和执行本地操作的设置

标记	操作监视组件正在运行	操作监视组件未运行
Operations Manager 标记		
off	不将警报通知发送到 HPOM。	不将警报通知发送到 HPOM。
on	将警报通知发送到 HPOM。	不将警报通知发送到 HPOM。
本地操作标记		
off	不执行本地操作。	不执行本地操作。
always	即使操作监视组件正在运行也执行本地操作。	执行本地操作。
on	将本地操作发送到 HPOM。	执行本地操作。

警报处理中的错误

发送警报时发生的最后一个错误记录在 agdb 中。要查看 agdb 的内容，请输入：

```
agsysdb -l
```

将显示以下信息：

```
PA alarming status:

OVO messages : on      Last Error : none
Exec Actions : on
Analysis system: <主机名>, Key=<IP 地址>
PerfView : no Last Error : <错误号>
SNMP : yes Last Error : <错误号>
```

为警报分析历史数据

可以使用 utility 程序的 analyze 命令在日志文件数据中查找警报条件（请参见第 5 章 utility 命令）。这与前面说明的实时警报的处理不同，因为您要将日志文件中的历史数据与 alarmdef 文件中的警报定义进行比较，以确定触发了哪些警报条件。

历史数据中的警报信息示例

以下示例显示了分析历史数据中的警报条件时报告的内容。

对于第一个示例，警报定义中定义了 **START**、**END** 和 **REPEAT** 语句。每次警报满足其所有条件指定持续时间时，都列出警报启动事件。当不再满足这些条件时，列出警报结束事件。如果满足警报条件的时间长到可以不用结束第一个警报而生成另一个警报，则列出重复事件。

列出的每个事件都显示日期和时间、警报号和警报事件。*不*执行 **EXEC** 操作，但会用任何请求的参数代替 **EXEC** 操作列出来。

```
05/10/99 11:15  ALARM [1] START
CRITICAL: CPU test 99.97%
05/10/99 11:20  ALARM [1] REPEAT
WARNING: CPU test 99.997%
05/10/99 11:25  ALARM [1] END
RESET: CPU test  22.86%
EXEC: end.script
```

如果使用的是彩色工作站，会突出显示以下输出：

```
CRITICAL statements are RED
MAJOR statements are MAGENTA
MINOR statements are YELLOW
WARNING statements are CYAN
NORMAL statements are GREEN
```

下面的示例显示了列出警报事件之后显示的警报摘要。第一列列出警报号，第二列列出发生警报条件的次数，第三列列出警报条件的总持续时间。

```
Performance Alarm Summary:

  Alarm   Count   Minutes
      1     574      2865
      2       0         0

Analysis coverage using "alarmdef":
Start: 05/04/99 08:00    Stop: 05/06/99 23:59
Total time analyzed:  Days: 2  Hours: 15  Minutes: 59
```

警报定义组件

当一个或多个定义的条件持续指定时长时，即发生警报。警报定义可以包括警报启动或结束时要执行的操作。

条件是两个或更多项之间的比较。被比较项可以是度量名称、常量或变量。例如：

```
ALARM gbl_cpu_total_util > 95 FOR 5 MINUTES
```

可以指定警报启动、结束或重复时要执行的操作。操作可以是以下某项：

- ALERT，将消息发送到 Operations Manager 或将 SNMP 陷阱发送到 Network Node Manager
- EXEC，执行 UNIX 命令，或
- PRINT，使用 `utility` 程序处理时将消息发送到 `stdout`。

例如：

```
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"
```

可以使用布尔逻辑、多实例数据（比如，应用程序）循环和变量创建更复杂的操作。（有关详细信息，请参见下面的警报语法参考部分）。

还可以使用 INCLUDE 语句识别要使用的其他警报定义文件。例如，可能希望将警报定义分为几个小文件。

警报语法参考

此部分介绍警报语法中可用的语句。有关如何使用语法创建有用的警报定义的示例，可能要查看 alarmdef 文件。

警报语法

```
ALARM  条件 [[AND,OR] 条件 ]
  FOR 持续时间 [SECONDS, MINUTES]
  [TYPE=" 字符串 "]
  [SERVICE=" 字符串 "]
  [SEVERITY= 整数 ]
  [START 操作 ]
  [REPEAT EVERY 持续时间 [SECONDS, MINUTES] 操作 ]
  [END 操作 ]
[RED, CRITICAL, ORANGE, MAJOR, YELLOW, MINOR, CYAN, WARNING,
  GREEN, NORMAL, RESET] ALERT 消息
EXEC "UNIX 命令 "
PRINT 消息
IF 条件
   THEN 操作
   [ELSE 操作 ]
{APPLICATION, PROCESS, DISK, LVOLUME, TRANSACTION, NETIF, CPU,
   FILESYSTEM} LOOP 操作
INCLUDE " 文件名 "
USE " 数据源名称 "
[VAR] 名称 =  值
ALIAS 名称 = " 替换的名称 "
SYMPTOM  变量 [ TYPE = {CPU, DISK, MEMORY, NETWORK}]
  RULE 条件  PROB 概率
  [RULE 条件 PROB 概率 ]
    .
    .
```

约定

- 大括号 ({ }) 表示其中的一个选项是必需的。
- 方括号 ([ ]) 表示可选项。
- 方括号或大括号中由逗号分隔的项是选项。只能选择一个。
- 斜体表示要替换的变量名称。
- 所有语法关键字都大写。

常见元素

警报语法的多个语句中都会用到以下元素，介绍如下。

- 注释
- 复合语句

- 条件
- 常量
- 表达式
- 度量名称
- 消息

注释

可以在注释前双正斜杠 (//) 或英镑标记 (#)。在这两种情况下，注释都在行尾结束。例如：

```
# 任何文本或字符
```

或

```
// 任何文本或字符
```

复合语句

复合语句允许将语句列表作为单个语句执行。复合语句是大括号 ({}) 内的语句列表。复合语句与 IF 语句、LOOP 语句以及 ALARM 语句的 START、REPEAT 和 END 子句结合使用。复合语句不能包括 ALARM 和 SYMPTOM 语句。

```
{
 语句
 语句
}
```

在下面的示例中， highest_cpu 定义变量。highest_cpu 值会保存下来，并只在 highest_cpu 值被更大的 highest_cpu 值超过时通知您。

```
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
```

条件

条件定义为两个项之间的比较。

```
项 1 {>, <, >=, <=, ==, !=} 项 2
  [AND, OR[ 项 3 {>, <, >=, <=, ==, !=} 项 4]]
```

其中 “==” 表示 “等于”，“!=” 表示 “不等于”。

ALARM、IF 和 SYMPTOM 语句中会使用条件。项可以是度量名称、数字常量、用引号括起来的字母数字字符串、别名或变量。比较字母数字项时，只能使用 == 或 != 运算符。

常量

常量可以是数字或字母数字。字母数字常量两边必须加双引号。例如：

```
345
345.2
"Time is"
```

常量在表达式和条件中很有用。例如，您可能想将度量与条件内的常量数字值进行比较，以在值过高时生成警报，比如

```
gbl_cpu_total_util > 95
```

表达式

算术表达式对两个或更多操作数执行一次或多次算术运算。在要使用数字值的地方都可以使用表达式。逻辑算术运算符包括：

```
+, -, *, /
```

可以使用圆括号控制先求值的表达式部分。

例如：

```
Iteration + 1
gbl_cpu_total_util - gbl_cpu_user_mode_util
( 100 - gbl_cpu_total_util ) / 100.0
```

度量名称

在警报定义中指定度量名称时，要替换成当前度量值。度量名称必须严格按照度量定义中的形式输入，但不区分大小写。有关度量定义可参见*性能收集组件 Performance Collection Component Dictionary of Operating Systems Performance Metrics*。

如果度量来自于非 SCOPE 数据源 （比如 DSI 度量），建议使用完全限定度量名称。

指定完全限定度量的格式是：

```
data_source:instance(class):metric_name
```

SCOPE 数据源中的全局度量不需要限定。例如：

```
metric_1
```

SCOPE 数据源中定义的每个应用程序都可使用的应用程序度量需要应用程序名称。例如，

```
application_1:metric_1
```

对于 application、process、disk、netif、transaction、lvolume、cpu 和 filesystem 等多实例数据类型，必须将度量与数据类型名称关联，使用 LOOP 语句时例外。为此，请指定后跟冒号的数据类型名称，然后是度量名称。例如， other_apps:app_cpu_total_util 指定应用程序 other_apps 的 CPU 总利用率。

指定完全限定多实例度量并在别名中使用别名时，如果一个别名具有类标识符，建议使用以下示例中所示的语法：

```
alias my_fs="/dev/vg01/lvol1(LVOLUME)"
alarm my_fs:LV_SPACE_UTIL > 50 for 5 minutes
```

如果使用有嵌入空格的应用程序名称，必须用下划线 (_) 代替空格。例如，application 1 必须更改为 application_1。有关使用包含特殊字符的名称或大小写很重要的名称的详细信息，请参见第 171 页的 ALIAS 语句。

如果有一个名为 “other” 的磁盘和一个名为 “other” 的应用程序，则需要指定类及实例：

```
other (disk):metric_1
```

提取的日志文件中的全局度量 （这里 scope_extract 是数据源名称）会如下指定：

```
scope_extract:application_1:metric_1
```

DSI 度量会如下指定：

```
dsi_data_source:dsi_class:metric_name
```

任何包含特殊字符 （比如，星号）的度量名称都必须使用别名才能指定。

消息

消息是由 PRINT 或 ALERT 语句发送的信息。它可以包括加引号的字母数字字符串、数字常量、表达式和变量的任意组合。消息中的元素用逗号隔开。例如：

```
RED ALERT "cpu utilization=", gbl_cpu_total_util
```

数字常量、度量和表达式可以进行宽度和小数位数格式化。*宽度*指定字段应格式化为多宽，*小数位数*指定要使用几位小数。数字值右对齐。-（减号）指定左对齐。字母数字字符串总是左对齐。例如：

```
度量名称 [|[-] 宽度 [| 小数位数 ]]
gbl_cpu_total_util|6|2    格式化为 '100.00'
(100.32 + 20)|6           格式化为 '   120'
gbl_cpu_total_util|-6|0   格式化为 '100   '
gbl_cpu_total_util|10|2   格式化为 '    99.13'
gbl_cpu_total_util|10|4   格式化为 '  99.1300'
```

ALARM 语句

ALARM 语句定义一个或一组条件以及条件为 true 的持续时间。在 ALARM 语句中，可以定义警报条件启动、重复和结束时要执行的操作。可能想要定义为警报的条件或事件包括：

- 全局交换空间快满的状况持续 5 分钟

- 内存分页速率过高持续 1 个间隔
- 前十分钟 CPU 一直以 75% 的利用率运行

语法

```
ALARM  条件 [[AND,OR] 条件 ]
  FOR 持续时间 {SECONDS, MINUTES}
  [TYPE=" 字符串 "]
  [SERVICE=" 字符串 "]
  [SEVERITY= 整数 ]
  [START 操作 ]
  [REPEAT EVERY 持续时间 {SECONDS, MINUTES} 操作 ]
  [END 操作 ]
```

- **ALARM** 语句必须是顶层语句。它不能嵌套在任何其他语句中。但是，可以在单个 ALARM 语句中包括多个 ALARM 条件。如果条件用 **AND** 连接，所有条件都必须为 true 才能触发警报。如果条件用 **OR** 连接，只要满足任何一个条件就会触发警报。
- **TYPE** 是一个加引号的字符串，最多可包含 38 个字符。如果要发送警报，可以使用 **TYPE** 对警报分类并指定要使用的图形模板名称。
- **SERVICE** 是一个加引号的字符串，最多可包含 200 个字符。如果要使用将要与 ITO 5.0 一起发行的 ServiceNavigator，可以将性能收集组件警报与 ServiceNavigator 中定义的服务链接起来（请参见《HP Operations ServiceNavigator 概念和配置指南》*(HP Operations ServiceNavigator Concepts and Configuration Guide)*。

  ```
  SERVICE=" 服务 ID"
  ```

- **SEVERITY** 是 0 到 32767 之间的整数。
- **START**、**REPEAT** 和 **END** 是分别用于指定满足警报条件、再次满足条件或停止条件时要采取的操作的*关键字*。ALARM 语句中应至少有一个 **START**、**REPEAT** 或 **END**。每个关键字后面都会跟着一项*操作*。
- *操作* - ALARM START、REPEAT 或 END 最常用的操作是 ALERT 语句。但是，也可以使用 EXEC 语句发送消息邮件或运行批处理文件，或者在用 `utility` 程序分析历史日志文件时使用 PRINT 语句。除了不允许再使用一个 ALARM 语句外，任何语法语句都可以使用。

  START、REPEAT 和 END 操作可以是复合语句。例如，可以使用复合语句提供 ALERT 和 EXEC。
- *条件* - 条件定义为两个项之间的比较。

  ```
  项 1 {>, <, >=, <=, ==, !=} 项 2
     [AND, OR[ 项 3 {>, <, >=, <=, ==, !=} 项 4]]
  ```

  其中 “==” 表示 “等于”，“!=” 表示 “不等于”

  项可以是度量名称、数字常量、用引号括起来的字母数字字符串、别名或变量。比较字母数字项时，只能使用 == 或 != 运算符。

  可在子条件之间指定 “OR” 和 “AND” 运算符来使用复合条件。例如：

  ```
  ALARM gbl_cpu_total_util > 90 AND
  gbl_pri_queue > 1 for 5 minutes
  ```

- 也可以*不*在子条件之间指定 “OR” 和 “AND” 运算符而使用复合条件。例如：

```
ALARM gbl_cpu_total_util > 90
gbl_cpu_sys_mode_util > 50 for 5 minutes
```

将在这两个条件都为 true 时生成警报。

FOR *持续时间* **SECONDS, MINUTES** 指定条件必须一直为 true 才能触发警报的时间段。

指定小于一分钟的持续时间，尤其是系统上有多个数据源时务必谨慎。如果必须以很短的间隔轮询每个数据源中的数据，会严重影响性能。持续时间必须是警报条件中提及的度量的最长收集间隔的倍数。

对于 scope 数据，持续时间是五分钟；但是，进程数据的持续时间是一分钟。对于 DSI 数据，持续时间是五秒或更长。

- **REPEAT EVERY** *持续时间* **SECONDS, MINUTES** 指定重复警报之前的时间段。

如何使用

警报循环开始于所有用 AND 连接的条件或某个用 OR 连接的条件为 true 并至少持续指定时长的第一个间隔。那时，警报生成器执行 START *操作*，并在每个后续间隔检查 REPEAT 条件。如果经过了足够长的时间，则执行 REPEAT 子句的*操作*。（这将持续到一个或多个警报条件变为 false 为止。）这样就完成警报循环并执行 END 语句 （如有）。

若要发出警报通知，请在 START 和 END 语句中使用 ALERT 语句。如果不指定 END ALERT，警报生成器自动将警报发送到 Operations Manager，将 SNMP 陷阱发送到 Network Node Manager。

示例

以下 ALARM 示例在交换空间利用率很高的状况持续 5 分钟时发送红色警报。它与默认 alarmdef 文件中的警报条件类似。务必先删除默认警报条件再将此示例添加到 alarmdef 文件，否则后续警报消息可能造成混乱。

```
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"
```

此 ALARM 示例测试 gbl_swap_space_util 度量是否超过 90。根据警报生成器的配置，ALERT 可以通过 SNMP 陷阱发送到 Network Node Manager，也可以作为消息发送到 Operations Manager。

REPEAT 语句每 15 分钟检查 gbl_swap_space_util 条件一次。只要度量仍然大于 90，REPEAT 语句就会每 15 分钟发送一次消息 “swap utilization is still very high”。

当 gbl_swap_space_util 条件小于 90 时，发送 RESET ALERT 语句和 “End of swap utilization condition” 消息。

以下示例在 ALARM 语句中定义复合操作。此示例显示了如何在发生事件时导致发送消息邮件。

```
ALARM gbl_cpu_total_util > 90 FOR 5 MINUTES
  START
  {
    RED ALERT "Your CPU is busy."
    EXEC "echo 'cpu is too high'| mailx root"
  }
  END
     RESET ALERT "CPU no longer busy."
```

ALERT 可以触发将 SNMP 陷阱发送到 Network Node Manager 或将消息发送到 Operations Manager。EXEC 可以触发将邮件消息作为性能收集组件系统的本地操作发送，具体取决于警报生成器的配置。

默认情况下，如果 Operations Manager 代理程序正在运行，将不执行本地操作。而是将本地操作作为消息发送到 Operations Manager。

以下两个示例显示了多个条件的运用。ALARM 语句中可以有多个测试条件。在这种情况下，每个语句都必须为 true 才会发送 ALERT。

以下 ALARM 示例测试 gbl_cpu_total_util 度量和 gbl_cpu_sys_mode_util 度量。如果这两个条件都为 true， RED ALERT 语句将发送红色警报。任一测试条件变为 false 时，就发送 RESET。

```
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."
```

下面的 ALARM 示例测试 gbl_cpu_total_util 度量和 gbl_cpu_sys_mode_util 度量。如果有一个条件为 true， RED ALERT 语句将发送红色警报。

```
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"
```

不要在相同警报中使用不同间隔记录的度量。例如，不应在一个语句中按全局度量值 （以 5 分钟间隔记录）对进程 （以 1 分钟间隔记录）执行循环，如下所示：

```
IF global_metric THEN
   PROCESS LOOP...
```

不同间隔不能按您的预期同步，因此结果是无效的。

对于 GlancePlus，请在进程循环中使用进程度量以便为所有进程发送警报。

ALERT 语句

ALERT 语句允许将消息发送到 Network Node Manager 或 Operations Manager。ALERT 语句通常用作 ALARM 内的一项操作。它也可以用于 IF 语句内，只要检测到条件就发送消息，而不是持续时间过去之后发送消息。如果将 ALERT 用在 ALARM 或 IF 语句之外，每个间隔都会发送一次消息。

语法

[RED, CRITICAL, ORANGE, MAJOR, YELLOW, MINOR, CYAN, WARNING, GREEN, NORMAL, RESET] *ALERT 消息*

- **RED** 表示 **CRITICAL**， **ORANGE** 表示 **MAJOR**， **YELLOW** 表示 **MINOR**， **CYAN** 表示 **WARNING**，**GREEN** 表示 **NORMAL**。这些关键字将警报符号转化为与警报条件关联的颜色。
- **RESET** - ALARM 条件结束时，发送 **RESET** ALERT 及消息。如果尚未在警报定义中定义重置， ALARM 条件结束时只发送 **RESET** ALERT，而不发送消息。
- *消息* - 用于创建消息的字符串和数字值的组合。数字值可以用参数 [|[-] *宽度* [| *小数位数*]] 格式化。*宽度*指定字段应格式化为多宽，*小数位数*指定要使用几位小数。数字值右对齐。-（减号）指定左对齐。字母数字字符串总是左对齐。

如何使用

ALERT 也可以触发将 SNMP 陷阱发送到 Network Node Manager 或将消息发送到 Operations Manager，具体取决于警报生成器的配置。 对于发送到 Operations Manager 的警报消息，消息浏览器中会以蓝色显示 WARNINGS。

示例

典型 ALERT 语句为：

```
RED ALERT "CPU utilization = ", gbl_cpu_total_util
```

如果安装了 Network Node Manager，此语句将在 Network Node Manager 的警报浏览器窗口创建严重级别为严重的警报。

EXEC 语句

EXEC 语句可用于指定要对本地系统执行的系统 （UNIX 或 Windows）命令。例如，使用 EXEC 语句在每次满足特定条件时将邮件发送给 IT 管理员。

EXEC 应用于 ALARM 或 IF 语句内，以便仅在满足指定条件时执行命令。如果将 EXEC 语句用于 ALARM 或 IF 语句之外，将以不可预测的间隔执行操作。

语法

EXEC " *系统命令* "

系统命令 - 要对本地系统执行的命令。

不要在 EXEC 语句中使用嵌入双引号 (")。这样做会导致 perfalarm 无法将警报发送到 HPOM。而应使用嵌入单引号 (')。例如：

```
EXEC "echo 'performance problem detected' "
EXEC "mkdir c:\\directory\\filename"
```

EXEC 语句的语法要求用双引号将文件的路径名称括起来。但是，如果路径名称包含空格，路径名称必须用单引号括起来，而且必须再在外面加上双引号。

示例：

```
EXEC "'C:\\Program Files\\Mail Program\\SendMail.exe'"
```

只要 EXEC 语句的系统命令参数包含单引号，就必须用单引号将程序括起来，因为用 EXEC 语句运行命令时会将第一对单引号 (') 转换成双引号 (")。

示例：

```
EXEC "'echo' 'test execution'"
```

在上面的示例中， echo 是用单引号括起来的程序，因为它包含带单引号的参数 （此示例中为 test execution）。此外，按照 EXEC 语句的语法，命令的整个字符串两边必须加双引号。

不要在 EXEC 语句中使用嵌入双引号 (")，否则 perfalarm 无法将警报发送到 HPOM。而应使用嵌入单引号 (')。

例如：

```
EXEC "'echo' 'dialog performance problem'"
```

在上面的示例中， echo 是用单引号括起来的程序，因为它包含带单引号的参数 （此示例中为 dialog performance problem）。此外，按照 EXEC 语句的语法，命令的整个字符串两边必须加双引号。

如何使用

EXEC 可以触发本地系统上的本地操作，具体取决于警报生成器的配置。例如，可以打开或关闭本地操作。如果警报生成器配置为将信息发送到 Operations Manager，通常不会执行本地操作。

示例

在以下示例中，当 gbl_disk_util_peak 度量超过 20 时， EXEC 语句执行 UNIX mailx 命令。

```
IF gbl_disk_util_peak > 20 THEN
  EXEC "echo 'high disk utilization detected'| mailx root"
```

下面的示例显示了网络数据包速率超过每秒 1000 的平均值且持续 15 分钟时， EXEC 语句将邮件发送给系统管理员。

```
ALARM gbl_net_packet_rate > 1000 for 15 minutes
    TYPE = "net busy"
    SEVERITY = 5
    START
      {
      RED ALERT "network is busy"
      EXEC "echo 'network busy condition detected'| mailx root"
      }
    END
    RESET ALERT "NETWORK OK"
```

使用带有高开销的命令或脚本且经常执行的 EXEC 语句时，务必要谨慎。

警报生成器会执行命令，并等到命令完成后再继续。建议不要指定需要很长时间才能完成的命令。

PRINT 语句

PRINT 语句可用于从 utility 程序使用 analyze 函数打印消息。警报生成器会忽略 PRINT 语句。

语法

PRINT *消息*

- *消息* - 创建消息的字符串和数字值的组合。数字值可以用参数 [|[-] *宽度* [| *小数位数*]] 格式化。*宽度*指定字段应格式化为多宽，*小数位数*指定要使用几位小数。消息的字母数字部分必须加上引号。数字值右对齐。- （减号）指定左对齐。字母数字字符串总是左对齐。

示例

```
PRINT "The total time the CPU was not idle is",
              gbl_cpu_total_time |6|2, "seconds"
```

执行时，此语句打印如下消息：

```
The total time the CPU was not idle is 95.00 seconds
```

IF 语句

使用 IF 语句可以定义 IF-THEN 逻辑的条件。IF 语句应用于 ALARM 语句内。但是，它可以单独使用，也可以用于 alarmdef 文件中任何需要 IF-THEN 逻辑的位置。

如果在 ALARM 语句之外指定 IF 语句，则无法控制其执行频率。

语法

IF *条件* **THEN** *操作* **[ELSE** *操作***]**

- **IF** *条件* - 条件定义为两个项之间的比较。

  项 1 {>, <, >=, <=, ==, !=} 项 2
  [AND, OR[项 3 {>, <, >=, <=, ==, !=} 项 4]]

  其中 “==” 表示 “等于”，“!=” 表示 “不等于”。

  项可以是度量名称、数字常量、用引号括起来的字母数字字符串、别名或变量。比较字母数字字符串时，只能使用 == 或 != 运算符。

- *操作* - 任何操作，或设置变量。（ALARM 在这种情况下无效。）

如何使用

IF 语句测试*条件*。如果*条件*为 true，执行 THEN 之后的*操作*。如果*条件*为 false，*操作*取决于可选的 ELSE 子句。如果已指定 ELSE 子句，执行子句之后的*操作*；否则 IF 语句不执行任何操作。

示例

在此示例中，计算 CPU 瓶颈症状，所生成的瓶颈概率用于定义青色或红色 ALERT。根据警报生成器的配置， ALERT 触发将 SNMP 陷阱发送到 Network Node Manager 或将消息 “End of CPU Bottleneck Alert” 及占用的 CPU 百分比一同发送到 Operations Manager。

```
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"
```

不要在相同语句中使用不同间隔记录的度量。例如，不应在一个语句中按全局度量值（以 5 分钟间隔记录）对进程（以 1 分钟间隔记录）执行循环，如下所示：

```
IF global_metric THEN
  PROCESS LOOP ...
```

不同间隔不能按您的预期同步，因此结果是无效的。

LOOP 语句

LOOP 语句遍历多实例数据类型，并执行为每个实例定义的*操作*。

语法

```
{APPLICATION, PROCESS, LVOLUME, DISK, CPU, FILESYSTEM, TRANSACTION,
NETIF, LOGICAL}
LOOP
    操作
```

- **APPLICATION**, **PROCESS**, **LVOLUME**, **DISK**, **CPU**, **FILESYSTEM**, **TRANSACTION**, **NETIF**, **LOGICAL** - 包含多实例数据的性能收集组件数据类型。
- *操作* - PRINT、EXEC、ALERT，或设置变量。

如何使用

当 LOOP 语句循环访问数据类型的每个实例时，度量值会更改。例如，如果正在使用 utility 程序的 analyze 命令，以下 LOOP 语句会将每个应用程序的名称打印到 stdout。

```
APPLICATION LOOP
  PRINT app_name
```

一个 LOOP 可以嵌套在另一个 LOOP 语句中，但最多只能嵌套五层。

为了使 LOOP 执行， LOOP 语句必须引用一个或多个 LOOP 语句中定义的相同数据类型的度量。

示例

可以使用 LOOP 语句在所有活动应用程序间循环。

以下示例显示了如何确定每个间隔 CPU 利用率最高的应用程序。

```
highest_cpu = 0
 APPLICATION loop
   IF app_cpu_total_util > highest_cpu THEN
    {
      highest_cpu = app_cpu_total_util
      big_app = app_name
     }
  ALERT "Application ", app_name, " has the highest cpu util at
",highest_cpu_util|5|2, "%"
 ALARM highest_cpu > 50
   START
     RED ALERT  big_app, " is the highest CPU user at ", highest_cpu, "%"
   REPEAT EVERY 15 minutes
     CYAN ALERT big_app, " is the highest CPU user at ", highest_cpu, "%"
   END
 RESET ALERT "No applications using excessive cpu"
```

INCLUDE 语句

使用 INCLUDE 语句可以包括默认 alarmdef 文件和另一个警报定义文件。

语法

INCLUDE " *文件名* **"**

其中*文件名*是另一个警报定义文件的名称。文件名必须总是完全限定名称。

如何使用

可以使用 INCLUDE 语句将逻辑上截然不同的警报定义组拆分成单独文件。

示例

例如，如果在单独的文件中有一些事务度量的警报定义，并命名为

```
trans_alarmdef1
```

可以将以下行添加到 alarmdef1 文件的警报定义中以包括它：

```
INCLUDE "/var/opt/perf/trans_alarmdef1"
```

USE 语句

引用了默认 SCOPE 数据源以外的数据源时，可以添加 USE 语句简化 alarmdef 文件中度量名称的使用。这使您可以不必包括数据源名称而指定度量名称。

必须在 datasources 文件中定义数据源名称。如果遇到无效或不可用的数据源名称， alarmdef 文件的语法检查将失败。

在 alarmdef 文件中使用 **USE** 语句并不意味着该语句后面的所有度量名称都来自指定数据源。

语法

USE " *数据源名称* "

如何使用

警报生成器检查 alarmdef 文件的语法是否有效时，它生成文件中引用的所有数据源的顺序搜索列表。perfalarm 在遇到完全限定度量名称或 USE 语句时按顺序将条目添加到此数据源搜索列表。此列表随后用于对未完全限定的度量名称与合适的数据源名称进行匹配。使用 USE 语句可方便地将数据源添加到 perfalarm 搜索列表，进而允许在 alarmdef 文件中使用缩短的度量名称。有关度量名称语法的论述，请参见本章前面第 158 页的度量名称。

perfalarm 的默认度量名称和数据源匹配行为是先在 SCOPE 数据源中查找度量名称。 当 perfalarm 在 alarmdef 文件中遇到第一个度量名称时，执行隐含的 USE "SCOPE" 语句。此功能启用默认的 SCOPE 数据源搜索路径，以便在 alarmdef 文件中可以引用 SCOPE 度量，而无需完全限定度量名称。请参见下一页的以下示例。

```
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"
```

perfalarm 检查包含上述语句的 alarmdef 文件的语法时，会遇到度量“gbl_cpu_total_util”，它会尝试查找其数据源。perfalarm 的数据源搜索列表中还没有任何数据源，因此它执行隐含的 **USE** "SCOPE" 语句，然后搜索 SCOPE 数据源以查找度量名称。找到匹配项后，perfalarm 继续检查 alarmdef 文件的其余部分。

当 perfalarm 遇到 USE "ORACLE7" 语句时，将 ORACLE7 数据源添加到数据源搜索列表。遇到“ActiveTransactions”度量名称时，perfalarm 从 SCOPE 数据源开始按顺序搜索数据源列表。SCOPE 不包含该度量名称，因此接下来搜索 ORACLE7 数据源，并找到匹配项。

如果 perfalarm 在任何数据源中都找不到度量名称的匹配项，则打印错误消息，并终止 perfalarm。

要更改默认搜索行为，可以将 USE 语句添加到 alarmdef 文件开头、所有度量名称引用的前面。这会导致将 USE 语句中指定的数据源添加到数据源搜索列表中 SCOPE 数据源的前面。将先搜索 USE 语句中的数据源再搜索 SCOPE 数据源，以查找度量名称的匹配项。请参见以下示例。

一旦使用 USE 语句引用数据源，就没有任何方式可以更改搜索顺序或从搜索列表中删除数据源。

```
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"
```

在上面的示例中，alarmdef 文件中的语句顺序发生了变化。在引用任何度量名称之前，先定义 USE "ORACLE7" 语句，因此 ORACLE7 数据源作为第一个数据源添加到数据源搜索列表。当 perfalarm 遇到第一个度量名称“gbl_cpu_total_util”时，执行隐含的 USE "SCOPE" 语句。因为“gbl_cpu_total_util”度量名称不是完全限定名称，perfalarm 从 ORACLE7 开始按顺序搜索数据源列表。ORACLE7 不包含该度量名称，因此接下来搜索 SCOPE 数据源，并找到匹配项。

perfalarm 继续检查 alarmdef 文件的其余部分。当 perfalarm 遇到“ActiveTransactions”度量时，从 ORACLE7 开始按顺序搜索数据源列表。找到匹配项后，perfalarm 继续搜索 alarmdef 文件的其余部分。如果 perfalarm 在任何数据源中都找不到度量名称（非完全限定）的匹配项，则打印错误消息，并终止 perfalarm。

当多个数据源包含相同度量名称时，应注意如何使用 USE 语句。perfalarm 按顺序搜索数据源列表。如果您从使用相同度量名称的不同数据源定义警报条件，必须用数据源名称限定度量名称以确保从正确的数据源检索度量值。如以下示例所示，警报语句中的度量名称都包含数据源。

```
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 语句

VAR 语句可用于定义变量并向变量分配值。

语法

[VAR] *名称* **=** *值*

- *名称* - 变量名称必须以字母开头，可包括字母、数字和下划线字符。变量名称不区分大小写。
- *值* - 如果值是字母数字字符串，必须在两边加引号。

如何使用

VAR 将值分配给用户变量。如果变量之前不存在，则创建变量。

定义后，变量可用于 alarmdef 文件的任何位置。

示例

可以通过将值分配给变量来定义变量。以下示例通过分配零值定义数字变量 *highest_CPU_value*。

```
highest_CPU_value = 0
```

下面的示例通过分配空字符串值定义字母数字变量 *my_name*。

```
my_name = ""
```

ALIAS 语句

ALIAS 语句可用于在度量名称的任何部分（类、实例或度量）有区分大小写的名称或有包含特殊字符的名称时用别名替换。只有在上述情况下才应使用 ALIAS 语句。

语法

ALIAS 名称 **= "** *替换的名称* **"**

- *名称* - 名称必须以字母开头，可包括字母、数字和下划线字符。
- *替换的名称* - 必须用 ALIAS 语句进行替换才能使警报生成器唯一可识别的名称。

如何使用

由于 alarmdef 文件的处理方式，如果度量名称的任何部分（类、实例或度量名称）只能通过大小写唯一识别，则需要创建别名。还需要为包含特殊字符的所有名称创建别名。例如，如果有名为“BIG”和“big”的应用程序，需要将“big”别名化以确保它们被视为不同的应用程序。必须在 alarmdef 文件中要替换的名称的*第一个*实例之前的某个位置定义别名。

示例

因为在语法上不能使用特殊字符或区分大小写，使用应用程序名称“AppA”和“appa”会因无法区分这两者而导致错误。可以将“AppA”别名化赋予其一个唯一可识别的名称。例如：

```
ALIAS appa_uc = "AppA"
ALERT "CPU alert for AppA.util is",appa_uc:app_cpu_total_util
```

如果要对带有类标识符的实例使用别名，请在别名中同时包括实例名称和类名。以下示例显示了实例名称“other”、类名“APPLICATION”的别名。

```
ALIAS my_app="other(APPLICATION)"
   ALERT my_app:app_cpu_total_util > 50 for 5 minutes
```

SYMPTOM 语句

SYMPTOM 提供了基于一组条件设置单个变量值的方式。只要任一条件为 true，它就将概率值加到 SYMPTOM 变量值上。

语法

SYMPTOM *变量*
  RULE *条件* **PROB** *概率*
  [RULE *条件* **PROB** *概率* **]**
      .
      .
      .

- 关键字 **SYMPTOM** 和 **RULE** 只能用于 SYMPTOM 语句，不能用于其他语法语句中。SYMPTOM 语句必须是顶层语句，不能嵌套在任何其他语句内。其他语句只能在所有对应的 RULE 语句都结束后才能跟在 SYMPTOM 后面。
- *变量*是将成为此症状名称的变量名称。SYMPTOM 语句中定义的变量名称可以在其他语法语句中使用，但不得在这些语句中更改变量值。
- **RULE** 是 SYMPTOM 语句的选项，不能单独使用。可以根据需要在 SYMPTOM 语句中使用任意多个 **RULE** 选项。SYMPTOM 变量在每个间隔按照 RULE 计算。
- *条件*定义为两个项之间的比较。

  项 1 {>, <, >=, <=, ==, !=} 项 2
  [项 3 {>, <, >=, <=, ==, !=} 项 4]

  其中 “==” 表示 “等于”，“!=” 表示 “不等于”。

  项可以是度量名称、数字常量、用引号括起来的字母数字字符串、别名或变量。比较字母数字项时，只能使用 == 或 != 运算符。
- *概率*是数字常量。值为 true 的每个 SYMPTOM RULE 的概率加起来即为 SYMPTOM 值。

如何使用

“测量” 和 “值” 之间的条件为 true 的所有概率的总和就是症状发生的概率。

示例

```
SYMPTOM CPU_Bottleneck
RULE gbl_cpu_total_util > 75  PROB 25
RULE gbl_cpu_total_util > 85  PROB 25
RULE gbl_cpu_total_util > 90  PROB 25
RULE gbl_run_queue      >  3  PROB 50
IF CPU bottleneck > 50 THEN
CYAN ALERT "The CPU symptom is: ", CPU_bottleneck
```

警报定义示例

以下示例显示了警报定义的典型用法。

CPU 问题示例

此示例指示一旦 CPU 利用率超过 90% 达到 5 分钟，且 CPU 运行队列超过 3 个达到 5 分钟，就触发将 SNMP 陷阱发送到 Network Node Manager，或将消息发送到 Operations Manager （具体取决于警报生成器的配置）。

```
ALARM gbl_cpu_total_util > 90 AND
  gbl_run_queue > 3 FOR 5 MINUTES
  START
  CYAN ALERT "CPU too high at", gbl_cpu_total_util, "%"
REPEAT EVERY 20 MINUTES
{
  RED ALERT "CPU still to high at ", gbl_cpu_total_util, "%"
  EXEC "/usr/bin/pager -n 555-3456"
}
  END
  RESET ALERT "CPU at ", gbl_cpu_total_util, "% - RELAX"
```

如果 20 分钟后这两个条件仍为 true，就可能在 Network Node Manager 警报浏览器窗口中创建严重级别为严重的警报。然后，运行程序通知系统管理员。

当任一警报条件不为 true 时，删除警报符号并发送消息，显示全局 CPU 利用率、警报结束时间和 RELAX 注释。

交换空间利用率示例

在此示例中，只要交换空间利用率超过 95% 达到 5 分钟，ALERT 就触发将 SNMP 陷阱发送到 Network Node Manager，或将消息发送到 Operations Manager （具体取决于警报生成器的配置）。

```
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"
```

基于时间的警报示例

可以指定警报条件可以活动的时间间隔。例如，如果要运行定期执行的系统维护作业，可以为正常运行时间指定警报条件，并为系统维护时间指定一组不同的警报条件。

在此示例中，仅在一天当中的 8:00AM 到 5:00PM 触发警报。

```
start_shift = "08:00"
end_shift = "17:00"
ALARM gbl_cpu_total_util > 80
    TIME > start_shift
    TIME < end_shift for 10 minutes
```

```
TYPE = "cpu"
START
  CYAN ALERT "cpu too high at ", gbl_cpu_total_util, "%"
REPEAT EVERY 10 minutes
  RED ALERT"cpu still too high at ", gbl_cpu_total_util, "%"
END
  IF time == end_shift then
  {
  IF gbl_cpu_total_util > 80 then
     RESET ALERT "cpu still too high, but at the end of shift"
  ELSE
     RESET ALERT  "cpu back to normal"
  ELSE
```

磁盘实例警报示例

可以通过识别特定磁盘实例名称和对应度量名称，为特定磁盘生成警报。

以下警报语法示例为特定磁盘实例生成警报。当磁盘实例中使用特殊字符时，需要别名。

```
ALIAS diskname=""
ALARM diskname:bydsk_phys_read > 1000 for 5 minutes
TYPE="Disk"
  START
     RED ALERT "Disk  "
  REPEAT EVERY 10 MINUTES
     CYAN ALERT "Disk  cyan alert"
  END
     RESET ALERT "Disk  reset alert"
```

自定义警报定义

在警报定义文件 alarmdef 中指定生成警报的条件。首次安装性能收集组件时，alarmdef 文件包含一组默认警报定义。可以使用这些默认警报定义，也可以自定义警报定义以符合您的需要。

可以按如下方式自定义 alarmdef 文件：

1 根据需要修订警报定义。可以查看本章其他部分的警报定义语法示例。

2 保存文件。

3 使用性能收集组件 utility 程序验证警报定义：

   a 输入 **utility**

   b 在提示符处，输入

   checkdef

   此命令检查警报语法，并在发现文件有问题时显示错误或警告。

4 为使新警报定义生效，请输入：

   ovpa restart alarm

   或

   mwa restart alarm

   这会导致警报生成器停止、重新启动然后读取自定义的 alarmdef 文件。

可以对每个性能收集组件系统使用一组独有的警报定义，也可以选择对一组系统使用一组相同警报定义以标准化该组系统的监视。

如果 alarmdef 文件非常大，性能收集组件警报生成器和 utility 程序可能不会按预期工作。此问题是否发生取决于系统资源的可用性。

了解性能警报的最佳方法是试验添加新警报定义或更改默认警报定义。

10 RTMA 组件的 Adviser

只有启用 HP Ops OS Inst to Realtime Inst LTU 或 Glance Pak Software LTU 后，才能使用 adviser 功能。

adviser 功能可用于在 RTMA 组件收集的特定度量的值超过 （或低于）设置的阈值时生成和查看警报。adviser 功能由 **adviser 脚本**和 **padv** 实用程序组成。adviser 脚本帮助您创建规则，在监视的系统的性能出现下降迹象时生成警报。 `padv` 实用程序帮助您在感兴趣的系统上运行 adviser 脚本。

此主题着重讨论 RTMA 组件可以使用的 adviser 功能。GlancePlus 软件提供可与 adviser 实用程序一起使用的其他功能。有关结合使用 adviser 功能与 GlancePlus 软件的信息，请参见 *GlancePlus 的联机帮助*。

警报和症状

警报可用于突出显示度量条件。adviser 脚本可用于为 RTMA 组件监视的度量定义阈值。当度量值超出设置的阈值时， RTMA 组件以警报消息的形式生成警报。此消息以 `stdout` 的形式发送到 **padv** 实用程序。

一旦满足您指定的条件就可能触发警报。警报基于您指定的任何时间段，可以是一个间隔或更长。

症状是影响系统性能的条件的组合。

adviser 通过观察具有相应阈值的不同度量并将值添加到这些度量造成瓶颈的概率，计算出一个表示瓶颈存在的联合概率的值。

运行 adviser 脚本

运行 `padv` 命令时， HP Operations Agent 扫描该命令指定的脚本并执行必需的操作。如果不用 `padv` 命令指定任何脚本文件， adviser 实用程序将从以下默认脚本文件检索必需的信息：

- *在 Windows 上：%ovdatadir%*`\perf\perfd`
- *在 UNIX 和 Linux 上：*`/var/opt/perf/perfd`

如果要运行包括特定于操作系统的诊断和操作的脚本，请使用以下默认脚本：

- *在 Windows 上：%ovdatadir%*\perf\perfd\os\<*操作系统类型*>\adv
- *在 UNIX 和 Linux 上：*/var/opt/perf/perfd/os/<*操作系统类型*>/adv

在此实例中，<*操作系统类型*>是您要运行脚本的节点的操作系统类型。

运行 adviser 脚本后，可以实现以下操作：

- 根据生成的警报将系统状态打印到文本文件
- 在运行 padv 命令的命令控制台中查看系统的实时状态。

使用 adviser

要使用 adviser 组件监视系统的实时运行状况，请执行以下步骤：

1. 按照要求配置 adviser 脚本。可在以下目录中找到示例脚本：
   - *在 Windows 上：%ovinstalldir%*\examples\adviser
   - *在 UNIX 或 Linux 上：*/opt/perf/examples/adviser
2. 标识要运行脚本的节点。
3. 确保 perfd 进程在标识的系统上运行。
4. 运行以下命令：

   padv -s <*脚本名称*> **-n** <*系统名称*>

   adviser 脚本开始在指定系统上运行，并根据脚本文件的配置创建结果。

在多个系统上运行 adviser 脚本

可使用 mpadv 命令在多个系统上运行 adviser 脚本。要使用 mpadv 命令，请执行以下步骤：

1. 标识要运行脚本的节点。
2. 创建列出所有标识的系统的名称的文本文件。
3. 在本地系统上保存该文本文件。
4. 按照要求配置 adviser 脚本。可在以下目录中找到示例脚本：
   - *在 Windows 上：%ovinstalldir%*\examples\adviser
   - *在 UNIX 或 Linux 上：*/opt/perf/examples/adviser
5. 确保 perfd 进程在标识的系统上运行。
6. 运行以下命令：

   mpadv -l <*系统列表文本文件*>**-s**<*脚本名称*>

   adviser 脚本开始在 < *系统列表文本文件* > 文件中指定的系统上运行，并根据脚本文件的配置创建结果。

adviser 语法

adviser 语法是简单的脚本语言，可用于设置警报并定义症状条件。

以下目录中提供了默认语法文件 `adviser.syntax`：

- *在 Windows 上：%ovdatadir%*`\perf`
- *在 UNIX 和 Linux 上：*`/var/opt/perf`

可编辑语法文件以定义自己的警报和症状。

语法约定

- 大括号 ({ }) 表示其中的一个选项是必需的。
- 方括号 ([ ]) 表示可选项。
- 方括号或大括号中由逗号分隔的项是选项。只能选择一个。
- 斜体表示将替换的变量名称。
- adviser 语法关键字必须始终大写。

注释

语法：

[*任何文本或字符*]

或

// [*任何文本或字符*]

可以在注释前加双正斜杠 (//) 或 # 号。在这两种情况下，注释都在行尾结束。

条件

条件定义为两个度量名称、用户变量或数字常量之间的比较。

```
项 1 {>, <, >=, <=, ==, !=} 项 2 [OR 项 3 \
                {>, <, >=, <=, ==, !=} 项 4]
```

或：

```
项 1 {>, <, >=, <=, ==, !=} 项 2 [AND 项 3 \
                {>, <, >=, <=, ==, !=} 项 4]
```

（“==”表示 “等于”，“!=”表示 “不等于”。）

ALARM 语句和 IF 语句中会使用条件。它们可用于比较两个数字度量、变量或常量，也可以用在两个字符串度量名称、用户变量或字符串常量之间。对于字符串条件，只能使用 == 或 != 运算符。

可在子条件之间指定 OR 或 AND 运算符来使用复合条件。

示例：

```
gbl_swap_space_reserved_util > 95
proc_proc_name == "test" OR proc_user_name == "tester"
proc_proc_name != "test" AND
                  proc_cpu_sys_mode_util > highest_proc_so_far
```

常量

常量可以是字母数字或数字。字母数字常量两边必须加双引号。有两类数字常量：整数和实数。整数常量只能包含数字和可选的符号标志。实数常量还可以包含小数点。

示例：

345	整数数字
345.2	实数数字
“Time is”	字母数字文本

表达式

使用表达式计算数字值。条件或操作中可使用表达式。

表达式可以包含：

- 数字常量
- 数字度量名称
- 数字变量
- 以上各项的算术组合
- 用括号将上述各项分组的组合

示例：

```
Iteration + 1
3.1416
gbl_cpu_total_util - gbl_cpu_user_mode_util
( 100 - gbl_cpu_total_util ) / 100.0
```

adviser 语法中的度量名称

在 adviser 语法中，可在任何位置直接引用度量。可以在 adviser 语法中使用以下类型的度量：

- 全局度量 （gbl_ 或 tbl_ 前缀）
- 应用程序度量 （app_ 前缀）
- 进程度量 （proc_ 前缀）
- 磁盘度量 （bydsk_ 前缀）
- “按 CPU”度量 （bycpu_ 前缀）
- 文件系统度量 （fs_ 前缀）
- 逻辑卷度量 （lv_ 前缀）
- 网络接口度量 （bynetif_ 前缀）
- 交换度量 （byswp_ 前缀）
- ARM 度量 （tt_ 或 ttbin_ 前缀）
- PRM 度量 （prm_ 前缀）
- 位置域度量 （ldom_ 前缀）

在 LOOP 语句上下文中，只能使用进程、逻辑卷、磁盘、文件系统、 LAN 和交换度量。

度量可包含字母数字 （如 gbl_machine 或 app_name）或数字数据，可以反映几个不同种类的测量。例如，度量名称的结尾表示测量的对象：

- `a _util` 度量测量利用率百分比
- `a _rate` 度量测量每秒单位
- `a _queue` 度量测量等待资源的进程或线程数

如果不能确定特定度量的测量单位，请参考度量定义文档。

除非使用 LOOP 语句，否则必须将应用程序度量与特定应用程序关联。为此，请指定后跟冒号的应用程序名称，然后是度量名称。例如， other_apps:app_cpu_total_util 指定应用程序 other_apps 的 CPU 总利用率。有关在语法中使用应用程序度量的详细信息，请参考 ALIAS 语句描述。

parm 文件定义的应用程序名称可能包含特殊字符和嵌入空格。为了在语法中使用这些名称 （应用程序名称必须与变量名称的形式匹配） ，名称不区分大小写，嵌入空格转换成下划线。这意味着定义为“Other Apps”的应用程序名称在语法中可能引用为 `other_apps`。对于用特殊字符定义的应用程序名称，必须用 ALIAS 语句指定备选名称。

显式限定时，可在语法中的任何位置引用应用程序度量。未限定的应用程序度量只能在 LOOP 语句上下文中引用。这是隐式限定应用程序或进程度量的迭代语句。

只能在 LOOP 语句上下文中引用进程度量。无法显式引用进程。

打印列表

打印列表是格式正确的表达式、度量名称、用户变量或常量的任意组合。请参见正确格式的示例。

表达式示例：

表达式 [| 宽度 [| 小数位数]]

度量名称或用户变量示例:

度量名称 [| 宽度 [| 小数位数]]

或

用户变量[| 宽度[| 小数位数]]

度量名称或用户变量必须是字母数字。

常量示例：

常量不需要格式化。

格式化的示例：

```
gbl_cpu_total_util|6|2     格式化为 '100.00'
(100.32 + 20)|6            格式化为 '120'
gbl_machine|5              格式化为 '7013/'
"User Label"               格式化为 "User Label"
```

变量

变量必须以字母开头，可包括字母、数字和下划线字符。变量不区分大小写。

通过将值分配给变量来定义变量。以下示例通过分配零值定义数字变量 highest_CPU_value。

```
highest_CPU_value = 0
```

以下示例通过分配空字符串值定义字母数字变量 my_name。

```
my_name = ""
```

adviser 语法语句

ALARM 语句

使用 ALARM 语句在系统上发生您定义的特定事件时通知您。adviser 脚本可使用 ALARM 语句通过发送到发出 padv 命令的控制台的消息通知您。

语法：

```
ALARM 条件  [FOR 持续时间 {SECONDS, MINUTES, INTERVALS}]
  [ 条件 [FOR 持续时间 {SECONDS, MINUTES, INTERVALS}]  ] ...

[START 语句 ]
```

```
[REPEAT [EVERY 持续时间 [SECONDS, MINUTES, INTERVAL, INTERVALS]]
  语句 ]

[END 语句 ]
```

ALARM 语句必须是顶层语句。它不能嵌套在任何其他语句中。

但是，可以在单个 ALARM 语句中包括若干 ALARM 条件，在这种情况下，所有条件都必须为 true 才会触发警报。还可以使用复合语句，该语句在警报循环期间内的恰当时间执行。

START、 REPEAT 和 END 是 ALARM 语句关键字。这些关键字中的每一个都指定一条语句。在 ALARM 语句中必须有 START、 REPEAT 或 END，并且必须以正确顺序列出。

警报循环开始于所有警报条件已为 true 并至少持续指定时长的第一个间隔。那时 adviser 脚本执行 START 语句，并在每个后续间隔检查 REPEAT 条件。如果经过了足够长的时间，则执行 REPEAT 子句的语句。这将持续到一个或多个警报条件变为 false 为止。这样就完成警报循环并执行 END 语句。

如果 REPEAT 语句中省略了 EVERY 规范， adviser 脚本将在每个间隔执行 REPEAT 语句。

ALARM 示例：典型的 ALARM 语句

以下 ALARM 示例设置在信标表将满时发出红色警报。它类似于默认语法中的预定义警报。务必先删除默认警报再将它添加到语法，否则后续警报消息可能造成混乱。

```
ALARM tbl_sem_table_util > 90 FOR 1 MINUTE

  START RED ALERT "Semaphore Table is nearly full"

  REPEAT EVERY 30 SECONDS
    RED ALERT "Semaphore Table still nearly full"

  END RESET ALERT "End of Semaphore Table full condition"
```

此 ALARM 示例测试度量 tbl_sem_table_util，看它是否大于 90。如果大于，则 RED ALERT 语句将严重性为 RED 的消息发送到发出 padv 命令的控制台。

REPEAT 语句每 30 秒检查 tbl_sem_table_util 条件一次。只要该条件大于 90， REPEAT 就通知 adviser 保持 RED ALERT 条件，并将 Semaphore Table still nearly full 消息发送到发出 padv 命令的控制台。

tbl_sem_table_util 条件低于 90 时，RESET ALERT 语句在发出 padv 命令的控制台中显示消息 End of Semaphore Table full condition。

ALARM 示例：使用多个条件

ALARM 语句中可以有多个测试条件。在这种情况下，每个语句都必须为 true 才会激活警报按钮。例如：

```
ALARM gbl_cpu_total_util > 90 FOR 2 MINUTES
  gbl_cpu_sys_mode_util > 50 FOR 1 MINUTES
    START RED ALERT
      "The CPU is busy and System Mode CPU utilization is high."
    END RESET ALERT "The CPU alert is now over."
```

此 ALARM 示例测试度量 gbl_cpu_total_util 和 CPU_Bottleneck。如果这两个条件都为 true， RED ALERT 语句将设置严重警报。任一测试条件变为 false 时，就执行 RESET 语句。

ALARM 示例：交换空间

```
//GLOBAL SWAP ALARM
symp_swap_util = gbl_swap_space_used / gbl_swap_space_avail
ALARM symp_swap_util > 0.9
  START
    RED ALERT "GLOBAL SWAP space is nearly full"
  END RESET ALERT "GLOBAL SWAP space crisis is over"
```

新变量 symp_swap_util 表示交换空间利用率。交换空间利用率超过 90% 时， adviser 脚本创建警报。在下一间隔，如果 symp_swap_util 低于 90%，警报条件变成 false，则重置 ALARM。

ALARM 示例：黄色警报

```
ALARM Symp_Global_Cpu_Bottleneck >  50 FOR 2 MINUTES

  START YELLOW ALERT "CPU Bottleneck probability= ",
    Symp_Global_Cpu_Bottleneck, "% for the last 2 minutes"

  REPEAT every 2 minutes
    YELLOW ALERT "CPU Bottleneck probability= ",
      Symp_Global_Cpu_Bottleneck, "% for the last 2 minutes"

  END
    RESET ALERT " CPU Bottleneck Yellow Alert over, probability=",
      Symp_Global_Cpu_Bottleneck, "%"
```

ALARM 测试 SYMPTOM 变量，此变量在 SYMPTOM 语句 Symp_Global_Cpu_Bottleneck 中定义。如果 SYMPTOM 变量大于 50 持续了 2 分钟，则 ALARM 将 YELLOW ALERT 发送到 padv 命令控制台来通知您。

ALARM 每 2 分钟重复一次，直到 ALARM 条件变为 false。那时 END 语句将重置 ALERT。

ALARM 示例：CPU 问题

```
ALARM

  gbl_cpu_total_util > 90 FOR 30 SECONDS

  gbl_run_queue > 3 FOR 30 SECONDS

  START YELLOW ALERT "CPU AT ", gbl_cpu_total_util,
    "% at ", gbl_stattime

  REPEAT EVERY 300 SECONDS {
    RED ALERT "CPU AT ", gbl_cpu_total_util
    exec "/usr/bin/pager -n 555-3456"
  }

  END ALERT "CPU at ", gbl_cpu_total_util, "% at ",
    gbl_stattime, "- RELAX"
```

此示例指示一旦 CPU 利用率超过 90% 达到 30 秒，且 CPU 运行队列超过 3 个达到 30 秒，就在 padv 命令控制台中生成黄色警报。

如果这两个条件都一直为 true，则生成红色警报并调用呼叫系统管理员的程序。

ALERT 语句

ALERT 语句用于在 padv 命令控制台中放置消息。一旦 ALARM 检测到问题，便可以运行 ALERT 语句，以指定严重性将消息发送到 padv 命令控制台。

可以结合使用 ALERT 语句与 ALARM 语句。

语法：

```
[(RED 或 CRITICAL), (YELLOW 或 WARNING), RESET] ALERT 打印列表
```

RED 和 YELLOW 分别与 CRITICAL 和 WARNING 同义。

ALERT 示例

ALERT 语句示例：

```
RED ALERT "CPU utilization = ", gbl_cpu_total_util,
  " at ", gbl_stattime
```

运行此语句时， padv 命令控制台中将写入消息，例如：

```
CPU utilization = 85.6 at 14:43:10
```

ALIAS 语句

使用 ALIAS 语句将变量分配给包含特殊字符或嵌入空格的应用程序名称。

语法：

```
ALIAS 变量 = "别名"
```

ALIAS 示例

因为不能在语法中使用特殊字符或嵌入空格，在下面的 PRINT 语句中使用应用程序名称 “other user root” 会导致错误。使用 ALIAS，您仍可以在语法中使用 “other user root” 或其他带空格和特殊字符的字符串。

```
ALIAS otherapp = “other user root”


PRINT "CPU for other root login processes is: ",
  otherapp:app_cpu_total_util
```

ASSIGNMENT 语句

使用 ASSIGNMENT 语句将数字值或字母数字值、表达式分配给用户变量。

语法：

```
[VAR] 变量 = 表达式
[VAR] 变量 = 字母数字项
[VAR] 变量 = 字母数字项
```

ASSIGNMENT 示例

用户变量在第一次赋值时确定是数字还是字母数字。不能在赋值语句中混合使用不同类型的变量。

此示例将字母数字的应用程序名称分配给新用户变量：

```
myapp_name = other:app_name
```

此示例不正确，因为它将数字值分配给以前定义为字母数字 （在示例 1 中）的用户变量：

```
myapp_name = 14
```

此示例将数字值分配给新用户变量：

```
highest_cpu = gbl_cpu_total_util
```

此示例不正确，因为它将字母数字文本分配给以前定义为数字 （在示例 3 中）的用户变量：

```
highest_cpu = "Time is"
```

复合语句

复合语句与 IF 语句、LOOP 语句以及 ALARM 语句的 START、REPEAT 和 END 子句结合使用。可使用复合语句执行语句列表。

语法

{

语句

语句

}

通过在大括号 ({}) 内将语句列表归为一组来构造复合语句。随后在语法中可将复合语句视为单个语句。

复合语句不能包括 ALARM 和 SYMPTOM 语句。（复合是一种语句类型而非关键字。）

复合语句示例

```
highest_cpu = highest_cpu
IF gbl_cpu_total_util > highest_cpu  THEN
  // Begin compound statement
  {
    highest_cpu = gbl_cpu_total_util
    PRINT "Our new high CPU value is ", highest_cpu, "%"
  }
  // End compound statement
```

在此示例中，`highest_cpu = highest_cpu` 定义名为 `highest_cpu` 的变量。adviser 脚本会保存 `highest_cpu` 值，并只在 `highest_cpu` 值被更大的值超过时通知您。

在该示例中，如果用 highest_cpu = 0 替换 highest_cpu = highest_cpu，那么 highest_cpu 值会在每个间隔重置为零。

每个间隔都会通知您 `highest_cpu` 的值。

EXEC 语句

使用 EXEC 语句从 adviser 语法内部执行 UNIX 命令。例如，如果要在每次满足特定条件时将邮件消息发送到 MIS 员工，就可以使用 EXEC 命令。

语法

`EXEC` 打印列表

生成的打印列表提交到操作系统以便执行。

因为您指定的 EXEC 命令可能每个更新间隔都执行一次，使用带有高开销的操作系统命令或脚本的 EXEC 语句时务必要谨慎。

EXEC 示例

在以下示例中， EXEC 在每个间隔执行 UNIX mailx 命令。

```
EXEC "echo 'gpm mailed you a message' | mailx root"
```

在以下示例中，仅当 gbl_disk_util_peak 度量超过 20 时， EXEC 才执行 UNIX mailx 命令。

```
IF gbl_disk_util_peak > 20 THEN
  EXEC "echo 'gpm detects high disk utilization' | mailx   root"
```

IF 语句

使用 IF 语句测试在 adviser 脚本语法中定义的条件。

语法：

```
IF 条件  THEN 语句 [ELSE 语句 ]
```

IF 语句测试条件。如果为 true，执行 THEN 后面的语句。如果条件为 false，则操作取决于可选的 ELSE 子句。

如果已指定 ELSE 子句，则执行它后面的语句。否则，IF 语句不执行任何操作。该语句可以是告诉 adviser 脚本执行多条语句的复合语句。

IF 示例

```
IF gbl_cpu_total_util > 90 THEN
  PRINT "The CPU is running hot at: ", gbl_cpu_total_util
ELSE IF gbl_cpu_total_util < 20 THEN
  PRINT "The CPU is idling at: ", gbl_cpu_total_util
```

在此示例中， IF 语句检查条件 (gbl_cpu_total_util > 90)。如果条件为 true，则 “The CPU is running hot at: " 和使用的 CPU 百分比一起显示在 padv 命令控制台中。

如果 (gbl_cpu_total_util > 90) 条件为 false， ELSE IF 转到下一行，并检查条件 (gbl_cpu_total_util < 20)。如果该条件为 true，则 “The CPU is idling at:” 和使用的 CPU 百分比一起显示在 padv 命令控制台中。

LOOP 语句

使用 LOOP 语句查找有关系统的信息。例如，可以查找使用 CPU 的百分比最高的进程，或使用率最高的交换区。可以用 LOOP 语句和使用您要收集其信息的系统条件度量名称的对应语句查找此信息。

语法：

```
{APPLICATION, APP, CPU, DISK, DISK_DETAIL, FILESYSTEM, FS, FS_DETAIL, LAN,
LOGICALVOLUME, LV, LV_DETAIL, NETIF, NFS, NFS_BYSYS_OPS, NFS_OP, PRM,
PRM_BYVG, PROCESS, PROC, PROC_FILE, PROC_REGION, PROC_SYSCALL, SWAP,
SYSTEMCALL, SC, THREAD, TRANSACTION, TT, TTBIN, TT_CLIENT, TT_INSTANCE,
TT_UDM, TT_RESOURCE, TT_INSTANCE_CLIENT, TT_INSTANCE_UDM, TT_CLIENT_UDM,
LDOM, PROC_LDOM}
```

```
LOOP 语句
```

LOOP 可以嵌套在其他语法语句中，但最多只能嵌套五层。该语句可能是复合语句，包含每次循环迭代时都要执行的语句块。 BREAK 语句允许从 LOOP 语句转义。

如果在语法中有收集特定数据的 LOOP 语句，而系统上没有对应的度量数据， adviser 脚本将跳过该 LOOP，并继续下一条语法语句或指令。例如，如果已定义 LOGICAL VOLUME LOOP，但系统上没有逻辑卷， adviser 脚本将跳过该 LOGICAL VOLUME LOOP，并继续下一条语法语句。

平台上不存在的循环会产生语法错误。

当 LOOP 语句循环访问每个间隔时，语句中使用的度量值会更改。例如，以下 LOOP 语句为系统上的每个活动应用程序执行 PRINT 语句一次，从而打印每个应用程序的名称。

APP LOOP

```
  PRINT app_name
```

在 HP_UX 11.23 等线程操作系统上，adviser 脚本支持线程循环。线程循环可嵌套在进程循环中，以检查特定进程的每个线程。如果在线程循环内引用 PROC_ 度量，它可能返回意外的结果 （线程信息）。

线程循环还可以在进程循环外。在这种情况下，它会查系统上的所有活动线程。不应在线程循环中嵌套进程循环。

由于 LOOP 语句在每个间隔启动，使用它们要小心，因为可能影响性能。使用嵌套 LOOP 语句时尤其应小心。

APPLICATION LOOP 示例

使用 APPLICATION LOOP 语句在所有活动应用程序间循环。

在 APPLICATION LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或应用程序 (app_) 度量。

以下示例使用 APPLICATION LOOP 查找某个间隔内 CPU 利用率最高的应用程序。

```
big_app = ""
highest_cpu = 0
APPLICATION LOOP
  IF (app_cpu_total_util > highest_cpu) THEN
  {
    highest_cpu = app_cpu_total_util
    big_app = app_name
  }
  IF (highest_cpu > 20) THEN
    YELLOW ALERT "The application ", big_app,
      " is the highest CPU user at", highest_cpu, "%"
```

查找应用程序之后，如果 CPU 值大于 20， adviser 脚本将包含 app_name 和 CPU 值的消息写入 padv 命令控制台。

CPU LOOP 示例

使用 CPU LOOP 语句循环查看有关系统上 CPU 使用情况的数据。在 CPU LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或 “按 CPU” (bycpu_) 度量。

此示例打印系统上每个 CPU 的 CPU 使用百分比。

```
Print "----------", glb_stattime, "----------"
CPU LOOP
  PRINT "CPU # ", bycpu_id, " used ", bycpu_cpu_total_util, " % CPU"
```

在有两个 CPU 的系统上，两个间隔生成的输出为：

```
----------10:52:01----------
CPU #             0 used   0.6 % CPU
CPU #             1 used   3.4 % CPU
----------10:52:11----------
CPU #             0 used   0.4 % CPU
CPU #             1 used    2.3 % CPU
```

DISK LOOP 示例

使用 DISK LOOP 语句在配置的磁盘设备间循环。使用此 LOOP 时，adviser 脚本检查 “按磁盘的 IO (IO by Disk)” 窗口中显示的特定磁盘信息。在 DISK LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或 “按磁盘” 度量。

此示例打印系统上每个磁盘的物理写速率。

```
PRINT "---------------------", gbl_stattime, "--------------------"
DISK LOOP
  PRINT bydsk_devname, " write rate: ", bydsk_phys_write_rate
```

在有三个磁盘的系统上，两个间隔生成的输出为：

```
---------------------11:00:23-----------------------
/dev/hdisk0          write rate:    2.4
/dev/hdisk1          write rate:    0.0
/dev/cd0             write rate:    0.0

---------------------11:00:33-----------------------
/dev/hdisk0          write rate:    0.0
/dev/hdisk1          write rate:    0.0
/dev/cd0             write rate:     0.0
```

FILE SYSTEM LOOP 示例

FILE SYSTEM LOOP 设计为在配置的文件系统间循环，并允许 adviser 脚本报告 “按文件系统的 IO (IO By File System)” 窗口中可访问的信息。在 FILE SYSTEM LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或 “按文件系统的 IO” (fs_) 度量。

以下示例报告有三个设备的系统上每个文件系统设备利用的空间。

```
PRINT "--------------------", gbl_stattime, "----------------------"
FS LOOP
  PRINT fs_devname, " is ", fs_space_util, "% full at ",
    fs_max_size," megabytes"
```

在有三个文件系统的系统上，两个间隔的输出为：

```
----------------------11:11:28------------------------
/dev/hd4              is   77.9% full at       32 megabytes
/dev/hd2              is   94.9% full at      928 megabytes
/dev/hd9var           is   93.9% full at       56 megabytes


----------------------11:11:38------------------------
/dev/hd4              is   77.9% full at       32 megabytes
/dev/hd2              is   94.9% full at      928 megabytes
/dev/hd9var           is   93.6% full at       56 megabytes
```

NFS BY OPERATION LOOP 示例

使用 NFS BY OPERATION LOOP 在执行的 NFS 操作间循环。使用此 LOOP 时， adviser 脚本检查特定的 NFS 操作。在 NFS_OP LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或 “按操作” 度量。

以下示例打印执行的服务器和客户端操作：

```
PRINT “--------------------” , gbl_stattime, “----------------------”
NFS_OP LOOP
  PRINT byop_server_count," server and ",byop_client_count,
    " client ",byop_name," operations performed"
```

在作为 NFS 服务器不执行任何活动、但其用户在另一台 NFS 服务器上执行目录列表操作的系统上，生成的输出为：

```
----------------------14:55:41------------------------
    0 server and      0 client null           operations performed
    0 server and      2 client getattr        operations performed
    0 server and      0 client setattr        operations performed
    0 server and      0 client root           operations performed
    0 server and    886 client lookup         operations performed
```

```
    0 server and    884 client readlink       operations performed
    0 server and      0 client read           operations performed
    0 server and      0 client writecache     operations performed
    0 server and      0 client write          operations performed
    0 server and      0 client create         operations performed
    0 server and      0 client remove         operations performed
    0 server and      0 client rename         operations performed
    0 server and      0 client link           operations performed
    0 server and      0 client symlink        operations performed
    0 server and      0 client mkdir          operations performed
    0 server and      0 client rmdir          operations performed
    0 server and     28 client readdir        operations performed
    0 server and      1 client statfs         operations performed
```

NETWORK INTERFACE LOOP 示例

使用 NETWORK INTERFACE LOOP 在配置的 LAN 设备间循环。

```
# This version will only work with hp-ux 11.x.If you want it to
# work for 10.20 you need to remove the "BYNETIF_QUEUE," string
# below as that metric is only available from 11.x glance.
# The following string variable should be changed to the interface
# of interest.  For example:
#   netif_to_examine = "lan0"
# If you want to see all interfaces, leave it an empty string (""):
#   netif_to_examine = ""
# initialize variables:

headers_printed = headers_printed
netif loop {
# print information for the selected interface or if null then all:
IF (BYNETIF_NAME == netif_to_examine) or
    (netif_to_examine == "") THEN
  {

# print headers the first time through the loop:
     IF headers_printed == 0 THEN
    {
      print "Time    Interface  InPkts OutPkts  OutQ  Colls  Errs"
      print "   "
```

```
      headers_printed = 1
    }
# print one line per interface reported:
    print GBL_STATTIME, " ", BYNETIF_NAME|8,
      BYNETIF_IN_PACKET, BYNETIF_OUT_PACKET,
      BYNETIF_QUEUE, BYNETIF_COLLISION, BYNETIF_ERROR
    # (note that some interface types do not report collisions or
    # errors)
}
}
print "  "
```

生成的输出为：

```
Time     Interface  InPkts OutPkts  OutQ  Colls  Errs

22:43:42 lan3          49      3     0      0      0
22:43:42 lan0           0      0     0      0      0
22:43:42 lan1           0      0     0      0      0
22:43:42 lan2           0      0     0      0      0
22:43:42 lo0            0      0     0      0      0

22:43:47 lan3         329      2     0      0      0
22:43:47 lan0           0      0     0      0      0
22:43:47 lan1           0      0     0      0      0
22:43:47 lan2           0      0     0      0      0
22:43:47 lo0            0      0     0      0      0
```

LOGICAL VOLUME 示例

使用 LOGICAL VOLUME LOOP 在配置的逻辑卷间循环。在 LOGICAL VOLUME LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或逻辑卷度量。

```
 PRINT "-----------------", gbl_stattime,    "-------------------"
  LV LOOP
    PRINT "Volume ", lv_dirname, " was read at a rate of ",
      lv_read_rate, " per second"
```

在有逻辑卷的系统上，两个间隔生成的输出为：

```
----------------------------11:46:50-----------------------------
```

```
Volume /dev/vg00            was read at a rate of    0.0 per second
Volume /dev/vg00/group      was read at a rate of    0.0 per second
Volume /dev/vg00/lvol3      was read at a rate of  314.3 per second

----------------------------11:47:00-----------------------------
Volume /dev/vg00            was read at a rate of    0.0 per second
Volume /dev/vg00/group      was read at a rate of    0.0 per second
Volume /dev/vg00/lvol3      was read at a rate of   70.6 per second
```

PRM LOOP 示例

仅限 HP-UX。使用 PRM LOOP 循环查看 Process Resource Manager (PRM) 组中找到的信息。在 PRM LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或应用程序度量。

以下 PRM LOOP 示例检查是否有高运行队列和任何超过其 CPU 权利的 PRM 组。

```
IF gbl_run_queue > 3 THEN {
  print " "

  print "--- High run queue = ", gbl_run_queue, " at ", gbl_stattime,
    " ---"
  prm loop {
      IF app_prm_state > 2 THEN
      IF app_cpu_total_util > app_prm_cpu_entitlement THEN
        print "    Note PRM group ", app_name_prm_groupname,
          " exceeds entitlement."
  }
}
```

每个间隔打印的输出为：

```
--- High run queue =   3.4 at 15:53:29 ---
    Note PRM group Testing exceeds entitlement.
```

PRM_BYVG LOOP 示例

仅限 HP-UX。使用 PRM_BYVG LOOP 在某个卷组的 PRM 组之间循环。（请注意，PRM 信息仅可用于 PRM 配置文件中指定的卷组。） PRM_BYVG LOOP 必须嵌套在 LV LOOP 中。以下示例按 PRM 组显示磁盘资源使用情况统计信息。

```
PRM loop {
  disk_state = app_prm_disk_state
}
```

```
IF disk_state == 0 THEN{
  print "  Disk manager state: Not Installed"
}

else IF disk_state == 1 THEN {
  print "  Disk manager state: Reset"
}

else IF disk_state == 2 THEN {
  print "  Disk manager state: Disabled"
}

else IF disk_state == 3 THEN {

  print "  Disk manager state: Enabled"

  lv loop {

    IF lv_type == "G" THEN {

      print "  Volume Group: ", lv_dirname
      print "                        %        %          KB"
      print "PRM Group     PRMID entitled achieved   transferred"
      print "-------------------------------------------------"

      prm_byvg loop {
        print prm_byvg_prm_groupname|13, prm_byvg_prm_groupid|5,
          prm_byvg_group_entitlement|8, prm_byvg_group_util|8,
          prm_byvg_transfer
      }
      print "   "
    }
  }
}
```

每个间隔的输出为：

```
Disk manager state: Enabled
Volume Group: /dev/vg00

                  %        %        KB

PRM Group     PRMID entitled achieved transferred
-------------------------------------------------
PRM_SYS           0       0     100         8
OTHERS            1      50       0         0
tools             2      50       0         0
```

PROCESS LOOP 示例

使用 PROCESS LOOP 语句在所有活动进程间循环。在 PROCESS LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或进程 (proc_) 度量。以下示例使用 PROCESS LOOP 查找某个间隔内 CPU 利用率最高的进程。

```
big_proc_id = 0
big_proc_name = ""
big_proc_cpu = 0
PROCESS LOOP
IF proc_cpu_total_util > big_proc_cpu THEN {
  big_proc_cpu = proc_cpu_total_util
  big_proc_name = proc_proc_name
  big_proc_id = proc_proc_id
}

IF big_proc_cpu > 10 THEN
  YELLOW ALERT "Possible loop, process ", big_proc_name,
    " pid ", big_proc_id|6|0, " using ", big_proc_cpu, " % CPU"
```

SWAP LOOP 示例

使用 SWAP LOOP 在配置的交换区之间循环，并允许 adviser 脚本报告来自“交换空间 (Swap Space)”窗口的信息。在 SWAP LOOP 中，可使用表 (tbl_)、全局 (gbl_) 或“按交换空间”(byswp_) 度量。

以下示例报告有两个交换设备的系统上可用的交换空间。

```
PRINT "--------------------", gbl_stattime, "----------------------"
SWAP LOOP
  PRINT BYSWP_SWAP_SPACE_NAME, " has ", BYSWP_SWAP_SPACE_USED,
    " used out of", BYSWP_SWAP_SPACE_AVAIL, " megabytes "
```

在有一个交换区的系统上，两个间隔打印的输出为：

```
----------------------15:31:59-------------------------
/dev/hd6              has     37 used out of   128 megabytes

----------------------15:32:09-------------------------
/dev/hd6              has     37  used out of  128 megabytes
```

SYSTEM CALL LOOP 示例

使用 SYSTEM CALL LOOP 在系统上的调用之间循环。使用 SYSTEM CALL LOOP 时，adviser 脚本检查“系统调用 (System Call)”窗口中提供的信息。在 SYSTEM CALL LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或系统调用 (syscall_) 度量。

以下示例检查是否有高系统调用率，然后打印最频繁的调用。

```
IF gbl_syscall_rate > 6000 THEN {
  print " "
  print "--- High syscall rate = ", gbl_syscall_rate, " at ",
    gbl_stattime, " ---"
  highestrate = 0

  systemcall loop {
    IF syscall_call_rate > highestrate THEN {
      highestrate = syscall_call_rate
      highestcall = syscall_call_name
    }
  }
  print "    Most frequent syscall was ", highestcall, " at",
    highestrate, " per second"
}
```

输出为：

```
--- High syscall rate =   6750.6 at 15:50:27 ---

    Most frequent syscall was gettimeofday at 6632.90 per second
```

TT LOOP 示例

使用 TT LOOP 在最后一个间隔期间记录的事务信息之间循环。使用此 LOOP 时，adviser 脚本检查“事务跟踪 (Transaction Tracking)”窗口中显示的特定事务信息。在 TT LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或事务跟踪 (tt_) 度量。

以下示例打印系统上每个注册的事务名称的已完成事务数和平均响应时间。

```
PRINT "-------------------", gbl_stattime, "----------------------"
TT LOOP
  PRINT tt_name, " had ", tt_count, " transactions; ",
    "response time ", tt_wall_time_per_tran, " secs"
```

在有四个事务的系统上，两个间隔生成的输出为：

```
-------------------------13:24:44--------------------------

First_Transaction  had   1 transactions; response time   1.000355 secs
Second_Transaction had   1 transactions; response time   2.000221 secs
Third_Transaction  had   1 transactions; response time   3.000231 secs
Fourth_Transaction had   0 transactions; response time   0.000000 secs

-------------------------13:24:54--------------------------

First_Transaction  had   3 transactions; response time   1.000383 secs
Second_Transaction had   1 transactions; response time   2.000216 secs
Third_Transaction  had   0 transactions; response time   0.000000 secs
Fourth_Transaction had   0 transactions; response time   0.000000 secs
```

TTBIN LOOP 示例

使用必须嵌套在 TT LOOP 中的 TTBIN LOOP 在系统上每个活动事务的响应时间二进制文件之间循环。使用此 LOOP 时，adviser 脚本检查“事务图 (Transaction Graph)”窗口中显示的特定事务信息。在 TTBIN LOOP 中，可使用全局 (gbl_)、表 (tbl_)、事务跟踪或事务跟踪二进制文件度量。

以下示例打印在间隔期间有任何已完成事务的每个事务名称的响应时间二进制文件。

```
PRINT “--------------------” , gbl_stattime, “----------------------”
TT LOOP
 IF (tt_count > 0) THEN
 {
  print "Transaction ", tt_name, " had ", tt_count, " transactions"
    lower_bin_limit = 0
    TTBIN LOOP
    {
      IF (ttbin_trans_count > 0) THEN {
        print "  ", ttbin_trans_count, " were between ",
```

```
          lower_bin_limit, " and ", ttbin_upper_range, " seconds"
        lower_bin_limit = ttbin_upper_range


            }
        }
  }
```

在有四个事务的系统上，两个间隔打印的输出为：

```
-------------------------13:46:31--------------------------
Transaction First_Transaction    had      4 transactions
       2 were between     1.00 and   2.000000 seconds
Transaction Second_Transaction   had      1 transactions
       1 were between     2.00 and   3.000000 seconds
Transaction Third_Transaction    had      1 transactions
       1 were between     3.00 and   5.000000 seconds

-------------------------13:46:41--------------------------
Transaction First_Transaction    had      3 transactions
       1 were between     1.00 and   2.000000 seconds
Transaction Second_Transaction   had      1 transactions
       1 were between     2.00 and   3.000000 seconds
Transaction Fourth_Transaction   had      1 transactions
       1 were between     3.00 and   5.000000 seconds
```

TT LOOP ARM 示例

使用 ARM 2.0 后， TT_CLIENT、 TT_INSTANCE 和 TT_UDM 循环可以嵌套在 TT LOOP 中。对于给定事务， TT_CLIENT LOOP 列出相关事务、 TT_INSTANCE LOOP 列出最多 2048 个事务实例，TT_UDM LOOP 则列出用户测量。在 TT LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或事务跟踪度量。

下面的示例显示如何使用多个循环来查看任意给定事务实例的用户测量。

示例 1：查找 SLO 违例

```
# The following example loops through all transactions looking for
# SLO violations, then prints the UDM information for all
# instances:

print "---------------------", GBL_STATTIME, "--------------------"
```

```
tt loop {

  IF tt_slo_count > 0 THEN {
    print "  "
   print "SLO violation count:", tt_slo_count,
      " for transaction:", tt_name, " user:", tt_uname,
      " app:", tt_app_name, " threshold: ", tt_slo_threshold

    tt_instance loop {
      starttime = gbl_stattime - gbl_interval
      IF tt_instance_stop_time > starttime THEN {

        # found a completed instance in the transaction, print info:
        print "instance pid:", tt_instance_proc_id,
          " wall time:", tt_instance_wall_time

        tt_instance_udm loop {
          print "   ", tt_instance_user_measurement_name|44,
            " value= ", tt_instance_user_measurement_value
        }
      }
    }
  }
}
```

以下是一个间隔的输出：

```
----------------------------17:19:03----------------------------
SLO violation count:    1 for transaction:Client_tra00       user:gracel
app:Client_Appl0         threshold:   5.000000
instance pid: 12137 wall time: 13.0407

SLO violation count:    1 for transaction:Server_transaction   user:joe
app:Server_Application   threshold:   5.000000
instance pid: 12137 wall time: 13.0358
   Metric #1 - Type 1 is a COUNTER32            value=           32
   Metric #2 - Type 4 is a GAUGE32              value=           37
   Metric #3 - Type 2 is a COUNTER64            value=     19088743
   Metric #4 - Type 9 is a STRING8              value=     String 8
   Metric #5 - Type 3 is a COUNTER32/DIVISOR32  value=        2.000
```

```
   Metric #6 - Type 8 is a NUMERICID64          value=     19088434
   The last field is always a STRING32          value=            0
instance pid: 12137 wall time:  3.0291
   Metric #1 - Type 1 is a COUNTER32            value=           32
   Metric #2 - Type 4 is a GAUGE32              value=           37
   Metric #3 - Type 2 is a COUNTER64            value=     19088743
   Metric #4 - Type 9 is a STRING8              value=     String 8
   Metric #5 - Type 3 is a COUNTER32/DIVISOR32  value=       21.333
   Metric #6 - Type 8 is a NUMERICID64          value=     19088434
   The last field is always a STRING32          value=            0
instance pid: 12137 wall time:  3.0256
   Metric #1 - Type 1 is a COUNTER32            value=           32
   Metric #2 - Type 4 is a GAUGE32              value=           37
   Metric #3 - Type 2 is a COUNTER64            value=     19088743
   Metric #4 - Type 9 is a STRING8              value=     String 8
   Metric #5 - Type 3 is a COUNTER32/DIVISOR32  value=       21.333
   Metric #6 - Type 8 is a NUMERICID64          value=     19088434
   The last field is always a STRING32          value=            0
instance pid: 12137 wall time:  2.0201
   Metric #1 - Type 1 is a COUNTER32            value=           32
   Metric #2 - Type 4 is a GAUGE32              value=           37
   Metric #3 - Type 2 is a COUNTER64            value=     19088743
   Metric #4 - Type 9 is a STRING8              value=     String 8
   Metric #5 - Type 3 is a COUNTER32/DIVISOR32  value=       21.333
   Metric #6 - Type 8 is a NUMERICID64          value=     19088434
   The last field is always a STRING32          value=            0
instance pid: 12137 wall time:  1.0101
   Metric #1 - Type 1 is a COUNTER32            value=           32
   Metric #2 - Type 4 is a GAUGE32              value=           37
   Metric #3 - Type 2 is a COUNTER64            value=     19088743
   Metric #4 - Type 9 is a STRING8              value=     String 8
   Metric #5 - Type 3 is a COUNTER32/DIVISOR32  value=       21.333
   Metric #6 - Type 8 is a NUMERICID64          value=     19088434
   The last field is always a STRING32          value=            0
```

示例 2：ARM 2.0 语法

```
# The following example prints info for all completed transactions
```

```
# during the interval.

print "-------------------", GBL_STATTIME,  "--------------------"

header_printed = 0
tt loop {

  tt_instance loop {

    starttime = GBL_STATTIME - GBL_INTERVAL

    IF TT_INSTANCE_STOP_TIME > starttime THEN {

      IF header_printed == 0 THEN {
        print "  "
        print "TranID  StartTime                 StopTime",
          "                   "
        header_printed = 1
      }

      print TT_TRAN_ID|6, "  ", TT_INSTANCE_START_TIME, "  ",
        TT_INSTANCE_STOP_TIME
      print "          TranName: ",TT_NAME|40
    }
  }
}
```

以下是一个间隔的输出:

```
-----------------------------17:21:24-----------------------------
TranID  StartTime                 StopTime
     3  Wed Jun  3 17:21:07 1998  Wed Jun  3 17:21:20 1998
          TranName: Client_tra00

     7  Wed Jun  3 17:21:07 1998  Wed Jun  3 17:21:20 1998
          TranName: Server_transaction

     7  Wed Jun  3 17:21:17 1998  Wed Jun  3 17:21:20 1998
          TranName: Server_transaction
```

```
     7  Wed Jun  3 17:21:17 1998  Wed Jun  3 17:21:20 1998
          TranName: Server_transaction

     7  Wed Jun  3 17:21:18 1998  Wed Jun  3 17:21:20 1998
          TranName: Server_transaction

     7  Wed Jun  3 17:21:19 1998  Wed Jun  3 17:21:20 1998
          TranName:  Server_transaction
```

LDOM LOOP 示例

*仅限 HP-UX 11iv2 和更高版本。*使用 LDOM LOOP 语句循环查看系统上位置域中的数据。在 LDOM LOOP 中，可使用全局 (gbl_)、表 (tbl_) 或 ldom (ldom_) 度量。

此示例打印系统上每个位置域的 LDOM 内存使用情况：

```
print " "
PRINT "----------------------", gbl_stattime, "-----------------------"
print "LDOM Phys         Num             Mem      Mem    Mem"
print "  ID   ID Active CPUs Type      Avail     Free Used %"
PRINT "-------------------------------------------------------"
LDOM LOOP
{
 print LDOM_ID, " ", LDOM_PHYS_ID, " ", LDOM_ACTIVE, " ", LDOM_NUM_CPU, " ",
LDOM_MEM_TYPE, " ", LDOM_MEM_AVAIL, " ", LDOM_MEM_FREE, " ", LDOM_MEM_UTIL
}
PRINT "-------------------------------------------------------"
```

在有三个 LDOM 的系统上，两个间隔生成的输出为：

```
----------------------03:30:27-----------------------
LDOM Phys         Num             Mem      Mem    Mem
  ID   ID Active CPUs Type      Avail     Free Used %
-------------------------------------------------------
   0    0      1    4 LOCAL       0mb      0mb    0.0
   1    2      1    4 LOCAL       0mb      0mb    0.0
  na   na      1    0 GLOBAL    8.0gb    5.5gb   30.5
-------------------------------------------------------

----------------------03:30:32-----------------------
LDOM Phys         Num             Mem      Mem    Mem
  ID   ID Active CPUs Type      Avail     Free Used %
```

```
-------------------------------------------------
   0     0       1     4 LOCAL        0mb       0mb     0.0
   1     2       1     4 LOCAL        0mb       0mb     0.0
  na    na       1     0 GLOBAL     8.0gb     5.5gb    30.5
-------------------------------------------------
```

PROC_LDOM LOOP 示例

仅限 HP-UX 11iv2 和更高版本。使用 PROC_LDOM LOOP 在进程可从中获取内存的位置域之间循环。

PROC_LDOM LOOP 必须嵌套在 PROCESS LOOP 中。

以下示例显示可从不同 LDOM 获取内存的 scopeux 进程：

```
print " "

PRINT "----------------------", gbl_stattime,
"-----------------------"

print "进程            LDOM LDOM          RSS        RSS        RSS        RSS
LDOM"

print "Name             ID Type        Total     Shared    Private   Weighted
Mem %"

PRINT
"--------------------------------------------------------------------------------
-----"


PROCESS LOOP

{

if PROC_PROC_NAME == "scopeux" then

{

PROC_LDOM LOOP

{

print PROC_PROC_NAME,"  ",PROC_LDOM_ID," ", PROC_LDOM_TYPE," ",
PROC_LDOM_TOTAL," ", PROC_LDOM_SHARED," ", PROC_LDOM_PRIVATE," ",
PROC_LDOM_WEIGHTED," ", PROC_LDOM_PCT

}

}

}

PRINT
"--------------------------------------------------------------------------------
-----"
```

在有三个为 scopeux 进程过滤的 LDOM 的系统上，两个间隔的输出为：

```
-----------------------------------04:53:50---------------------------------
----
```

```
Process            LDOM LDOM          RSS        RSS        RSS        RSS   LDOM
Name                ID Type        Total     Shared    Private   Weighted  Mem %
-------------------------------------------------------------------------------
----
scopeux              0 LOCAL         0kb        0kb        0kb        0kb    0.0
scopeux              1 LOCAL         0kb        0kb        0kb        0kb    0.0
scopeux             na GLOBAL     31.9mb     23.7mb      8.2mb     13.4mb    0.2
-------------------------------------------------------------------------------
----

-----------------------------------04:53:55---------------------------------
----
Process            LDOM LDOM          RSS        RSS        RSS        RSS   LDOM
Name                ID Type        Total     Shared    Private   Weighted  Mem %
-------------------------------------------------------------------------------
----
scopeux              0 LOCAL         0kb        0kb        0kb        0kb    0.0
scopeux              1 LOCAL         0kb        0kb        0kb        0kb    0.0
scopeux             na GLOBAL     31.9mb     23.7mb      8.2mb     13.4mb    0.2
-------------------------------------------------------------------------------
----
```

PRINT 语句

使用 PRINT 语句将您要收集的数据打印到 stdout （padv 命令控制台）。您可能想用 PRINT 语句记录度量或算出的变量。

语法：

PRINT 打印列表

PRINT 示例

```
PRINT "The Application OTHER has a total CPU of ",

  other:app_cpu_total_util, "%"
```

调用此语句时，将类似以下内容的消息打印到 padv 命令控制台：

The Application OTHER has a total CPU of 89%

SYMPTOM 语句

语法:

SYMPTOM 变量 [TYPE = {CPU, DISK, MEMORY, NETWORK}]

RULE 测量 {>, <, >=, <=, ==, !=} 值 PROB 概率

[RULE 测量 {>, <, >=, <=, ==, !=} 值 PROB 概率]

.

.

.

关键字 SYMPTOM 和 RULE 只能用于 SYMPTOM 语句，不能用于其他语法语句中。SYMPTOM 语句必须是顶层语句，不能嵌套在其他任何语句内。

变量是将成为此症状名称的变量名称。 SYMPTOM 语句中定义的变量名称可以在其他语法语句中使用，但不得在这些语句中更改变量值。

RULE 是 SYMPTOM 语句的选项，不能单独使用。可以根据需要在 SYMPTOM 语句中使用任意多个 RULE 选项。

SYMPTOM 变量在每个间隔按照 RULE 计算。

- “测量”是作为 RULE 一部分计算的变量或度量的名称
- “值”是与测量进行比较的常量、变量或度量
- “概率”是数字常量、变量或度量

值为 true 的所有 SYMPTOM RULE 的概率加起来即为 SYMPTOM 值。该 SYMPTOM 值随即显示在 padv 命令控制台中的消息中。

“测量”和 “值”之间的条件为 true 的所有概率的总和就是症状发生的概率。

SYMPTOM 示例

语法：

```
SYMPTOM CPU_Bottleneck TYPE=CPU
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
SYMPTOM CPU_Level TYPE=CPU
RULE gbl_cpu_sys_mode_util > 40  PROB 25
RULE gbl_cpu_sys_mode_util > 50  PROB 25
RULE gbl_cpu_sys_mode_util > 60  PROB 25
RULE gbl_cpu_sys_mode_util > 70  PROB 50
```

上面定义的总概率 (PROB) 最高的 CPU 症状就是决定 padv 命令控制台中消息的严重级别的症状。

SYMPTOM 示例：全局 CPU 瓶颈

```
SYMPTOM Symp_Global_Cpu_Bottleneck TYPE=CPU
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 75
```

例如，如果该间隔的 CPU 利用率 (gbl_cpu_total_util) 是 93%，运行队列是 2，则前三个规则将都为 true，这样 25 会加到概率中三次。由于第四条规则不是 true，不会添加 75。因此 Symp_Global_Cpu_Bottleneck 变量在该间隔的值为 75（百分比）。

11 在 Windows 上使用性能收集组件

要访问性能收集组件图形用户界面，请在以下文件夹中单击 HP Operations Agent 软件图标：

开始 → 程序 → HP → Operations Agent → 性能收集组件

图 2　　性能收集组件主窗口

本章介绍以下使用性能收集组件图形界面执行的任务：

- 数据类型和类
- 汇总级别
- 要提取或导出的数据范围
- 提取日志文件数据和导出日志文件数据
- 将日志文件数据存档
- 调整日志文件大小

- 扫描日志文件以获取信息
- 分析日志文件
- 配置导出模板
- 配置用户选项
- 配置收集参数
- 配置警报定义
- 检查性能收集组件状态
- 生成性能计数器收集

在开始使用性能收集组件执行涉及提取、导出数据或将数据存档的任务之前，请阅读以下内容。它们分别讨论了如何选择数据类型、汇总级别以及要提取、导出或存档的数据范围。

数据类型和类

可以选择提取或导出以下 scopent 日志文件数据类型：

数据类型	测量类型
global	系统范围信息或全局信息
application	每个用户定义的应用程序中的进程
configuration	系统配置使用情况
process	所选的感兴趣进程
disk	磁盘设备使用情况
filesystem	逻辑磁盘使用情况
logicalsystem	逻辑系统使用情况
cpu	CPU 使用情况
netif	网络接口设备使用情况
transaction	事务跟踪数据

可以根据数据类选择导出 DSI 日志文件数据。每个类都表示一个传入数据源，由记录在一起的一组相关数据项 （度量）组成。

汇总级别

要导出数据，必须在导出日志文件数据时指定所需的汇总级别 - 详细信息、摘要或详细信息和摘要。

- “详细信息”指定从除进程数据类型以外的其他所有数据类型中导出 5 分钟详细信息数据，从进程数据类型中导出 1 分钟详细信息数据。
- “摘要”指定导出 1 小时内汇总的数据。
- “详细信息和摘要”导出的数据量最大。

汇总影响导出数据的大小。例如，每小时摘要数据大概是 5 分钟详细信息数据大小的十分之一。

要提取或导出的数据范围

可以根据记录数据的日期和时间选择提取或导出特定数据。例如，可能想要从特定日期开始的 30 天内每天 （星期一到星期日） 8:00 am 到 8:00 pm 记录的数据。

如果不指定要导出数据的特定范围，将使用默认开始日期提取或导出数据，即日志文件中最晚日期之前的 30 天的日期。如果数据少于 30 天，则为日志文件中最早记录的日期。

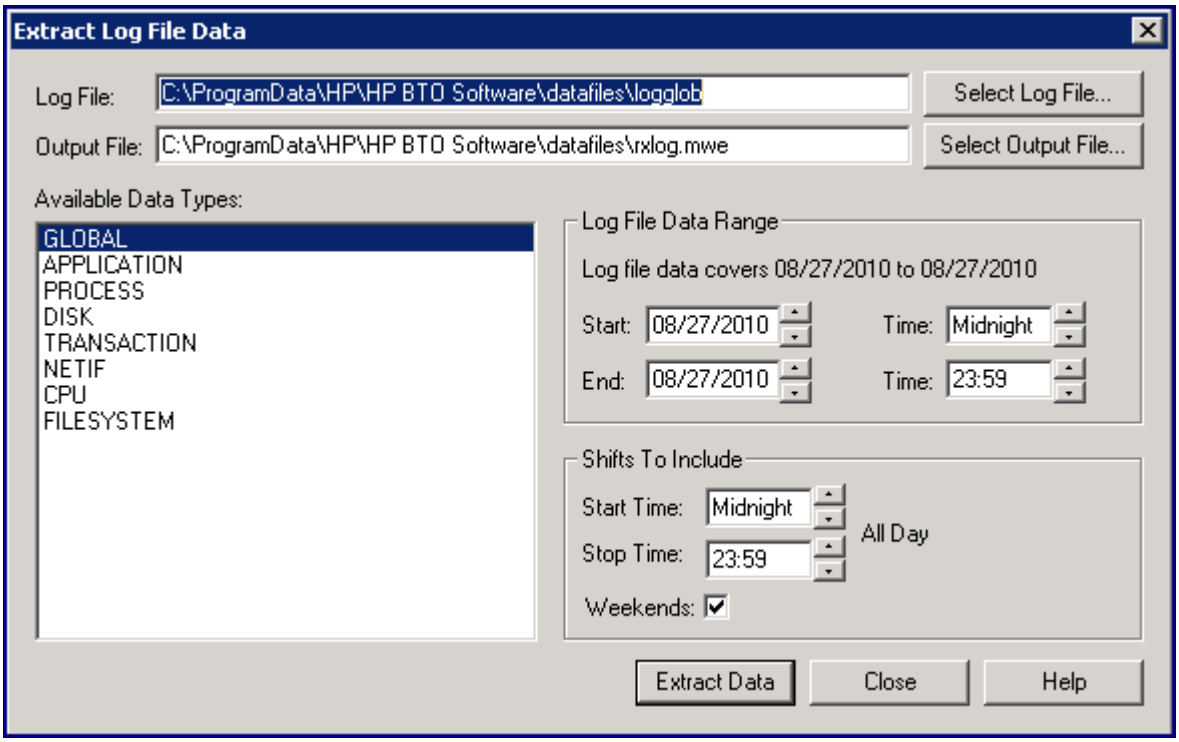
还可以将提取或导出限制为星期一到星期五、符合工作轮换的特定时段 （比如 8:00 am 到 5:00 pm）记录的数据。如果不指定轮换，则默认为提取或导出包括周末在内的每天 24 小时的数据。

提取日志文件数据

数据收集器 scopent 连续收集数据，并将数据记录到原始日志文件。您可以将默认全局日志文件集 logglob 中的特定数据提取到提取的日志文件，供稍后存档或供 HP Performance Manager 等分析程序进行分析。还可以从现有的提取日志文件提取数据。但*不能*提取 DSI 日志文件数据。

指定 logglob 时，将自动打开日志文件集中的所有其他原始日志文件。例如，不需要打开 logproc 日志文件提取进程数据；打开 logglob 即可访问原始日志文件集中的所有数据类型。

图 3　　提取日志文件数据对话框



当启动性能收集组件后显示“提取日志文件数据 (Extract Log File Data)”或“导出日志文件数据 (Export Log File Data)”对话框时，“日志文件 (Log File)”框中将显示默认全局日志文件名称 `logglob` 指示当前活动的日志文件。`logglob` 一直保持活动状态直到关闭性能收集组件或选择其他日志文件为止。选择其他日志文件时，“日志文件 (Log File)”框中显示该文件的名称作为当前活动的日志文件。

提取日志文件数据

要提取日志文件数据，请执行以下步骤：

1. 在主窗口的**日志文件 (Logfile)** 菜单中单击**提取 (Extract)**。将出现“提取日志文件数据 (Extract Log File Data)”对话框，显示当前活动日志文件的名称和当前选择的输出文件的名称。可以轻松地指定其他日志文件和输出文件。
2. 选择要提取的数据类型后，选择要提取的日志文件数据范围和要包括的轮换。单击**提取数据 (Extract Data)** 按钮启动提取进程。

有关提取日志文件数据的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择“**提取日志文件数据 (Extract log file data)**”。

导出日志文件数据

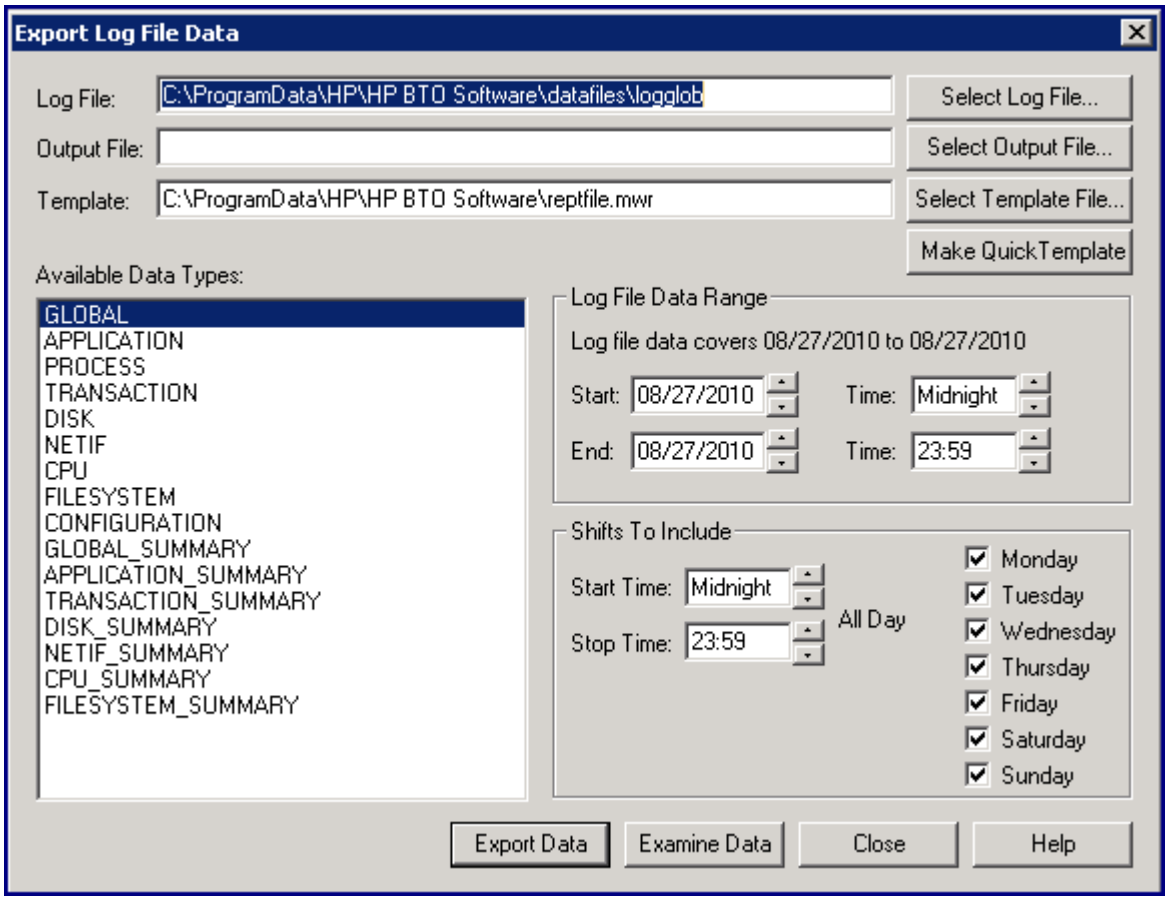
可以从原始或提取的 `scopent` 日志文件或从 DSI 日志文件将特定数据导出到电子表格和其他报表工具可以使用的格式化导出文件中。

export 函数*不*从日志文件中删除任何数据。

以下部分讨论：

- 可以为导出数据指定的文件属性
- 可用于简化导出任务的导出模板
- 默认导出文件
- 导出任务概述

图 4　　导出日志文件数据对话框



文件属性

可以为导出数据分配各种文件属性，包括文件格式、表示缺少的数据的值、字段分隔符、列标题、摘要间隔的分钟数、布局、数据类型以及要包括在导出文件中的特定度量。这些属性可以保存在导出模板文件中，也可以直接使用“制作快速模板 (Make Quick Template)”对话框指定。（有关详细信息，请参见第 220 页的制作快速导出模板。）

文件格式

导出文件的输出格式可以是 ASCII、数据文件、二进制或 WK1 （电子表格）：

- ASCII 格式是可打印的字符数据，适于打印报告或由用户编写的程序或批处理文件作后处理。
- 数据文件格式与 ASCII 类似，但所有非数字项两边加了双引号 (" ")。数据文件格式对于将数据传输到大多数电子表格和图形包很有用。
- 二进制格式不可打印。这种格式占用的空间更小，因为数字值表示为二进制整数。它是最适合输入到用户编写的分析程序的格式，因为它需要的转换最少，度量准确性最高。
- WK1 （电子表格）格式与 Microsoft Excel 和其他电子表格、数据库和图形产品兼容。

缺少值

导出文件可以包含代替源日志文件中缺少的数据的数据值。当度量对某些 scopent 收集器版本不可用时，会发生缺少数据的情况。另外，应用程序、磁盘和事务的多布局导出格式会在输出记录中为每个应用程序、磁盘或事务保留空间。如果在间隔内没有为特定条目记录数据，其数据为“缺少”。

字段分隔符

在 ASCII 和数据文件格式导出文件中的两个度量之间可以插入字段分隔符。分隔符可以是可打印或不可打印的 （比如，制表符）。

默认分隔符是空格，但很多程序需要用逗号作为字段分隔符。

摘要分钟数

可以指定每个摘要间隔的分钟数。分钟数的范围是 5 到 1440 分钟 （一天）。

标题

导出文件中可以包括列标题。文件中的第一条记录是导出的数据（WK1 格式文件除外）。但是，如果在文件中包括标题，ASCII 和数据文件格式的文件有导出文件标题（如已指定）及写在文件中第一条数据记录*前面*的每列度量的列标题。二进制格式文件中的标题写在文件中第一条记录的前面，并包括度量描述。

WK1 文件始终包含标题。

多布局

可以为应用程序或磁盘等多实例数据类型指定多个布局 （每个记录输出）。

单布局为时间间隔期间的每个活动应用程序或磁盘写入一条记录。多布局针对每个时间间隔写入一条记录，该记录的一部分为每个配置的应用程序或磁盘保留。

导出文件标题

可以为导出文件指定标题。标题可以包括文字字符串和替换关键字。导出标题字符串中的以下项可以替换：

`!date`	创建导出文件的日期
`!time`	创建导出文件的时间
`!logfile`	从其获取数据的日志文件的名称
`!collector`	收集器程序 （`scopent` 或 `dsilog`）的名称和版本
`!class`	所请求数据的类型
`!system_id`	收集 `scopent` 原始或提取的日志文件数据的系统标识符 （对 DSI 日志文件数据无效）

例如，可以输入以下字符串：

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

该字符串将生成类似以下内容的标题：

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

导出文件模板

导出任务使用为导出文件定义文件属性的导出模板。默认文件属性来自于 <rpmtools>\data\reptfile.mwr 文件，它指定：

- ASCII 文件格式
- 0 （零）表示缺少值
- 字段分隔符是空格
- 60 分钟摘要
- 包括标题
- 导出中包括给定数据类型或类的建议度量集

可以指定其他导出模板文件，或制定特定文件属性规范 （请参见第 220 页的制作快速导出模板）。

还可以使用“配置导出模板 (Configure Export Templates)”对话框创建自定义的导出模板（请参见第 222 页的配置导出模板）。

默认导出文件

如果不指定输出文件包括导出数据，导出任务会根据指定的数据类型和汇总级别在 < *磁盘驱动器* >:\Program Files\HP\HP BTO Software\data\datafiles 目录中创建默认输出文件。

scopent 数据

导出 scopent 日志文件数据时，向导出文件分配以下默认文件名。

xfrdGLOBAL.ext	全局详细信息数据
xfrsGLOBAL.ext	全局摘要数据
xfrdAPPLICATION.ext	应用程序详细信息数据
xfrsAPPLICATION.ext	应用程序摘要数据
xfrdPROCESS.ext	进程详细信息数据
xfrdDISK.ext	磁盘设备详细信息数据
xfrsDISK.ext	磁盘设备摘要数据
xfrdCPU.ext	CPU 详细信息数据
xfrsCPU.ext	CPU 摘要数据
xfrdFILESYSTEM.ext	文件系统详细信息数据
xfrsFILESYSTEM.ext	文件系统摘要数据
xfrdNETIF.ext	netif 详细信息数据
xfrsNETIF.ext	netif 摘要数据
xfrdTRANSACTION.ext	事务跟踪详细信息数据
xfrsTRANSACTION.ext	事务跟踪摘要数据
xfrdCONFIGURATION.ext	配置详细信息数据
xfrdLOGICAL.ext	逻辑系统详细信息数据文件
xfrsLOGICAL.ext	逻辑系统摘要数据文件

完成导出任务后，可以在“导出日志文件数据 (Export Log File Data)”对话框中单击**检查数据 (Examine Data)** 按钮查看输出导出文件的内容。

默认文件名根据数据类型名称创建。前缀是 xfrd 或 xfrs 则取决于数据是详细信息数据还是摘要数据。扩展名 (.ext) 是导出模板文件中指定的文件格式：asc（ASCII）、bin（二进制）、dat（数据文件）或 wk1（电子表格）。

例如：

xfrdNETIF.wk1 为电子表格格式，包含 NETIF 数据类型的详细数据。

xfrsAPPLICATION.asc 为 ASCII 格式，包含应用程序数据类型的摘要数据。

DSI 数据

导出 DSI 日志文件数据时，默认文件名根据类名创建。前缀是 xfrd 或 xfrs 则取决于数据是详细信息数据还是摘要数据。扩展名是导出模板文件中指定的文件格式：asc （ASCII）、 dat （数据文件）或 wk1 （电子表格）。

例如：

xfrdACCTG.wk1 为电子表格格式，包含 ACCTG 类的详细数据。

xfrsPERSONL.asc 为 ASCII 格式，包含 PERSONL 类的摘要数据。

导出日志文件数据

要导出日志文件数据，请执行以下步骤：

1 在主窗口的“日志文件 (Logfile)”菜单中单击**导出 (Export)**。将出现“导出日志文件数据 (Export Log File Data)”对话框，显示当前活动日志文件的名称和当前选择的导出模板文件的名称。可以指定其他日志文件和导出模板文件。

如果指定输出文件，所有选择的数据类型或类的数据将放在相同文件中。如果不指定输出文件，将根据指定的数据类型或类和汇总级别创建默认输出文件。（请参见第 217 页的默认导出文件。）

2 选择一个或多个要导出的数据类型或类后，选择要导出的日志文件数据范围和要包括的轮换。

3 要覆盖默认导出模板中指定的文件属性，或选择要在导出文件中包括的特定度量，请单击**制作快速模板 (Make Quick Template)** 按钮。

4 完成制作快速模板的步骤或为导出文件选择特定度量后，返回到 “导出日志文件数据 (Export Log File Data)”对话框。单击**导出数据 (Export Data)** 按钮启动导出进程。

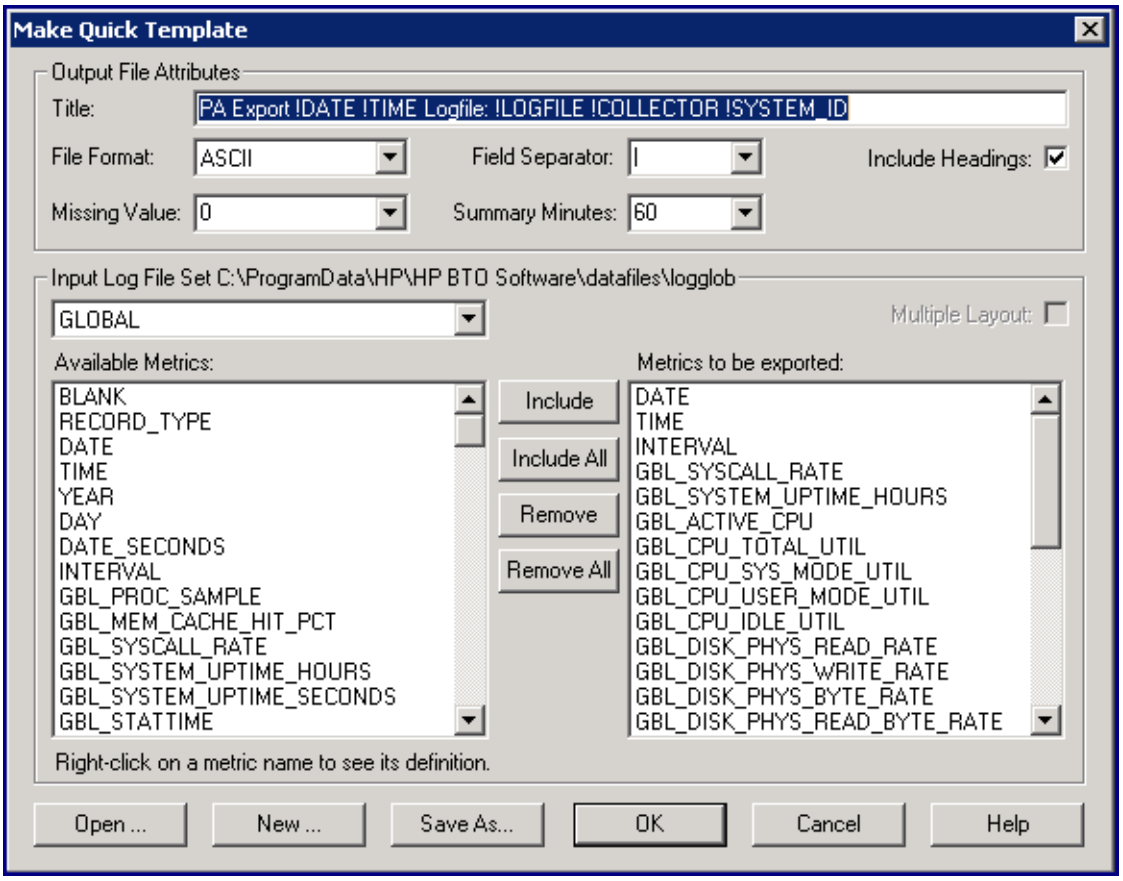
5 完成导出后，可以单击**检查数据 (Examine Data)** 按钮查看导出文件的内容。

有关导出日志文件数据的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择 “**导出日志文件数据 (Export log file data)**”。

制作快速导出模板

使用 “导出日志文件数据 (Export Log File Data)”对话框中的 “制作快速模板 (Make Quick Template)”功能，可以选择要在导出文件中包括的特定度量和更改选择导出的导出模板中指定的任何文件属性和度量。

图 5　　制作快速模板对话框



要制作快速模板，请执行以下步骤：

1. 在“导出日志文件数据 (Export Log File Data)”对话框中单击**制作快速模板 (Make Quick Template)** 按钮。将出现“制作快速模板 (Make Quick Template)”对话框，显示选择导出的导出文件的标题。
2. “输出文件属性 (Output File Attributes)”下面的框根据选择导出的导出模板中设置的文件属性显示导出文件的当前设置。可以修改其中的任何设置。
3. 要使用其他导出模板文件，请单击**打开 (Open)** 按钮。
4. 还可以单击**新建 (New)** 按钮清除“制作快速模板 (Make Quick Template)”对话框中的所有现有设置并创建全新的导出模板文件。

选择要导出的度量

- 选择要导出的度量的数据类型或类之后，可以选择要在导出中包括的特定度量。每个数据类型或类都有自己的度量集，在“可用度量 (Available Metrics)”下面列出。“要导出的度量 (Metrics to be exported)”下面列出的度量是由当前导出模板指定的要在导出中包括的度量。可以使用该列表、从列表中删除度量或选择其他要在导出中包括的可用度量。
- 如果选择 `application`、`disk`、`cpu`、`filesystem`、`netif` 或 `transaction` 数据类型，请选中**多布局 (Multiple Layout)** 复选框生成多个布局（每个记录输出），或保持未选中状态生成单个布局。

保存选择

进行选择之后，可以：

- 使用对文件属性和要导出的度量列表所做的选择继续导出过程，但“不”保存任何模板文件的更改。

- 将选择保存到新导出模板文件，供将来导出使用。原始导出模板文件将保持不变。

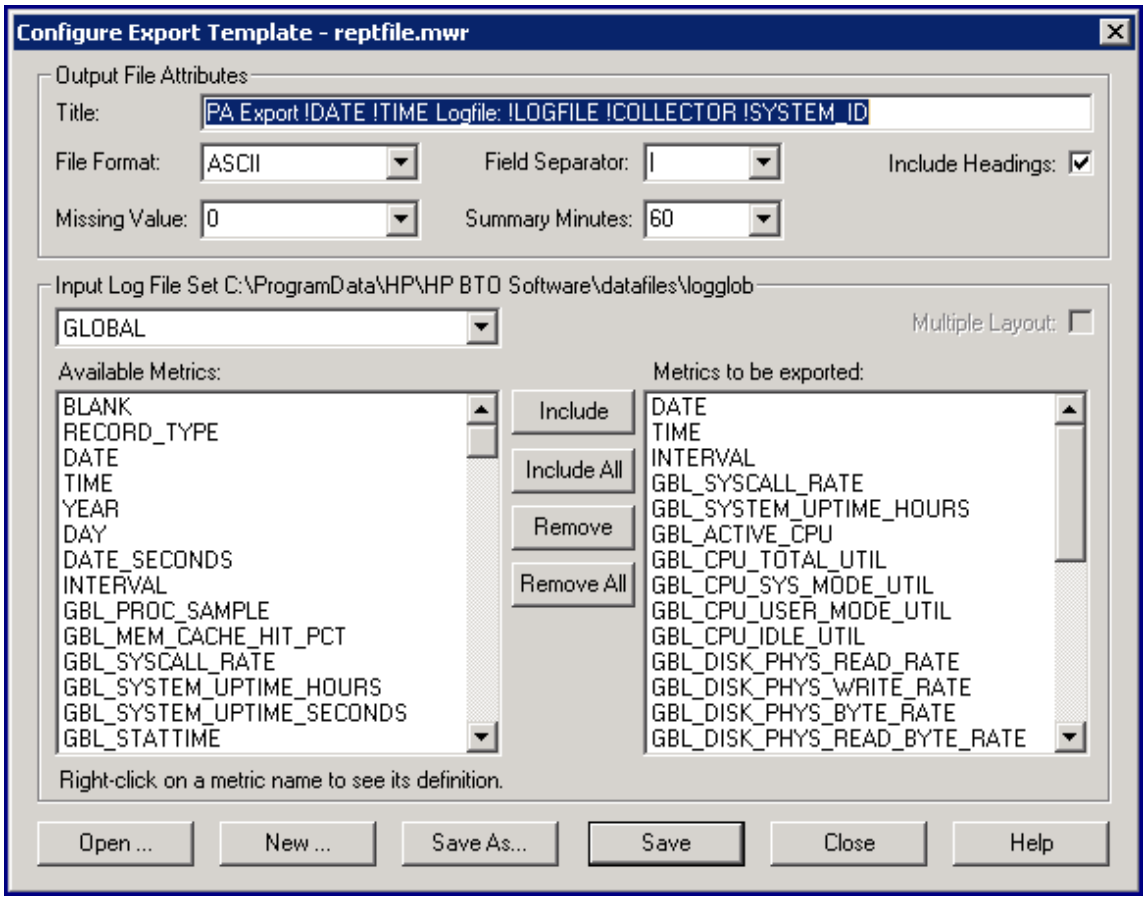
如果将选择保存到新导出模板文件，指定新模板文件名时必须包括 .mwr 文件扩展名。例如，mytmplte.mwr

有关制作快速导出模板的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择 “**制作快速模板 (Make Quick Template)**”。

配置导出模板

使用 “配置 (Configure)”菜单的**导出模板 (Export Templates)** 命令，可以自定义现有导出模板文件或新建导出模板文件。使用 “配置导出模板 (Configure Export Template)”对话框可以选择要在模板中包括的新文件属性和特定度量。

图 6　　配置导出模板对话框



要配置导出模板，请执行以下步骤：

1　在主窗口的 “配置 (Configure)” 菜单中单击**导出模板 (Export Templates)**。将出现 “配置导出模板 (Configure Export Template)” 对话框，对话框标题中将显示当前打开的导出模板文件的名称。要编辑其他导出模板文件，请单击**打开 (Open)** 按钮。

2　“输出文件属性 (Output File Attributes)” 下面的框根据要配置的导出模板中设置的文件属性显示导出文件的当前设置。可以修改其中的任何设置。

3 还可以单击**新建 (New)** 按钮清除 “配置导出模板 (Configure Export Template)” 对话框中的所有现有设置并创建全新的导出模板文件。

选择要导出的度量

- 选择要导出的度量的数据类型或类之后，可以选择要在导出中包括的特定度量。每个数据类型或类都有自己的度量集，在 “可用度量 (Available Metrics)” 下面列出。“要导出的度量 (Metrics to be exported)” 下面列出的度量是由当前导出模板指定的要在导出中包括的度量。可以使用该列表、从列表中删除度量或选择其他要在导出中包括的可用度量。
- 如果选择 `application`、`disk`、`cpu`、`filesystem`、`netif` 或 `transaction` 数据类型，请选中**多布局 (Multiple Layout)** 复选框生成多个布局 （每个记录输出），或保持未选中状态生成单个布局。

保存选择

有三种选择保存编辑后的模板：

- 将更改保存到当前模板文件。
- 将更改保存到新模板文件，这种情况下原始模板文件将保持不变。
- 取消更改以避免更改任何模板文件。

如果将选择保存到新导出模板文件，指定新文件名时必须包括 .mwr 文件扩展名。

如果在更改模板文件*后*未保存更改就单击**关闭 (Close)** 按钮，会提示您取消或保存更改。单击**取消 (Cancel)** 按钮返回 “配置导出模板 (Configure Export Template)” 对话框。

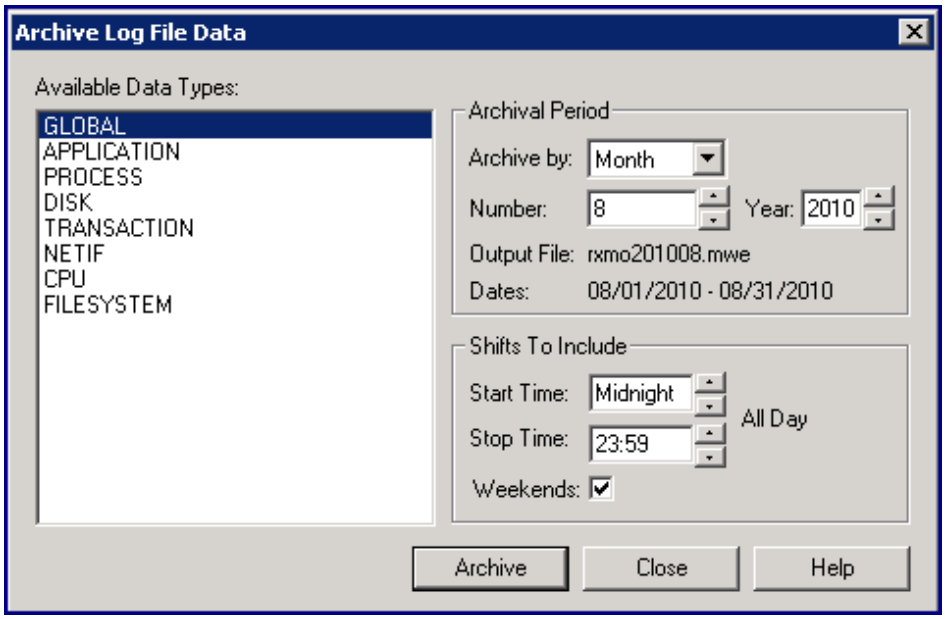
有关配置导出模板文件的逐步说明，请从 “帮助 (Help)” 菜单选择**帮助主题 (Help Topics)**，选择 “**如何…？ (How Do I…?)**”，再选择 “**配置导出模板文件 (Configure an export template file)**”。

将日志文件数据存档

使用 “日志文件 (Logfile)” 菜单的**存档 (Archive)** 命令，可以提取 scopent 日志文件数据的所选部分进行存档和用于将来数据分析。

若用于存档，只能从原始日志文件提取数据。
提取的数据自动放在 < *磁盘驱动器* >:Program Files\HP\HP BTO Software \data\datafiles 目录的存档输出文件中，其名称应反映所选的存档周期。这些文件可以复制到磁带进行脱机存储，然后删除以释放磁盘空间。

图 7　　将日志文件数据存档对话框



存档周期

可以根据特定的按周、按月或按年周期记录的数据选择要提取的数据。

还可以将提取限制为包括或不包括周末（星期六和星期日）、符合工作轮换的特定时段记录的数据。如果不指定轮换，则默认为提取每天 24 小时的数据。默认情况下包括周末。

追加存档数据

存档操作提供一项特殊功能。根据选择的存档周期 （按周、按月或按年），自动检查该存档周期上一个输出文件，确认它是否包括截止最后一天提取的数据。如果不包括，则将数据追加到文件完成上一个存档周期的提取。

例如，您在 1999 年 5 月 7 日开始执行 1999 年 5 月的月度数据存档，创建了名为 rxmo199905.mwe 的输出文件，该文件包含 5 月 1 日到当天 （5 月 7 日）的数据。

在 1999 年 6 月 4 日，调用了另一个按月存档周期。在为 6 月份创建 rxmo199906.mwe 文件之前，检查上个月的 rxmo199905.mwe 文件。当发现文件未完成时，将数据追加到该文件以完成到 1999 年 5 月 31 日为止的数据提取。然后，创建 rxmo199906.mwe 文件保存 1999 年 6 月 1 日到当天（6 月 4 日）的数据。

只要每月 （每周、每年）至少调用一次另一个按月 （按周、按年）存档周期，此功能都会先完成每个存档周期的文件，再创建下一个存档周期的文件。

存档提示

以下是有关将日志文件数据存档的一些建议：

- 每月一次指定按月存档周期并将原始日志文件中的所有详细信息数据提取到单个提取的日志文件。
- 如果系统每月生成超过 64 MB 的数据，可能需要按周提取数据，或从提取中取消进程详细信息数据。
- 按年提取全局摘要和应用程序摘要数据，可最大程序减少所需磁盘空间。然后使用此存档文件进行长期趋势分析。

要将日志文件数据存档，请执行以下步骤：

1 在主窗口的 “日志文件 (Logfile)” 菜单中单击**存档 (Archive)**。将出现 “将日志文件数据存档 (Archive Log File Data)” 对话框。要存档的数据将从原始日志文件集中提取出来。

2 从 “可用数据类型 (Available Data Types)” 列表选择要存档的数据类型后，指定存档周期（周、月或年），并指定任何要包括的轮换。

3 单击**存档 (Archive)** 按钮启动存档进程。

有关将日志文件数据存档的逐步说明，请从 “帮助 (Help)” 菜单选择**帮助主题 (Help Topics)**，选择 “**如何…？ (How Do I…?)**”，再选择 “**将日志文件数据存档 (Archive log file data)**”。

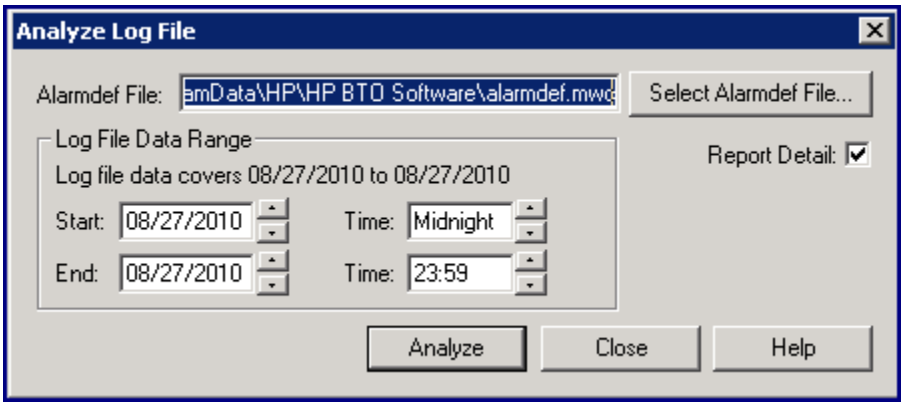
分析日志文件

使用 “日志文件 (Logfile)” 菜单的**分析日志文件 (Analyze Log File)** 命令，可以对照警报定义文件中的警报定义对原始日志文件集中的数据进行分析，并报告任何生成的警报活动。

可通过此任务评估警报定义是否与系统上收集的历史数据很好地匹配。它还可用于确定警报定义是否将在分析系统上生成过多或过少的警报。

要分析的原始日志文件引自性能收集组件默认数据源 SCOPE。要分析其他日志文件，请在警报定义文件中放入 USE 语句，指定引用该日志文件的数据源的名称。

图 8　　分析日志文件对话框



要分析的数据范围

可以分析在特定时间段收集的日志文件数据。如果不指定要分析的数据的特定范围，将使用默认开始日期分析数据，即日志文件中最晚日期之前的 30 天的日期，如果数据少于 30 天，则为日志文件中最早记录的日期。

分析报告

在执行此任务时，它生成可打印的报告，列出警报事件和警报摘要。（只有在 “分析日志文件 (Analyze Log File)” 对话框中选中了 “报告详细信息 (Report Detail)” 框时，才会列出警报事件。）

- 警报事件包括警报 START、END 和 REPEAT 状态以及关联的 PRINT 语句中的任何文本。如果 PRINT 语句中的任何文本作为条件列出 （在 IF 语句中）并为 true，则还包括该文本。不执行 EXEC 语句，但列出 EXEC 语句，以便您可以看到已执行的操作。
- 警报摘要显示发生的警报数计数和每个警报处于活动状态 (on) 的时间。计数包括 alarm starts 和 repeats，但不包括 alarm ends。

要分析日志文件，请执行以下步骤：

1. 在主窗口的“日志文件 (Logfile)”菜单中单击**分析 (Analyze)**。将出现**分析日志文件 (Analyze Log File)** 对话框，显示当前选择的警报定义文件的名称。
2. 要使用其他警报定义文件，请单击**选择 alarmdef 文件 (Select Alarmdef File)** 按钮。
3. 选择要分析的日志文件数据的范围。
4. 如果要在分析报告中包括警报事件，请选中**报告详细信息 (Report Detail)** 框。否则，只生成警报摘要。
5. 单击**分析 (Analyze)** 按钮启动分析。分析结果将显示在 MeasureWare Agent 报告查看器窗口中。

有关分析日志文件的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…? (How Do I…?)**”，再选择 “**分析日志文件 (Analyze a log file)**”。

扫描日志文件

使用 “日志文件 (Logfile)”菜单的**扫描日志文件 (Scan Log File)** 命令，可以扫描 `scopent` 日志文件并创建有关其内容的报告。可以扫描整个日志文件，也可以扫描日志文件中特定时间段收集的数据部分。

扫描所生成的报告由 12 个部分组成。报告的以下四个部分总是打印。

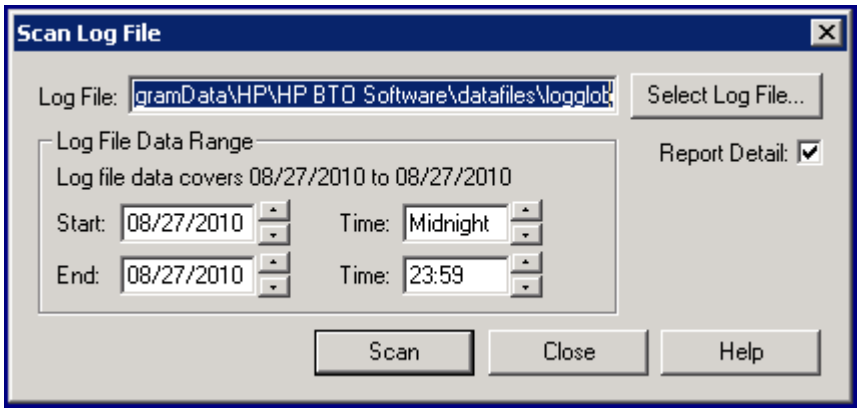
- 进程摘要报告
- 收集器覆盖时间摘要
- 日志文件内容摘要
- 日志文件空余空间摘要

报告的以下八个部分仅在选中了 “扫描日志文件 (Scan Log File)”对话框中的**报告详细信息 (Report Detail)** 时打印。

- 初始 `parm` 文件全局信息和系统配置信息
- 初始 `parm` 文件应用程序定义
- `parm` 文件全局更改

- parm 文件应用程序 / 更改通知
- 收集器关闭时间通知
- 特定于应用程序的摘要报告

图 9　　扫描日志文件对话框



要扫描日志文件，请执行以下步骤：

1　在主窗口的“日志文件 (Logfile)”菜单中单击**扫描日志文件 (Scan Log File)**。将出现“扫描日志文件 (Scan Log File)”对话框，突出显示当前打开的日志文件的名称。

2　要扫描其他日志文件，请单击**选择日志文件 (Select Log File)** 按钮。

3　要扫描特定时间段记录的数据，请在“日志文件数据范围 (Log File Data Range)”下面选择或输入该时间段的开始和结束日期和时间。

4　如果需要完整的扫描报告，请选中**报告详细信息 (Report Detail)** 框。否则，只生成一部分报告。

5　要启动扫描进程，请单击**扫描 (Scan)** 按钮。扫描结果显示在性能收集组件报告查看器窗口中。

有关扫描日志文件的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…? (How Do I…?)**”，再选择 “**扫描日志文件 (Scan a log file)**”。

调整日志文件大小

使用 “日志文件 (Logfile)” 菜单的**调整日志文件大小 (Resize Log File)** 命令，可以更改原始 scopent 日志文件的大小。可以使用兆字节大小调整特定文件的大小或使用文件应保留数据的天数调整大小。

原始日志文件的最大大小在 parm 文件的 size 参数中指定。调整日志文件大小使您能够更好地控制回滚日志文件数据的频率。

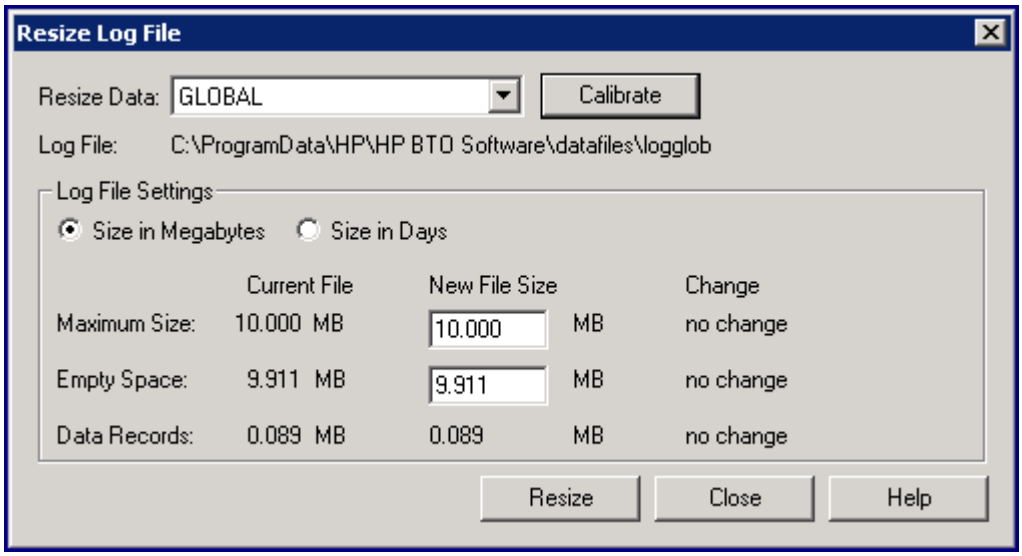
可以选择调整以下任何类型的数据大小：global、application、process、device 或 transaction，它们分别对应原始日志文件 logglob、logappl、logproc、logdev 和 logtran。然后选择如何执行大小调整：以 MB 为单位或按天数。根据选择的调整大小的类型，“调整日志文件大小 (Resize Log File)” 对话框的 “日志文件设置 (Log File Settings)” 框将显示以下项：

- “最大大小 (Maximum Size)” 字段显示当前文件大小、新文件大小和调整大小操作所做的更改。
- “空余空间 (Empty Space)” 字段显示当前文件的空间量、完成调整大小进程后文件中所需的空间量和所做的更改。这些值用于确定在调整大小进程中是否必须删除当前日志文件的一些数据。
- “数据记录 (Data Records)” 字段显示当前日志文件中包括的数据记录量，以及调整大小后的日志文件中的新数据记录量。

日志文件大小以 MB 为单位维护。通常，以天为单位指定大小比以 MB 为单位指定更方便。如果选择 “以天为单位的大小 (Size in Days)”，对话框中的所有单位都更改为 “天”。从 MB 到天的换算基于每种数据类型的 “每天兆字节” 值。最初，此换算使用预计值。

单击**校准 (Calibrate)** 按钮可以获得更准确的值。校准功能实际测量现有日志文件，以获取更准确的每天兆字节值。如果以 MB 为单位指定大小，则不需要换算，也不需要使用校准功能。

图 10　　调整日志文件大小对话框



在调整日志文件大小之前，*必须*停止 scopent 收集器。要停止 scopent，请执行第 37 页的停止和重新启动数据收集中的步骤。

在停止 scopent 前尝试调整日志文件大小不会影响现有日志文件。要调整日志文件大小，请执行以下步骤：

1　停止 scopent 后，在主窗口的“日志文件 (Logfile)”菜单中选择**调整日志文件大小 (Resize Log File)**，以显示“调整日志文件大小 (Resize Log File)”对话框。

2　在“调整数据大小 (Resize Data)”框中，选择要调整大小的数据类型：**global**、**application**、**process**、**device** 或 **transaction**。

3　选择**以 MB 为单位的大小 (Size in Megabytes)** 或**以天为单位的大小 (Size in Days)**。根据您的选择，将显示“当前文件 (Current File)”和“新文件大小 (New File Size)”。

4　要基于显示的“新文件大小 (New File Size)”执行大小调整操作，请单击**调整大小 (Resize)** 按钮启动调整大小进程。

要更准确地估计以天为单位调整大小时要在日志文件中增加多少额外空间，请执行以下步骤：

1　单击**校准 (Calibrate)** 按钮。不久即显示最近 30 天内在文件中记录的数据记录的实际数量和大小。

2 单击**关闭 (Close)** 按钮返回 “调整日志文件大小 (Resize Log File)” 对话框。如果以天为单位调整大小，将显示基于校准的新 “当前文件 (Current File)” 和 “新文件大小 (New File Size)” 值。

3 单击**调整大小 (Resize)** 按钮调整日志文件大小。

4 在执行另一个任务之前，使用第 37 页的停止和重新启动数据收集中的步骤启动 `scopent`。

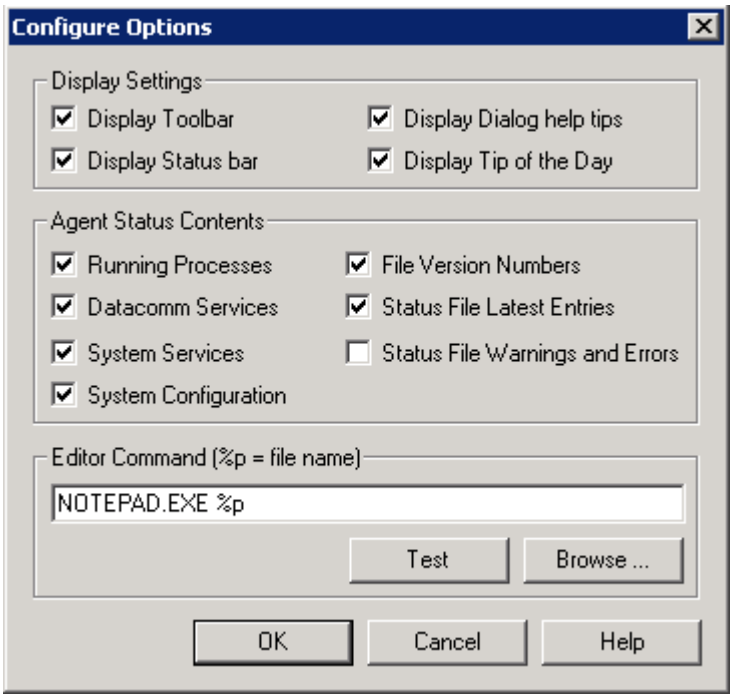
有关调整日志文件大小的逐步说明，请从 “帮助 (Help)” 菜单选择**帮助主题 (Help Topics)**，选择 “**如何…？ (How Do I…?)**”，再选择 “**调整日志文件大小 (Resize a log file)**”。

配置用户选项

使用 “配置 (Configure)” 菜单的**选项 (Options)** 命令，可以控制工具栏、状态栏、对话框帮助提示的显示以及使用性能收集组件时主窗口上的当天提示的显示。

还可以使用**选项 (Options)** 命令配置修改收集参数和警报定义文件的编辑器或字处理器，并选择从 “代理程序 (Agent)” 菜单选择**状态命令 (Status command)** 时要查看的状态类型信息。

图 11　　配置选项对话框



要配置用户选项，请执行以下步骤：

1. 在主窗口的“配置 (Configure)”菜单中，单击**选项 (Options)** 命令显示“配置选项 (Configure Options)”对话框。
2. 选中**显示工具栏 (Display Toolbar)** 复选框，在主窗口中显示工具栏。
3. 选中**显示状态栏 (Display Status Bar)** 复选框，在每个对话框底部和主窗口中显示当前状态。
4. 选中**显示对话框帮助提示 (Display Dialog Help Tips)** 复选框，在对话框中显示帮助提示。
5. 选中“显示当天提示 (Display Tip of the Day)”复选框，在打开性能收集组件主窗口时显示当天的提示。

要进行配置，请执行以下步骤：

1 在“编辑器命令 (Editor Command)”框中，使用 .exe 文件扩展名输入编辑器的目录路径和文件名 （例如， C:\MSOffice\winword\winword.exe）。

2 单击**浏览 (Browse)** 按钮，显示可从中选择编辑器的“选择文本编辑器 (Select a Text Editor)”对话框。

3 单击**测试 (Test)** 按钮，确保选择的编辑器已配置，然后单击**确定 (OK)**。

要配置查看哪些代理程序状态信息，请在 “代理程序状态内容 (Agent Status Contents)”下面选择一个或多个选项框，然后单击**确定 (OK)**。

有关配置用户选项的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…? (How Do I…?)**”，再选择 “**配置用户选项 (Configure user options)**”。

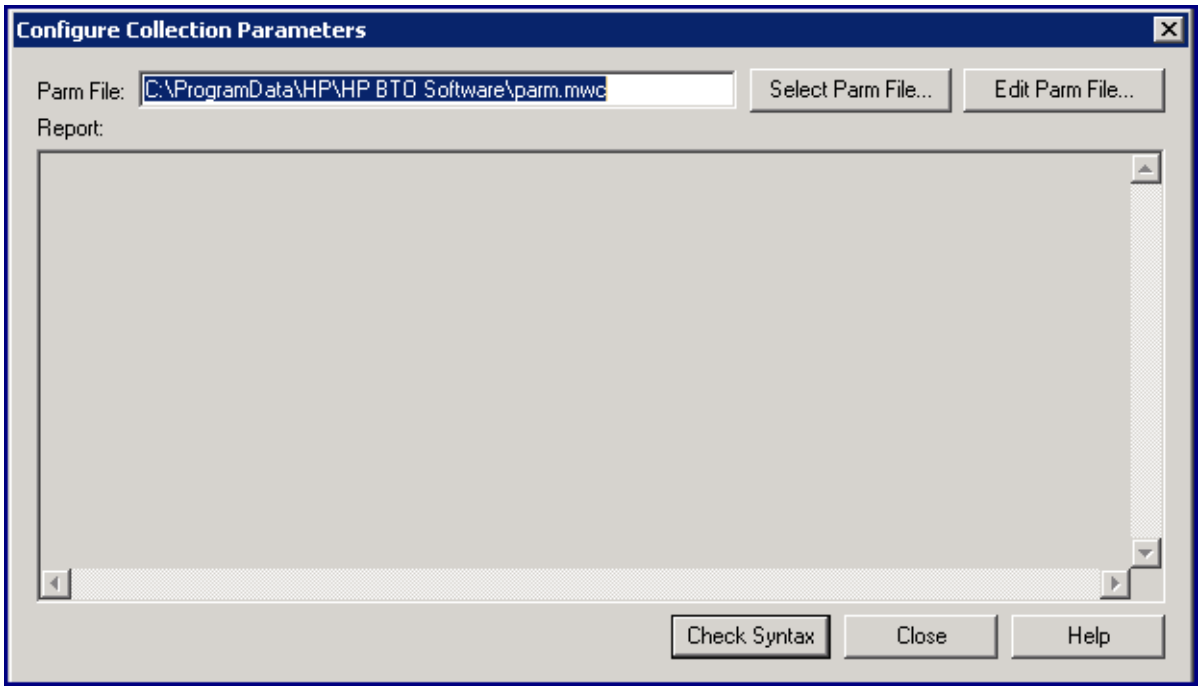
配置收集参数

使用“配置 (Configure)”菜单的**收集参数 (Collection Parameters)** 命令，可以检查 scopent 用于数据收集的 parm 文件的语法。可以检查 parm 文件设置是否有语法错误和警告，查看定义应用程序可用的空间量。

如果发现任何警告或错误并要更正它们，或如果要更改或添加 parm 文件参数，可以使用编辑 parm 文件 (Edit Parm File) 功能轻松地修改 parm 文件。

有关 parm 文件及其参数的详细描述，请参见第 15 页的管理数据收集。

图 12　　配置收集参数对话框

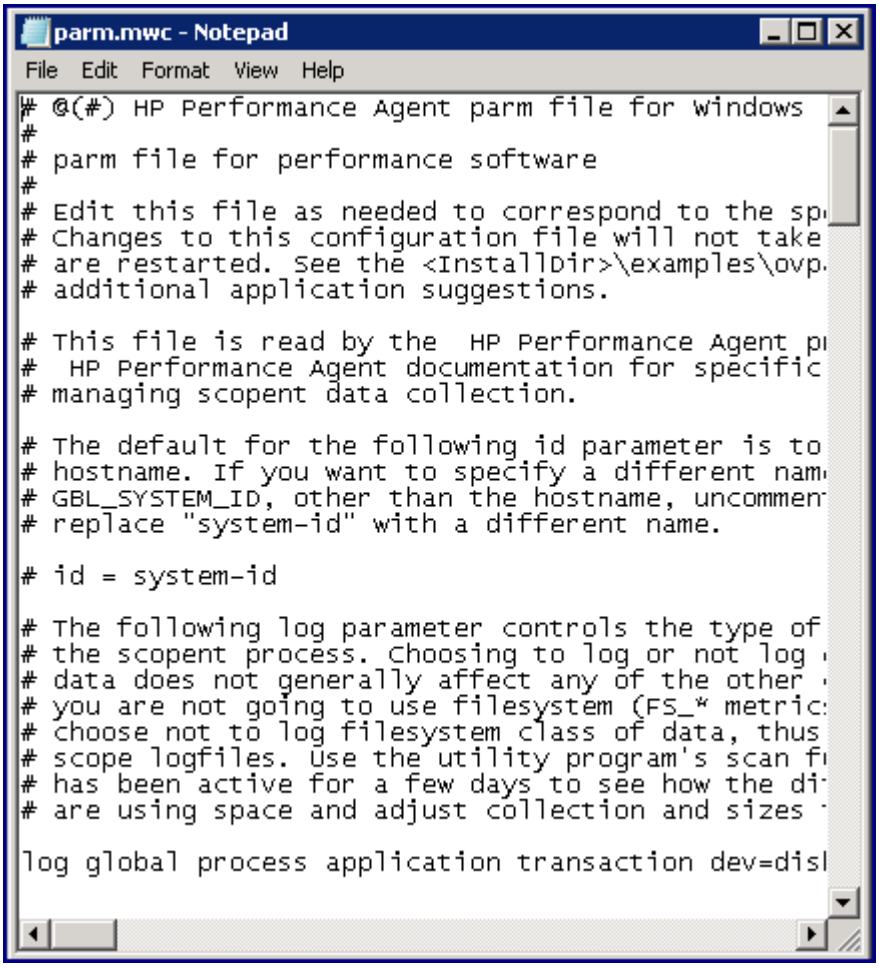


要检查语法，请执行以下步骤：

1. 在性能收集组件主窗口的“配置 (Configuration)”菜单中单击**收集参数 (Collection Parameters)**。将出现“配置收集参数 (Configure Collection Parameters)”对话框，在“parm 文件 (Parm File)”框中显示当前打开的 parm.mwc 文件的名称。
2. 要检查其他 parm 文件，请单击**选择 parm 文件 (Select Parm File)** 按钮。
3. 要检查 parm 文件语法，请单击**检查语法 (Check Syntax)** 按钮。任何生成的警告或错误都显示在性能收集组件报告查看器窗口中。
4. 要修改 parm 文件的任何部分，请单击**编辑 parm 文件 (Edit Parm File)** 按钮。可以将“编辑 parm 文件 (Edit Parm File)”和 “配置收集参数 (Configure Collection Parameters)”对话框放在屏幕上，以便可以同时使用它们。

有关检查 parm 文件语法的逐步说明，请从 “帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择**“如何…？ (How Do I…?)”** ，再选择 **“检查收集参数文件的语法 (Check the syntax of a collection parameters file)”**。

图 13　　修改收集参数文件窗口



要修改 parm 文件，请执行以下步骤：

1　在性能收集组件主窗口的“配置　(Configuration)”菜单中，单击**收集参数　(Collection Parameters)**，再在“配置收集参数 (Configure Collection Parameters)”对话框中单击**编辑 parm 文件 (Edit Parm File)** 按钮。当前打开的 parm 文件的内容将显示在之前指定的编辑器或字处理器中。（要指定编辑器或字处理器，请参见第 233 页的配置用户选项。）

2 在对文件进行任何更改之前，请参见第 17 页的 parm 文件了解要遵循的一些规则和约定。

3 根据需要修改文件，并以文本格式保存文件。

在性能收集组件运行时，不能用 Windows **写字板**编辑器修改 parm 文件。必须停止性能收集组件，使用**写字板**，然后重新启动性能收集组件。但是，在性能收集组件运行时，可以使用**记事本**编辑器修改文件。

在继续另一个任务之前，*必须*激活对 parm 文件所做的任何更改。执行以下步骤：

1 在性能收集组件主窗口的 “代理程序 (Agent)” 菜单中，单击**启动 / 停止 (Start/Stop)** 打开 “MeasureWare 服务 (MeasureWare Services)” 窗口。

2 选中**数据收集 (Data Collection)** 复选框。

3 单击**刷新 (Refresh)** 按钮。

4 单击**关闭 (Close)** 按钮返回主窗口。

有关修改 parm 文件的逐步说明，请从 “帮助 (Help)” 菜单选择**帮助主题 (Help Topics)**，选择 “**如何…？ (How Do I…?)**”，再选择 “**修改收集参数文件 (Modify a collection parameters file)**”。

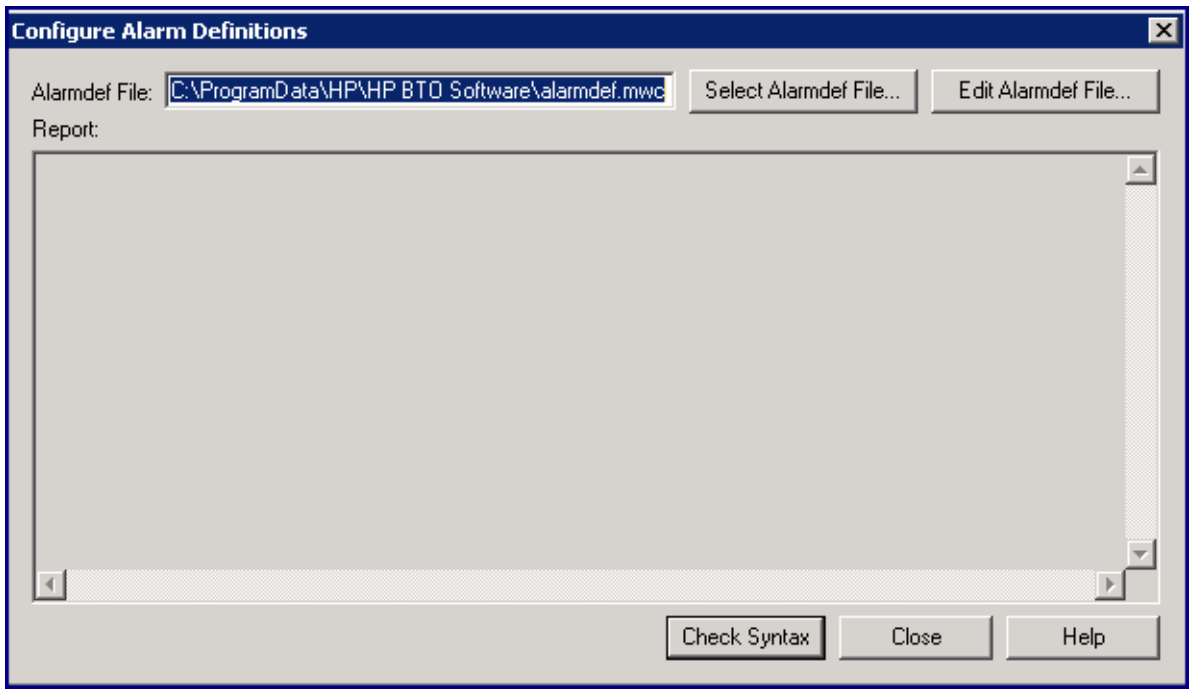
如果使用写字板、记事本或 Microsoft Word 修改 parm.mwc 文件，然后使用**另存为 (Save As)** 命令保存文件，那么文件名会自动添加默认 .txt 扩展名。这样，您将有一个名为 parm.mwc.txt 的文件。要保留 parm.mwc 文件名，请使用**另存为 (Save As)** 命令将文件另存为文本文件，并在文件名两边加双引号 (")。例如："parm.mwc"。

配置警报定义

使用 “配置 (Configure)” 菜单的**警报定义 (Alarm Definitions)** 命令，可以检查警报定义文件 (alarmdef.mwc) 中的警报定义的语法。确定警报定义语法正确时，可以对照警报定义分析日志文件，检查历史日志文件数据中是否有警报 （请参见第 227 页的分析日志文件）。

如果发现任何警告或错误并要更正它们，或如果要添加或删除警报定义，可以使用 “配置警报定义 (Configure Alarm Definitions)” 对话框中的**编辑 alarmdef 文件 (Edit Alarmdef File)** 按钮轻松地修改警报定义文件。

图 14　　配置警报定义对话框



要检查语法，请执行以下步骤：

1. 在性能收集组件主窗口的 “配置 (Configure)” 菜单中单击**警报定义 (Alarm Definitions)**。将出现 “配置警报定义 (Configure Alarm Definitions)” 对话框，显示当前打开的警报定义文件的名称。
2. 要检查其他警报定义文件，请单击**选择 alarmdef 文件 (Select Alarmdef File)** 按钮。
3. 单击**检查语法 (Check Syntax)** 按钮启动语法检查进程。几秒钟后检查结果 （包括任何警告或错误）将显示在性能收集组件报告查看器窗口中。

4 要修改警报定义文件的任何部分，请单击**编辑 alarmdef 文件 (Edit Alarmdef File)** 按钮。

有关检查警报定义文件语法的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择“**检查警报定义文件的语法 (Checking the syntax of an alarm definitions file)**”。

修改警报定义文件

要修改 alarmdef 文件，请执行以下步骤：

1 在性能收集组件主窗口的“配置 (Configure)”菜单中单击**警报定义 (Alarm Definitions)**，再在“配置警报定义 (Configure Alarm Definitions)”对话框中单击**编辑 alarmdef 文件 (Edit Alarmdef File)** 按钮。当前打开的警报定义文件的内容将显示在之前指定的编辑器或字处理器中。（要配置编辑器或字处理器，请参见第 233 页的配置用户选项。）

2 在对文件进行任何更改之前，请参见第 156 页的警报语法参考了解有关警报定义的详细信息。

3 根据需要修改文件，并以文本格式保存文件。

如果使用写字板、记事本或 Microsoft Word 修改警报定义文件，然后使用**另存为 (Save As)** 命令保存文件，那么文件名会自动添加默认 `.txt` 扩展名。这样，您将有一个名为 `alarmdef.mwc.txt` 的文件。要保留 `alarmdef.mwc` 文件名，请使用**另存为 (Save As)** 命令将文件另存为文本文件，并在文件名两边加双引号 (")。例如，`"alarmdef.mwc"`。

激活更改

在继续另一个任务之前，*必须*激活对警报定义文件所做的任何更改。执行以下步骤：

1 在性能收集组件主窗口的“代理程序 (Agent)”菜单中，单击**启动 / 停止 (Start/Stop)** 打开“MeasureWare 服务 (MeasureWare Services)”窗口。

2 选中**警报定义 (Alarm Definitions)** 复选框。

3 单击**刷新 (Refresh)** 按钮。

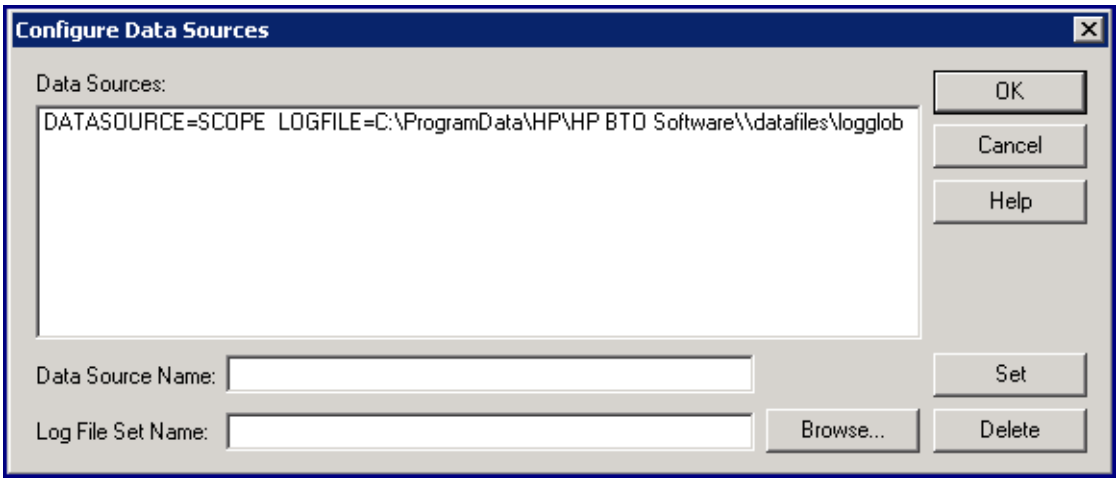
4 单击**关闭 (Close)** 按钮返回主窗口。

有关修改警报定义文件的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择 “**修改警报定义文件 (Modify an alarm definitions file)**”。

配置数据源

性能收集组件对每个特定数据源 （比如 scopent 日志文件或 DSI 日志文件）使用数据源。每个数据源由单个日志文件集组成。数据源在 < 数据目录 >\conf\perf 目录中的 datasources 文件中配置。安装性能收集组件后首次启动它时，已配置了名为 SCOPE 的默认数据源，并提供 scopent 日志文件集。

图 15　　配置数据源对话框



数据源文件格式

datasources 文件中放入的每个条目都表示一个由单个日志文件集组成的数据源。条目指定辨别数据仓库服务器所依据的数据源名称以及找到它包括的数据的位置。条目不区分大小写。语法是：

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

- **datasource** 是关键字。*datasource_name* 是用于识别警报定义或分析软件中所用的数据源的名称。数据源名称必须是唯一的。 *datasource_name* 的最大长度是 64 个字符。
- **logfile** 是关键字。*logfile_set* 是识别日志文件集的完全限定名称。它可以是 scopent 创建的原始日志文件集、extract 任务创建的提取日志文件或 DSI 日志文件集。如果指定包含嵌入空格的日志文件路径名称，必须在路径名称的两边加上双引号 (")。

指定 scopent 日志文件集时，使用 logglob 文件名即可。不需要指定其他原始日志文件名称，因为它们是作为单个日志文件集访问的。

指定 DSI 日志文件集时也是如此。指定 DSI 根文件的名称即可。不需要指定 DSI 日志文件集中的任何其他文件。

从远程位置配置数据源

指定驻留在网络共享位置的日志文件时，需要使用通用命名约定 (UNC)。在系统启动时，性能收集组件服务自动启动，而远程连接的文件系统的驱动器映射要等到用户登录后才能建立。因此，任何使用驱动器映射名称引用远程系统上的日志文件的数据源都会导致 coda 生成无效数据源错误。如果登录*后*启动性能收集组件服务，这时驱动器映射已建立，数据源会得到处理。

以下是三个数据源条目示例：

示例 1:

以下示例显示了驻留在默认 <*磁盘驱动器*>:\Program Files\HP\HP BTO Software\data\datafiles\ 目录中的默认 SCOPE 数据源。

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

示例 2:

在以下示例中，使用通用命名约定 (UNC) 指定驻留在网络共享位置的日志文件集。

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

示例 3:

以下示例显示了驻留在路径名称包括嵌入空格的目录中的 SCOPE 数据源。

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

要配置数据源，请在性能收集组件主窗口的“配置 (Configure)”菜单中单击**数据源 (Data Sources)**。

将出现 “配置数据源 (Configure Data Sources)”对话框，列出当前数据源条目。每个条目表示一个数据源。

要进行修改，请执行以下步骤：

1 在 “数据源 (Data Sources)”列表中选择数据源。

2 单击**日志文件集名称 (Log File Set Name)** 框，修改日志文件集名称，然后单击**设置 (Set)** 按钮。

要添加新数据源，请执行以下步骤：

1 单击**数据源名称 (Data Source Name)** 框，输入新名称。

2 单击**日志文件集名称 (Log File Set Name)** 框，输入新的完全限定日志文件集名称，然后单击**设置 (Set)** 按钮。

或者

3 单击**浏览 (Browse)** 按钮选择现有数据源。

要删除数据源，请执行以下步骤：

1 在 “数据源 (Data Sources)”列表中选择数据源。

2 单击**删除 (Delete)** 按钮

3 完成数据源文件配置后，单击**确定 (OK)**。

激活更改：

在继续另一个任务之前，*必须*激活对数据源所做的任何更改。执行以下步骤：

1. 在性能收集组件主窗口的“代理程序 (Agent)”菜单中，选择**启动 / 停止 (Start/Stop)** 打开“MeasureWare 服务 (MeasureWare Services)”窗口。
2. 单击**停止服务 (Stop Services)** 按钮停止 MeasureWare 服务。
3. **停止服务 (Stop Services)** 按钮变暗后，单击**启动服务 (Start Services)** 按钮。
4. 单击**关闭 (Close)** 按钮返回主窗口。

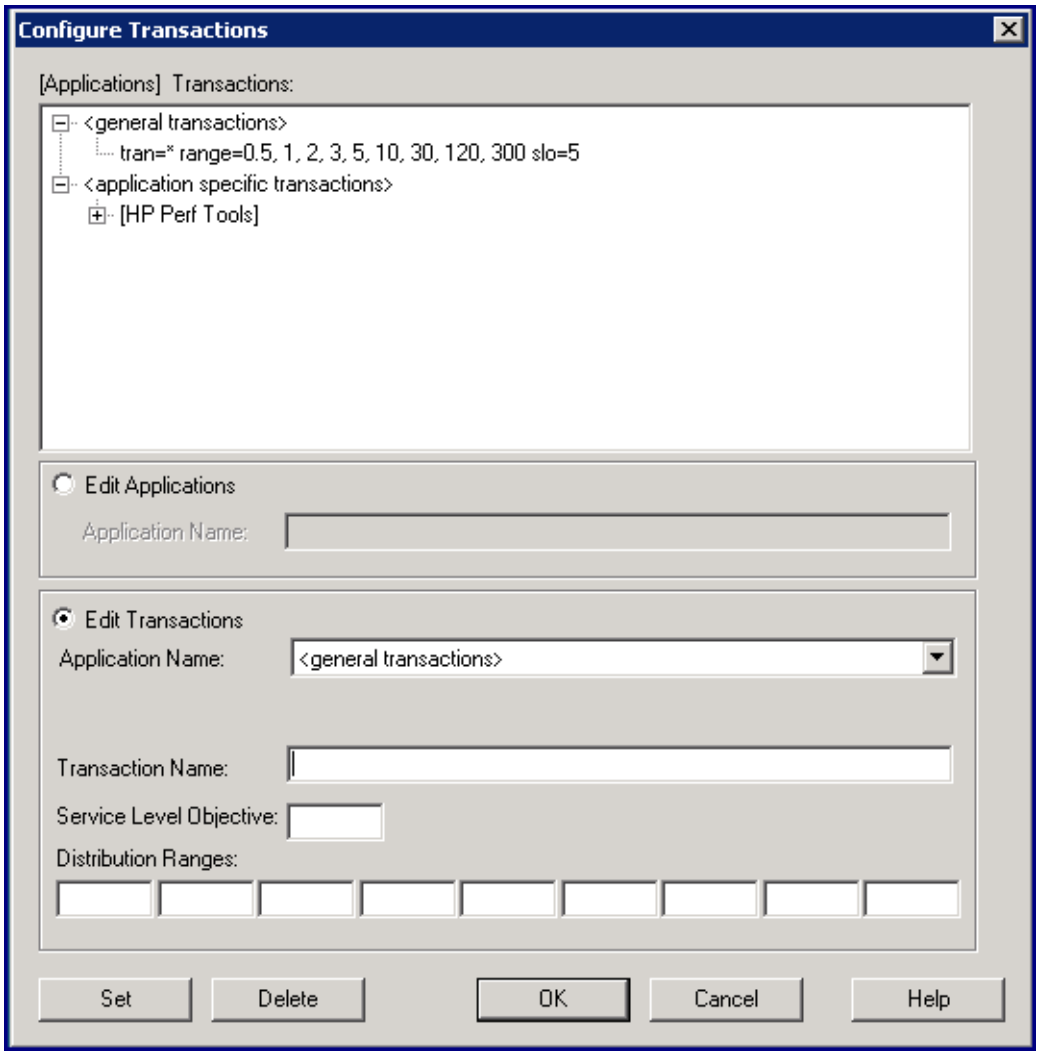
有关修改数据源文件的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择“**修改数据源文件 (Modify a data source file)**”。

配置事务

使用事务配置文件 `ttdconf.mwc` 可以自定义应用程序的事务数据收集。该文件定义事务名称、性能分布范围和每个事务要满足的服务级别目标。还可以选择定义特定于应用程序的事务。

默认 `ttdconf.mwc` 文件包括三个条目。两个条目定义性能收集组件 `scopent` 收集器使用的事务，另外一个条目 `tran=*` 注册已通过应用程序响应测量 (ARM) API 函数调用检测的应用程序的所有事务。

图 16　　配置事务对话框



如果要在系统中添加的新应用程序使用的是默认 `ttdconf.mwc` 文件的 `tran=*` 条目中的服务级别目标和范围值，则无需执行任何操作来合并新事务。新事务会自动应用所有默认值。

但是，如果要在系统中添加的应用程序其事务具有自己的唯一服务级别目标和分布范围值，则必须将这些事务添加到 ttdconf.mwc 文件。

ttdconf.mwc 文件中的条目顺序不相关。首先搜索精确匹配项。如果找不到，则使用末尾带星号 (*) 的最长匹配。

在对文件进行任何更改之前，请参见第 339 页的什么是事务跟踪？，了解有关配置文件格式、事务和应用程序名称、性能分布范围以及服务级别目标的描述。要进行配置，请在性能收集组件主窗口的“配置 (Configure)”菜单中，单击**事务 (Transactions)** 显示“配置事务 (Configure Transactions)”对话框。使用此对话框可以执行以下任务：

1 添加常规事务

2 添加特定于应用程序的事务

3 修改事务的性能分布范围或服务级别目标

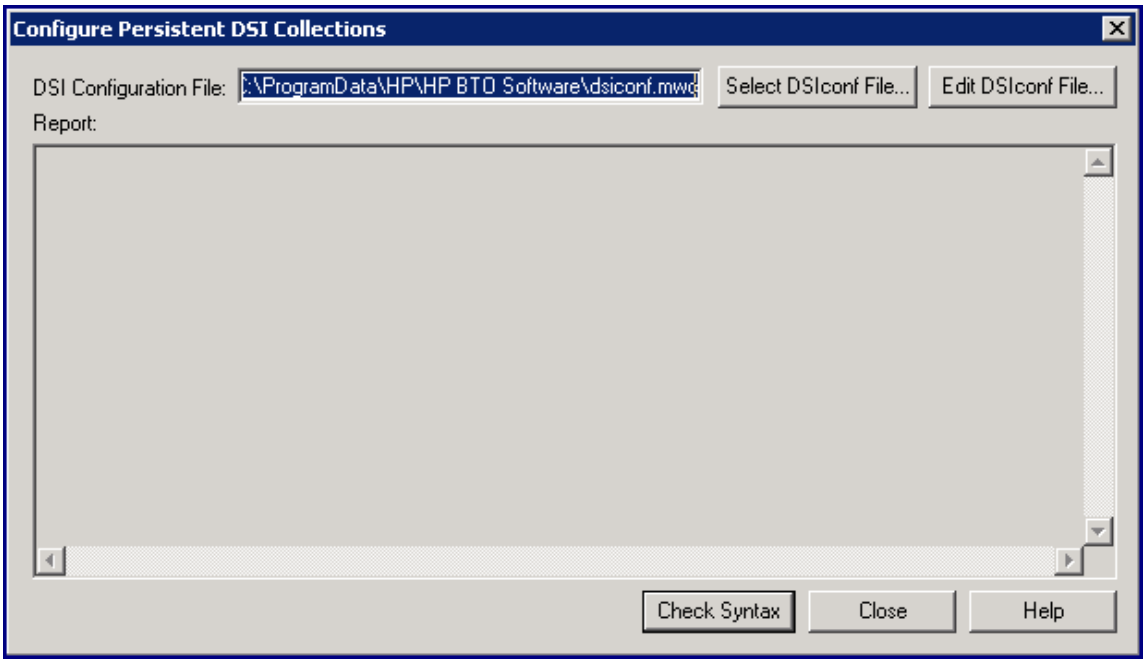
4 删除事务

有关配置这些任务的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…？ (How Do I…?)**”，再选择“**配置事务 (Configure transactions)**”。

配置持续 DSI 收集

使用“配置 (Configure)”菜单的**持续 DSI 收集 (Persistent DSI Collections)** 命令，可以检查 DSI 配置文件 dsiconf.mwc 的语法或修改该文件。dsiconf.mwc 文件用于配置持续记录从外部源收集到性能收集组件的数据。有关详细信息，请参见第 255 页的数据源集成概述。

图 17　　配置持续 DSI 收集对话框



要检查 DSI 配置文件的语法，请执行以下步骤：

1. 在性能收集组件主窗口的“配置 (Configure)”菜单中单击**持续 DSI 收集 (Persistent DSI Collections)**。“配置持续 DSI 收集 (Configure Persistent DSI Collections)”对话框显示当前打开的 `dsiconf.mwc` 文件的名称。
2. 要检查其他 `dsiconf.mwc` 文件，请单击**选择 DSIconf 文件 (Select DSIconf File)** 按钮。
3. 要检查文件语法，请单击**检查语法 (Check Syntax)** 按钮。任何生成的警告或错误都显示在性能收集组件报告查看器窗口中。
4. 要修改文件的任何部分，请单击**编辑 DSIconf 文件 (Edit DSIconf File)** 按钮。可以将“编辑 DSIconf 文件 (Edit DSIconf File)”和“配置持续 DSI 收集 (Configure Persistent DSI Collections)”对话框放在屏幕上，以便可以同时使用它们。

有关检查 DSI 配置文件语法的逐步说明，请从“帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择“**如何…? (How Do I…?)**”，再选择“**检查 DSI 配置文件的语法 (Check the syntax of a DSI configuration file)**”。

要修改 DSI 配置文件，请执行以下步骤：

1 在性能收集组件主窗口的 “配置 (Configure)” 菜单中单击**持续 DSI 收集 (Persistent DSI Collections)**，再在 “配置持续 DSI 收集 (Configure Persistent DSI Collections)” 对话框中单击**编辑 DSIconf 文件 (Edit DSIconf File)** 按钮。在以前指定的编辑器或字处理器中显示当前打开的 `dsiconf.mwc` 文件的内容。（要指定编辑器或字处理器，请参见第 233 页的配置用户选项。）

2 在对文件进行任何更改之前，请参见第 209 页的在 Windows 上使用性能收集组件了解要遵循的规则和约定。

3 根据需要修改文件，并以文本格式保存文件。

在继续另一个任务之前，*必须*激活对 `dsiconf.mwc` 文件所做的任何更改。执行以下步骤：

1 在性能收集组件主窗口的 “代理程序 (Agent)” 菜单中，单击**启动 / 停止 (Start/Stop)** 打开 “MeasureWare 服务 (MeasureWare Services)” 窗口。

2 选中**持续 DSI 收集 (Persistent DSI Collections)** 复选框。

3 单击**刷新 (Refresh)** 按钮。

4 单击**关闭 (Close)** 按钮返回主窗口。

有关修改 DSI 配置文件的逐步说明，请从 “帮助 (Help)” 菜单选择**帮助主题 (Help Topics)**，选择 “**如何操作…？ (How Do I…?)**”，再选择 “**修改 DSI 配置文件 (Modify a DSI configuration file)**”。

如果使用 WordPad、Notepad 或 Microsoft Word 修改 `dsiconf.mwc` 文件，然后使用**另存为 (Save As)** 命令保存文件，那么文件名中会自动添加默认 `.txt` 扩展名。因而，您有一个名为 `dsiconf.mwc.txt` 的文件。要保留 `dsiconf.mwc` 文件名，请使用**另存为 (Save As)** 命令将文件另存为文本文件，并在文件名两边加双引号 (")。例如：`"dsiconf.mwc"`

检查性能收集组件状态

使用“代理程序 (Agent)”菜单的**状态 (Status)** 命令可以检查性能收集组件进程的当前状态。状态信息由 `perfstat` 程序生成。

您可以从“配置 (Configure)”菜单选择**选项 (Options)** 命令，并在“配置选项 (Configure Options)”对话框中选择以下任意选项，指定要在状态报告中包括哪些特定信息。

正在运行的进程

列出性能收集组件当前正在运行的后台和前台进程。同时还列出应该在运行但*未*运行的所有后台进程。

Datacomm 服务

Datacomm 服务定位性能收集组件 datacomm 服务并与其通信。它们显示警报生成器数据库服务器 (`agdbserver`) 进程是否正在运行以及是否作出了响应。如果未启用数据通信，它们会等待 datacomm 服务作出响应，并在 30 秒后生成此信息。

系统服务

显示 scope 收集器、事务管理器和测量接口等性能收集组件系统服务的当前状态。

系统配置

系统名称、操作系统版本和处理器类型。

文件版本号

性能收集组件文件的版本号。注明任何缺少的重要文件。

状态文件最新条目

来自每个性能工具状态文件的几个最新条目。

状态文件警告和错误

列出性能工具状态文件中所有包含 “Error”或 “Warning”的行。如果长时间忽略警告，可能会生成很长的列表。

要列出当前状态，请在性能收集组件主窗口的 “代理程序 (Agent)”菜单中单击**状态 (Status)**。性能收集组件报告查看器将显示您在 “配置选项 (Configure Options)”对话框中选择的信息。

要获取所有状态信息的完整报告，请在 “代理程序 (Agent)”菜单中单击**报告 (Report)**。性能收集组件报告查看器将显示所有状态信息的完整列表。

有关检查性能收集组件状态的逐步说明，请从 “帮助 (Help)”菜单选择**帮助主题 (Help Topics)**，选择 “**如何…？ (How Do I…?)**” ，再选择 “**检查性能收集组件进程的状态 (Check status of Performance Collection Component processes)**”。

还可以从 Windows 命令提示符处运行 `perfstat` 程序。

生成性能计数器收集

性能收集组件提供对 Windows 性能计数器的访问，这些计数器用于测量系统性能或系统上的应用程序或设备性能。使用扩展的收集生成器和管理器 (ECBM) 选择特定性能计数器以生成数据收集。

生成性能计数器收集

要生成收集，请在性能收集组件主窗口的 “代理程序 (Agent)” 菜单中选择**扩展的收集 (Extended Collections)**。将出现 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)” 窗口，在左窗格显示 Windows 对象列表。有关生成收集的说明，请在 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)” 窗口的 “帮助 (Help)” 菜单中选择**帮助主题 (Help Topics)**。

生成 Windows 性能计数器收集之后，使用底部的 “扩展的收集管理器 (Extended Collection Manager)” 窗格注册、启动和停止新收集和现有收集。

管理性能计数器收集

要管理数据收集，请使用 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)” 底部的 “扩展的收集管理器 (Extended Collection Manager)” 窗格。最初不显示任何收集，因为必须注册收集后才能开始收集数据。

注册或存储创建的收集后，“扩展的收集管理器 (Extended Collection Manager)” 窗格将显示当前收集列表。“扩展的收集管理器 (Extended Collection Manager)” 窗格还显示每个收集的状态，并允许您查看收集本身的信息 （属性） 。有关管理收集的说明，请在 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)” 窗口的**帮助 (Help)** 菜单中选择**帮助主题 (Help Topics)**。

使用扩展的收集生成器和管理器的提示

- `< 安装目录 >\paperdocs\mwa\C\monxref.txt` 文件包括性能收集组件度量对 Windows 性能计数器和命令的交叉引用。通过扩展的收集生成器和管理器为已由性能收集组件收集的度量记录数据会产生额外的系统开销。
- 使用扩展的收集生成器创建收集时，将为 Windows 性能计数器分配默认度量名称，供性能收集组件内部使用。这些默认名称通常没有意义或难以解译。为使度量名称更有意义，或让它们与其源应用程序提供的度量名称匹配，请将度量名称从 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)” 窗口的左窗格拖到右窗格后，右键单击或双击该度量名称修改度量属性。（有关详细说明，请参见扩展的收集生成器和管理器联机帮助。）

- 如果启动 61 个或更多个收集，超过 60 个的收集会显示错误状态。这可能导致其他收集出现问题。
- 如果从配置了 **Wolfpack** 的系统中收集逻辑磁盘度量，必须重新启动收集，以收集任何注册收集时不存在的新磁盘实例的数据。
- 删除收集后需要重新启动性能收集组件收集删除才能成功。如果不重新启动性能收集组件，可能会在删除操作期间收到错误。此错误通常表明一些文件未成功删除。如重新启动性能收集组件后仍留有一些文件和目录，可能需要手动删除它们。
- 扩展的收集生成器和管理器可能会报告缓存计数器的一些度量缺少值。在某些情况（如度量值溢出）下会发生此问题。同时还会向 ECBM 状态文件发送一则消息。重新启动收集可以解决问题。

有关扩展的收集生成器和管理器的概念解释，以及有关创建和查看数据收集的说明，请参见扩展的收集生成器和管理器联机帮助。要查看联机帮助，请从桌面选择**开始 → 程序 → HP → Operations Agent → 性能收集组件 → ECB-ECM 联机帮助 (ECB-ECM Online Help)**。可以在性能收集组件主窗口的“代理程序 (Agent)”菜单中选择**扩展的收集 (Extended Collections)**，再在“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口的“帮助 (Help)”菜单中选择**帮助主题 (Help Topics)**。此外，还可通过选择 “扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”中出现的对话框中的**帮助 (Help)** 按钮来访问联机帮助。

从命令行管理 ECBM

可以使用 Windows 命令提示符从 `<rpmtools>\bin` 目录运行 ECBM 程序。

可以使用以下命令从命令行管理收集：

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

要显示各种选项，请输入以下命令：

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

要启动停止的收集，请输入以下命令：

```
mwcmcmd start <收集名称>
```

要从可变实例策略启动新收集，请输入以下命令：

```
mwcmcmd start <策略名称> <收集名称> <实例> [选项]
```

以下选项可用：

```
-i <采样间隔>          - 更改采样间隔（秒）
-l <日志文件路径名称>  - 更改默认日志位置
-a <警报文件>          - 更改警报定义文件
```

要停止活动的收集，请输入以下命令：

```
mwcmcmd stop <收集名称>
```

以下是注册策略文件的命令：

```
mwcmcmd register <策略文件> <收集/策略名称> [选项]
```

以下选项仅在注册固定实例策略文件时可用：

```
-i <采样间隔>          - 更改采样间隔（秒）
-l <日志文件路径名称>  - 更改默认日志位置
-a <警报文件>          - 更改警报定义文件
```

要删除单个收集：

```
mwcmcmd delete <收集/策略名称> [选项]
```

以下选项仅在删除收集时可用：

```
-p <存档路径>          - 将日志文件存档到指定路径
-r                     - 重新启动 Performance Agent
```

要删除多个收集或策略：

```
mwcmcmd delete { <收集 / 策略名称> | -c | -all }
    -c      - 删除所有收集
    -a      - 删除所有收集和策略
```

一次删除多个策略 / 收集时，性能收集组件会自动重新启动，并删除所有关联的日志文件。

要列出所有注册的收集和策略，请输入以下命令：

```
mwcmcmd list
```

要列出某个收集或策略的属性，请输入以下命令：

```
mwcmcmd properties <收集 / 策略名称>
```

要列出策略中的可变实例对象，请输入：

```
mwcmcmd objects <策略名称>
```

12 数据源集成概述

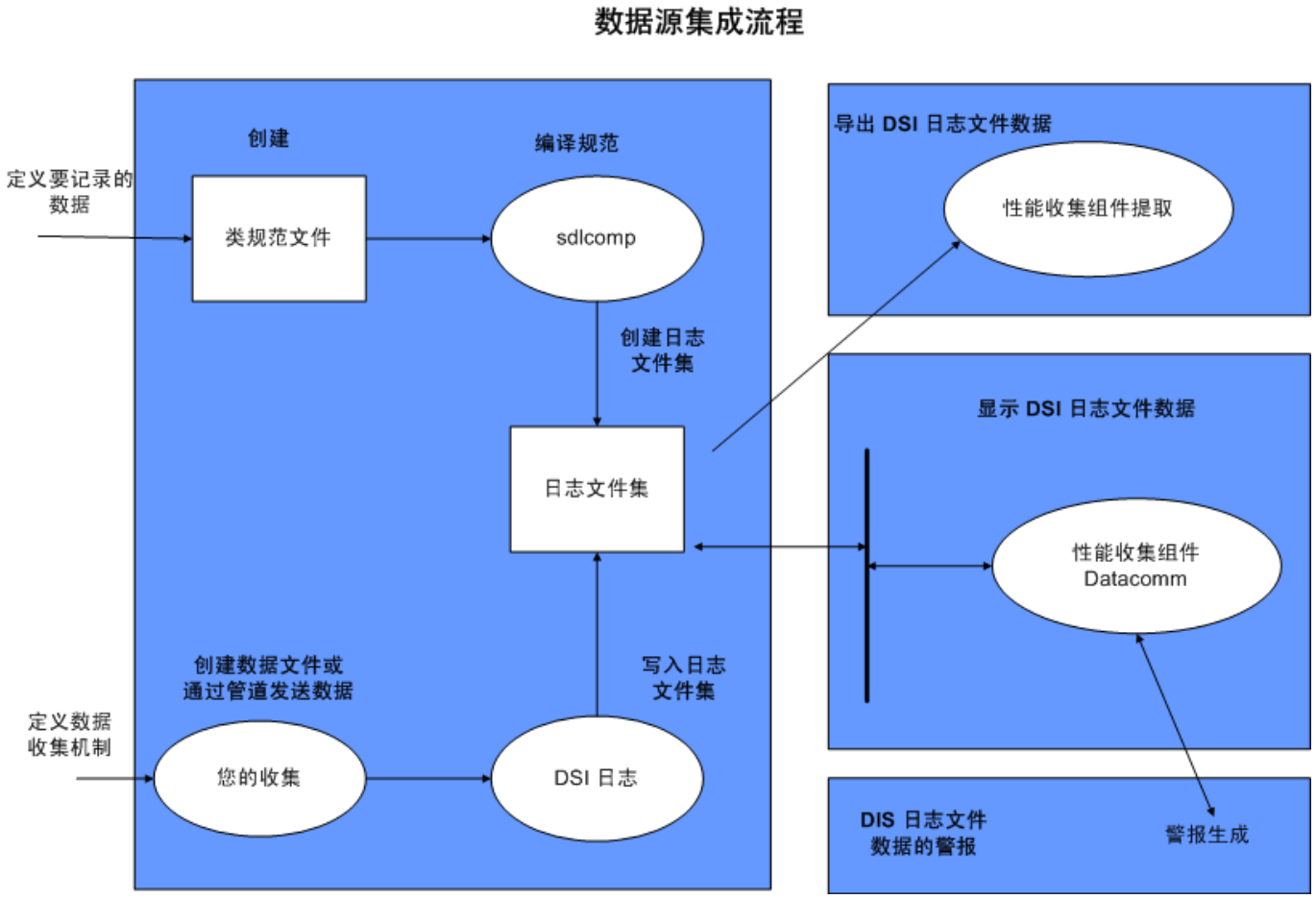
数据源集成 (DSI) 技术使您可以使用 HP Operations Agent 记录数据、定义警报以及访问来自性能收集组件的 scope 收集器记录的度量之外的新数据源的度量。度量可以从数据库、 LAN 监视器和最终用户应用程序等数据源获得。

用 DSI 记录的数据可与 scope 收集器记录的标准性能度量一起显示在 HP Performance Manager 中。 DSI 记录的数据也可以用 性能收集组件 extract 程序导出，以便显示在电子表格或类似的分析包中。

DSI 的工作原理

下图显示了如何创建 DSI 日志文件以及如何将其用于记录和管理数据。DSI 日志文件包含在性能收集组件 scope 收集器外部收集的自描述数据。下页更详细地描述了 DSI 流程。

图 18　　数据源集成流程



使用 DSI 记录数据包括以下任务：

创建类规范

首先为要记录的数据的每个类创建并编译规范。该规范描述数据的类以及要记录到该类的各个度量。使用 DSI 编译器 `sdlcomp` 编译规范时，将创建一组空日志文件接受来自 `dsilog` 程序的数据。此过程将创建包含根文件、描述文件及一个或多个数据文件的日志文件集。

收集和记录数据

然后您就可以通过启动感兴趣的进程收集要记录的数据。可以直接通过管道将收集进程的输出发送给 `dsilog` 程序，也可以从存储数据的文件发送。`dsilog` 按照规范处理数据，并将它写入相应的日志文件。`dsilog` 可用于指定传入数据的形式和格式。

您在 DSI 流程中提供的数据应包含多条数据记录。一条记录由包含在单行中的度量值组成。如果将数据发送到 DSI 时采取每次发送一条记录，停止进程，然后发送另一条记录的方式，则 `dsilog` 可以追加，但无法汇总数据。

使用数据

可使用 Performance Manager 显示 DSI 日志文件数据。或者可使用 性能收集组件 `extract` 程序导出数据以供其他分析工具使用。还可以配置在 DSI 度量超过定义的条件时要发出的警报。

13 使用数据源集成

本章概述如何使用 DSI，包含以下信息：

- 规划数据收集
- 在类规范文件中定义日志文件格式
- 创建空的日志文件集
- 将数据记录到日志文件集
- 使用记录的数据

有关 DSI 类规范和 DSI 程序的详细参考信息，请参见第 14 章 DSI 类规范参考和第 15 章 DSI 程序参考。

规划数据收集

创建 DSI 类规范文件并启动记录进程前，需要确定以下主题：

- 充分了解环境，知道在计算资源的管理中哪几类数据会有用。
- 提供了哪些数据？
- 数据在哪里？
- 如何收集数据？
- 数据项之间的分隔符是什么？为了 `dsilog` 能正确处理，输入流中的度量值必须用空格分隔（默认）或用户定义的分隔符分隔。
- 收集的频率如何？
- 维护日志需要多大的空间？
- 用于访问数据的程序或进程的输出是什么？
- 您希望在哪些条件下生成哪些警报？
- 用类规范和 `dsilog` 进程进行记录时，有哪些选项可用？

定义日志文件格式

清楚了解要收集哪类数据后，创建类规范以定义要记录的数据以及将包含记录的数据的日志文件集。在类规范中输入以下信息：

- 数据类名和 ID 号
- 可替代类名的标签名称（可选）。（例如，如果有标签名称，则可在 Performance Manager 中使用它。）
- 滚出旧数据以便为新数据留出空间时要发生的操作。有关详细信息，请参见日志文件的组织方式。
- 度量名称和其他说明信息，如度量值可精确到几位小数。
- 想要每小时只记录有限的记录数时，希望如何汇总数据。

下面是类规范的示例：

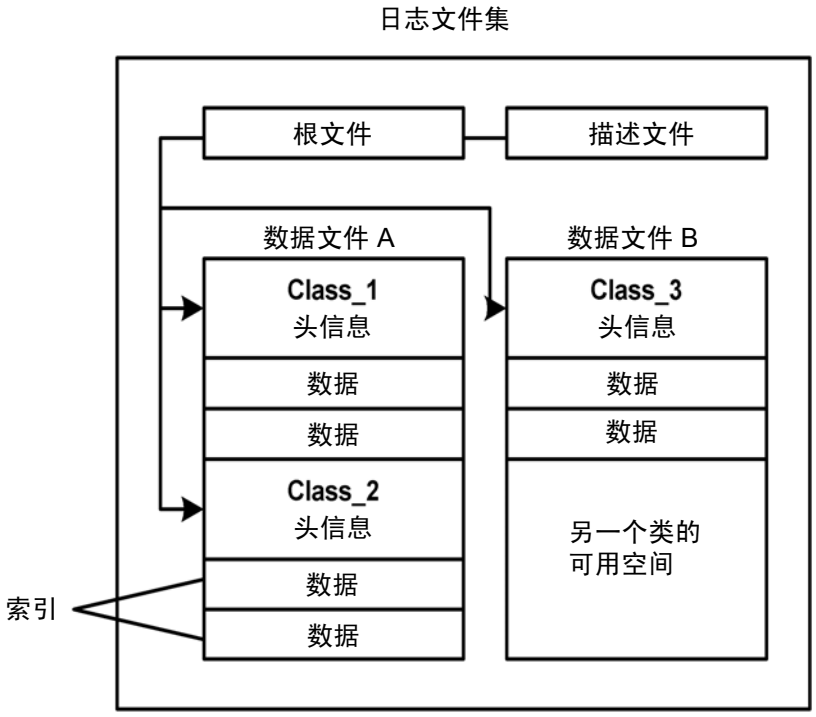
```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
ROLL BY HOUR
RECORDS PER HOUR 120;

METRICS
RUN_Q_PROCS    = 106
LABEL   "Procs in run q"
PRECISION 0;

BLOCKED_PROCS   = 107
LABEL   "Blocked Processes"
PRECISION 0;
```

可以在类规范文件中包括一个或多个类。完成类规范文件后，命名该文件，然后保存。运行 DSI 编译器 sdlcomp 时，使用此文件创建日志文件集。有关类规范和度量描述语法的详细信息，请参见第 14 章 DSI 类规范参考

日志文件的组织方式

日志文件按类的方式组织。每个类都表示一个传入数据源，由记录在一起的一组数据项或度量组成。一个类中的每个数据记录或数据行都表示该组度量的值的一个示例。

类的数据存储在磁盘上属于日志文件集的日志文件中。日志文件集包含根文件、描述文件及一个或多个日志文件。来自某一类的所有数据始终保存在单个数据文件中。但是，向 sdlcomp 编译器提供日志文件集名称时，可将多个类一起存储在单个日志文件集中，也可分别存储在单独的日志文件集中。下图演示如何在单个日志文件集中存储两个类。



因为每个类都创建为循环的日志文件，所以即使已指定应将多个类存储在单个日志文件集中，也可单独设置每个类的存储容量。达到存储容量时，该类将 “滚动”，即删除该类中的最早记录，为新数据留出空间。

您可以指定在该类滚动时要执行的操作，如将旧数据导出到存档文件。

创建日志文件集

DSI 编译器 sdlcomp 使用类规范文件创建或更新空的日志文件集。随后用该日志文件集从 dsilog 程序接收记录的数据。

要创建日志文件集，请完成以下任务：

1 用相应的变量和选项运行 sdlcomp。例如，

sdlcomp [-maxclass *值*] *规范文件*
[*日志文件集* [*日志文件*]] [选项]

2 检查输出中是否有错误，并根据需要进行更改。

有关 sdlcomp 的详细信息，请参见第 15 章的编译器语法。

测试类规范文件和记录进程 （可选）

DSI 使用程序 sdlgendata，您可使用该程序对照所生成数据的传入源测试类规范文件。随后可以检查此进程的输出，以验证 DSI 可以按照规范记录数据。有关 sdlgendata 的详细信息，请参见第 15 章的用 sdlgendata 测试记录进程。

要测试用于记录进程的类规范文件：

1 将 sdlgendata 生成的数据提供给 dsilog 程序。语法是：

sdlgendata *logfile_set class* **|** **dsilog** *logfile_set class* **-vo**

2 检查输出，查看类规范文件是否与数据收集进程的格式匹配。如果 sdlgendata 程序输出与您的程序不同，表示您的输出格式有误，或类规范文件有误。

3 开始收集真实数据之前，从测试进程删除所有日志文件。

将数据记录到日志文件集

创建日志文件集并测试（可选）之后，根据需要更新性能收集组件配置文件，然后运行 dsilog 程序记录传入数据。

1. 更新数据源配置文件 datasources，将 DSI 日志文件添加为生成警报的数据源。
2. 如果要对特定的 DSI 度量发出警报，请修改警报定义文件 alarmdef。有关详细信息，请参见第 15 章的定义 DSI 度量的警报。
3. 或者，通过管道将数据（可能由 sdlgendata 生成以匹配类规范）发送到带 -vi 选项设置的 dsilog 程序来测试记录进程。
4. 检查数据，确保记录正确。
5. 测试之后，删除已测试的数据。
6. 从命令行启动收集进程。
7. 使用相应的变量和选项设置从收集进程通过管道将数据发送到 dsilog（或以其他方式发送到 stdin）。例如：

   < *带变量的程序或进程* >| dsilog *logfile_set class*

dsilog 程序设计为接收连续的数据流。因此，构造脚本以使 dsilog 能够接收连续的输入数据很重要。不要编写为新输入数据点创建新 dsilog 进程的脚本。这可能导致时间戳重复写入 dsilog 文件，从而导致读取文件时 Performance Manager 和 perfalarm 出现问题。有关有问题的脚本和推荐的脚本的信息，请参见第 16 章数据源集成示例

有关 dsilog 选项的详细信息，请参见第 15 章的 dsilog 记录进程。

使用记录的数据

创建 DSI 日志文件后，可使用性能收集组件的 `extract` 程序导出数据。还可以配置在 DSI 度量超过定义的条件时要发出的警报。

下面是使用记录的 DSI 数据的方式：

- 导出数据供电子表格等报表工具使用。
- 使用 Performance Manager 等分析工具显示导出的 DSI 数据。
- 使用 HP Operations Manager 或 HP Network Node Manager 监视警报。

不能从 DSI 日志文件创建提取的日志文件。

14 DSI 类规范参考

本章提供有关以下内容的详细参考信息：

- 类规范
- 类规范语法
- 类规范中的度量描述

类规范

对于每个传入数据源，都必须创建类规范文件以描述用于存储传入数据的格式。要创建文件，请使用下一部分类规范语法中描述的类规范语言。类规范文件包含：

- 类描述，它将名称和数字 ID 分配给传入数据集，确定将存储的数据量，并指定何时滚动数据以为新数据留出空间。
- 每个数据项的度量描述。度量描述指定并描述数据项。它还指定当为该类配置的时间间隔内有多条记录到达时，要应用于数据 (RECORDS PER HOUR) 的摘要级别。

要生成类规范文件，请使用任何可将文件另存为 ASCII 文本文件的编辑器或字处理器。指定在运行 sdlcomp 以编译类规范文件时该文件的名称。编译类规范时，它自动创建或更新用于存储数据的日志文件集。

可使用类规范确定每小时将为该类存储的记录数，并指定当到达的记录比想存储的多时要使用的汇总方法。例如，如果已请求每小时存储 12 条记录（每五分钟存储一条记录），但每分钟都有记录到达，则可以对某些数据项取平均值，并对另一些加总，以维持连续计数。

DSI 编译器 sdlcomp 用以下名称为日志文件集 （名为 logfile_set_name）创建文件：

```
logfile_set_name 和 logfile_set_name.desc
```

sldcomp 用以下默认名称为类 （名为 class_name）创建文件：

```
logfile_set_name.class_name
```

不要使用与这些命名约定冲突的类规范文件名，否则 sdlcomp 将失败。

类规范语法

方括号 [ ] 中显示的语法语句是可选的。大括号 { } 中显示的多个语句表示必须选择一条语句。斜体字表示您输入的变量名或数字。逗号可用于分隔语法语句，增强清晰性，但不能直接放在分号前，因为这标记着类规范的结束和每个度量规范的结束。语句不区分大小写。

用户定义的描述（如*度量标签名称*或*类标签名称*）不能与 DSI 类规范语法的任何关键字元素相同。

注释以 # 或 // 开头。行内 # 或 // 后的所有内容都被忽略。请注意，类描述和每个度量描述的后面必须有分号。下面是类规范每一部分的详细信息和示例。

```
CLASS 类名 = 类 ID 号
[LABEL "类标签名称"]

 [INDEX BY {HOUR | DAY | MONTH} MAX INDEXES 数字
 [[ROLL BY {HOUR | DAY | MONTH} [ACTION "操作" ]

 [ CAPACITY { 最大记录数 } ]
[ RECORDS PER HOUR 数字 ]
;

METRICS

metric_name = metric_id_number
[ LABEL "metric_label_name" ]
[ TOTALED | AVERAGED | SUMMARIZED BY metric_name ]
 [ MAXIMUM  metric_maximum_number ]
[PRECISION {0 | 1 | 2 | 3 | 4 | 5} ]
[TYPE TEXT LENGTH "length"]
;
```

类描述

要创建类描述，请将名称分配给来自特定数据源的一组度量，指定类的容量，并指定超过容量时如何滚动类中的数据。

类描述必须以 CLASS 关键字开头。类规范中的最后参数后必须有分号。

语法

CLASS *类名* **=** *类 ID 号*

[LABEL " *类标签名称* **"]**

[INDEX BY { HOUR | DAY | MONTH } MAX INDEXES *数字*
[[ROLL BY { HOUR | DAY | MONTH } [ACTION " *操作* **"]**

[CAPACITY { 最大记录数 **}]**

[RECORDS PER HOUR *数字* **]**

;

默认设置

类描述的默认设置是：

```
LABEL （类名）
INDEX BY DAY
MAX INDEXES 9
RECORDS PER HOUR 12
```

要使用默认设置，请只输入 CLASS 关键字及*类名*和数字*类 ID 号*。

CLASS

类名和类 ID 标识来自特定数据源的一组度量。

语法

CLASS *类名* **=** *类 ID 号*

如何使用

*类名*和*类 ID 号*必须符合以下要求：

- *类名*是字母数字，最多可包含 20 个字符。名称必须以字母字符开头，可包含下划线（但不能包含特殊字符）。
- *类 ID 号*必须是数字，最多可包含六位数。
- *类名*和*类 ID 号*都不区分大小写。

- *类名*和*类 ID 号*在定义的所有类中都必须是唯一的，而且不能与性能收集组件的 parm 文件中定义的任何应用程序相同。（有关 parm 文件的信息，请参见《HP Operations Agent for UNIX 用户手册》的第 2 章。）

示例

```
CLASS VMSTAT_STATS = 10001;
```

LABEL

类标签将类作为整体进行标识。在 Performance Manager 中用它代替类名。

语法

[LABEL " *类标签名称* " **]**

如何使用

*类标签名称*必须符合以下要求：

- 必须包含在双引号中。
- 最多可包含 48 个字符。
- 不能与 DSI 类规范语法的任何关键字元素 （如 CAPACITY、 ACTION 等）相同。
- 如果它包含双引号，则前面要有反斜杠 (\)。例如，如果标签是 "my" data，则输入 **"\"my\" data"**。
- 如果没有指定标签，则使用*类名*作为默认值。

示例

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data";
```

INDEX BY、 MAX INDEXES 和 ROLL BY

INDEX BY、MAX INDEXES 和 ROLL BY 设置可用于指定如何存储数据，以及何时丢弃数据。用这些设置指定要存储的数据块、最大存储块数以及数据达到其最大索引值时要丢弃的数据块大小。

语法

[INDEX BY {HOUR | DAY | MONTH} MAX INDEXES *数字* **]**
[[ROLL BY {HOUR | DAY | MONTH} [ACTION " *操作* **"]]**

如何使用

INDEX BY 设置允许在达到类容量时，从类中滚出数据块。 INDEX BY 和 RECORDS PER HOUR 选项可用于间接设置类的容量 （如后面的控制日志文件大小中所述）。

INDEX BY 设置不能超过 ROLL BY 设置。例如， INDEX BY DAY 不能与 ROLL BY HOUR 一起使用，但 INDEX BY HOUR 可以与 ROLL BY DAY 一起使用。

如果未指定 ROLL BY，则使用 INDEX BY 设置。达到容量时，将释放最早的滚动间隔内记录的所有记录，以供重用。

在丢弃 （滚动）数据之前，将执行所有指定的 ACTION。这一可选 ACTION 可用于在从类删除数据之前将它们导出到另一位置。有关导出数据的信息，请参见第 15 章 DSI 程序参考。

滚动操作备注

ACTION 语句中指定的 UNIX 命令不能在后台运行。另外，不要在 ACTION 语句中指定将导致长时延迟的命令，因为在延迟期间不会记录新数据。

如果命令超过一行，则用双引号标记每行的开始和结束。必要时在引号内包含空格，以确保连接各行时不同命令行选项仍是分开的。

如果命令包含双引号，则前面要有反斜杠 (\)。

ACTION 语句不能超过 199 个字符。

在 ACTION 语句中，可使用宏定义要滚出日志文件的数据的时间窗口。这些宏由 dsilog 展开。可使用 PT_START 指定要滚出的数据块的开始 （UNIX 时间，即自 1970 年 1 月 1 日 00:00:00 起经过的秒数），用 PT_END 指定数据的结束 （UNIX 时间）。与 extract 程序结合使用以在覆盖数据之前导出数据时，这些选项尤其有用。

如果使用宏，则对照 199 个字符的限制使用其展开长度。

示例

以下示例可能有助于阐明 INDEX BY、 MAX INDEXES 与 ROLL BY 子句之间的关系。

以下示例将 CAPACITY 间接设置为 144 条记录 (1*12*12)。

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
RECORDS PER HOUR 12;
```

以下示例将 CAPACITY 间接设置为 1440 条记录 (1*12*120)。

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
RECORDS PER HOUR 120;
```

以下示例显示 ROLL BY HOUR。

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
ROLL BY HOUR
RECORDS PER HOUR 120;
```

以下示例导致在覆盖数据之前，当前标识为滚动 （除周末）的所有数据都导出到名为 sys.sdl 的文件。请注意，最后一个示例靠最后的两行包含在双引号中，表示它们构成单个命令。

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
ROLL BY HOUR
ACTION "extract -xp -l sdl_new -C SYS_STATS "
"-B $PT_START$ -E $PT_END$ -f sys.sdl, purge -we 17 "
RECORDS PER HOUR 120;
```

其他示例

下面的建议索引设置可帮助您考虑要存储的数据量。

INDEX BY	MAX INDEXES	存储的数据量
HOUR	72	3 天
HOUR	168	7 天
HOUR	744	31 天
DAY	365	1 年
MONTH	12	1 年

下表详细说明了使用 ROLL BY 进行的设置

INDEX BY	MAX INDEXES	ROLL BY	含义
DAY	9	DAY	将在日志文件中存储 9 天的数据。记录第 10 天的数据之前，滚出第 1 天的数据。这些是索引和最大索引的默认值。
HOUR	72	HOUR	将在日志文件中存储 72 小时 （3 天）的数据。记录第 73 小时的数据之前，滚出第 1 个小时的数据。此后，在后续的每个小时的开始，滚出 “最早的”那个小时的数据。
HOUR	168	DAY	将在日志文件中存储 168 小时（7 天）的数据。记录第 169 小时（第 8 天）的数据之前，滚出第 1 天的数据。此后，在后续的每一天的开始，滚出 “最早的”那一天的数据。

INDEX BY	MAX INDEXES	ROLL BY	含义
HOUR	744	MONTH	将在日志文件中存储 744 小时（31 天）的数据。记录第 745 小时 （第 32 天）的数据之前，滚出第 1 个月的数据。此后，在记录第 745 小时的数据之前，滚出“最早的”那个月的数据。 例如， dsilog 开始于 4 月 15 日，记录数据至 5 月 16 日 （744 小时）。在记录第 745 小时 （5 月 17 日的第 1 个小时）的数据之前， dsilog 将滚出 4 月 （4 月 15 - 30 日）的数据。

INDEX BY	MAX INDEXES	ROLL BY	含义
DAY	30	DAY	将在日志文件中存储 30 天的数据。记录第 31 天的数据之前，滚出第 1 天的数据。此后，在后续的每一天的开始，滚出 “最早的”那一天的数据。 例如，dsilog 开始于 4 月 1 日，记录全月的数据，然后，要记录 5 月 1 日 （第 31 天）的数据时，将滚出 4 月 1 日的数据。
DAY	62	MONTH	将在日志文件中存储 62 天的数据。记录第 63 天的数据之前，滚出第 1 个月的数据。此后，在记录第 63 天的数据之前，滚出 “最早的”那个月的数据。 例如，如果 dsilog 开始于 3 月 1 日，记录 3 月和 4 月的数据，日志文件中将有 61 天的数据。一旦 dsilog 记录 5 月 1 日的数据（第 62 天），日志文件将满。dsilog 必须滚出整个 3 月的数据才能记录 5 月 2 日的数据。
MONTH	2	MONTH	将在日志文件中存储两个月的数据。记录第 3 个月的数据之前，滚出第 1 个月的数据。此后，在后续的每个月的开始，滚出 “最早的”那个月的数据。 例如，dsilog 开始于 1 月 1 日，记录 1 月和 2 月的数据。dsilog 必须滚出 1 月的数据才能记录 3 月的数据。

控制日志文件大小

确定每个类中要存储的数据量，以及为了给新数据留出空间而需丢弃的数据量。

类容量根据 INDEX BY（小时、天或月）、RECORDS PER HOUR 和 MAX INDEXES 计算。以下示例显示不同设置的结果。

在此示例中，类容量是 288 （24 条索引 * 12 条记录 / 小时）。

```
INDEX BY HOUR
MAX INDEXES 24
RECORDS PER HOUR 12
```

在此示例中，类容量是 504 （7 天 * 24 小时 / 天 * 3 条记录 / 小时）。

```
INDEX BY DAY
MAX INDEXES 7
RECORDS PER HOUR 3
```

在此示例中，类容量是 14,880 （2 个月 * 31 天 / 月 * 24 小时 / 天 * 10 条记录 / 小时）。

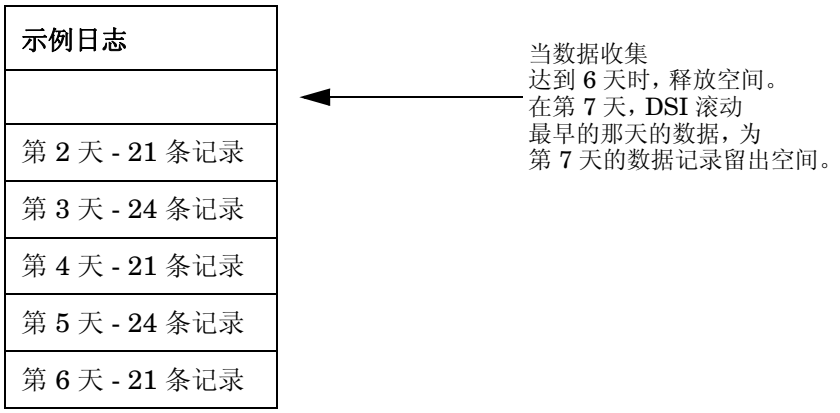
```
INDEX BY MONTH
MAX INDEXES 2
RECORDS PER HOUR 10
```

如果不指定 INDEX BY、 RECORDS PER HOUR 和 MAX INDEXES 的值， DSI 将使用类描述的默认设置。请参见本章前面的类描述下的 “默认设置”。

可使用 ROLL BY 选项确定每次达到类记录容量时要丢弃的数据量。ROLL BY 的设置受 INDEX BY 设置的限制， ROLL BY 单位 （小时、天、月）不能小于 INDEX BY 单位。

以下图例演示了下例的数据是如何滚动的

```
INDEX BY DAY
MAX INDEXES 6
ROLL BY DAY
```



在上面的示例中，类容量按以下设置限制为 6 天的数据：

```
MAX INDEXES 6。
```

而以下设置指示删除一天的数据：

```
ROLL BY DAY。
```

第 7 天的数据到达时，将丢弃最早的那天的数据。请注意，在记录进程开始时，不会丢弃数据。只有在第 7 天结束时，该类第一次满后，才会每天发生一次滚动。

RECORDS PER HOUR

RECORDS PER HOUR 设置确定每小时写入日志文件的记录数。RECORDS PER HOUR 的默认数是 12，与性能收集组件每 5 分钟一次 （60 分钟 /12 条记录 = 每 5 分钟记录一次）的数据采样测量间隔保持一致。

默认数或您输入的数字可能要求记录进程先汇总数据，然后才能成为日志文件的一部分。度量描述中指定了用于汇总每个数据项的方法。有关详细信息，请参见本章后面的汇总方法。

语法

[RECORDS PER HOUR *数字* **]**

如何使用

记录进程使用此值汇总传入数据，生成指定的记录数。例如，如果数据每分钟到达一次，而您将 RECORDS PER HOUR 设置为 6 （每 10 分钟一次），则对 10 个数据点进行汇总，将每条记录写入类。某些常见的 RECORDS PER HOUR 设置如下所示：

```
RECORDS PER HOUR 6 --> 1 条记录 /10 分钟
RECORDS PER HOUR 12 --> 1 条记录 /5 分钟
RECORDS PER HOUR 60 --> 1 条记录 / 分钟
RECORDS PER HOUR 120 --> 1 条记录 /30 秒
```

备注

RECORDS PER HOUR 在 dsilog 中可以用 -s seconds 选项覆盖。但是，覆盖原始设置可能导致 Performance Manager 绘制数据图形时出现问题。

如果 dsilog 在整个记录间隔内没有收到度量数据，则为该度量记录一条缺少数据的指示符。在 dsilog 中可使用 -asyn 选项强制 DSI 使用记录的最后一个值。有关 -asyn 选项的描述，请参见第 15 章的 dsilog 记录进程。

示例

在此示例中，每 10 分钟写入一条记录。

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
RECORDS PER HOUR 6;
```

CAPACITY

CAPACITY 是类中要存储的记录数。

语法

[CAPACITY { *最大记录数* **}]**

如何使用

类容量根据 RECORDS PER HOUR、INDEX BY 和 MAX INDEXES 中的设置得出。除非指定的容量大于从这些其他设置得出的值，否则将忽略 CAPACITY 设置。如果发生这种情况，则增大 MAX INDEXES 设置来提供指定的容量。

示例

```
INDEX BY DAY
MAX INDEXES 9
RECORDS PER HOUR 12
CAPACITY 3000
```

在上面的示例中，得到的类容量是 2,592 条记录 （9 天 * 24 小时 / 天 * 12 条记录 / 小时）。

3000 大于 2592，因此 sdlcomp 将 MAX INDEXES 增大到 11，产生 3168 的类容量。编译之后，通过运行带 -decomp 选项的 sdlutil 可以看到生成的 MAX INDEXES 和 CAPACITY 值。

度量描述

类规范文件中的度量描述用于定义类的各个数据项。度量描述等于度量名称加数字标识符，它指定由于每小时到达的记录多于已用 RECORDS PER HOUR 设置指定的值而必须汇总数据时要使用的方法。

用户定义的描述 （如*度量标签名称*）不能与 DSI 类规范语法的任何关键字元素相同。

请注意， dsilog 格式文件最多只能有 100 个度量。

```
METRICS

metric_name = metric_id_number
[LABEL "metric_label_name"]
[TOTALED | AVERAGED | SUMMARIZED BY metric_name]
[MAXIMUM metric_maximum_number]
[PRECISION {0 | 1 | 2 | 3 | 4 | 5} ]
TYPE TEXT LENGTH "length"
```

对于数字度量，可指定汇总方法 （TOTALED、 AVERAGED、 SUMMARIZED BY）和 PRECISION。对于文本度量，只能指定 TYPE TEXT LENGTH。

METRICS

度量名称和 ID 号标识收集的度量。

语法

METRICS
metric_name **=** *metric_id_number*

如何使用

度量部分必须以 METRICS 关键字开头，然后才是第一个度量定义。每个度量的度量名称都必须符合以下要求：

- 不能超过 20 个字符。
- 必须以字母字符开头。
- 只能包含字母数字字符和下划线。
- 不区分大小写。

度量还有一个度量 ID 号，它不能超过 6 个字符。

*度量名称*和*度量 ID 号*在类中定义的所有度量中都必须是唯一的。*class_name*:*metric_name* 组合对该系统必须唯一，不能与任何 *application_name*:*metric_name* 相同。

每个度量描述与下一个描述由分号 (;) 分隔。

您可以重用其他任何类中的度量名称，只要该类的数据存储在同一日志文件集，且定义也相同（请参见第 13 章的日志文件的组织方式）。要重用已在同一日志文件集中的另一个类中定义的度量定义，只需指定 *metric_name*，而无需 *metric_id_number* 或其他任何规范。如果要将任何选项设置成与之前定义的度量不同，必须赋予该度量唯一的名称和数字标识符，并重新定义。

类规范这部分中度量名称的顺序决定导出记录的数据时字段的顺序。如果传入数据的顺序不同于此规范中列出的顺序，或者您不想记录传入数据流中的所有数据，请参见第 15 章 DSI 程序参考了解如何将度量映射到正确位置。

时间戳度量自动作为第一个度量插入每个类中。如果希望时间戳显示在导出数据中的不同位置，请在希望它出现的位置包括内部定义的度量定义的简短形式 (DATE_TIME;)。要省略该时间戳而使用作为传入数据一部分的 UNIX 时间戳（自 1970 年 1 月 1 日 00:00:00 起经过的秒数），请在启动 dsilog 进程时选择 -timestamp 选项。

最简单的度量描述使用度量名称作为标签，并使用 AVERAGED、MAXIMUM 100 和 PRECISION 3 小数位的默认设置，它需要以下描述：

```
METRICS
metric_name = metric_id_number
```

必须使用 sdlcomp 编译每个类，然后用 dsilog 进程记录该类的数据，而不管是否已重用度量名称。

示例

```
VM;
```

VM 是重用已在同一日志文件集中的另一类中定义的度量定义的示例。

LABEL

度量标签标识 Performance Manager 图形和导出的数据中的度量。

语法

[LABEL " *度量标签名称* **"]**

如何使用

指定包含在双引号中的文本字符串，以标记图形和导出的数据中的度量。最多可包含 48 个字符。如果未指定标签，则用度量名称标识度量。

备注

如果标签包含双引号，则前面要有反斜杠 (\)。例如，如果标签是 "my" data，则输入 **"\"my\" data"**。

*度量标签名称*不能与 DSI 类规范语法的任何关键字元素（如 CAPACITY、ACTION 等）相同。

示例

```
METRICS
RUN_Q_PROCS  = 106
LABEL  "Procs in run q";
```

汇总方法

汇总方法确定当记录数超过 CLASS 部分的 RECORDS PER HOUR 选项中设置的数字时，如何汇总数据。例如，您可能想将出现的计数加总，但对比率取平均值。汇总方法仅对数字度量有效。

语法

[{TOTALED | AVERAGED | SUMMARIZED BY *metric_name*}]

如何使用

度量不按时间取平均值，而按该类中的另一个度量取平均值时，应使用 SUMMARIZED BY。例如，假定您已定义度量 TOTAL_ORDERS 和 LINES_PER_ORDER。如果每五分钟将这些度量提供给记录进程一次，但每小时只写一次记录，要将 LINES_PER_ORDER 正确地汇总为（总行数 / 总订购数），记录进程必须每五分钟执行以下计算：

- 在每个五分钟间隔结束时用 LINES_PER_ORDER 乘 TOTAL_ORDERS，并将结果保存在总行数的内部连续计数中。
- 保存 TOTAL_ORDERS 的连续计数。
- 在该小时结束时，用 TOTAL_ORDERS 除总行数。

要指定此类计算，将 LINES_PER_ORDER 指定为 SUMMARIZED BY TOTAL_ORDERS。

如果未指定汇总方法，则度量默认为 AVERAGED。

示例

```
METRICS
ITEM_1_3 = 11203
LABEL    "TOTAL_ORDERS"
TOTALED;
ITEM_1_5   = 11205
LABEL   "LINES_PER_ORDER"
SUMMARIZED BY ITEM_1_3;
```

PRECISION

PRECISION 标识用于度量值的小数位数。如果未指定 PRECISION，将根据指定的 MAXIMUM 计算。如果两者皆未指定，默认的 PRECISION 值是 3。此设置仅对数字度量有效。

语法

[PRECISION{0|1|2|3|4|5}]

如何使用

PRECISION 设置确定可记录的最大值。使用 PRECISION 0 表示整数。

PRECISION	小数位数	最大可接受数	MAXIMUM
0	0	2,147,483,647	> 10,000
1	1	214,748,364.7	1001 到 10,000
2	2	21,474,836.47	101 到 1,000
3	3	2,147,483.647	11 到 1,000
4	4	214,748.3647	2 到 10
5	5	21,474.83647	1

示例

```
METRICS
RUN_Q_PROCS        = 106
LABEL   "Procs in run q"
PRECISION 1;
```

TYPE TEXT LENGTH

三个关键字 TYPE TEXT LENGTH 指定度量是文本而不是数字。文本定义为除 ^d、\n 或分隔符（如果有）以外的任何字符。

因为 dsilog 输入信息的数据项之间的默认分隔符是空格，如果文本包含嵌入空格，将需要更改分隔符。如第 15 章 DSI 程序参考中所述用 dsilog -c char 选项指定不同的分隔符。

语法

[TYPE TEXT LENGTH *长度* **]**

如何使用

*长度*必须大于 0 小于 4096。

备注

汇总方法、MAXIMUM 和 PRECISION 不能用文本度量指定。文本不能汇总，这意味着 dsilog 将采用间隔内的第一个记录值并忽略其余的值。

示例

```
METRICS
text_1 = 16
LABEL "first text metric"
TYPE TEXT LENGTH 20;
```

类规范示例

```
CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
ROLL BY HOUR
RECORDS PER HOUR 120;


METRICS

RUN_Q_PROCS       = 106
LABEL   "Procs in run q"
PRECISION 0;

BLOCKED_PROCS   = 107
LABEL   "Blocked Processes"
PRECISION 0;

SWAPPED_PROCS     = 108
LABEL     "Swapped Processes"
PRECISION 0;

AVG_VIRT_PAGES    = 201
LABEL     "Avg Virt Mem Pages"
PRECISION 0;

FREE_LIST_SIZE    = 202
LABEL     "Mem Free List Size"
PRECISION 0;

PAGE_RECLAIMS     = 303
LABEL     "Page Reclaims"
PRECISION 0;


ADDR_TRANS_FAULTS = 304
LABEL     "Addr Trans Faults"
PRECISION 0;

PAGES_PAGED_IN    = 305
LABEL     "Pages Paged In"
PRECISION 0;



PAGES_PAGED_OUT   = 306
LABEL     "Pages Paged Out"
PRECISION 0;

PAGES_FREED       = 307
LABEL     "Pages Freed/Sec"
PRECISION 0;
```

```
MEM_SHORTFALL     = 308
LABEL     "Exp Mem Shortfall"
PRECISION 0;

CLOCKED_PAGES     = 309
LABEL     "Pages Scanned/Sec"
PRECISION 0;

DEVICE_INTERRUPTS = 401
LABEL     "Device Interrupts"
PRECISION 0;

SYSTEM_CALLS      = 402
LABEL     "System Calls"
PRECISION 0;

CONTEXT_SWITCHES  = 403
LABEL     "Context Switches/Sec"
PRECISION 0;

USER_CPU          = 501
LABEL     "User CPU"
PRECISION 0;

SYSTEM_CPU        = 502
LABEL     "System CPU"
PRECISION 0;

IDLE_CPU          = 503
LABEL     "Idle CPU"
PRECISION 0;
```

15 DSI 程序参考

本章提供有关以下内容的详细参考信息：

- sdlcomp 编译器
- 配置文件 datasources 和 alarmdef
- dsilog 记录进程
- 用 性能收集组件 extract 程序导出 DSI 数据
- sdlutil 数据源管理实用程序

sdlcomp 编译器

sdlcomp 编译器检查类规范文件中是否有错误。如果未发现错误，它将在您指定的日志文件集的描述文件中添加类描述和度量描述。它还在日志文件集的根文件中设置指针，指向要用于数据存储的日志文件。如果日志文件集或日志文件不存在，则由编译器创建。

可通过在编译器命令中指定完整路径，将 DSI 文件放在系统上的任何位置。但是，一旦指定了路径，就*不能*将 DSI 日志文件移到其他目录。SDL62 是相关的类规范错误消息，具体请参见第 17 章的 SDL 错误消息。 DSI 用于类规范错误消息的格式是前缀 SDL（自描述日志文件）后跟消息号。

编译器语法

```
sdlcomp [-maxclass 值] specification_file
   [logfile_set[log file]] [选项]
```

变量和选项	定义
-maxclass *值*	可用于指定创建新日志文件集时要提供的最大类数。如果以现有日志文件集的名称使用它，则忽略此选项。不管是否使用，每个新添加的类都会占用约 500 个字节的磁盘空间开销。如果不指定 -maxclass，默认值是 10。
specification_file	是包含类规范的文件的名称。如果它不在当前目录中，则必须使用完全限定名称。
logfile_set	是应添加此类的日志文件集的名称。
log file	是日志文件集中将包含此类数据的日志文件。如果不指定日志文件，则为此类新建一个日志文件并自动命名该日志文件。
-verbose	将编译器输出的详细描述打印到 stdout。
-vers	显示版本信息。
-?	显示语法描述。
-u	允许每秒记录多条记录。使用此选项只能记录未汇总的数据。

编译器输出示例

给定以下命令行：

```
->sdlcomp vmstat.spec sdl_new
```

以下代码是成功编译的输出示例。请注意， `vmstat.spec` 就是上一章中所示的规范文件示例。

```
sdlcomp
Check class specification syntax.

CLASS VMSTAT_STATS = 10001
LABEL "VMSTAT data"
INDEX BY HOUR
MAX INDEXES 12
ROLL BY HOUR
RECORDS PER HOUR 120;

METRICS

RUN_Q_PROCS       = 106
LABEL   "Procs in run q"
PRECISION 0;

BLOCKED_PROCS   = 107
LABEL   "Blocked Processes"
PRECISION 0;

SWAPPED_PROCS     = 108
LABEL     "Swapped Processes"
PRECISION 0;

AVG_VIRT_PAGES    = 201
LABEL     "Avg Virt Mem Pages"
PRECISION 0;

FREE_LIST_SIZE    = 202
LABEL     "Mem Free List Size"
PRECISION 0;

PAGE_RECLAIMS     = 303
LABEL     "Page Reclaims"
PRECISION 0;
ADDR_TRANS_FAULTS = 304
LABEL     "Addr Trans Faults"
PRECISION 0;

PAGES_PAGED_IN    = 305
LABEL     "Pages Paged In"
PRECISION 0;

PAGES_PAGED_OUT   = 306
LABEL     "Pages Paged Out"
PRECISION 0;
```

```
PAGES_FREED       = 307
LABEL     "Pages Freed/Sec"
PRECISION 0;

MEM_SHORTFALL     = 308
LABEL     "Exp Mem Shortfall"
PRECISION 0;

CLOCKED_PAGES     = 309
LABEL     "Pages Scanned/Sec"
PRECISION 0;

DEVICE_INTERRUPTS = 401
LABEL     "Device Interrupts"
PRECISION 0;

SYSTEM_CALLS      = 402
LABEL     "System Calls"
PRECISION 0;

CONTEXT_SWITCHES  = 403
LABEL     "Context Switches/Sec"
PRECISION 0;

USER_CPU          = 501
LABEL     "User CPU"
PRECISION 0;

SYSTEM_CPU        = 502
LABEL     "System CPU"
PRECISION 0;

IDLE_CPU          = 503
LABEL     "Idle CPU"
PRECISION 0;
Note:  Time stamp inserted as first metric by default.

Syntax check successful.

Update SDL sdl_new.
Open SDL sdl_new
Add class VMSTAT_STATS.
Check class VMSTAT_STATS.

Class VMSTAT_STATS successfully added to log file set.
```

有关错误消息和恢复的说明，请参见第 17 章错误消息。

配置文件

开始记录数据前，您可能需要更新两个性能收集组件配置文件：

- `/var/opt/OV/conf/perf/datasources`
- `/var/opt/perf/alarmdef` - 有关使用 `alarmdef` 配置文件的信息，请参见下一部分定义 DSI 度量的警报。

定义 DSI 度量的警报

可使用性能收集组件定义 DSI 度量的警报。当 DSI 度量满足或超过您定义的条件时，这些警报会通知您。要定义警报，请指定满足或超过时将触发警报通知或操作的条件。为通过 DSI 记录的数据定义警报的方式与为其他性能收集组件度量定义的方式相同 - 都在性能收集组件系统上的 `alarmdef` 文件中进行。 `alarmdef` 文件位于性能收集组件的 `var/opt/perf/` 配置目录中。

任何时候在警报定义中指定 DSI 度量名称时，都是使用完全限定名称；即前面有 *datasource_name* 和 *class_name*，如下所示：

datasource_name:*class_name*:`metric_name`

- *datasource_name* 是用于在 `datasources` 文件中配置数据源的名称。
- *class_name* 是用于标识数据源的类规范中类的名称。如果类规范中度量名称是唯一的（未重用），则不需要输入 *class_name*。
- `metric_name` 是数据源的类规范中的数据项。

但是，如果选择不完全限定度量名称，则需要在 `alarmdef` 文件中包括 USE 语句以标识要使用的数据源。有关 USE 语句的详细信息，请参见 《HP Operations Agent for UNIX 用户手册》的第 7 章“性能警报”。

要激活对 `alarmdef` 文件的更改以使警报生成器可以读取，请在命令行中输入 **`ovpa restart alarm`** 命令。

有关警报定义语法、如何处理警报以及自定义警报定义的详细信息，请参见《HP Operations Agent for UNIX 用户手册》的第 7 章。

警报处理

`dsilog` 记录数据时，会将它与 `alarmdef` 文件中的警报定义进行比较，以确定是否满足或超过条件。如果是，将触发警报通知或操作。

您可以配置接收警报通知的目标以及是否执行本地操作。警报通知可发送到 Performance Manager 中央分析系统，您可在该系统中绘制体现系统性能特征的度量图。 SNMP 陷阱可发送到 HP Network Node Manager。可以在性能收集组件系统上执行本地操作。警报信息还可发送到 Operations Manager。

dsilog 记录进程

`dsilog` 进程要求您设计自己的程序或使用已存在的程序来访问数据。然后可以通过管道将此数据发送到 `dsilog`，后者将数据记录到日志文件集。必须对定义的每个类使用单独的记录进程。

dsilog 预期从 stdin 接收数据。要启动记录进程，可如以下示例所示，通过管道将您用于收集数据的进程的输出发送到 dsilog。

```
vmstat 60 | dsilog logfile_set class
```

命令行中只能有一个管道（|）。这是因为如果使用两个管道， UNIX 缓冲将保存第一条命令的输出，直到写入 8000 个字符才继续执行第二条命令并通过管道发送到日志文件。

还可以使用 fifo（命名管道）。例如，

```
mkfifo -m 777 myfifo
dsilog logfile_set class -i myfifo &
vmstat 60 > myfifo &
```

& 导致进程在后台运行。

请注意，如果您打算运行很多 dsilog 进程，可能需要增大 UNIX 内核参数 shmmni 和 nflocks 的值。 shmmni 指定共享内存段的最大数； nflocks 指定系统上文件锁定的最大数。它们的默认值都是 200。每个活动的 DSI 日志文件集都使用一个共享内存段 (shmmni) 和一个或多个文件锁定 (nflocks)。在 HP-UX 上，可以使用 System Administration and Maintenance 实用程序 (SAM) 更改 shmmni 和 nflocks 的设置。

语法

dsilog logfile_set class [*选项* **]**

下面几页描述了 dsilog 参数和选项。

表 1 dsilog **参数和选项**

变量和选项	定义
logfile_set	是要存储数据的日志文件集的名称。如果它不在当前目录中，则必须使用完全限定名称。
class	是要记录的类的名称。
-asyn	指定数据将以 RECORDS PER HOUR 速率异步到达。如果记录间隔内没有数据到达，则重复上一个记录间隔的数据。但是，如果 dsilog 尚未记录任何数据，记录的度量值将被视为缺少数据。这会导致在数据图形显示中绘制水平线，如果导出数据，则在每个记录中重复数据。
-c char	使用指定的字符作为字符串分隔符。不能使用以下字符作为分隔符：小数点、负号、 ^z、 \n。如果任何文本度量中有嵌入空格，则必须使用此选项指定唯一分隔符。
-f *格式文件*	指定对将输入到记录进程的数据进行描述的文件。如果不指定此选项， dsilog 在以下情况下从类规范获得输入格式。有关详细信息，请参见本章后面的创建格式文件。 输入记录中的每个数据项都对应于类规范中定义的度量。 类规范中定义度量的顺序与输入记录中作为数据项显示度量的顺序相同。 如果输入记录中的数据项比度量定义多， dsilog 将忽略所有多出的数据项。
-f *格式文件*（续）	如果类规范列出的度量定义比输入数据项多，导出数据时字段会显示 “缺少” 数据，在分析软件中绘制数据图时该度量将无任何可用数据。 格式文件中的字段数限制为 100。

表 1　　dsilog 参数和选项

变量和选项	定义
-i fifo 或 ASCII 文件	指示输入应来自指定的 fifo 或 ASCII 文件。如果不使用此选项，输入则来自 stdin。如果使用此方法，请在启动收集进程之前启动 dsilog。有关使用 fifo 的详细信息，请参见手册页 mkfifo。有关示例，另请参阅第 16 章数据源集成示例。
-s seconds	是汇总数据的间隔秒数。 -s 选项覆盖类规范中的 RECORDS PER HOUR 的汇总间隔和汇总速率默认设置。如果此选项存在，它将覆盖 RECORDS PER HOUR 的值。 零 (0) 值关闭汇总，表示记录所有传入数据。使用 -s 0 选项要小心，因为 dsilog 将在到达该点时对日志数据加盖时间戳。这可能导致 Performance Manager 和 perfalarm 出现问题，因为使用固定间隔的时间戳时它们的工作状态最佳。如果 Performance Manager 将访问该日志文件，则最好不要用 -s 0 选项。
-t	将记录的所有内容以 ASCII 格式打印到 stdout。
-timestamp	指示记录进程不要提供时间戳，而是使用输入数据中已提供的时间戳。传入数据中的时间戳必须用 UNIX 时间戳格式（自 1970 年 1 月 1 日 00:00:00 起经过的秒数）表示当地时间。
-vi	过滤通过 dsilog 的输入，并将错误写入 stdout 而不是日志文件。它不将记录的实际数据写入 stdout（请参见下面的 -vo 选项）。此选项可用于检查输入有效性。
-vo	过滤通过 dsilog 的输入，并将记录的实际数据和错误写入 stdout 而不是日志文件。此选项可用于检查数据汇总的有效性。
-vers	显示版本信息
-?	显示语法描述。

dsilog 如何处理数据

dsilog 程序扫描每个输入数据字符串，将分隔的字段解析为各个数字度量或文本度量。预测将如何处理数据的关键规则是输入字符串的有效性。有效的输入字符串要求任何指定的度量类型（数字或文本）之间都有分隔符。空格是默认分隔符，但可以用 dsilog -c char 命令行选项指定不同的分隔符。

任何提供给 DSI 的记录的末尾都*必须*有换行字符，这样 DSI 才能正确解译它。

用 sdlgendata 测试记录进程

开始记录数据之前，可用 sdlgendata 程序测试编译的日志文件集和记录进程。 sdlgendata 发现类的度量（如类规范中所述），并为类中的每个度量生成数据。

语法

sdlgendata logfile_set class [*选项* **]**

下面说明了 sdlgendata 参数和选项。

表 2 sdlgendata **参数和选项**

变量和选项	定义
logfile_set	是要为其生成数据的日志文件集名称。
class	是要为其生成数据的数据类。
-timestamp [*数字*]	为时间戳提供数据。如果提供了负数或未提供数字，则对 timestamp 使用当前时间。如果使用正数，则时间开始于 0，并对每条新的数据记录递增*数字*。
-wait *数字*	导致在生成的记录之间等待*数字*秒。
-cycle 数字	在*数字*个周期后循环数据。
-vers	显示版本信息。
-?	显示语法描述。

通过管道将 sdlgendata 输出发送到带 -vi 或 -vo 选项的 dsilog，以在开始用自己的进程或程序记录之前验证输入 (-vi) 和输出 (-vo)。

完成测试之后，删除该测试创建的所有日志文件。否则，这些文件保留为日志文件测试的一部分。

使用以下命令通过管道将数据从 sdlgendata 发送到记录进程。-vi 选项指定丢弃数据并将错误写入 stdout。按 **CTRL+C** 或其他中断控制字符停止数据的生成。

```
sdlgendata logfile_set class -wait 5 | dsilog \
logfile_set class -s 10 -vi
```

上面的命令生成如下所示的数据：

```
dsilog
I:  744996402  1.0000  2.0000  3.0000   4.0000   5.0000   6.0000    7.0000
I:  744996407  2.0000  3.0000  4.0000   5.0000   6.0000   7.0000    8.0000
I:  744996412  3.0000  4.0000  5.0000   6.0000   7.0000   8.0000    9.0000
I:  744996417  4.0000  5.0000  6.0000   7.0000   8.0000   9.0000   10.0000
I:  744996422  5.0000  6.0000  7.0000   8.0000   9.0000  10.0000   11.0000
I:  744996427  6.0000  7.0000  8.0000   9.0000  10.0000  11.0000   12.0000
I:  744996432  7.0000  8.0000  9.0000  10.000   11.000   12.0000   13.0000
I:  744996437  8.0000  9.0000  10.0000  11.0000  12.0000  13.0000   14.0000
```

还可以使用 dsilog 的 -vo 选项检查真实数据的输入和汇总后的输出，而不实际记录它。以下命令按 5 秒间隔将 vmstat 通过管道发送到 dsilog，然后将它汇总到 10 秒。

```
->vmstat 5 | dsilog logfile_set class -s 10 -vo
dsilog
I: 744997230  0.0000  0.0000  21.0000  2158.0000  1603.0000  2.0000  2.0000
I: 744997235  0.0000  0.0000  24.0000  2341.0000  1514.0000  0.0000  0.0000
interval marker
L: 744997230  0.0000  0.0000  22.5000  2249.5000  1558.5000  1.0000  1.0000

I: 744997240  0.0000  0.0000  23.0000  2330.0000  1513.0000  0.0000  0.0000
I: 744997245  0.0000  0.0000  20.0000  2326.0000  1513.0000  0.0000  0.0000
interval marker
L: 744997240  0.0000  0.0000  21.5000  2328.0000  1513.0000  0.0000  0.0000

I: 744997250  0.0000  0.0000  22.0000  2326.0000  1513.0000  0.0000  0.0000
I: 744997255  0.0000  0.0000  22.0000  2303.0000  1513.0000  0.0000  0.0000
interval marker
L: 744997250  0.0000  0.0000  22.0000  2314.5000  1513.0000  0.0000  0.0000

I: 744997260  0.0000  0.0000  22.0000  2303.0000  1512.0000  0.0000  0.0000
I: 744997265  0.0000  0.0000  28.0000  2917.0000  1089.0000  9.0000 33.0000
interval marker
L: 744997260  0.0000  0.0000  25.0000  2610.0000  1300.5000  4.5000 16.5000

I: 744997270  0.0000  0.0000  28.0000  2887.0000  1011.0000  3.0000  9.0000
I: 744997275  0.0000  0.0000  27.0000  3128.0000   763.0000  8.0000  6.0000
interval marker
L: 744997270  0.0000  0.0000  27.5000  3007.5000   887.0000  5.5000 12.5000
```

还可以用 dsilog -vo 选项使用旧数据的文件进行测试，只要数据包含自己的 UNIX 时间戳（自 1970 年 1 月 1 日 00:00:00 起经过的秒数）。要使用旧数据的文件，请输入如下命令：

dsilog -timestamp -vo < *旧文件* **>**

创建格式文件

在以下情况下，创建格式文件将数据输入映射到类规范：

- 数据输入包含类规范中未包括的数据。
- 传入数据的度量顺序与类规范中指定的顺序不同。

格式文件是 ASCII 文本文件，可以用 vi 或任何文本编辑器创建。使用 dsilog 中的 -f 选项指定格式文件的完全限定名称。

因为记录进程通过搜索分隔符 （默认的空格或用户使用 dsilog -c 选项定义的分隔符）后的第一个有效字符开始处理下一个度量，格式文件只是告诉记录进程跳过哪些字段，以及将哪些度量名称与未跳过的字段关联。

$numeric 告诉记录进程跳过一个数字度量字段并转到下一个。$any 告诉记录进程跳过一个文本度量字段并转到下一个。请注意，格式文件字段数限制为 100。

例如，如果传入数据流包含以下信息：

```
ABC 987 654 123 456
```

而您要只将第一个数字字段记录到名为 metric_1 的度量中，格式文件可能如下所示：

```
$any metric_1
```

它告诉记录进程只记录第一个数字字段中的信息，并丢弃其余数据。要只记录第三个数字字段中的信息，格式文件可能如下所示：

```
$any $numeric $numeric metric_1
```

要以相反顺序记录所有四个数字数据项，格式文件可能如下所示：

```
$any metric_4 metric_3 metric_2 metric_1
```

如果传入数据流包含以下信息：

```
/users    15.9    3295    56.79%    xdisk1 /dev/dsk/
c0d0s*
```

而您要只将第一个文本度量和前两个数字字段分别记录到名为 *text_1*、*num_1* 和 *num_2* 的度量字段中，格式文件可能如下所示：

```
text_1   num_1   num_2
```

它告诉记录进程只记录前三个字段中的信息，并丢弃其余数据。

要记录所有数据，但丢弃第三个度量后的 “%”，格式文件可能如下所示：

```
text_1   num_1   num_2   num_3   $any   text_2   text_3
```

由于您要记录数字字段，而 “%”被视为文本字段，您必须跳过它才能正确记录后面的文本字段。

要以不同顺序记录数据项，格式文件可能如下所示：

```
text_3   num_2   num_1   num_3   $any   text_2   text_1
```

请注意，如果在类规范中将 *text_1* 定义为六个字符，将导致只记录 *text_3* 的前六个字符。要将 *text_3* 的*整体*记录为第一个值，请更改类规范并改变数据流以留出额外空间。

更改类规范

要更改类规范文件，必须如下所示重新创建整个日志文件集：

1. 停止 dsilog 进程。
2. 如果要保存数据，或将旧数据与您要记录的新数据集成，则使用 UNIX 时间戳选项从现有日志文件导出数据。有关如何执行该操作的信息，请参见本章后面的导出 DSI 数据。
3. 运行 sdlutil 删除日志文件集。有关如何执行该操作的信息，请参见本章后面的用 sdlutil 管理数据。
4. 更新类规范文件。
5. 运行 sdlcomp 重新编译类规范。
6. 也可以使用 dsilog 中的 -i 选项与步骤 2 中导出的旧数据集成。您可能需要用 -f *格式文件*选项操作数据使之与新数据匹配
7. 运行 dsilog 开始根据新的类规范记录。
8. 只要尚未更改日志文件集的名称或位置，就不需要更新 datasources 文件。

导出 DSI 数据

要从 DSI 日志文件导出数据，请使用性能收集组件 `extract` 程序的 `export` 函数。有关使用 `extract` 导出数据的详细信息，请参见《HP Operations Agent for UNIX 用户手册》的第 5 章和第 6 章。下一页提供了使用命令行参数导出 DSI 数据的示例。

可使用几种方法找出可从 DSI 日志文件导出的类和度量。可以用 `sdlutil` 列出此信息，如本章后面的用 sdlutil 管理数据中所述。也可以用 `extract guide` 命令创建导出模板文件，该文件列出 DSI 日志文件中的类和度量。随后可以用 `vi` 编辑、命名和保存该文件。导出模板文件用于指定导出格式，如 《HP Operations Agent for UNIX 用户手册》的第 5 章和第 6 章中所述。

您*必须*是根用户或日志文件的创建者才能导出 DSI 日志文件数据。

使用 extract 导出 DSI 日志文件数据的示例

```
extract -xp -l logfile_set -C class [ 选项 ]
```

可以用 `extract` 命令行选项执行以下操作：

- 指定导出输出文件。
- 设置要导出的第一个和最后一个间隔的开始和结束日期和时间。
- 只导出特定时间 （轮换）之间的数据。
- 排除一周内某几天的数据 （如周末）。
- 指定报告中放在度量之间的分隔符。
- 选择是否显示无数据到达的间隔的标题和空白记录，以及应对缺少的数据或空数据显示的值。
- 以 UNIX 格式或日期和时间格式显示导出的日期 / 时间。
- 设置额外的汇总级别。

在 Performance Manager 中查看数据

为了在 Performance Manager 中显示 DSI 日志文件中的数据，需要将 DSI 日志文件配置为性能收集组件数据源。开始记录数据之前，请通过将数据源添加到性能收集组件系统上的 `datasources` 文件来配置它。

可以用 Performance Manager 集中查看、监视、分析、比较和预测 DSI 数据中的趋势。Performance Manager 有助于您识别当前和潜在的问题。它提供在用户生产率受到影响之前就解决问题所需的信息。

用 sdlutil 管理数据

要管理 DSI 日志文件中的数据，请使用 `sdlutil` 程序执行以下任何任务：

- 在 `stdout` 中列出当前定义的类和度量信息。可将输出重定向到文件。
- 在 `stdout` 中列出类的完整统计信息。
- 显示列出的所有度量的度量描述。
- 列出某个日志文件集中的文件。
- 从日志文件集删除类和数据。
- 根据日志文件集中的信息重新创建类规范。
- 显示版本信息。

语法

`sdlutil logfile_set [` *选项* **`]`**

变量和选项	定义
`logfile_set`	是通过编译类规范创建的日志文件集的名称。
`-classes` *类列表*	提供列出的所有类的类描述。如果不列出类，则提供所有类的类描述。用空格分隔*类列表*中的项。
`-stats` *类列表*	提供列出的所有类的完整统计信息。如果不列出类，则提供所有类的完整统计信息。用空格分隔*类列表*中的项。
`-metrics` *度量列表*	提供*度量列表*中所有度量的度量描述。如果不列出度量，则提供所有度量的度量描述。用空格分隔*度量列表*中的项。
`-id`	显示日志文件使用的共享内存段 ID。
`-files`	列出日志文件集中的所有文件。
`-rm all`	从日志文件中删除所有类和数据及其数据和共享内存 ID。

变量和选项	定义
-decomp *类列表*	根据日志文件集中的信息重新创建类规范。结果写入 stdout，如果要更改文件并重用它，应将结果重定向到文件。用空格分隔*类列表*中的项。
-vers	显示版本信息。
-?	显示语法描述。

16 数据源集成示例

数据源集成是一种非常强大和灵活的技术。 DSI 的实现可以很简单直观，也可以很复杂。

本章包含使用 DSI 执行以下任务的示例：

- 编写 `dsilog` 脚本
- 记录 `vmstat` 数据
- 记录 `sar` 数据
- 记录 `who` 字数统计

编写 dsilog 脚本

dsilog 代码设计为接收连续的数据行的流作为输入。 dsilog 按照每个类的规范指令对这一输入流进行汇总，在每个请求的汇总间隔记录一个汇总数据行。日志中写入的时间戳符合预期的汇总速率（每小时记录数）时， Performance Manager 和 perfalarm 的工作状态最佳。允许 dsilog 执行汇总时，会自动匹配。

为每个到达的输入行都执行 dsilog 进程，这可能导致 Performance Manager 和 perfalarm 出现问题。建议不要采用此方法。

- 有问题的 dsilog 脚本
- 推荐的 dsilog 脚本

示例 1 - 有问题的 dsilog 脚本

在以下脚本中，为每个到达的输入行执行新的 dsilog 进程。

```
while :
do
   feed_one_data_row | dsilog sdlname classname
   sleep 50
done
```

示例 2 - 推荐的 dsilog 脚本

在以下脚本中，一个 dsilog 进程接收连续的输入数据流。 feed_one_data_row 写为一个函数，该函数为单个 dsilog 进程提供连续的数据流。

```
# Begin data feed function
feed_one_data_row()
  {
   while :
   do
#  Perform whatever operations necessary to produce one row
#  of data for feed to a dsilog process
   sleep 50
   done
  }
#  End data feed function


#  Script mainline code
   feed_one_data_row | dsilog sdlname classname
```

记录 vmstat 数据

此示例显示如何使用默认设置来设置数据源集成，以记录由 vmstat 报告的前两个值。您可以阅读这一部分概览数据源集成的工作流程，也可以执行每个任务在系统上创建对应的 DSI 日志文件。

实现数据源集成所需的步骤包括：

- 创建类规范文件。
- 编译类规范文件。
- 启动 dsilog 记录进程。

创建类规范文件

类规范文件是创建用于描述类或传入数据集以及类中需要记录为度量的每个数字的文本文件。该文件可用您选择的文本编辑器创建。数据源集成的这一示例文件应在 /tmp/ 目录中创建。

以下示例显示类规范文件，它是描述在名为 VMSTAT_STATS 的类中进行记录的前两个 vmstat 数字所必需的。因为只在此类中定义了两个度量，记录进程将忽略每个 vmstat 输出记录的其余内容。文件中每行后面都有注释行对它们进行说明。

```
CLASS VMSTAT_STATS = 10001;
  # Assigns a unique name and number to vmstat class data.
  # The semicolon is required to terminate the class section
  # of the file.

METRICS
  # Indicates that everything that follows is a description
  # of a number (metric) to be logged.

RUN_Q_PROCS = 106;
      # Assigns a unique name and number to a single metric.
      # The semicolon is required to terminate each metric.

BLOCKED_PROCS = 107;
      # Assigns a unique name and number to another metric.
# The semicolon is required to terminate each metric.
```

编译类规范文件

使用 sdlcomp 编译类规范文件时，将检查文件是否有语法错误。如果未发现错误，sdlcomp 将创建或更新一组日志文件保存该类的数据。

使用赋予类规范文件的文件名，然后为*日志文件集名称*指定一个容易记住其日志文件包含哪类数据的名称。在下面的命令和编译器输出示例中，/tmp/vmstat.spec 用作文件名，/tmp/VMSTAT_DATA 用于日志文件集。

```
-> sdlcomp /tmp/vmstat.spec /tmp/VMSTAT_DATA
```

```
sdlcomp X.01.04
Check class specification syntax.

CLASS VMSTAT_STATS = 10001;

METRICS
RUN_Q_PROCS        = 106;
BLOCKED_PROCS      = 107;

NOTE: Time stamp inserted as first metric by default.

Syntax check successful.

Update SDL VMSTAT_DATA.
Shared memory ID used by vmstat_data=219

Class VMSTAT_STATS successfully added to log file set.
```

此示例在 /tmp/ 目录中创建名为 VMSTAT_DATA 的日志文件集，此日志文件集除了包括数据文件以外，还包括根文件和描述文件。此日志文件集即可接受记录的数据。如果类规范文件中有语法错误，则显示指示问题的消息，而不创建日志文件集。

启动 dsilog 记录进程

现在可以通过管道直接将 vmstat 的输出发送到 dsilog 记录进程。使用以下命令：

```
vmstat 60 | dsilog /tmp/VMSTAT_DATA VMSTAT_STATS &
```

此命令每 60 秒运行 vmstat，并直接将输出发送到 VMSTAT_DATA 日志文件集中的 VMSTAT_STATS 类。该命令在后台运行。还可以用 remsh 从远程系统提供 vmstat。

请注意，记录进程开始时会生成以下消息：

```
Metric null has invalid data
Ignore to end of line, metric value exceeds maximum
```

出现这条消息的原因是 dsilog 无法记录 vmstat 输出中的标头行。尽管屏幕上显示这条消息，但 dsilog 会继续运行，开始用第一个有效输入行记录数据。

访问数据

可使用 sdlutil 程序报告类的内容：

```
sdlutil /tmp/VMSTAT_DATA -stats VMSTAT_STATS
```

默认情况下，每 5 分钟汇总和记录数据一次。

可使用 extract 程序命令行参数从类中导出数据。例如：

```
extract -xp -l /tmp/VMSTAT_DATA -C VMSTAT_STATS -ut -f stdout
```

请注意，您必须是根用户或日志文件的创建者才能导出 DSI 数据。

从一个文件记录 sar 数据

此示例显示如何使用标准 sar （系统活动报告）实用程序设置若干 DSI 数据收集来提供数据。

使用系统实用程序时，准确理解该实用程序如何报告数据很重要。例如，注意以下两个 sar 命令之间的区别：

```
sar -u 1 1

   HP-UX hpptc99 A.11.00 E 9000/855    04/10/99

   10:53:15    %usr    %sys    %wio   %idle
   10:53:16       2       7       6      85

sar -u 5 2

   HP-UX hpptc99 A.11.00 E 9000/855    04/10/99

   10:53:31    %usr    %sys    %wio   %idle
   10:53:36          4       5       0      91
   10:53:41          0       0       0      99

Average             2       2       0      95
```

正如您看到的，指定大于 1 的迭代值会导致 sar 显示间隔的平均值。此平均值可能对您无关紧要，但它会影响 DSI 类规范文件和数据转换。您应知道 sar 或其他系统实用程序的输出在不同的 UNIX 平台上执行时可能不同。您应在非常熟悉要使用的实用程序后再创建 DSI 类规范文件。

上面的第一个示例使用 sar 通过 sar 的 -u 选项监视 CPU 利用率。如果查看 sar 的手册页，您会看到 -u 选项报告了以用户模式运行的时间 (%usr)、以系统模式运行的时间 (%sys)、有某些进程等待块 I/O 的空闲时间 (%wio) 及其他空闲时间 (%idle)。因为我们对监视长时间段内的 CPU 活动更感兴趣，我们使用不显示平均值的 sar 形式。

创建类规范文件

要完成的第一个任务是创建 DSI 类规范文件。以下是可用于描述传入数据的类规范的示例：

```
#  sar_u.spec
#
#  sar -u class definition for HP systems.
#
#  ==> 1 minute data; max 24 hours; indexed by hour; roll by day
#
```

```
CLASS sar_u = 1000
LABEL "sar -u data"
INDEX BY         hour
MAX INDEXES      24
ROLL BY          day
ACTION "./sar_u_roll $PT_START$ $PT_END$"
RECORDS PER HOUR 60;

METRICS

hours_1 = 1001
LABEL "Collection Hour"
PRECISION 0;

minutes_1 = 1002
LABEL "Collection Minute"
PRECISION 0;

seconds_1 = 1003
LABEL "Collection Second"
PRECISION 0;

user_cpu = 1004
LABEL "%user"
AVERAGED
MAXIMUM 100
PRECISION 0;

sys_cpu = 1005
LABEL "%sys"
AVERAGED
MAXIMUM 100
PRECISION 0;


wait_IO_cpu = 1006
LABEL "%wio"
AVERAGED
MAXIMUM 100
PRECISION 0;

idle_cpu = 1007
LABEL "%idle"
AVERAGED
MAXIMUM 100
PRECISION 0;
```

编译类规范文件

下一个任务是用以下命令编译类规范文件。

```
sdlcomp sar_u.spec sar_u_log
```

sar -u 命令的输出是系统标头行、空白行、选项标头行以及由时间戳后跟要捕获的数据组成的数据行。我们唯一应注意的就是最后一行。因此，我们需要一种机制，只从 sar -u 命令保存输出的最后一行，并将该数据提供给 DSI。

dsilog 预期从 stdin 接收数据。要启动记录进程，可将您使用的进程的输出通过管道发送到 dsilog。但是，命令行中只能有一个管道 (|)。如果使用两个管道， UNIX 缓冲将保存第一条命令的输出，直到写入 8000 个字符才继续执行第二条命令并通过管道发送到日志文件。这样，执行类似下面的命令将不起作用：

```
sar -u 60 1 | tail -1 | dsilog
```

因此我们使用 fifo 作为 DSI 的输入源。但是，这样做也并非毫无问题。

假定我们要使用以下脚本：

```
#!/bin/ksh                    sar_u_feed

# sar_u_feed script that provides sar -u data to DSI via
# a fifo(sar_u.fifo)

while :                          # (infinite loop)
do

# specify a one minute interval using tail to extract the
# last sar output record(contains the time stamp and data),
# saving the data to a file.

/usr/bin/sar -u 60 1 2>/tmp/dsierr | tail -1 > /usr/tmp/sar_u_data

# Copy the sar data to the fifo that the dsilog process is
# reading.

cat /usr/tmp/sar_u_data > ./sar_u.fifo

done
```

遗憾的是，如果按原样运行，此脚本不会产生所需结果。这是因为 cat 命令打开 fifo，写入数据记录，然后关闭 fifo。关闭指示 dsilog 不会继续将数据写入日志，因此 dsilog 写入这条数据记录后即终止。我们需要一个虚拟进程来使 fifo “保持”打开状态。因此，我们需要一个虚拟 fifo 以及为输入打开虚拟 fifo 并为输出打开 sar_u.fifo 的进程。这样就会使 sar_u.fifo 保持打开状态，从而阻止 dsilog 终止。

启动 DSI 记录进程

现在让我们以分步方法将 sar -u 数据输入 dsilog。

1 创建两个 fifo；其中一个是用于使另一个真正的输入 fifo “保持打开状态”的虚拟 fifo。

```
# Dummy fifo.
mkfifo ./hold_open.fifo
# Real input fifo for dsilog
mkfifo ./sar_u.fifo
```

2 用 -i 选项启动 dsilog，指定来自 fifo 的输入。在启动 sar 数据馈送 (sar_u_feed) 之前先启动 dsilog，这很重要。

```
dsilog ./sar_u_log sar_u \
-i ./sar_u.fifo &
```

3 启动使输入 fifo 保持打开状态的虚拟进程。

```
cat ./hold_open.fifo  \
 > ./sar_u.fifo &
```

4 启动 **sar** 数据馈送脚本 (sar_u_feed)。

```
./sar_u_feed &
```

5 sar_u_feed 脚本将不断向 dsilog 提供数据，直到它终止，或使 fifo 保持打开状态的 cat 终止。我们的类规范文件指示 sar_u_log 将按小时建立索引，包含最多 24 小时，在下一天（按日滚动）的开始执行脚本 sar_u_roll。

```
!/bin/ksh                sar_u_roll
#
# Save parameters and current date in sar_u_log_roll_file.
# (Example of adding comments/other data to the roll file).

mydate=`date`
echo "$# $0 $1 $2" >> ./sar_u_log_roll_file
echo $mydate       >> ./sar_u_log_roll_file

extract -l ./sar_u_log -C sar_u -B $1 -E $2 -1 -f \
stdout -xp >> ./sar_u_log_roll_file
```

6 滚动脚本将要滚出的数据保存在 ASCII 文本文件中，该文本文件可用文本编辑器查看，也可以打印到打印机。

从多个文件记录 sar 数据

如果您不只对 CPU 利用率感兴趣，可以用一个类规范文件描述数据，也可以为每个选项创建一个类规范文件并将它们编译到一个日志文件集中。第一个示例显示编译到单个日志文件集中的单独的类规范文件。

在此示例中，我们将监视 CPU 利用率、缓冲区活动
(sar -b) 和系统调用 (sar -c)。以这种方式记录数据需要三个类规范文件、三个 dsilog 进程、三个 dsilog 输入 fifo 和三个脚本来提供 sar 数据。

创建类规范文件

下面是这些选项中每一个的类规范文件。

```
#  sar_u_mc.spec
#
#  sar -u class definition for log files on HP systems.
#
#  ==> 1 minute data; max 24 hours; indexed by hour; roll by day
#

CLASS sar_u = 1000
LABEL "sar -u data"
INDEX BY         hour
MAX INDEXES      24
ROLL BY          day
ACTION "./sar_u_mc_roll $PT_START$ $PT_END$"
RECORDS PER HOUR 60;

METRICS

hours_1 = 1001
LABEL "Collection Hour"
PRECISION 0;

minutes_1 = 1002
LABEL "Collection Minute"
PRECISION 0;

seconds_1 = 1003
LABEL "Collection Second"
PRECISION 0;
user_cpu = 1004
LABEL "%user"
AVERAGED
MAXIMUM 100
PRECISION 0;
```

```
sys_cpu = 1005
LABEL "%sys"
AVERAGED
MAXIMUM 100
PRECISION 0;

wait_IO_cpu = 1006
LABEL "%wio"
AVERAGED
MAXIMUM 100
PRECISION 0;

idle_cpu = 1007
LABEL "%idle"
AVERAGED
MAXIMUM 100
PRECISION 0;

#  sar_b_mc.spec
#
#  sar -b class definition for log files on HP systems.
#
#  ==> 1 minute data; max 24 hours; indexed by hour; roll by day
#

CLASS sar_b = 2000
LABEL "sar -b data"
INDEX BY         hour
MAX INDEXES      24
ROLL BY          day
ACTION "./sar_b_mc_roll $PT_START$ $PT_END$"
RECORDS PER HOUR 60;

METRICS

hours_2 = 2001
LABEL "Collection Hour"
PRECISION 0;

minutes_2 = 2002
LABEL "Collection Minute"
PRECISION 0;

seconds_2 = 2003
LABEL "Collection Second"
PRECISION 0;
```

```
bread_per_sec = 2004
LABEL "bread/s"
PRECISION 0;

lread_per_sec = 2005
LABEL "lread/s"
PRECISION 0;

read_cache = 2006
LABEL "%rcache"
MAXIMUM 100
PRECISION 0;

bwrit_per_sec = 2007
LABEL "bwrit/s"
PRECISION 0;

lwrit_per_sec = 2008
LABEL "lwrit/s"
PRECISION 0;

write_cache = 2009
LABEL "%wcache"
MAXIMUM 100
PRECISION 0;

pread_per_sec = 2010
LABEL "pread/s"
PRECISION 0;

pwrit_per_sec = 2011
LABEL "pwrit/s"
PRECISION 0;

#  sar_c_mc.spec
#
#  sar -c class definition for log files on HP systems.
#
#  ==> 1 minute data; max 24 hours; indexed by hour; roll by day
#

CLASS sar_c = 5000
LABEL "sar -c data"
INDEX BY         hour
MAX INDEXES      24
ROLL BY          day
ACTION "./sar_c_mc_roll $PT_START$ $PT_END$"
RECORDS PER HOUR 60;
```

```
METRICS

hours_5 = 5001
LABEL "Collection Hour"
PRECISION 0;

minutes_5 = 5002
LABEL "Collection Minute"
PRECISION 0;

seconds_5 = 5003
LABEL "Collection Second"
PRECISION 0;

scall_per_sec = 5004
LABEL "scall/s"
PRECISION 0;

sread_per_sec = 5005
LABEL "sread/s"
PRECISION 0;

swrit_per_sec = 5006
LABEL "swrit/s"
PRECISION 0;

fork_per_sec = 5007
LABEL "fork/s"
PRECISION 2;

exec_per_sec = 5008
LABEL "exec/s"
PRECISION 2;

rchar_per_sec = 5009
LABEL "rchar"
PRECISION 0;

wchar_per_sec = 5010
LABEL "wchar/s"
PRECISION 0;
```

下面是提供 sar 数据所需的两个额外脚本。

```
#!/bin/ksh

# sar_b_feed script that provides sar -b data to DSI via
# a fifo (sar_b.fifo)

while :                       # (infinite loop)
do

# specify a one minute interval using tail to extract the
# last sar output record(contains the time stamp and data),
# saving the data to a file.

/usr/bin/sar -b 60 1 2>/tmp/dsierr | tail -1 &> \
/usr/tmp/sar_b_data


# Copy the sar data to the fifo that the dsilog process is reading.

cat /usr/tmp/sar_b_data > ./sar_b.fifo

done

#!/bin/ksh                           sar_c_feed

# sar_c_feed script that provides sar -c data to DSI via
# a fifo(sar_c.fifo)

while :                                 # (infinite loop)
do

# specify a one minute interval using tail to extract the
# last sar output record(contains the time stamp and data),
# saving the data to a file.

/usr/bin/sar -c 60 1 2>/tmp/dsierr | tail -1 > /usr/tmp/sar_c_data


# Copy the sar data to the fifo that the dsilog process is reading.


cat /usr/tmp/sar_c_data > ./sar_c.fifo

done
```

编译类规范文件

将三个规范文件编译到一个日志文件集中：

```
sdlcomp ./sar_u_mc.spec sar_mc_log
sdlcomp ./sar_b_mc.spec sar_mc_log
sdlcomp ./sar_c_mc.spec sar_mc_log
```

启动 DSI 记录进程

回到 sar 数据的分步方法：

1 创建四个 fifo；其中一个是用于使三个真正的输入 fifo “保持打开状态”的虚拟 fifo。

```
# Dummy fifo.
mkfifo ./hold_open.fifo

# sar -u input fifo for dsilog.
mkfifo ./sar_u.fifo

# sar -b input fifo for dsilog.
mkfifo ./sar_b.fifo

# sar -c input fifo for dsilog.
mkfifo ./sar_c.fifo
```

2 用 -i 选项启动 dsilog，指定来自 fifo 的输入。在启动 sar 数据馈送之前先启动 dsilog，这很重要。

```
dsilog ./sar_mc_log sar_u \
-i ./sar_u.fifo &

dsilog ./sar_mc_log sar_b \
-i ./sar_b.fifo &

dsilog ./sar_mc_log sar_c \
-i ./sar_c.fifo &
```

3 启动使输入 fifo 保持打开状态的虚拟进程。

```
cat ./hold_open.fifo \
 > ./sar_u.fifo &

cat ./hold_open.fifo \
 > ./sar_b.fifo &

cat ./hold_open.fifo \
> ./sar_c.fifo &
```

4 启动 sar 数据馈送脚本。

```
./sar_u_feed &

./sar_b_feed &

./sar_c_feed &
```

为多个选项记录 sar 数据

使用 sar 将数据提供给 DSI 的最后一个示例使用一个规范文件，从多个 sar 选项 (ubycwavm) 定义数据。

```
#  sar_ubycwavm.spec
#
#  sar -ubycwavm class definition for HP systems.
#
#  ==> 1 minute data; max 24 hours; indexed by hour; roll by day
#

CLASS sar_ubycwavm = 1000
LABEL "sar -ubycwavm data"
INDEX BY         hour
MAX INDEXES      24
ROLL BY          day
ACTION "./sar_ubycwavm_roll $PT_START$ $PT_END$"
RECORDS PER HOUR 60;

METRICS
hours = 1001
LABEL "Collection Hour"
PRECISION 0;

minutes = 1002
LABEL "Collection Minute"
PRECISION 0;

seconds = 1003
LABEL "Collection Second"
PRECISION 0;

user_cpu = 1004
LABEL "%user"
AVERAGED
MAXIMUM 100
PRECISION 0;

sys_cpu = 1005
LABEL "%sys"
AVERAGED
MAXIMUM 100
PRECISION 0;

wait_IO_cpu = 1006
LABEL "%wio"
AVERAGED
MAXIMUM 100
PRECISION 0;
```

```
idle_cpu = 1007
LABEL "%idle"
AVERAGED
MAXIMUM 100
PRECISION 0;

bread_per_sec = 1008
LABEL "bread/s"
PRECISION 0;

lread_per_sec = 1009
LABEL "lread/s"
PRECISION 0;

read_cache = 1010
LABEL "%rcache"
MAXIMUM 100
PRECISION 0;

bwrit_per_sec = 1011
LABEL "bwrit/s"
PRECISION 0;

lwrit_per_sec = 1012
LABEL "lwrit/s"
PRECISION 0;

write_cache = 1013
LABEL "%wcache"
MAXIMUM 100
PRECISION 0;
pread_per_sec = 1014
LABEL "pread/s"
PRECISION 0;

pwrit_per_sec = 1015
LABEL "pwrit/s"
PRECISION 0;

rawch = 1016
LABEL "rawch/s"
PRECISION 0;

canch = 1017
LABEL "canch/s"
PRECISION 0;
```

```
outch = 1018
LABEL "outch/s"
PRECISION 0;

rcvin = 1019
LABEL "rcvin/s"
PRECISION 0;

xmtin = 1020
LABEL "xmtin/s"
PRECISION 0;

mdmin = 1021
LABEL "mdmin/s"
PRECISION 0;

scall_per_sec = 1022
LABEL "scall/s"
PRECISION 0;

sread_per_sec = 1023
LABEL "sread/s"
PRECISION 0;

swrit_per_sec = 1024
LABEL "swrit/s"
PRECISION 0;

fork_per_sec = 1025
LABEL "fork/s"
PRECISION 2;

exec_per_sec = 1026
LABEL "exec/s"
PRECISION 2;

rchar_per_sec = 1027
LABEL "rchar/s"
PRECISION 0;

wchar_per_sec = 1028
LABEL "wchar/s"
PRECISION 0;
```

```
swpin = 1029
LABEL "swpin/s"
PRECISION 2;

bswin = 1030
LABEL "bswin/s"
PRECISION 1;

swpot = 1031
LABEL "swpot/s"
PRECISION 2;

bswot = 1032
LABEL "bswot/s"
PRECISION 1;
blks = 1033
LABEL "pswch/s"
PRECISION 0;

iget_per_sec = 1034
LABEL "iget/s"
PRECISION 0;

namei_per_sec = 1035
LABEL "namei/s"
PRECISION 0;

dirbk_per_sec = 1036
LABEL "dirbk/s"
PRECISION 0;

num_proc = 1037
LABEL "num proc"
PRECISION 0;

proc_tbl_size = 1038
LABEL "proc tbl size"
PRECISION 0;

proc_ov = 1039
LABEL "proc ov"
PRECISION 0;
```

```
num_inode = 1040
LABEL "num inode"
PRECISION 0;

inode_tbl_sz = 1041
LABEL "inode tbl sz"
PRECISION 0;

inode_ov = 1042
LABEL "inode ov"
PRECISION 0;

num_file = 1043
LABEL "num file"
PRECISION 0;

file_tbl_sz = 1044
LABEL "file tbl sz"
PRECISION 0;

file_ov = 1045
LABEL "file ov"
PRECISION 0;

msg_per_sec = 1046
LABEL "msg/s"
PRECISION 2;

LABEL "sema/s"
PRECISION 2;
```

此时，我们需要查看生成的输出：

```
sar -ubycwavm 1 1:
HP-UX hpptc16 A.09.00 E 9000/855    04/11/95

12:01:41    %usr    %sys    %wio    %idle
              bread/s  lread/s  %rcache   bwrit/s   lwrit/s  %wcache  pread/s
         pwrit/s
         rawch/s canch/s outch/s   cvin/s  xmtin/s  mdmin/s
         scall/s sread/s swrit/s   fork/s   exec/s  rchar/s wchar/s
         swpin/s bswin/s swpot/s  bswot/s  pswch/s
          iget/s namei/s dirbk/s
         text-sz  ov  proc-sz  ov  inod-sz  ov  file-sz  ov
           msg/s  sema/s
```

```
12:01:42    22      48       30      0
             0     342      100     33      81      59   0    0
             0       0      470      0       0       0
           801     127       71      1.00    1.00    975872 272384
          0.00     0.0     0.00    0.0     251
            28     215      107
         N/A   N/A  131/532    0  639/644   0   358/1141   0
         40.00    0.00
```

此输出与 sar -u 输出类似，只是多了几行标头和数据。我们将再次使用 tail 提取数据行，但需要将它表示为“一条”数据记录进入 dsilog。以下脚本捕获数据并使用 tr（转换字符）实用程序“除去”换行，使 dsilog 将它视为单独的一行输入数据。

```
#!/bin/ksh                    Sar_ubycwavm_feed

# Script that provides sar data to DSI via a fifo(sar_data.fifo)

while :                        # (infinite loop)
do

# specify a one minute interval using tail to extract the
# last sar output records (contains the time stamp and data)
# and pipe that data to tr to strip the new lines converting
# the eight lines of output to one line of output.


/usr/bin/sar -ubycwavm 60 1 2>/tmp/dsierr | tail -8 | \
tr "\012" " " > /usr/tmp/sar_data

# Copy the sar data to the fifo that the dsilog process is reading.

cat /usr/tmp/sar_data > ./sar_data.fifo

# Print a newline on the fifo so that DSI knows that this is
# the end of the input record.

print "\012" > ./sar_data.fifo

done
```

分步过程和之前的 sar -u 示例差不多，不同之处在于日志文件集名称、类名、fifo 名称 (sar_ubycwavm.fifo) 和上面列出用于提供 sar 数据的脚本。

记录系统用户数

下个示例使用 who 监视系统用户数。我们再次从类规范文件开始。

```
#  who_wc.spec
#
#  who word count DSI spec file
#

CLASS who_metrics = 150
LABEL "who wc data"
INDEX BY         hour
MAX INDEXES      120
ROLL BY          hour
RECORDS PER HOUR 60;

METRICS
who_wc = 151
label "who wc"
averaged
maximum 1000
precision 0;
```

编译规范文件以创建日志文件：

```
sdlcomp ./who_wc.spec ./who_wc_log.
```

与 sar 不同，您不能用 who 指定间隔或迭代值，因此我们创建至少提供间隔控制的脚本。

```
#!/bin/ksh         who_data_feed

while :
do
    # sleep for one minute (this should correspond with the
    # RECORDS PER HOUR clause in the specification file).

sleep 60

    # Pipe the output of who into wc to count
    # the number of users on the system.

who | wc -l > /usr/tmp/who_data

# copy the data record to the pipe being read by dsilog.

cat /usr/tmp/who_data > ./who.fifo

done
```

我们再次需要 fifo 和一个脚本向 dsilog 提供数据，因此我们回到分步过程。

1 创建两个 fifo；其中一个是用于使另一个真正的输入 fifo “保持打开状态”的虚拟 fifo。

```
# Dummy fifo.
mkfifo ./hold_open.fifo

# Real input fifo for dsilog.
mkfifo ./who.fifo
```

2 用 -i 选项启动 dsilog，指定来自 fifo 的输入。在启动 who 数据馈送之前先启动 dsilog，这很重要。

```
dsilog ./who_wc_log who_metrics \
-i ./who.fifo &
```

3 启动使输入 fifo 保持打开状态的虚拟进程。

```
cat ./hold_open.fifo \
> ./who.fifo &
```

4 启动 who 数据馈送脚本 (who_data_feed)。

```
./who_data_feed  &
```

17 错误消息

有三类 DSI 错误消息：类规范、 dsilog 记录进程和常规。

- 类规范错误消息的格式为前缀 SDL 后跟消息号。
- dsilog 记录进程消息的格式为前缀 DSILOG 后跟消息号。
- 常规错误消息可能由上面的任一方式生成，也可能由其他任务生成。这些消息包含负号 (-) 前缀和消息号。

本章列出了 DSI 错误消息。SDL 和 DSILOG 错误消息以数字顺序列出，还提供了从错误状态恢复需执行的操作。常规错误消息是自说明性的，因此没有给出恢复操作。

SDL 错误消息

SDL 错误消息是自描述日志文件类规范错误消息，格式是 SDL< *消息号* >。

消息 SDL1

ERROR: Expected equal sign, "=".

此处应为 "="。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL2

ERROR: Expected semi-colon, ";".

分号 (;) 标记类规范和每个度量规范的结束。如果在应使用分号的位置发现不正确或拼错的单词，也可能看到此消息。

例如：如果输入

```
class xxxxx = 10
   label "this is a test"
   metric 1000;
```

而不是

```
class xxxxx = 10
   label "this is a test"
   capacity 1000;
```

就会看到此错误消息，并且它指向单词 "metric"。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL3

```
ERROR: Precision must be one of {0, 1, 2, 3, 4, 5}
```

精度决定了将数字内部转换为整数和转换回度量值的数字表示时使用的小数位数。

操作：有关详细信息，请参见第 14 章的 PRECISION。

消息 SDL4

```
ERROR: Expected quoted string.
```

应为文本字符串。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL5

```
ERROR: Unterminated string.
```

字符串必须以双引号结尾。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL6

```
NOTE: Time stamp inserted at first metric by default.
```

时间戳度量自动作为第一个度量插入每个类中。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL7

```
ERROR: Expected metric description.
```

度量部分必须以 METRICS 关键字开头，然后才是第一个度量定义。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL8

```
ERROR: Expected data class specification.
```

类规范的类部分必须以 CLASS 关键字开头。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL9

```
ERROR: Expected identifier.
```

应是度量或类的标识符。标识符必须以字母字符开头，可包含字母数字字符或下划线，不区分大小写。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL10

```
ERROR: Expected positive integer.
```

数字格式不正确。

操作：只输入正整数的数字。

消息 SDL13

```
ERROR: Expected specification for maximum number of indexes.
```

最大索引数是计算类容量所必需的。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL14

```
ERROR: Syntax Error.
```

输入的语法不正确。

操作：检查语法，并根据需要进行更正。有关详细信息，请参见第 14 章的类规范语法。

消息 SDL15

```
ERROR: Expected metric description.
```

缺少度量描述。

操作：输入度量描述，以定义类的各个数据项。有关详细信息，请参见第 14 章的类规范语法。

消息 SDL16

```
ERROR: Expected metric type.
```

每个度量都必须有 *metric_name* 和数字的 *metric_id*。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL17

```
ERROR: Time stamp metric attributes may not be changed.
```

时间戳度量自动作为第一个度量插入每个类中。您可以更改时间戳的位置，或删除它并使用 UNIX 时间戳。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL18

```
ERROR: Roll action limited to 199 characters.
```

`ROLL BY` 操作的上限是 199 个字符。

操作：有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL19

```
ERROR: Could not open specification file (file).
```

在命令行 `sdlcomp specification_file` 中，无法打开规范文件。该错误跟在下一行中，如：

```
$/usr/perf/bin/sdlcomp /xxx
ERROR: Could not open specification file /xxx.
```

操作：验证该文件是否可读。如果可读，检查文件的名称并验证输入是否正确。

消息 SDL20

```
ERROR: Metric descriptions not found.
```

度量描述格式不正确。

操作：务必以 METRICS 关键字开始类语句的度量部分。有关详细信息，请参见第 14 章的度量描述。

消息 SDL21

```
ERROR: Expected metric name to begin metric description.
```

可能缺少度量名称，或者度量描述格式不正确。

操作：可能缺少度量名称，或者度量描述格式不正确。

消息 SDL24

```
ERROR: Expected MAX INDEXES specification.
```

指定 INDEX BY 时， MAX INDEXES 值是必需的。

操作：输入所需值。有关详细信息，请参见第 14 章的 INDEX BY、MAX INDEXES 和 ROLL BY。

消息 SDL25

```
ERROR: Expected index SPAN specification.
```

缺少一个 INDEX BY 值。

操作：指定 INDEX BY 时输入限定符。有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL26

```
ERROR: Minimum must be zero.
```

该数字不能小于 0。

消息 SDL27

```
Expected positive integer.
```

缺少正值。

操作：只输入正整数的数字。

消息 SDL29

```
ERROR: Summarization metric does not exist.
```

使用了 SUMMARIZED BY 作为汇总方法，但未指定 *metric_name*。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL30

```
ERROR: Expected 'HOUR', 'DAY', or 'MONTH'.
```

输入缺少限定符。

操作：必须输入上面三个限定符中的一个。有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL33

```
ERROR: Class id number must be between 1 and 999999.
```

类 ID 必须是数字，最多可包含 6 位数。

操作：为类输入最多不超过六位的类 ID 号。有关详细信息，请参见第 14 章的类规范语法。

消息 SDL35

```
ERROR: Found more than one index/capacity statement.
```

每个 CLASS 部分只能有一个 INDEX BY 或 CAPACITY 语句。

操作：按照第 14 章的类规范语法中的格式限制完成输入。

消息 SDL36

```
ERROR: Found more than one metric type statement.
```

每个度量定义只能有一个 METRICS 关键字。

操作：有关格式信息，请参见第 14 章的度量描述。

消息 SDL37

```
ERROR: Found more than one metric maximum statement.
```

每个度量定义只能有一个 MAXIMUM 语句。

操作：有关格式信息，请参见第 14 章的度量描述。

消息 SDL39

```
ERROR: Found more than one metric summarization specification.
```

每个度量定义只能有一个汇总方法 （TOTALED、 AVERAGED 或 SUMMARIZED BY）。

操作：有关详细信息，请参见第 14 章的汇总方法。

消息 SDL40

```
ERROR: Found more than one label statement.
```

每个度量或类定义只能有一个 LABEL。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL42

```
ERROR: Found more than one metric precision statement.
```

每个度量定义只能有一个 PRECISION 语句。

操作：有关详细信息，请参见第 14 章的 PRECISION。

消息 SDL44

```
ERROR: SCALE, MINIMUM, MAXIMUM, (summarization) are inconsistent with text
metrics
```

类规范语法的这些元素只对数字度量有效。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL46

```
ERROR: Inappropriate summarization metric (!).
```

不能按时间戳度量汇总。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL47

```
ERROR:Expected metric name.
```

每个 METRICS 语句都必须包含 *metric_name*。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL48

```
ERROR: Expected positive integer.
```

CAPACITY 语句要求使用正整数。

操作：有关详细信息，请参见第 14 章的 CAPACITY。

消息 SDL49

```
ERROR: Expected metric specification statement.
```

第一个度量定义前面必须有 METRICS 关键字。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL50

```
Object name too long.
```

metric_name 或 *class_name* 最多只能包含 20 个字符。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL51

```
ERROR: Label too long (max 20 chars).
```

*类标签*或*度量标签*最多只能包含 20 个字符。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL53

```
ERROR: Metric must be between 1 and 999999.
```

度量 ID 最多只能包含 6 位数。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL54

```
ERROR: Found more than one collection rate statement.
```

每个类描述只能有一个 RECORDS PER HOUR 语句。

操作：有关详细信息，请参见第 14 章的 RECORDS PER HOUR。

消息 SDL55

```
ERROR: Found more than one roll action statement.
```

每个类规范只能有一个 ROLL BY 语句。

操作：有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL56

```
ERROR: ROLL BY option cannot be specified without INDEX BY option.
```

ROLL BY 语句前面必须有 INDEX BY 语句。

操作：有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL57

```
ERROR: ROLL BY must specify time equal to or greater than INDEX BY.
```

因为滚动间隔要根据索引间隔来识别要丢弃的数据，ROLL BY 时间必须大于或等于 INDEX BY 时间。

操作：有关详细信息，请参见第 14 章的 INDEX BY、 MAX INDEXES 和 ROLL BY。

消息 SDL58

```
ERROR: Metric cannot be used to summarize itself.
```

SUMMARIZED BY 度量不能与 *metric_name* 相同。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL62

```
ERROR: Could not open SDL (name).
```

此错误后有说明性消息。它可能是文件系统错误，如：

```
$/usr/perf/bin/sdlutil xxxxx -classes
ERROR: Could not open SDL xxxxx.
ERROR: Could not open log file set.
```

或者可能是内部错误，如：

```
$/usr/perf/bin/sdlutil xxxxx -classes
ERROR: Could not open SDL xxxxx.
ERROR: File is not SDL root file or the
description file is not accessible.
```

如果移动了日志文件，也可能看到此错误。因为路径名称信息存储在 DSI 日志文件中，日志文件不能移动到其他目录。

操作：如果上面的描述或后续消息未指出明显的问题，则用 `sdlutil` 删除日志文件集并重建。

消息 SDL63

```
ERROR: Some files in log file set (name) are missing.
```

已检查组成日志文件集的文件的列表，未发现成功操作所需的一个或多个文件。

操作：除非您确切地知道发生了什么，否则最佳做法就是使用 `sdlutil` 删除日志文件集并重新开始。

消息 SDL66

```
ERROR: Could not open class (name).
```

后面会有说明性消息。

操作：除非问题很明显，否则用 `sdlutil` 删除日志文件集并重新开始。

消息 SDL67

```
ERROR: Add class failure.
```

后面会有说明性消息。

编译器无法将新类添加到日志文件集。

操作：如果日志文件集中的所有正确类皆可访问，请指定新的或不同的日志文件集。如果不能访问，则用 `sdlutil` 删除日志文件集并重新开始。

消息 SDL72

```
ERROR: Could not open export files (name).
```

无法打开导出的数据要写入的文件。

操作：检查导出文件路径是否存在，以及它有哪些权限。

消息 SDL73

```
ERROR: Could not remove shared memory ID (name).
```

后面会有说明性消息。

操作：要删除共享内存 ID，您必须是创建日志文件集的用户或根用户。用 UNIX 命令 `ipcrm -m` *ID* 删除共享内存 ID。

消息 SDL74

```
ERROR: Not all files could be removed.
```

并非日志文件集中的所有文件都能删除。

后面会有说明性消息。

操作：执行以下操作列出文件和共享内存 ID：

```
sdlutil (logfile set) -files
sdlutil (logfile set) -id
```

要删除文件，请使用 UNIX 命令 **rm** *文件名*。要删除共享内存 ID，请使用 UNIX 命令 **ipcrm -m** *ID*。请注意，只有在关闭日志文件集时 sdlutil 未正确删除共享内存 ID，它才会存在并需要删除。

消息 SDL80

ERROR: Summarization metric (metric) not found in class.

之前未在 METRIC 部分定义 SUMMARIZED BY 度量。

操作：有关详细信息，请参见第 14 章的度量描述。

消息 SDL81

ERROR: Metric id (id) already defined in SDL.

度量 ID 只需定义一次。要重用已在另一个类中定义的度量定义，只需指定 *metric_name*，而无需 *metric_id* 或其他任何规范。

操作：有关详细信息，请参见第 14 章的 METRICS。

消息 SDL82

ERROR: Metric name (name) already defined in SDL.

metric_name 只需定义一次。要重用已在另一个类中定义的度量定义，只需指定 *metric_name*，而无需 *metric_id* 或其他任何规范。

操作：有关详细信息，请参见第 14 章的 METRICS。

消息 SDL83

ERROR: Class id (id) already defined in SDL.

类 ID 只需定义一次。检查拼写，确保输入正确。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL84

ERROR: Class name (name) already defined in SDL.

class_name 只需定义一次。检查拼写，确保输入正确。

操作：有关详细信息，请参见第 14 章的类规范语法。

消息 SDL85

ERROR: Must specify class to de-compile.

使用 -decomp 时，必须指定*类列表*。

操作：有关详细信息，请参见第 15 章的用 sdlutil 管理数据。

消息 SDL87

ERROR: You must specify maximum number of classes with -maxclass.

使用 -maxclass 选项时，必须指定创建新日志文件集时要提供的最大类数。

操作：有关详细信息，请参见第 15 章的 sdlcomp 编译器。

消息 SDL88

```
ERROR: Option \"!\" is not valid.
```

命令行输入无效。

操作：检查输入的内容，确保符合正确的语法。

消息 SDL89

```
ERROR: Maximum number of classes (!) for -maxclass is not valid.
```

`-maxclass` 数必须大于 0。

操作：有关详细信息，请参见第 15 章的 sdlcomp 编译器。

消息 SDL90

```
ERROR: -f option but no result file specified.
```

使用 `-f` 选项时，必须指定格式文件。

操作：使用 `-f` 选项时，必须指定格式文件。

消息 SDL91

```
ERROR: No specification file named.
```

类规范文件未分配名称。

操作：使用 `sdlcomp` 时，必须输入*规范文件*。有关详细信息，请参见第 15 章的 sdlcomp 编译器。

消息 SDL92

```
ERROR: No log file set named.
```

使用 `sdlcomp` 时，必须输入*日志文件集*。

操作：有关详细信息，请参见第 15 章的 sdlcomp 编译器。

消息 SDL93

```
ERROR: Metric ID already defined in class.
```

度量 ID 只需定义一次。

操作：要重用已在另一个类中定义的度量定义，只需指定 *metric_name*，而无需 *metric_id* 或其他任何规范。

有关详细信息，请参见第 14 章的度量描述。

消息 SDL94

```
ERROR: Metric name already defined in class.
```

度量名称只需定义一次。

操作：要重用已在另一个类中定义的度量定义，只需指定 *metric_name*，而无需 *metric_id* 或其他任何规范。有关详细信息，请参见第 14 章的度量描述。

消息 SDL95

ERROR: Text found after complete class specification.

sdlcomp 编译器发现了它未能识别为类规范一部分的文本。

操作：重新输入规范并重试。

消息 SDL96

ERROR: Collection rate statement not valid.

正确格式是 RECORDS PER HOUR(*数字*)。关键字必须符合此顺序，且不能缩写。

操作：更正关键字并遵循必需的格式。

消息 SDL97

ERROR: Expecting integer between 1 and 2,147,483,647.

必须使用此范围内的数字。

操作：输入属于该范围的数字。

消息 SDL98

ERROR: Action requires preceding ROLL BY statement.

输入顺序不正确，或类规范文件中缺少该输入。

操作：“操作”指定当日志文件滚动时将发生的操作。首先必须知道何时应滚动。ACTION 的前面必须有 ROLL BY。

例如：

```
class xxxxx = 10
   index by month max indexes 12
   action "ll *";
```

应为：

```
class xxxxx = 10
   index by month max indexes 12
   roll by month
   action "ll *";
```

消息 SDL99

ERROR: MAX INDEXES requires preceding INDEX BY statement.

输入顺序不正确，或类规范文件中缺少该输入。

操作：要指定最大索引数，程序需要知道建立索引的依据。MAX INDEXES 的前面必须有 INDEX BY 语句。

例如：

```
class xxxxx = 10
   max indexes 12
   label "this is a test";
```

应为：

```
class xxxxx = 10
   index by month
   max indexes 12
   label "this is a test";
```

消息 SDL100

WARNING: CAPACITY UNLIMITED not implemented, derived value used. (SDL-100)

消息 SDL101

ERROR: Derived capacity too large. (SDL-101)

消息 SDL102

ERROR: Text Length should not exceed 4096.

文本度量的最大允许长度是 4096。

消息 SDL103

ERROR: RECORDS PER HOUR should not be greater than 3600 for logging summarized data.

操作：只有记录未汇总的数据时， RECORDS PER HOUR 才能大于 3600。使用 -u 选项进行编译。

DSILOG 错误消息

DSILOG 错误消息是采用以下格式的 dsilog 记录进程消息：

DSILOG< *消息号* >。

消息 DSILOG1

ERROR: Self describing log file not specified.

操作：更正命令行并重试。

消息 DSILOG2

ERROR: Data class name not specified.

数据类必须是传递到 dsilog 的第二个参数。

操作：更正命令行并重试。

消息 DSILOG3

ERROR: Could not open data input file (name).

无法打开命令行中指定的文件。 UNIX 文件系统错误显示在错误消息的下一行中。

消息 DSILOG4

ERROR: OpenClass (\"name\") failed.

无法打开指定的类。它可能不在指定的日志文件集中，或其数据文件不可访问。

操作：后面会有说明性消息给出内部错误描述或文件系统错误。

消息 DSILOG5

`ERROR: Open of root log file (name) failed.`

无法打开日志文件集根文件。说明性消息中将显示原因。

消息 DSILOG6

`ERROR: Time stamp not defined in data class.`

类已生成，未包括时间戳。

操作：用 `sdlutil` 删除日志文件集并重新开始。

消息 DSILOG7

`ERROR: (Internal error) AddPoint ( ) failed.`

`dsilog` 尝试将记录写入数据文件，但未成功。后面会有说明性消息。

消息 DSILOG8

`ERROR: Invalid command line parameter (name).`

显示的参数未识别为有效的命令行选项，或者它在命令行中的位置不正确。

操作：更正命令行参数并重试。

消息 DSILOG9

`ERROR: Could not open format file (name).`

找不到指引传入度量与数据类中度量匹配的文件，或者该文件无法访问。后面会有说明性消息及 UNIX 文件系统错误。

操作：检查类规范文件，确认它存在。

消息 DSILOG10

`ERROR: Illegal metric name (name).`

格式文件包含的度量名称长度超过最大度量名称大小，或在其他方面不符合度量名称语法。

操作：更正类规范中的度量名称，并重新运行 `dsilog`。

消息 DSILOG11

`ERROR: Too many input metrics defined. Max 100.`

格式文件中只能指定 100 个度量

操作：输入应外部重新格式化至 `dsilog`，或将数据源拆分为两个或更多数据源。

消息 DSILOG12

```
ERROR: Could not find metric (name) in class.
```

数据类中找不到在格式文件中发现的度量名称。

操作：更正并重试。

消息 DSILOG13

```
ERROR: Required time stamp not found in input specification.
```

使用了 `-timestamp` 命令行选项，但格式文件未指定时间戳在传入数据中的位置。

操作：指定时间戳的位置。

消息 DSILOG14

```
ERROR: (number) errors, collection aborted.
```

设置收集时检测到严重错误。

操作：更正错误并重试。 `-vi` 和 `-vo` 选项也可用于在数据进入和将要记录数据时验证数据。

消息 DSILOG15

```
ERROR: Self describing log file and data class not specified.
```

命令行必须指定要记录数据的日志文件集和数据类。

操作：更正命令行输入并重试。

消息 DSILOG16

```
ERROR: Self describing log file set root file (name) could not be accessed.
error=(number).
```

无法打开日志文件集根文件。

操作：查看后面的说明性消息确定问题所在。

消息 （未编号）

```
Metric null has invalid data
Ignore to end of line, metric value exceeds maximum
```

`dsilog` 未为特定输入行记录任何数据时，出现此警告消息。当输入不符合 DSI 日志文件预期的格式时 （如输入中存在空白或标头行或度量值超过指定精度时），将出现此消息。在这种情况下，将跳过 （不记录）有问题的行。 `dsilog` 将继续记录下一个有效输入行的数据。

消息 DSILOG17

```
ERROR: Logfile set is created to log unsummarized data, could not log
summarized data.
```

操作：如果编译期间用 `-u` 选项创建日志文件集，则使用 `dsilog` 的 `-s 0` 选项记录。使用该选项表示记录的数据未汇总。

常规错误消息

错误	说明
-3	尝试添加超过 max-class 所允许数量的类。
-5	无法打开包含类数据的文件。
-6	无法读取文件。
-7	无法写入文件。
-9	尝试在未请求写访问权限的情况下写入日志文件。
-11	找不到类的指针。
-13	文件或数据结构未初始化。
-14	无法读取类描述文件。
-15	无法写入类描述文件。
-16	未在度量描述类中找到定义某类所需的所有度量。
-17	日志文件集中的文件的路径名称长度超过 1024 个字符。
-18	类名长度超过 20 个字符。
-19	文件不是日志文件集根文件。
-20	文件不是 lod 文件集的一部分。
-21	当前软件不能访问日志文件集。
-22	无法获取共享内存段或 ID。
-23	无法附加到共享内存段。
-24	无法打开日志文件集。
-25	无法确定当前工作目录。
-26	无法从类数据文件读取类标头。
-27	打开日志文件集中的文件失败。
-28	无法打开数据类。
-29	Lseek 失败。
-30	无法从日志文件读取。
-31	无法写入日志文件。
-32	删除失败。

错误	说明
-33	shmctl (REM_ID) 失败。
-34	日志文件集不完整：缺少根文件或描述文件。
-35	用于添加类的目标日志文件不在当前日志文件集中。

18 什么是事务跟踪?

本章描述：

提高性能管理

您可以通过 HP Operations Agent 和 HP GlancePlus 的事务跟踪功能提高管理系统性能的能力。

随着分布式任务关键业务应用数量的增加，应用程序和系统管理器需要有更多信息告诉它们分布式信息技术 (IT) 的执行情况。

- 您的应用程序是否曾停止响应？
- 应用程序响应时间是否无法接受？
- 达到了您的服务级别目标 (SLO) 吗？

性能收集组件和 GlancePlus 的事务跟踪功能使 IT 管理器可在业务事务方面实现其客户端 / 服务器 IT 环境的端到端可管理性。使用性能收集组件，您可以定义业务事务是什么，并捕获在*您的*业务环境中有意义的事务数据。

如果您的应用程序已通过标准应用程序响应测量 (ARM) API 调用检测，这些产品还可在多个供应商平台上提供广泛的事务跟踪和端到端管理能力。

事务跟踪的优点

- 提供从事务开始到结束的经过时间的客户端视图。
- 提供事务数据。
- 帮助您管理服务级别协议 (SLA)。

此部分的其余各节更详细地论述了这些主题。

事务时间的客户端视图

事务跟踪提供从事务开始到结束的经过时间的客户端视图。在信息技术 (IT) 环境中使用事务跟踪时，您会发现以下益处：

- 可以准确跟踪每个事务执行的次数。
- 可以看到事务完成耗费的时间，而不是像现在这样只能估计时间。
- 可以将事务时间与系统资源利用率关联。
- 可以在系统管理应用程序中使用您自己的业务可交付产品生产数据，比如用于容量规划、性能管理、记帐和内部计费的数据。
- 可以根据真实的工作单元 （您的事务）进行应用程序优化和详细的性能故障排除，而不是用抽象的系统定义和网络资源来表示实际工作。

事务数据

在应用程序中插入应用程序响应测量 (ARM) API 调用来标记每个业务事务的开始和结束时，可以使用以下资源和性能监视工具监视事务数据：

- 性能收集组件提供记录、报告和检测事务数据警报所需的注册功能。可在 Performance Manager、Glance 中查看事务数据，或从性能收集组件日志文件将数据导出到电子表格和其他报表工具可访问的文件来查看。
- Performance Manager 绘制性能数据的图形，以进行短期故障排除并用于查看趋势和长期分析。
- Glance 显示详细的实时数据，用于监视每一刻的系统和事务。
- Performance Manager、Glance、或 HP Operations Manager 消息浏览器可用于监视服务级别符合性警报。

第 22 章事务度量中描述了各个事务度量

服务级别目标

服务级别目标 (SLO) 源自业务应用程序用户需要的规定服务级别。 SLO 通常基于服务级别协议 (SLA) 的开发。信息技术资源管理器需要收集、监视、存储和报告从 SLO 获得的实际度量，以确定它们是否达到业务应用程序用户同意的服务级别。

SLO 可以像监视简单事务的响应时间那样简单，也可以像跟踪系统可用性一样复杂。

场景：实时订单处理

想象一家成功的电视购物频道，雇用了几百名电话接线员，他们从观众那里收到各类商品的订单。假定该企业使用计算机程序输入订单信息、检查商品可用性并且更新库存。我们可以用这家虚构的企业演示事务跟踪如何帮助组织兑现客户承诺、满足 SLO。

资源管理器可根据关键任务、客户满意度因素、生产力因素和最长响应时间确定要提供给其客户的服务级别。

第 23 章事务跟踪示例包含显示如何在订单处理应用程序示例中插入 ARM API 调用，以便可以用性能收集组件和 Glance 监视事务数据的伪代码示例。

实时订单处理的要求

要在上面所述的实时订单处理示例中满足 SLO，资源管理器必须跟踪完成以下关键任务所需的时间长度：

- 输入订单信息
- 查询商品可用性
- 更新库存

对客户来说，关键的客户满意度因素是接线员可以多快接受其订单。

对企业来说，关键生产力因素是接线员每个小时可以完成的订单数。

为达到客户满意度和生产力因素的要求，必须监视访问库存数据库、调整库存和写回记录的事务的响应时间，以确保符合确立的 SLO。例如，资源管理器可能已为此应用程序确立这样的 SLO：90% 的事务必须在 5 秒内完成。

准备订单处理应用程序

可在订单处理应用程序中插入 ARM API 调用，以创建库存响应和更新库存事务。请注意， ARM API 调用必须由应用程序的程序员在编译应用程序*前*插入。有关包括定义各种事务的 ARM API 调用的订单处理程序的示例 （以伪代码编写），请参见第 23 章事务跟踪示例。

有关通过 ARM API 调用检测应用程序的详细信息，请参见《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）。

监视事务数据

在系统上安装了已通过 ARM API 调用检测的应用程序后运行该应用程序时，可以用性能收集组件、 GlancePlus 或 Performance Manager 监视事务数据。

... 使用性能收集组件

可以用性能收集组件收集和记录指定事务的数据，监视 SLO 随时间变化的趋势，并在超过 SLO 时生成警报。一旦识别出这些趋势，就可以根据事务量分配信息技术成本。可以将性能收集组件警报配置为激活技术人员的寻呼机，以便立即调查和解决问题。有关详细信息，请参见第 24 章高级功能。

在 Performance Manager 中查看事务数据必须有性能收集组件。

... 使用 Performance Manager

Performance Manager 从性能收集组件接收警报和事务数据。例如，可以将性能收集组件配置为在订单处理应用程序检查库存的时间太长时，让 Performance Manager 接收警报并向资源管理器控制台发送警告，以提醒潜在的问题。

在 Performance Manager 中，可以从数据源的 “类列表 (Class List)” 窗口选择 **TRANSACTION**，然后选择各种事务的**图形事务度量**。有关详细信息，请参见 Performance Manager 联机帮助。

... 使用 GlancePlus

使用 GlancePlus 监视最新的事务响应时间，以及事务执行情况是否符合您确立的 SLO。GlancePlus 帮助您识别并解决可能正在影响事务性能的资源瓶颈。有关详细信息，请参见 GlancePlus 联机帮助，该文档可通过 GlancePlus “帮助 (Help)” 菜单访问。

使用 ARM 的准则

通过 ARM API 检测应用程序需要仔细规划。此外，如果了解 ARM 数据收集的功能和限制，则管理包含 ARM 检测的应用程序的环境会更容易。下面列出了不完全了解可能导致混淆的一些事项。

1. 要捕获 ARM 度量， `ttd` 和 `midaemon` 必须正在运行。对于性能收集组件， `scope` 收集器必须正在运行才能记录 ARM 度量。 `ovpa start` 脚本会启动所有必需的进程。同样，如果 `ttd` 和 `midaemon` 未启动， Glance 会启动它们。（请参见第 19 章的事务跟踪守护进程 (ttd)。）
2. 需要重新读取事务配置文件 `ttd.conf` 才能捕获任何新定义的事务名称。（请参见第 19 章的事务配置文件 (ttd.conf)。）
3. 必须重新启动性能收集组件、用户应用程序和 `ttd` 才能捕获任何*新的*或修改过的事务范围和服务级别目标 (SLO)。（请参见第 19 章的添加新应用程序。）
4. 性能收集组件忽略用户定义的度量中的字符串。只记录前六个非字符串的用户定义度量。（请参见第 24 章的如何使用数据类型。）
5. 如果指定某事务的警报条件，在该事务名称中使用短划线有限制。（请参见第 20 章的警报部分中的 “... 使用性能收集组件”）
6. 性能收集组件只显示应用程序名称和事务名称的前 60 个字符。（请参见第 19 章的指定应用程序和事务名称。）
7. 请限制检测过的唯一事务名称数。（请参见第 20 章的独有事务的限制。）
8. 不要让 ARM API 函数调用影响从最终用户角度执行应用程序。（请参见第 19 章的 ARM API 调用状态返回结果。）
9. 使用共享库链接。（请参见第 382 页的按平台分类的 C 编译器选项示例部分）

19 事务跟踪的工作原理

性能收集组件的以下组件与 GlancePlus 共同帮助您从已通过应用程序响应测量 (ARM) 调用检测的应用程序定义和跟踪事务数据。

- 测量接口守护进程 midaemon，它监视事务数据并将数据报告给其共享内存段，性能收集组件、Performance Manager 和 GlancePlus 可访问和报告其共享内存段中的信息。在 HP-UX 系统上， midaemon 还监视系统性能数据。
- 事务配置文件 /var/opt/perf/ttd.conf，用于定义事务并标识每个事务要监视的信息。
- 事务跟踪守护进程 ttd，它通过 midaemon 读取、注册和同步来自事务配置文件 ttd.conf 的事务定义。

支持 ARM 2.0

ARM 2.0 是应用程序响应测量的以前版本的超集。ARM 2.0 提供的新功能包括用户定义的度量、事务关联和记录代理程序。性能收集组件和 GlancePlus 支持用户定义的度量和事务关联，但*不*支持记录代理程序。

但是，您可能想用记录代理程序测试应用程序中的检测。ARM 2.0 Software Developers Kit (SDK) 中包含记录代理程序的源代码 `logagent.c`，您可从以下网站获取该工具包：

http://regions.cmg.org/regions/cmgarmw

有关使用记录代理程序的信息，请参见《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）。

《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）使用术语 “应用程序定义的度量”而不是 “用户定义的度量”。

支持 ARM API 调用

性能收集组件和 GlancePlus 支持下列应用程序响应测量 (ARM) API 调用。

`arm_init()`	指定并注册应用程序和 （可选）用户。
`arm_getid()`	指定和注册事务类，并提供相关的事务信息。定义用户定义的度量的上下文。
`arm_start()`	指示唯一事务实例开始。
`arm_update()`	更新唯一事务实例的值。
`arm_stop()`	指示唯一事务实例结束。
`arm_end()`	指示应用程序结束。

有关通过 ARM API 调用检测应用程序的信息以及调用及其参数的完整描述，请参见当前的《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）和 *arm (3)* 手册页。对于商业应用程序，请查看产品文档了解应用程序是否已通过 ARM API 调用检测。

有关必需库的重要信息，请参见本手册后面第 377 页的事务库。

arm_complete_transaction 调用

除了 ARM 2.0 API 标准以外， HP ARM 代理程序还支持 arm_complete_transaction 调用。此调用是特定于 HP 的对 ARM 标准的扩展，可用于在事务的开始无法由 arm_start 调用分隔时，标记已完成事务的结束。arm_complete_transaction 调用将已完成事务实例的响应时间视为参数。

除了指示事务实例结束以外，还可在可选的数据缓冲区中提供有关事务的其他信息。有关该可选数据的详细信息以及此调用及其参数的完整描述，请参见 *arm (3)* 手册页。

ARM 检测的应用程序示例

有关如何实现 ARM API 调用的示例，请参见 /< 安装目录 >/examples/arm/ 目录中的 ARM 检测的应用程序示例 armsample1.c、armsample2.c、armsample3.c 和 armsample4.c，及其生成脚本 Make.armsample。

- armsample1.c 显示简单的标准 ARM API 调用的使用。
- armsample2.c 也显示简单的标准 ARM API 调用的使用。它的结构类似于 armsample1.c，但它是交互式的。
- armsample3.c 提供如何使用 ARM API 版本 2.0 提供的用户定义的度量和事务相关器的示例。此示例模拟一个客户端 / 服务器应用程序，其中服务器和客户端都执行很多事务。（通常应用程序客户端和服务器组件应在独立的程序中，但为了简化将它们放在一起。）

  客户端程序启动事务，并从其 arm_start 调用请求 ARM 相关器。此相关器由客户端保存，然后传递到服务器，以便服务器调用 arm_start 时可以使用它。随后，在服务器上运行的性能工具就可以使用此相关器信息区分使用服务器的不同客户端。

  此程序中还显示了用于将用户定义的度量值传递到 ARM API 的机制。这样，您在性能工具中不仅可以看到响应时间和服务级别信息，还可以看到对应用程序本身很重要的数据。例如，事务可能在处理不同大小的请求，而请求的大小可能是用户定义的度量。响应时间过长时，此用户定义的度量就可用于查看较长的响应时间是否对应于较大的事务实例。
- armsample4.c 提供在 ARM 调用中使用用户定义的度量的示例。不同度量值可通过 arm_start、arm_update 和 arm_stop 调用传递。另外， tran 无法由 start/stop 调用分隔时，可以使用 arm_complete_transaction。

指定应用程序和事务名称

尽管 ARM 在 arm_init 和 arm_getid API 调用中允许使用最多包含 128 个字符的应用程序名称和事务名称，但性能收集组件最多只显示 60 个字符。前 60 个字符后的所有字符都不可见。但是，GlancePlus 允许您查看最多 128 个字符。

性能收集组件对提取或导出的事务数据中应用程序名称和事务名称的显示有一些限制。这些规则还应用于在 Performance Manager 中查看应用程序名称和事务名称。

应用程序名称*始终*优先于事务名称。例如，如果导出的事务数据包含 65 个字符的应用程序名称和 40 个字符的事务名称，*只*显示应用程序名称。应用程序名称的最后五个字符不可见。

另一个示例，如果应用程序名称包含 32 个字符，而事务名称有 40 个字符，则性能收集组件显示完整的应用程序名称，事务名称显示为被截断。总共显示 60 个字符。59 个字符分配给应用程序名称和事务名称，1 个字符分配给分隔这两个名称的下划线 (_)。下面就是应用程序名称“WarehouseInventoryApplication”和事务名称“CallFromWestCoastElectronicSupplier”在性能收集组件或 Performance Manager 中的显示：

```
WarehouseInventoryApplication_CallFromWestCoastElectronicSup
```

如果要用 Performance Manager 查看数据，则应用程序名称和事务名称的 60 个字符的组合必须唯一。

事务跟踪守护进程 (ttd)

事务跟踪守护进程 ttd 通过 midaemon 读取、注册和同步来自 ttd.conf 的事务定义。

当您用 ovpa start 命令启动性能收集组件的 scope 数据收集器时，ttd 启动。ttd 发出后以后台模式运行，错误写入文件 /var/opt/perf/status.ttd。

midaemon 也必须在运行，才能处理事务和收集与这些事务关联的性能度量 （请参见下页）。

我们强烈建议您不要停止 ttd。

如果必须停止 ttd，则在重新启动 ttd 和性能收集组件进程之前，还*必须*停止正在运行的任何 ARM 检测的应用程序。 ttd 必须正在运行才能捕获系统上所进行的所有 arm_init 和 arm_getid 调用。如果 ttd 停止后重新启动，将重复这些调用返回的事务 ID，这样会使 ARM 度量无效

用 ovpa 脚本启动性能收集组件进程以确保按正确顺序启动进程。 ovpa stop 将*不会*关闭 ttd。如果必须关闭 ttd 以重新安装任何性能软件，则使用命令 /< 安装目录 >/bin/ttd -k。但是，我们建议您不要停止 ttd，除非要重新安装性能收集组件。

如果系统上没有性能收集组件，GlancePlus 将启动 midaemon。然后 midaemon 启动 ttd（如果它未运行），*之后* midaemon 才开始处理任何测量数据。

有关完整的程序选项，请参见 ttd 手册页。

ARM API 调用状态返回结果

要注册事务， ttd 进程必须始终在运行。如果由于任何原因终止了 ttd，当它没有运行时，arm_init 或 arm_getid 调用将返回“失败”返回代码。如果随后重新启动 ttd，则新的 arm_getid 调用可能重新注册其他程序已在使用的相同事务 ID，从而导致记录无效的数据。

ttd 终止后重新启动时， ARM 检测的应用程序可能开始获取 ARM API 调用的返回值 -2 (TT_TTDNOTRUNNING) 和 EPIPE errno 错误。最初启动应用程序时，会在任何初始 ARM API 调用上创建客户端连接句柄。此客户端句柄允许应用程序与 ttd 进程通信。终止 ttd 时，此连接不再有效，下次应用程序尝试使用 ARM API 调用时，可能获取返回值 TT_TTDNOTRUNNING。此错误反映*上个* ttd 进程已不再运行，即使有另一个 ttd 进程在运行。（部分 ARM API 调用返回结果在 *arm (3)* 手册页中作了说明。）

要消除此错误，如果终止了 ttd，必须重新启动 ARM 检测的应用程序。首先，停止 ARM 检测的应用程序。然后重新启动 ttd（用 /< 安装目录 >/bin/ovpa start 或 /< 安装目录 >/bin/ttd），再重新启动应用程序。重新启动应用程序会在应用程序和 ttd 进程之间创建新的客户端连接句柄。

如果 midaemon 有错误，某些 ARM API 调用将不返回错误。例如，当 midaemon 用完了共享内存段的空间时，就会发生这种情况。性能度量 GBL_TT_OVERFLOW_COUNT 将为 > 0。如果发生溢出情况，可能要关闭正在运行的任何性能工具（除 ttd 以外）并重新启动 midaemon，用 -smdvss 选项为共享内存段指定更多空间。（有关详细信息，请参见 *midaemon* 手册页。）

我们建议您编写应用程序时确保即使发生 ARM 错误应用程序也继续执行。 ARM 状态不应影响程序执行。

可以通过 arm_getid 调用向 ttd 注册事务的活动客户端进程数限制为 maxfiles 内核参数的值。此参数控制每个进程打开的文件数。每个客户端注册请求都会导致 ttd 为 RPC 连接打开套接字（打开的文件）。客户端应用程序终止时套接字关闭。因此，此限制仅影响已通过 arm_getid 调用注册事务的活动客户端数。一旦达到此限制， ttd 就会向客户端 arm_getid 请求返回 TT_TTDNOTRUNNING。可以增加 maxfiles 内核参数值，将此限制提高到超过将向 ttd 注册事务的活动应用程序数。

测量接口守护进程 (midaemon)

测量接口守护进程 midaemon 是连续收集系统性能信息的低开销进程。要使性能收集组件收集事务数据或使 GlancePlus 报告事务数据， midaemon 必须正在运行。用 ovpa start 命令运行 scope 或启动 GlancePlus 时，它开始运行。

性能收集组件和 GlancePlus 都需要 midaemon 和 ttd 运行，这样才能注册和跟踪事务。 ovpa 脚本按正确顺序启动和停止性能收集组件的处理，包括 mideamon。 GlancePlus 启动 mideamon（如果它尚未运行）。 midaemon 启动 ttd（如果它尚未运行）。

有关 midaemon CPU 开销的信息，请参见本手册后面的 CPU 开销部分。

有关完整的程序选项，请参见 *midaemon* 手册页。

事务配置文件 (ttd.conf)

事务配置文件 /var/opt/perf/ttd.conf 用于定义应用程序名称、事务名称、性能分布范围及每个事务要满足的服务级别目标。 ttd 读取 ttd.conf 以确定如何注册每个事务。

可选择自定义 ttd.conf。性能收集组件附带的默认配置文件会导致监视任何应用程序中检测的*所有*事务。

如果您使用商业应用程序，并且不知道应用程序中的哪些事务已检测，请使用默认 ttd.conf 文件收集一些数据。然后查看数据，确定哪些事务可用。之后可以通过修改 ttd.conf 自定义该应用程序的事务数据收集。

添加新应用程序

如果将新的 ARM 检测的应用程序添加到使用 ttd.conf 文件中 tran=* 行的默认 slo 和 range 值的系统，则无需执行任何操作即可包含这些新事务。（有关 tran、range 和 slo 的描述，请参见配置文件关键字部分。）新事务将自动提取。ttd.conf 文件中 tran 行的 slo 和 range 值将应用于新事务。

添加新事务

对 ttd.conf 文件进行添加后，必须执行以下步骤才能使添加生效：

- 停止所有应用程序。
- 作为 **root** 用户执行 ttd -hup -mi 命令。

上面的操作导致 ttd.conf 文件被重新读取，并用 ttd 和 midaemon 注册新事务及其 slo 和 range 值。重新读取不会更改重新读取之前 ttd.conf 文件中的任何事务的 slo 或 range 值。

更改 range 或 slo 值

如果需要更改 ttd.conf 文件中现有事务的 slo 或 range 值，必须执行以下操作：

- 停止所有 ARM 检测的应用程序。
- 用 **ovpa stop** 停止 scope 收集器。
- 停止 Glance 的任何使用。
- 发出命令 **ttd -k**，停止 ttd。
- 对 ttd.conf 文件做出更改后：
- 用 **ovpa start** 重新启动 scope。
- 重新启动 ARM 检测的应用程序。

配置文件关键字

/var/opt/perf/ttd.conf 配置文件将事务名称与用表 1 中的关键字定义的事务属性关联。

表 1　　配置文件关键字

关键字	语法	使用
tran	**tran=** *事务名称*	必需
slo	**slo=** *秒*	可选
range	**range=** *秒[, 秒,...]*	可选

下面更详细地描述了这些关键字。

tran

用 tran 定义事务名称。此名称必须对应于在检测的应用程序中的 arm_getid API 调用中定义的事务。必须先使用 tran 关键字，然后才能指定可选属性 range 或 slo。tran 是配置文件中唯一的必需关键字。事务名称中结尾的星号 (*) 会导致在对此条目发出注册请求时执行通配符模式匹配。可在事务名称中使用短划线。但是，事务名称中不能使用空格。

事务名称最多可包含 128 个字符。但是，在性能收集组件中只能看到前 60 个字符。GlancePlus 可在特定屏幕中显示 128 个字符。

默认 ttd.conf 文件包含若干条目。前面的条目定义由性能收集组件数据收集器 scope 使用的事务，该收集器通过 ARM API 调用检测。文件还包含条目 tran=*，它注册通过 ARM API 或 Transaction Tracker API 调用检测的应用程序中的所有其他事务。

range

用 range 指定事务性能分布范围。性能分布范围用于区分耗费不同时长完成的事务，及查看每个时长产生的成功事务数。您定义的范围显示在 “GlancePlus 事务跟踪 (GlancePlus Transaction Tracking)” 窗口中。

为秒输入的每个值都表示该范围的事务时间的上限 （以秒为单位）。该值可以是整数或实数，最多可有六位小数。在 HP-UX 上，可达到 1 微秒 （.000001 秒）精度。但是在其他平台上，精度是 10 毫秒 （0.01 秒），因此只能识别小数点右边的前两个数字。

对于您定义的每个事务，最多支持十个范围。
可指定最多九个范围。一个范围保留作为溢出范围，此范围用于收集范围超过最大的用户定义范围的事务的数据。如果指定范围超过九个，则使用前九个范围，而忽略其他。

如果指定范围少于九个，则第一个未指定的范围成为溢出范围。其余的所有未指定范围都不使用。未指定范围度量报告为 0.000。第一个对应的未指定计数度量成为溢出计数。其余的未指定计数度量始终是零 (0)。

范围必须以升序 （请参见本章后面的示例）定义。

slo

用 slo 指定要用于监视性能服务级别协议 (SLA) 的服务级别目标 (SLO)，以秒为单位。

与 range 关键字一样，该值可以是整数或实数，最多可有六位小数。在 HP-UX 上，可达到 1 微秒（.000001 秒）精度。但是在其他平台上，精度是 10 毫秒（0.01 秒），因此只能识别小数点右边的前两个数字。

请注意，即使事务在 HP-UX 上可以 1 微秒精度排序，事务时间也是以 100 微秒的精度显示的。

配置文件格式

ttd.conf 文件可包含两类条目：常规事务和特定于应用程序的事务。

定义任何应用程序之前，应在 ttd.conf 文件中定义常规事务。这些事务将与定义的所有应用程序关联。默认 ttd.conf 文件包含一个常规事务条目和多个通过 ARM API 调用检测的 scope 收集器条目。

```
tran=* range=0.5, 1, 2, 3, 5, 10, 30, 120, 300 slo=5.0
```

（可选）每个应用程序都可以有一组自己的事务名称。这些事务将*只*与该应用程序关联。您指定的应用程序名称必须对应于在检测的应用程序中的 arm_init API 调用中定义的应用程序名称。每组特定于应用程序的条目都必须以包含在方括号中的应用程序名称开头。例如：

```
[AccountRec]
tran=acctOne range=0.01, 0.03, 0.05
```

应用程序名称最多可包含 128 个字符。但是，在性能收集组件中只能看到前 60 个字符。Glance 可在特定屏幕中显示 128 个字符。

如果有和 “常规”事务同名的事务，将使用应用程序下面列出的事务。

例如：

```
tran=abc range=0.01, 0.03, 0.05 slo=0.10
tran=xyz range=0.02, 0.04, 0.06 slo=0.08
tran=t* range=0.01, 0.02, 0.03

[AccountRec}
tran=acctOne range=0.04, 0.06, 0.08
tran=acctTwo range=0.1, 0.2
tran=t* range=0.03, 0.5

[AccountPay]

[GenLedg]
tran=GenLedgOne range=0.01
```

在以上示例中，前三个事务应用于指定的所有三个应用程序。

应用程序 [AccountRec] 有以下事务：acctOne、acctTwo、abc、xyz 和 t*。常规事务集中的一个条目也有名为 “t*” 的通配符事务。在这种情况下，将使用 AccountRec 应用程序的 “t*” 事务名称而忽略常规事务集中的那个。

应用程序 [AccountPay] 只有来自常规事务集的一个事务。

应用程序 [GenLedg] 有事务 GenLedgOne、 abc、 xyz 和 t*。

事务名称的顺序在应用程序中不会造成任何差别。

有关应用程序名称和事务名称的其他信息，请参见本章中的指定应用程序和事务名称部分。

配置文件示例

示例 1

```
tran=* range=0.5,1,2,3,5,10,30,12,30 slo=5.0
```

如果无条目与注册的事务名称匹配，“*” 条目将用作默认设置。这些默认设置在每个系统上都可通过修改 “*” 条目更改。如果缺少 “*” 条目，将使用与分配给上面的 “*” 条目的初始参数匹配的默认注册参数集。

示例 2

```
[MANufactr]
tran=MFG01 range=1,2,3,4,5,10 slo=3.0
tran=MFG02 range=1,2.2,3.3,4.0,5.5,10 slo=4.5
tran=MFG03
tran=MFG04 range=1,2.2,3.3,4.0,5.5,10
```

MANufctr 应用程序的 MFG01、MFG02 和 MFG04 事务都分别使用自己的唯一参数。MFG03 事务不需要跟踪时间分布或服务级别目标，因此它未指定这些参数。

示例 3

```
[Financial]
tran=FIN01
tran=FIN02 range=0.1,0.5,1,2,3,4,5,10,20 slo=1.0
tran=FIN03 range=0.1,0.5,1,2,3,4,5,10,20 slo=2.0
```

Financial 应用程序的 FIN02 和 FIN03 事务都分别使用自己的唯一参数。FIN01 事务不需要跟踪时间分布或服务级别目标，因此它未指定这些参数。

示例 4

```
[PERSONL]
tran=PERS*  range=0.1,0.5,1,2,3,4,5,10,20 slo=1.0
tran=PERS03 range=0.1,0.2,0.5,1,2,3,4,5,10,20 slo=0.8
```

PERSONL 应用程序的 PERS03 事务使用自己的唯一参数，其余人员事务则使用 PERSONL 应用程序的唯一默认参数集。

示例 5

```
[ACCOUNTS]
tran=ACCT_*    slo=1.0
tran=ACCT_REC range=0.5,1,2,3,4,5,10,20 slo=2.0
tran=ACCT_PAY range=0.5,1,2,3,4,5,10,20 slo=2.0
```

ACCOUNTS 应用程序的 ACCT_REC 和 ACCT_PAY 事务都分别使用自己的唯一参数，其余记帐事务则使用记帐应用程序的唯一默认参数集。只有应付帐款和应收帐款事务需要跟踪时间分布。事务名称的顺序在应用程序中不会造成任何差别。

使用 ARM 的开销注意事项

性能收集组件和 GlancePlus 的当前版本对支持 ARM 2.0 所需的其他数据的测量接口进行了修改。这些修改可能导致性能管理开销增加。规划应用程序的 ARM 检测时，应了解开销注意事项。

本章的其余部分讨论开销问题。

准则

下面是通过 ARM API 检测应用程序时需遵循的一些准则：

- 单独的事务 ID 总数应限制为不超过 4000。通常，对相同事务使用多个实例比对不同事务使用单个实例的成本低。请*只*注册那些将会主动监视的事务。
- 尽管 arm_start 和 arm_stop API 调用的开销非常小，但有大量事务实例时，它可能会增加。在大多数系统上，每秒超过几千个 arm_start 和 arm_stop 调用就可能严重影响总体性能。

- 请*只*在使用 ARM 2.0 功能时请求 ARM 相关器。（有关 ARM 相关器的详细信息，请参见《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）中的 “高级主题 （Advanced Topics）” 部分。用于生成、移动和监视相关器信息的开销大大高于监视未检测使用 ARM 2.0 相关器功能的事务。
- 字符串长度增加 （注册冗长事务名称、应用程序名称和用户定义字符串度量的应用程序）也会增加开销。

磁盘 I/O 开销

性能管理软件不会在系统上产生很大的磁盘开销。Glance 通常不将其数据记录到磁盘。性能收集组件的收集器守护进程 `scope` 生成磁盘日志文件，但 ARM 2.0 不会对其大小产生显著影响。`logtran scope` 日志文件用于存储 ARM 数据。

CPU 开销

通过 ARM 调用检测的程序通常不会因 ARM 调用而运行变慢。这是假定 `arm_getid` 调用的速率低于每秒一次调用， `arm_start` 和 `arm_stop` 调用的速率低于每秒几千次。应避免更频繁地调用 ARM API。

用于支持 ARM 的大多数额外 CPU 开销发生在性能工具程序和守护进程自身内部。 `midaemon` CPU 的开销略有增加，但不会比使用 ARM 1.0 时多百分之二。如果请求 `midaemon` 跟踪每个事务的资源度量，则每事务实例的开销可能是不跟踪每个事务资源度量时的两倍。（您可以在 `parm` 文件中设置 `log transaction=resource` 标记，启用每个事务资源度量的跟踪。）此外， Glance 和 `scope` 的 CPU 开销在使用通过 ARM 2.0 调用检测的应用程序的系统上会略微增加。只有那些广泛使用相关器和 / 或用户定义度量的 ARM 2.0 调用检测的应用程序会对 `midaemon`、`scope` 或 Glance 产生显著的性能影响。

使用量超过可用的默认共享内存时，可能发生 `midaemon` 溢出情况。这会导致：

- 一旦发生溢出情况， ARM 调用就没有返回代码。
- 显示错误度量，包括空白进程名称。
- 在 `status.mi` 中记录错误 （例如 “out of space”）。

内存开销

进行 ARM API 调用的程序不会对其内存虚拟集大小产生显著影响，除了用于传递 ARM 2.0 相关器和用户定义度量信息的空间外。这些缓冲区（在《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）中有说明）不应成为进程的内存需求的重要部分。

性能工具中有额外的虚拟集大小开销，用以支持 ARM 2.0。 `midaemon` 进程创建一个共享内存段，ARM 数据内部保留在其中，供性能收集组件和 GlancePlus 使用。此共享内存段的大小相对于 ARM 1.0 版本有所增长，以适应 ARM 2.0 使用的潜在需要。默认情况下，在大多数系统上，此共

享内存段的大小约为 11 MB。若非必需，此内存段并不全都驻留在物理内存中。因此，这不会对尚未受到内存限制的大多数系统产生显著影响。 midaemon 的内存开销可以使用特殊启动参数（请参见 *midaemon* 手册页）调整。

20 事务入门

本章提供了开始跟踪事务和服务级别目标所需的信息。有关详细的参考信息，请参见第 19 章事务跟踪的工作原理。有关示例，请参见第 23 章事务跟踪示例。

开始之前

性能收集组件在以下位置提供 libarm.* 共享库：

平台	路径
IBM RS/6000	/usr/lpp/perf/lib/
其他 UNIX 平台	/opt/perf/lib/

如果系统上没有安装性能收集组件，且您的平台上的上述路径中不存在 libarm.*，请参见本手册末尾第 382 页的按平台分类的 C 编译器选项示例。有关如何获取它的信息，另请参阅《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）中的 “ARM 共享库 (libarm)（The ARM Shared Library (libarm)）” 部分。有关 libarm 的描述，请参见本手册末尾第 377 页的 ARM 库 (libarm)。

设置事务跟踪

按下面的过程为应用程序设置事务跟踪。此部分的其余各节更详细地说明了这些步骤。

1. 通过确定要监视的关键事务以及您预期的响应级别定义 SLO （*可选*）。
2. 要在性能收集组件和 Performance Manager 中监视事务，请确保性能收集组件 parm 文件的事务记录已打开。然后启动或重新启动性能收集组件读取更新后的 parm 文件。

   在 GlancePlus 中查看事务*不*需要编辑 parm 文件。但是，ttd *必须*正在运行才能在 GlancePlus 中查看事务。启动 GlancePlus 时将自动启动 ttd。
3. 运行已通过本手册和 《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）中所述的 ARM API 调用检测的应用程序。

4 用性能收集组件或 Performance Manager 查看收集的事务数据，或用 GlancePlus 查看当前数据。如果数据在 Performance Manager 中不可见，关闭数据源然后重新连接。

5 自定义配置文件 `ttd.conf`，修改收集应用程序的事务数据的方式 （*可选*）。

6 对 `ttd.conf` 文件进行添加后，必须执行以下步骤才能使添加生效：

   a 停止所有 ARM 检测的应用程序。

   b 作为 **root** 用户执行 `ttd -hup -mi` 命令。

   这些操作导致重新读取 `ttd.conf` 文件，并用 `ttd` 和 `midaemon` 注册新事务及其 `slo` 和 `range` 值。重新读取不会更改重新读取之前 `ttd.conf` 文件中的任何事务的 `slo` 或 `range` 值。

7 如果需要更改 `ttd.conf` 文件中现有事务的 `slo` 或 `range` 值，请执行以下操作：

   a 停止所有 ARM 检测的应用程序。

   b 用 `ovpa stop` 停止 `scope` 收集器。

   c 停止 Glance 的所有使用。

   d 用 **`ttd -k`** 停止 `ttd`。

进行更改后：

   a 用 **`ovpa start`** 重新启动 `scope`。

   b 启动 ARM 检测的应用程序。

定义服务级别目标

实现事务跟踪的第一个步骤是确定满足客户期望必需的关键事务以及必需的事务响应级别。必需的响应级别就成为服务级别目标 (SLO)。在配置文件 `ttd.conf` 中定义服务级别目标。

定义服务级别目标可以很简单，只需查看信息技术部门的服务级别协议 (SLA)，了解需要监视哪些事务是否遵守 SLA 即可。如果没有 SLA，则可能要实现一个。但是，创建 SLA 不是跟踪事务必需的。

修改 parm 文件

如有必要，修改性能收集组件 `parm` 文件，将事务添加到记录以用于 Performance Manager 和性能收集组件的项的列表。如以下示例所示，在 `parm` 文件的日志参数中包括 `transaction` 选项：

```
log global application process transaction device=disk
```

`log transaction` 参数的默认值是 `no resource` 和 `no correlator`。要打开资源数据收集或相关器数据收集，请指定 `log transaction=resource` 或 `log transaction=correlator`。指定 `log transaction=resource, correlator` 可以同时记录它们。

必须先如下所述激活更新后的 parm 文件，才能收集事务数据用于性能收集组件和 Performance Manager：

性能收集组件状态	激活事务跟踪的命令
正在运行	ovpa restart
未运行	ovpa start

收集事务数据

启动应用程序。事务跟踪守护进程 ttd 和测量接口守护进程 midaemon 在您的应用程序运行时收集并同步事务数据。数据存储在性能收集组件或 GlancePlus 可以使用的 midaemon 共享内存段中。有关使用其中每个工具来查看应用程序的事务数据的信息，请参见第 363 页的监视性能数据。

错误处理

出于性能考虑，并非所有有问题的 ARM 或 Transaction Tracker API 调用都实时返回错误。例如，在以下情况下不会按预期返回错误：

- 用错误的 id 参数 （如未初始化的变量）调用 arm_start 时
- 之前未成功调用 arm_start 就调用 arm_stop 时

性能收集组件 - 要在通过 ARM 调用检测应用程序时调试这些问题，请运行应用程序足够长时间，以生成并收集足够的事务数据量。用性能收集组件收集此数据，然后用 extract 程序的 export 命令从 logtran 文件导出数据。检查数据，查看是否按预期记录所有事务。另外，检查 /var/opt/perf/status.ttd 文件是否有错误。

GlancePlus - 要在通过 ARM 调用检测应用程序时调试这些问题，请运行应用程序足够长时间，以生成足够的事务数据量，然后用 GlancePlus 查看是否按预期记录所有事务。

独有事务的限制

根据特定系统资源和内核配置，应用程序中允许的独特事务数可能有限制。此限制通常是几千个独有 arm_getid 调用。

midaemon 使用的共享内存段满时，独有事务数可能超过限制。如果发生这种情况，GlancePlus 中会显示溢出消息。尽管在性能收集组件中没有显示消息，但不会记录后续新事务的数据。（但是，请检查 /var/opt/perf/status.scope 是否有溢出消息。）后续新事务的数据在 GlancePlus 中不可见。将继续记录和报告已注册的事务。 GlancePlus 中的 GBL_TT_OVERFLOW_COUNT 度量报告无法测量的新事务数。

可通过停止后重新启动 midaemon 进程，用 -smdvss 选项指定更大的共享内存段大小来解决这种情况。可使用 midaemon -sizes 命令检查当前共享内存段大小。有关为您的系统优化 midaemon 的详细信息，请参见 *midaemon* 手册页。

自定义配置文件 （可选）

查看应用程序的事务数据之后，可能要自定义事务配置文件 /var/opt/perf/ttd.conf，修改收集应用程序的事务数据的方式。这是可选的，因为默认配置文件 ttd.conf 适用于应用程序中定义的所有事务。如果决定自定义 ttd.conf 文件，则在运行应用程序的同一系统上完成此任务。必须作为 **root** 用户登录才能修改 ttd.conf。

有关配置文件关键字 tran、range 和 slo 的信息，请参见第 19 章事务跟踪的工作原理。下面显示了如何使用每个关键字的一些示例：

```
tran=Example:     tran=answerid
                  tran=answerid*
                  tran=*

range=Example:    range=2.5,4.2,5.0,10.009

slo=Example:       slo=4.2
```

自定义配置文件以包括所有事务和每个关联的属性。请注意，使用 range 或 slo 关键字时前面必须有 tran 关键字。下面显示了 ttd.conf 文件的示例。

```
tran=*
tran=my_first_transaction slo=5.5

[answerid]
tran=answerid1 range=2.5, 4.2, 5.0, 10.009 slo=4.2

[orderid]
tran=orderid1 range=1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0
```

如果要对 ttd.conf 文件进行添加：

- 停止所有 ARM 检测的应用程序。
- 作为 **root** 用户执行 ttd -hup -mi 命令。

上面的操作倒置重新读取 ttd.conf 文件，并用 ttd 和 midaemon 注册新事务及其 slo 和 range 值。重新读取不会更改重新读取之前 ttd.conf 文件中的任何事务的 slo 或 range 值。

如果需要更改 ttd.conf 文件中现有事务的 slo 或 range 值，请执行以下操作：

1. 停止所有 ARM 检测的应用程序。
2. 用 **ovpa stop** 停止 scope 收集器。
3. 停止 Glance 的所有使用。
4. 用 **ttd -k** 停止 ttd。

进行更改后：

1 用 **ovpa start** 重新启动 scope。

2 启动 ARM 检测的应用程序。

监视性能数据

可以使用以下资源和性能管理产品监视事务数据：性能收集组件、 Performance Manager 和 GlancePlus。

... 使用性能收集组件

性能收集组件通过收集和记录长时期数据，使您能分析系统随时间变化的性能，并执行详细的趋势分析。来自性能收集组件的数据可以用 Performance Manager Agent 查看，或导出供多种其他性能监视、记帐、建模和规划工具使用。

用性能收集组件的 extract 程序可导出数据供电子表格和分析程序使用。还可以提取数据进行存档和分析。

要在性能收集组件中监视事务数据，性能收集组件和 ttd 必须正在运行。使用 ovpa 脚本启动性能收集组件可确保按正确顺序启动在 GlancePlus 中查看事务数据必需的 ttd 和 midaemon 进程。

... 使用 Performance Manager

Performance Manager 导入性能收集组件数据，您可将该数据转换成自定义的图形或数字格式。用 Performance Manager，您可以执行事务数据的历史趋势分析，进行更准确的预测。

您可以从 Performance Manager 数据源的“类列表 (Class List)”窗口选择 **TRANSACTION**，然后选择各种事务的图形事务度量。有关详细信息，请参见 Performance Manager 联机帮助，该文档可从 Performance Manager “帮助 (Help)”菜单访问。如果在 Performance Manager 中看不到预期的事务，则关闭当前数据源然后重新连接。

... 使用 GlancePlus

使用 GlancePlus 监视系统可帮助您识别资源瓶颈，并提供有关计算机系统的即时性能信息。GlancePlus 提供了两个窗口：“事务跟踪 (Transaction Tracking)”窗口显示有关您已定义的所有事务的信息，“事务图形 (Transaction Graph)”窗口显示有关单个事务的特定信息。例如，您可以对照您定义的 SLO 查看每个事务的执行情况。有关如何使用 GlancePlus 的详细信息，请参见联机帮助，该文档可从 “帮助 (Help)”菜单访问。

警报

可以使用以下资源和性能管理产品对事务数据发出警报：性能收集组件、Performance Manager 和 GlancePlus。

... 使用性能收集组件

要用性能收集组件生成警报，必须在其警报定义文件 `alarmdef` 中定义警报条件。可以将性能收集组件设置为以多种方式通知您警报条件，如发送电子邮件消息或呼叫您的寻呼机。

要通过 `alarmdef` 文件的语法检查，必须在日志文件中记录该应用程序名称和事务名称的数据，或在 `ttd.conf` 文件中注册这些名称。

对名称中有短划线 (-) 的事务定义警报条件时有限制。要消除此限制，请在 `alarmdef` 文件中用 `ALIAS` 命令重新定义事务名称。

... 使用 GlancePlus

可以配置 adviser 语法以对事务性能发出警报。例如，满足警报条件时，可指示 GlancePlus 将信息显示到 `stdout` 并执行 UNIX 命令 （如 `mailx`），或打开 GlancePlus 主窗口上的黄色或红色“警报 (Alarm)”按钮。有关 GlancePlus 中警报的详细信息，请在“编辑 adviser 语法 (Edit Adviser Syntax)”窗口从“帮助 (Help)”菜单选择**在此窗口上 (On This Window)**。

21 事务跟踪消息

将返回表 2 中列出的错误代码，应用程序开发人员在通过应用程序响应测量 (ARM) 或 Transaction Tracker API 调用检测应用程序时可使用它们：

表 2　　错误代码

错误代码	错误值	含义
-1	EINVAL	无效参数
-2	EPIPE	ttd（注册守护进程）未运行
-3	ESRCH	ttd.conf 文件中找不到事务名称
-4	EOPNOTSUPP	操作系统版本不受支持

通过 ARM 或 Transaction Tracker API 调用检测的应用程序正在运行时，发生的任何错误的返回代码可能来自事务跟踪守护进程 ttd。测量接口守护进程 midaemon 不产生任何错误返回代码。

如果发生 midaemon 错误，请参见 /var/opt/perf/status.mi 文件了解详细信息。

22 事务度量

ARM 代理程序作为 GlancePlus 和性能收集组件的共享组件提供，可生成很多不同的事务度量。要查看度量及其描述的完整列表：

- 对于已安装的 GlancePlus 度量，请使用 GlancePlus 联机帮助或参见 *GlancePlus for HP-UX Dictionary of Performance Metrics*，该文件位于：

  在 UNIX/Linux 上，`/<安装目录>/paperdocs/gp/C/` 下名为 `gp-metrics.txt`。

  安装目录是安装性能收集组件的目录。

- 对于特定平台的已安装性能收集组件度量，请参见该平台的 *HP Operations Agent Dictionary of Operating System Performance Metrics* 文件，后者位于：

  在 UNIX/Linux 上，`/<安装目录>/paperdocs/gp/C/` 下名为 `met<`*平台*`>.txt`。

  在 Windows 上，*%ovinstalldir%*`paperdocs\ovpa\C` 下名为 `met<`*平台*`>.txt`。

23 事务跟踪示例

本章包含的伪代码示例说明如何通过 ARM API 调用检测应用程序，以便应用程序中定义的事务可以用性能收集组件或 GlancePlus 监视。此伪代码示例对应于第 18 章什么是事务跟踪？中描述的实时订单处理场景

本章中包括几个事务配置文件示例，其中的一个对应于实时订单处理场景。

实时订单处理的伪代码

此伪代码示例包括用于为
第 18 章什么是事务跟踪？中所述的实时订单处理场景定义事务的 ARM API 调用。此例程会在接线员*每次*接电话处理客户订单时处理。此示例中以粗体文字突出显示包含 ARM API 调用的行。

```
routine answer calls()
{
*****************************************************
*  Register the transactions if first time in       *
*****************************************************
   if (transactions not registered)
   {
      appl_id = arm_init("Order Processing Application","*", 0,0,0)
      answer_phone_id = arm_getid(appl_id,"answer_phone","1st tran",0,0,0)
      if (answer_phone_id < 0)
         REGISTER OF ANSWER_PHONE FAILED - TAKE APPROPRIATE ACTION
      order_id = arm_getid(appl_id,"order","2nd tran",0,0,0)
      if (order_id < 0)
         REGISTER OF ORDER FAILED - TAKE APPROPRIATE ACTION
      check_id = arm_getid(appl_id,"check_db","3rd tran",0,0,0)
      if (check_id < 0)
         REGISTER OF CHECK DB FAILED - TAKE APPROPRIATE ACTION
      update_id = arm_getid(appl_id,"update","4th tran",0,0,0)
      if (update_id < 0)
         REGISTER OF UPDATE FAILED - TAKE APPROPRIATE ACTION

   } if transactions not registered
*****************************************************
*  Main transaction processing loop
*****************************************************
   while (answering calls)
   {
```

```
      if (answer_phone_handle = arm_start(answer_phone_id,0,0,0) < -1)
         TRANSACTION START FOR ANSWER_PHONE NOT REGISTERED
*****************************************************
*  At this point the answer_phone transaction has    *
*  started.  If the customer does not want to order, *
*  end the call; otherwise, proceed with order.      *
*****************************************************
      if (don't want to order)
          arm_stop(answer_phone_handle,ARM_FAILED,0,0,0)
          GOOD-BYE - call complete
      else
      {
*****************************************************
* They want to place an order - start an order now  *
*****************************************************
         if (order_handle = arm_start(order_id,0,0,0) < -1)
            TRANSACTION START FOR ORDER FAILED
  take order information: name, address, item, etc.
*****************************************************
* Order is complete - end the order transaction     *
*****************************************************
          if (arm_stop(order_handle,ARM_GOOD,0,0,0) < -1)
             TRANSACTION END FOR ORDER FAILED
*****************************************************
*  order taken - query database for availability    *
*****************************************************
         if (query_handle = arm_start(query_id,0,0,0) < -1)
            TRANSACTION QUERY DB FOR ORDER NOT REGISTERED
         query the database for availability
*****************************************************
* database query complete - end query transaction   *
*****************************************************
          if (arm_stop(query_handle,ARM_GOOD,0,0,0) < -1)
             TRANSACTION END FOR QUERY DB FAILED

*****************************************************
* If the item is in stock, process order, and       *
* update inventory.                                 *
*****************************************************
         if (item in stock)
            if (update_handle = arm_start(update_id,0,0,0) < -1)
               TRANSACTION START FOR UPDATE NOT REGISTERED
            update stock
*****************************************************
* update complete - end the update transaction      *
*****************************************************
            if (arm_stop(update_handle,ARM_GOOD,0,0,0) < -1)
               TRANSACTION END FOR ORDER FAILED
*****************************************************
* Order complete - end the call transaction         *
*****************************************************
         if (arm_stop(answer_phone_handle,ARM_GOOD,0,0,0) < -1)
            TRANSACTION END FOR ANSWER_PHONE FAILED
      } placing the order
   GOOD-BYE - call complete
```

```
   sleep("waiting for next phone call...zzz...")
   } while answering calls
   arm_end(appl_id, 0,0,0)
} routine answer calls
```

配置文件示例

此部分包含事务配置文件 /var/opt/perf/ttd.conf 的一些示例。有关 ttd.conf 文件和配置文件关键字的详细信息，请参见第 19 章事务跟踪的工作原理

示例 1（订单处理伪代码示例）

```
# The "*" entry below is used as the default if none of the
# entries match a registered transaction name.

tran=* range=0.5,1,1.5,2,3,4,5,6,7 slo=1
tran=answer_phone* range=0.5,1,1.5,2,3,4,5,6,7 slo=5
tran=order* range=0.5,1,1.5,2,3,4,5,6,7 slo=5
tran=query_db* range=0.5,1,1.5,2,3,4,5,6,7 slo=5
```

示例 2

```
# The "*" entry below is used as the default if none of the
# entries match a registered transaction name.

tran=* range=1,2,3,4,5,6,7,8 slo=5

# The entry below is for the only transaction being
# tracked in this application.  The "*" has been inserted
# at the end of the tran name to catch any possible numbered
# transactions.  For example "First_Transaction1",
# "First_Transaction2", etc.

tran=First_Transaction* range=1,2.2,3.3,4.0,5.5,10 slo=5.5
```

示例 3

```
# The "*" entry below is used as the default if none of the
# entries match a registered transaction name.

tran=*
tran=Transaction_One range=1,10,20,30,40,50,60 slo=30
```

示例 4

```
tran=FactoryStor* range=0.05, 0.10, 0.15 slo=3

# The entries below shows the use of an application name.
# Transactions are grouped under the application name.  This
# example also shows the use of less than 10 ranges and
# optional use of "slo."

[Inventory]
tran=In_Stock range=0.001, 0.004, 0.008
tran=Out_Stock range=0.001, 0.005
tran=Returns range=0.1, 0.3, 0.7

[Pers]
tran=Acctg range=0.5, 0.10, slo=5
tran=Time_Cards range=0.010, 0.020
```

24 高级功能

本章描述性能收集组件如何使用以下 ARM 2.0 API 功能：

- 数据类型
- 用户定义的度量
- scope 检测

如何使用数据类型

表 3 描述如何在性能收集组件中使用数据类型。它是对《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）的“高级主题（Advanced Topics）”部分中“数据类型定义（Data Type Definitions）”的补充。

表 3　　性能收集组件中数据类型的使用

ARM_Counter32	数据记录为 32 位整数。
ARM_Counter64	数据通过类型转换记录为 32 位整数。
ARM_CntrDivr32	进行计算，将结果记录为 32 位整数。
ARM_Gauge32	数据记录为 32 位整数。
ARM_Gauge64	数据通过类型转换记录为 32 位整数。
ARM_GaugeDivr32	进行计算，将结果记录为 32 位整数。
ARM_NumericID32	数据记录为 32 位整数。
ARM_NumericID64	数据通过类型转换记录为 32 位整数。
ARM_String8	忽略。
ARM_String32	忽略。

性能收集组件不记录字符串数据。因为性能收集组件每 5 分钟记录数据，并且记录的是该间隔的活动摘要，因此不能对应用程序提供的字符串进行汇总。

性能收集组件记录前六个可用用户定义的度量的最小、最大和平均值。如果 ARM 检测的应用程序传递 Counter32、String8、NumericID 32、Gauge32、Gauge64、Counter64、NumericID64、String32 和 GaugeDivr32，性能收集组件将以 32 位整数形式记录 Counter32、 NumericID32、

Gauge32、Gauge64、NumericID32 和 NumericID64 的五分钟间隔内的 Min、Max 和 Average 值。String8 和 String32 被忽略，因为字符串不能在性能收集组件中汇总。GaugeDivr32 也被忽略，因为只记录前六个可用的用户定义的度量。（有关更多示例，请参见下一部分用户定义的度量。）

用户定义的度量

此部分是对《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）中“高级主题（Advanced Topics）”下“应用程序定义的度量（Application-Defined Metrics）”的补充。它包含有关性能收集组件如何处理用户定义的度量（ARM 中称为“应用程序定义的度量”）的一些示例。表 4 中的示例显示当程序传递对应数据类型时记录的内容。

表 4　　特定程序数据类型的记录内容示例

... 程序传递的数据类型	... 记录内容
示例 1 String8 Counter32 Gauge32 CntrDivr32	 Counter32 Gauge32 CntrDivr32
示例 2 String32 NumericID32 NumericID64	 NumericID32 NumericID64

表 4　　特定程序数据类型的记录内容示例 （续）

... 程序传递的数据类型	... 记录内容
示例 3 NumericID32 String8 NumericID64 Gauge32 String32 Gauge64	NumericID32 NumericID64 Gauge32 Gauge64
示例 4 String8 String32	（无）
示例 5 Counter32 Counter64 CntrDivr32 Gauge32 Gauge64 NumericID32 NumericID64	Counter32 Counter64 CntrDivr32 Gauge32 Gauge64 NumericID32

因为性能收集组件不能对字符串汇总，因此未记录字符串。

在示例 1 中，只记录计数器、量表和计数器除数。

在示例 2 中，只记录数字。

在示例 3 中，只记录数字和量表。

在示例 4 中，未记录任何内容。

在示例 5 中，因为只记录前六个用户定义的度量，未记录 NumericID64。

scope 检测

scope 数据收集器已通过 ARM API 调用检测。性能收集组件启动时， scope 自动开始记录两个事务：Scope_Get_Process_Metrics 和 Scope_Get_Global_Metrics。这两个事务都将在 HP 性能工具应用程序中。

每五分钟记录事务数据一次，因此您会发现每个间隔内已完成五个 Get Process 事务 （每分钟一个事务）。检测 Scope_Get_Process_Metrics 事务以处理进程数据。如果系统上有 200 个进程，Scope_Get_Process_Metrics 事务所花的时间会比系统上仅有 30 个进程时长。

检测 Scope_Get_Global_Metrics 事务以收集所有五分钟数据（包括全局数据）。这包括 global、application、disk、transaction 及其他数据类型。

要停止记录进程数据和全局事务数据，请删除或注释掉 ttd.conf 文件中的 scope 事务条目。

25 事务库

此附录讨论：

- 应用程序响应测量库 (libarm)
- 按平台分类的的 C 编译器选项示例
- 应用程序响应测量 NOP 库 (libarmNOP)
- 使用 Java 包装

ARM 库 (libarm)

可以用性能收集组件和 GlancePlus 将环境设置为易于编译并使用 ARM 设备。开发所需的库位于 /opt/perf/lib/。有关编译的特定信息，请参见此附录的下一部分。

表 5 中列出的库文件位于 HP-UX 11.11 及未使用性能收集组件和 GlancePlus 的安装上：

表 5　　HP-UX 11.11 及未使用性能收集组件和 GlancePlus 的库文件

/opt/perf/lib/	libarm.0	ARM 的 HP-UX 10.X 兼容共享库 （非线程安全）。如果在 HP-UX 11 上执行用 -larm 在 10.20 上链接的程序，则 11.0 加载程序将自动引用此库。
	libarm.1	HP-UX 11 兼容共享库 （线程安全）。这将由 HP-UX 版本上用 -larm 链接的程序引用。如果 10.20 上链接的程序引用此库 （例如，如果它未用 -L /opt/perf/lib 链接），它可能中止并出现类似下面的错误：“/usr/lib/dld.sl: Unresolved symbol: _thread_once (code) from libtt.sl”。
	libarm.sl	libarm.1 的符号链接
	libarmNOP.sl	ARM 的 “无操作” 共享库 （API 调用成功但没有执行任何操作）；用于测试及未安装性能收集组件的系统。

表 5　　HP-UX 11.11 及未使用性能收集组件和 GlancePlus 的库文件（续）

/opt/perf/examples/arm	libarmjava.sl	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.sl	ARM 的 64 位共享库。
/opt/perf/lib/pa20_64/	请注意，这些文件将由用 +DD64 编译器选项在 HP-UX 11 上编译的程序自动引用。	
	libarm.sl	ARM 的 64 位共享库。
	libarmNOP.sl	ARM 的 64 位 “无操作”共享库 （API 调用成功但没有执行任何操作）；用于测试及未安装性能收集组件的系统。

表 6 中列出的其他库文件位于 IA64 HP-UX 安装上：

表 6　　HP-UX IA64 库文件

/opt/perf/lib/hpux32/	libarm.so.1	ARM 的 IA64/32 位共享库。
/opt/perf/lib/hpux64/	libarm.so.1	ARM 的 IA64/64 位共享库。
/opt/perf/examples/arm	libarmjava.so	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.so	ARM 的 64 位共享库。

因为 ARM 库调用 HP-UX，而 HP-UX 可能从一个版本的操作系统更改为下个版本，程序应使用 -larm 与共享库版本链接。不支持编译已通过 ARM API 调用检测的应用程序和与 ARM 库的存档版本链接 (-Wl, -a archive)。（有关其他信息，请参见第二章第 349 页的事务跟踪守护进程 (ttd)。）

安装了性能收集组件和 GlancePlus 的 AIX 操作系统上的库文件如下所示。

表 7　　AIX 库文件

/usr/lpp/perf/lib/	libarm.a	32 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
/usr/lpp/perf/lib	libarmNOP.a	ARM 的 32 位共享库。此库用于在未安装 Performance Agent/ 性能收集组件的系统上进行测试。
/usr/lpp/perf/lib64/	libarm.a	64 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
/usr/lpp/perf/lib64	libarmNOP.a	ARM 的 64 位共享库。此库用于在未安装 Performance Agent/ 性能收集组件的系统上进行测试。
/usr/lpp/perf/examples/arm	libarmjava.a	ARM 的 32 位共享库
/usr/lpp/perf/examples/arm/arm64	libarmjava.a	ARM 的 64 位共享库。
/usr/lpp/perf/lib/	libarmns.a	32 位存档的 ARM 库。功能与 32 位 libarm.a 相同。
/usr/lpp/perf/lib64/	libarmns.a	64 位存档的 ARM 库。功能与 64 位 libarm.a 相同。

安装了性能收集组件和 GlancePlus 的 Solaris 操作系统上的库文件如下所示。

表 8　　32 位程序的 Solaris 库文件

/opt/perf/lib/	libarm.so	32 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
	libarmNOP.so	ARM 的 32 位共享库。此库用于在未安装性能收集组件的系统上进行测试。

表 9　　Sparc 64 位程序的 Solaris 库文件

/opt/perf/lib/sparc_64/	libarm.so	64 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
	libarmNOP.so	ARM 的 64 位共享库。此库用于在未安装 Performance Agent/ 性能收集组件的系统上进行测试。
/opt/perf/examples/arm	libarmjava.so	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.so	ARM 的 64 位共享库。

表 10　　x86 64 位程序的 Solaris 库文件

/opt/perf/lib/x86_64/	libarm.so	64 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
	libarmNOP.so	ARM 的 64 位共享库。此库用于在未安装 Performance Agent 的系统上进行测试。
/opt/perf/examples/arm	libarmjava.so	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.so	ARM 的 64 位共享库。

必须使用 -xarch=generic64 命令行参数与为 32 位程序提供的其他参数一起编译 64 位程序。

安装了性能收集组件和 GlancePlus 的 Linux 操作系统上的库文件如下所示。

表 11　　Linux 库文件

/opt/perf/lib/	libarm.so	32 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
	libarmNOP.so	ARM 的 32 位共享库。此库用于在未安装性能收集组件的系统上进行测试。
/opt/perf/lib/	libarm.so	64 位共享 ARM 库 （线程安全）。此库由 -larm 链接的程序引用。
	libarmNOP.so	ARM 的 64 位共享库。此库用于在未安装性能收集组件的系统上进行测试。
/opt/perf/examples/arm	libarmjava.so	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.so	ARM 的 64 位共享库。

对于 Linux 2.6 IA 64 位，未实现 32 位 libarm.so 和 libarmjava.so。

按平台分类的 C 编译器选项示例

arm.h include 文件位于 /opt/perf/include/ 中。为方便起见，该文件也可通过符号链接从 /usr/include/ 访问。这意味着您不需要使用 “-I/opt/perf/include/”（尽管也可以这样做）。同样，libarm 驻留在 /opt/perf/lib/ 中，但从 /usr/lib/ 链接。生成 ARM 检测的应用程序时，应始终使用 “-L/opt/perf/lib/”。

- Linux：

  以下示例显示使用 ARM API 的 C 程序的编译命令。

  ```
  cc myfile.c -o myfile -I /opt/perf/include -L
  -Xlinker -rpath -Xlinker /opt/perf/lib
  ```

- Linux 上的 64 位程序：

  ```
  cc -m64 myfile.c -o myfile -I /opt/perf/include -L -Xlinker -rpath
  -Xlinker /opt/perf/lib64
  ```

- HP-UX：

  对于 IA64 平台上的 HP-UX 版本 11.2x，对于 32 位 IA ARM 检测的程序编译，将 -L 参数从 -L/opt/perf/lib 改为 -L/opt/perf/lib/hpux32；对于使用 ARM 的 64 位 IA 程序编译，将该参数改为 -L/opt/perf/lib/hpux64。

  以下示例显示使用 ARM API 的 C 程序的编译命令。

  ```
  cc myfile.c -o myfile -I /opt/perf/include -L /opt/perf/lib -larm
  ```

- Sun Solaris：

  以下示例适用于 Sun Solaris 上的性能收集组件和 GlancePlus：

  ```
  cc myfile.c -o myfile -I /opt/perf/include -L /opt/perf/lib -larm -lnsl
  ```

- Sun Solaris 上的 64 位 Sparc 程序：

  以下示例适用于 Sun Solaris 上的性能收集组件和 64 位程序：

  ```
  cc -xarch=generic64 myfile.c -o myfile -I /opt/perf/include -L /opt/perf/
  lib/sparc_64 -larm -lnsl
  ```

- Sun Solaris 上的 64 位 x86 程序：

  以下示例适用于 Sun Solaris 上的 Performance Agent 和 64 位程序：

  ```
  cc -xarch=generic64 myfile.c -o myfile -I /opt/perf/include -L /opt/perf/
  lib/x86_64 -larm -lnsl
  ```

- IBM AIX：

  IBM AIX 上的文件放置不同于其他平台（使用 /usr/lpp/perf/ 而不是 /opt/perf/），因此 IBM AIX 的示例与其他平台的示例不同：

  ```
  cc myfile.c -o myfile -I /usr/lpp/perf/include -L /usr/lpp/perf/lib -larm
  ```

- IBM AIX 上的 64 位程序：

  以下示例适用于 IBM AIX 上的 Performance Agent 和 64 位程序：

  ```
  cc -q64 myfile.c -o myfile -I /usr/lpp/perf/include -L /usr/lpp/perf/lib64
  -larm
  ```

对于 C++ 编译器，可能需要将 -D_PROTOTYPES 标记添加到编译命令，以便引用 arm.h 文件中的正确声明。

ARM NOP 库

性能收集组件和 Glance 库附带“无操作”库（名为 libarmNOP.*，其中 * 取决于操作系统平台，可为 sl、so 或 a）。此共享库除了返回每个 ARM API 调用的有效状态以外，不执行任何操作。这就允许已通过 ARM 检测的应用程序在未安装性能收集组件或 GlancePlus 的系统上运行。

要在未安装性能收集组件或 GlancePlus 的系统上运行 ARM 检测的应用程序，请将 NOP 库复制到合适的目录（通常是 /< 安装目录 >/lib/），并将它命名为 libarm.sl（根据平台不同，可能是 libarm.so 或 libarm.a）。安装了性能收集组件或 GlancePlus 时，它将用正确的函数库覆盖此 NOP 库（不像其他文件那样被删除）。这就确保了从系统删除性能收集组件或 GlancePlus 时，检测的程序不会中止。

使用 Java 包装

Java 本地接口 (JNI) 包装是为方便使用 Java 应用程序调用 HP ARM2.0 API 而创建的函数。这些包装 (armapi.jar) 包含在 ARM 示例程序中，位于 /< 安装目录 >/examples/arm/ 目录中。安装目录是安装性能收集组件的目录。

示例

Java 包装的示例位于 /< 安装目录 >/examples/arm/ 目录中。此位置还包含 README 文件，说明每个包装的功能。

设置应用程序 (arm_init)

要设置新应用程序，请创建新的 ARMApplication 实例，并传递此 API 的名称和描述。每个应用程序都需要用唯一名称标识。 ARMApplication 类使用 C 函数 arm_init。

语法：

```
ARMApplication myApplication =
new ARMApplication(" 名称 "," 描述 ");
```

设置事务 (arm_getid)

要设置新事务，可选择是否要使用用户定义的度量 (UDM)。 Java 包装使用 C 函数 arm_getid。

使用 UDM 设置事务

如果要使用 UDM，必须先定义新的 ARMTranDescription。ARMTranDescription 为 `arm_getid` 生成数据缓冲区。（另请参阅 `jprimeudm.java` 示例。）

语法：

```
ARMTranDescription myDescription =
new ARMTranDescription(" 事务名称 "," 详细信息 ");
```

如果不想使用详细信息，可以使用另一个构造函数：

语法：

```
ARMTranDescription myDescription =
new ARMTranDescription(" 事务名称 ");
```

添加度量

度量 1-6：

语法：

```
myDescription.addMetric(metricPosition, metricType,
metricDescription);
```

参数：

`metricPosition: 1-6`

`metricType: ARMConstants.ARM_Counter32`
`ARMConstants.ARM_Counter64 ARMConstants.ARM_CntrDivr32`
`ARMConstants.ARM_Gauge32 ARMConstants.ARM_Gauge64`
`ARMConstants.ARM_GaugeDivr32 ARMConstants.ARM_NumericID32`
`ARMConstants.ARM_NumericID64 ARMConstants.ARM_String8`

度量 7：

语法：

```
myDescription.addStringMetric(" 描述 ");
```

随即可以创建事务：

语法：

```
myApplication.createTransaction(myDescription);
```

设置度量数据

度量 1-6：

语法：

```
myTransaction.setMetricData(metricPosition, metric);
```

“度量”示例

```
ARMGauge32Metric metric = new ARMGauge32Metric(start);
ARMCounter32Metric metric = new ARMCounter32Metric(start);
ARMCntrDivr32Metric metric = new ARMCntrDivr32Metric(start, 1000);
```

度量 7：

语法：

```
myTransaction.setStringMetricData(text);
```

不使用 UDM 设置事务

不使用 UDM 设置事务时，可立即创建新事务。可选择是否指定详细信息。

指定详细信息

语法：

```
ARMTransaction myTransaction =
myApplication.createTransaction(" 事务名称 "," 详细信息 ";
```

不指定详细信息

语法：

```
ARMTransaction myTransaction =
myApplication.createTransaction(" 事务名称 ");
```

设置事务实例

要设置新事务实例，请用 ARMTransaction 的方法 `createTransactionInstance()` 创建 ARMTransactionInstance 的新实例。

语法：

```
ARMTransactionInstance myTranInstance =
myTransaction.createTransactionInstance();
```

启动事务实例 (arm_start)

要启动事务实例，可选择是否使用相关器。以下方法使用相关参数调用 C 函数 `arm_start`。

使用相关器启动事务实例

使用相关器时，必须区分获取和传递相关器。

请求相关器

如果事务实例要请求相关器，则按如下所示调用 （另请参阅 `jcorrelators.java` 示例）。

语法：

```
int status = myTranInstance.startTranWithCorrelator();
```

传递父相关器

如果已有先前事务的相关器，要将它传递到事务，语法如下：

语法

```
int status = startTran(parent);
```

参数

parent 是传递的相关器。在先前事务中，可以用方法 getCorrelator() 获取事务实例相关器。

请求和传递父相关器

如果已有先前事务的相关器，要将它传递到事务并请求相关器，语法如下：

语法：

```
int status = myTranInstance.startTranWithCorrelator(parent);
```

参数：

parent 是传递的相关器。在先前事务中，可以用方法 getCorrelator() 获取事务实例相关器。

检索相关器信息

可以用 getCorrelator() 方法如下检索事务实例相关器：

语法：

```
ARMTranCorrelator parent = myTranInstance.getCorrelator();
```

不使用相关器启动事务实例

不使用相关器时，可以如下启动事务实例：

语法：

```
int status = myTranInstance.startTran();
```

startTran 将唯一句柄返回给状态用于更新和停止。

更新事务实例数据

在启动到停止期间，可以更新事务实例的 UDM 任意次数。此部分包装使用相关参数调用 C 函数 arm_update。

使用 UDM 更新事务实例数据

使用 UDM 更新事务实例的数据时，首先必须为度量设置新数据。例如，

```
metric.setData(value)  for ARM_Counter32 ARM_Counter64, ARM_Gauge32,
ARM_Gauge64, ARM_NumericID32, ARM_NumericID64

metric.setData(value,value) for ARM_CntrDivr32 and , ARM_GaugeDivr32

metric.setData(string)  for ARM_String8 and ARM_String32
```

然后可以将度量数据设置成新的（类似设置度量数据部分中的示例）并调用更新：

语法：

```
myTranInstance.updateTranInstance();
```

不使用 UUM 更新事务实例数据

不使用 UDM 更新事务实例的数据时，只是调用更新。该操作发送“检测信号”指示事务实例仍在运行。

语法：

```
myTranInstance.updateTranInstance();
```

提供更大的不透明应用程序专用缓冲区

如果要使用第二个缓冲区格式，必须将字节数组传递给更新方法。（请参见《Application Response Measurement 2.0 API 指南》（*Application Response Measurement 2.0 API Guide*）。）

语法：

```
myTranInstance.updateTranInstance(byteArray);
```

停止事务实例 (arm_stop)

要停止事务实例，可以选择是否使用度量更新停止。

使用度量更新停止事务实例

要使用度量更新停止事务实例，请调用方法 `stopTranInstanceWithMetricUpdate`。

语法：

```
myTranInstance.stopTranInstanceWithMetricUpdate
(transactionCompletionCode);
```

参数：

事务完成代码可以是：

ARMConstants. ARM_GOOD	操作按预期正常运行时使用此值。
ARMConstants.ARM_ABORT	系统中存在基本故障时使用此值。
ARMConstants.ARM_FAILED	在事务正常工作但没有生成结果的应用程序中使用此值。

这些方法使用带请求的参数的 C 函数 arm_stop。

不使用度量更新停止事务实例

要不使用度量更新停止事务实例，请调用方法 stopTranInstance。

语法：

```
myTranInstance.stopTranInstance(transactionCompletionCode);
```

使用完成事务

Java 包装可以使用 arm_complete_transaction 调用。此调用可用于标记已运行指定纳秒数的事务的结束。这样就能实时集成在 ARM 代理程序外测量的事务响应时间。

除了指示事务实例结束以外，还可在可选的数据缓冲区中提供有关事务的其他信息 (UDM)。

（另请参阅 jcomplete.java 示例。）

使用完成事务 （使用 UDM）

语法：

```
myTranInstance.completeTranWithUserData(status,responseTime;
```

参数：

status	• ARMConstants. ARM_GOOD 操作按预期正常运行时使用此值。 • ARMConstants.ARM_ABORT 系统中存在基本故障时使用此值。 • ARMConstants.ARM_FAILED 在事务正常工作但没有生成结果的应用程序中使用此值。
responseTime	这是以纳秒为单位的事务响应时间。

使用完成事务 （不使用 UDM）

语法：

```
myTranInstance.completeTran(status,responseTime);
```

更多文档

有关 Java 类的更多信息，请参见 /< 安装目录 >/examples/arm/ 目录中的 doc 文件夹，它包含每个 Java 类的 html 文档。从 index.htm 开始。

26 记录和跟踪

可使用记录和跟踪机制诊断和排除 HP Operations Agent 中的故障。 HP Operations Agent 将错误、警告和常规消息存储在日志文件中，以便于分析。

跟踪机制帮助您跟踪代理程序操作中的具体问题；您可以将跟踪机制生成的跟踪文件传输到 HP Support 供进一步分析。

记录

HP Operations Agent 在节点上的 System.txt 文件中写入警告、错误消息和信息通知。System.txt 文件的内容揭示代理程序是否如预期在工作。可在以下位置找到 System.txt 文件:

在 Windows 上

*%ovdatadir%*log

在 UNIX/Linux 上

/var/opt/OV/log

此外， HP Operations Agent 在以下文件中添加性能收集组件和 coda 的状态详细信息：

在 Windows 上

- *%ovdatadir%*\status.scope
- *%ovdatadir%*\status.perfalarm
- *%ovdatadir%*\status.ttd
- %ovdatadir%\status.mi
- *%ovdatadir%*\status.perfd-< *端口* >

  在本实例中，< *端口* > 是 perfd 使用的端口。默认情况下， perfd 使用端口 5227。要更改 perfd 的默认端口，请参见第 48 页的配置 RTMA 组件。

- %ovdatadir%\log\coda.txt

在 UNIX/Linux 上

- /var/opt/perf/status.scope
- /var/opt/perf/status.perfalarm
- /var/opt/perf/status.ttd
- /var/opt/perf/status.mi

- /var/opt/perf/status.perfd
- *仅在 vMA 上。* /var/opt/perf/status.viserver
- /var/opt/OV/log/coda.txt

配置记录策略

System.txt 文件大小最大可达 1 MB，然后代理程序开始在新版本的 System.txt 文件中记录消息。可对 HP Operations Agent 的消息记录策略进行配置，限制 System.txt 文件的大小。

要修改默认记录策略，请执行以下步骤：

1 登录到节点。

2 转到以下位置：

在 Windows 上

*%ovdatadir%*conf\xpl\log

在 UNIX/Linux 上

/var/opt/OV/conf/xpl/log

3 用文本编辑器打开 log.cfg 文件。

4 BinSizeLimit 和 TextSizeLimit 参数控制 System.txt 文件的字节大小和字符数。默认情况下，这两个参数都设置为 1000000 （1 MB 和 1000000 个字符）。将默认值更改为所需值。

5 保存文件。

6 使用以下命令重新启动操作监视组件：

a **ovc -kill**

b **ovc -start**

跟踪

开始跟踪 HP Operations Agent 应用程序之前，必须执行一组先决条件任务，这些任务包括标识要跟踪的正确应用程序、设置跟踪类型和生成跟踪配置文件 （如有必要）。

开始跟踪 HP Operations Agent 进程之前，执行以下任务：

1 第 392 页的标识应用程序

2 第 394 页的设置跟踪类型

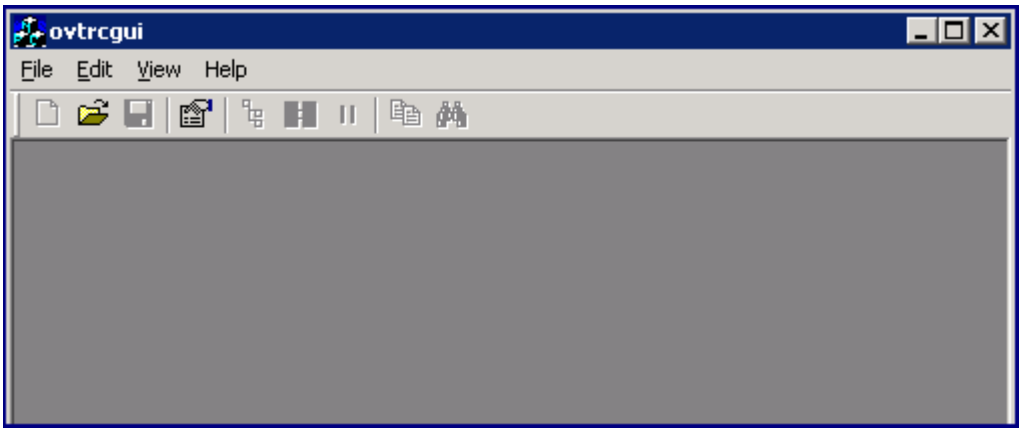
3 *可选。*第 397 页的创建配置文件

标识应用程序

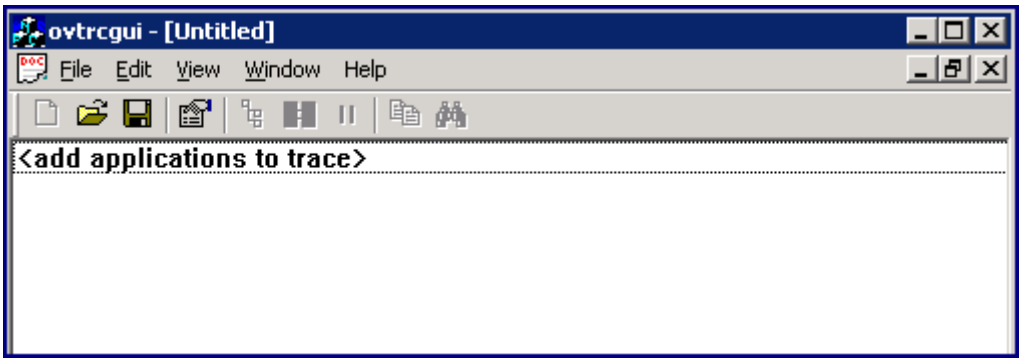
在受管系统上，标识要跟踪的 HP Software 应用程序。使用 ovtrccfg -vc 选项查看所有启用了跟踪的应用程序的名称，以及为每个启用了跟踪的应用程序定义的组件和类别的名称。

或者也可以用 ovtrcgui 实用程序查看启用了跟踪的应用程序的列表。要使用 ovtrcgui 实用程序查看启用了跟踪的应用程序的列表，请执行以下步骤：

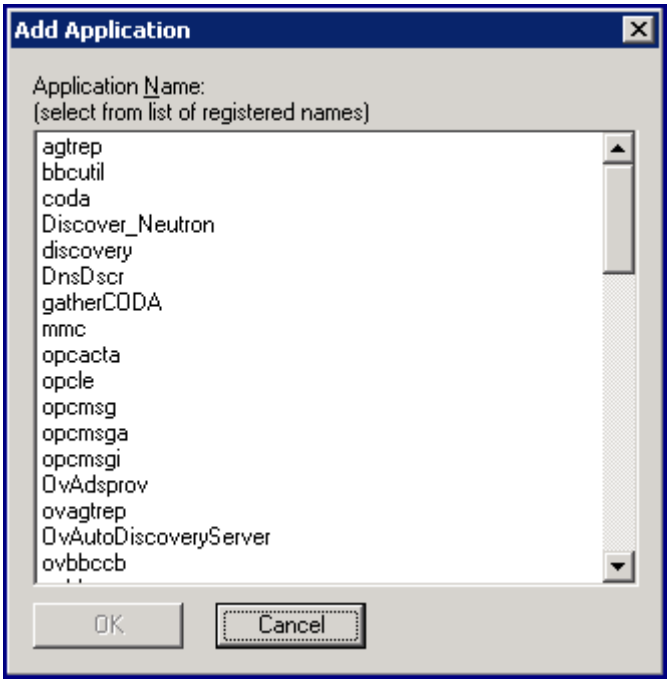
1 从 *%OvInstallDir%*\support 目录运行 ovtrcgui.exe 文件。将打开 ovtrcgui 窗口。



2 在 ovtrcgui 窗口中，单击**文件 (File) → 新建 (New) → 跟踪配置 (Trace Configuration)**。将打开新的跟踪配置编辑器。



3 在 ovtrcgui 窗口中，单击**编辑 (Edit) → 添加应用程序 (Add Application)**。或者右键单击编辑器，然后单击**添加应用程序 (Add Application)**。将打开 “添加应用程序 (Add Application)” 窗口。



“添加应用程序 (Add Application)” 窗口将显示可用的已启用跟踪的应用程序的列表。

设置跟踪类型

启用跟踪机制之前，先决定和设置要为应用程序配置的跟踪类型。要设置跟踪类型，请执行以下步骤：

确定要配置的跟踪类型 （静态或动态），然后执行以下步骤：

1 转到 < *数据目录* >/conf/xpl/trc/ 位置

2 找到 < *应用程序名称* >.ini 文件。如果文件存在，则转到步骤 3。如果 < *应用程序名称* >.ini 文件不存在，则执行以下步骤：

- 用文本编辑器创建新文件。
- 按给定顺序将以下属性添加到文件：DoTrace、 UpdateTemplate 和 DynamicTracing。

  不要将这些属性列在一行中。一个属性列一行。例如：

  ```
  DoTrace=
  UpDateTemplate=
  DynamicTracing=
  ```

- 保存文件。

3 用文本编辑器打开 < *应用程序名称* >.ini 文件。

4 要启用**静态跟踪**，请确保 DoTrace 属性设置为 ON， DynamicTracing 属性设置为 OFF。

5 要启用**动态跟踪**，请确保 DoTrace 和 DynamicTracing 属性都设置为 ON。

6 确保 UpdateTemplate 属性设置为 ON。

7 保存文件。

对于动态跟踪配置，甚至可以在应用程序启动后启用跟踪机制。对于静态跟踪配置，必须在应用程序启动前启用跟踪机制。

跟踪配置文件简介

语法

```
TCF Version <版本号>
APP: "<应用程序名称>"
SINK: File "<文件名>" "maxfiles=[1..100];maxsize=[0..1000];"
TRACE: "<组件名称>" "<类别名称>" <关键字列表>
```

以下部分详细说明了每一行语法。

TCF Version

TCF Version 行指定这是跟踪配置文件，并指定文件的版本号。它区分大小写，并且必须完全按如下所示指定：

语法：

```
TCF Version <版本号>
```

示例：

```
TCF Version 1.1
```

APP

应用程序行定义要跟踪的应用程序的名称。它必须以 APP 开头，后跟冒号 (:) 和一个空格，并且应用程序名称应包含在双引号 ("...") 中。可以指定跟踪多个应用程序。对要跟踪的每个应用程序重复此模式。

语法：

```
APP: "<应用程序名称>"
```

示例：

```
APP: "dbmanager"
APP: "opcmsg"
APP: "poller"
```

SINK

SINK 行指定接收跟踪输出的目标文件。目标必须是同一计算机上的文件。该行必须以 SINK: FILE 开头。该行上的参数应用空格分隔。

SINK: FILE 行有两个参数。

第一个参数是目标文件的名称，必须包含在双引号 ("...") 中。

第二个参数是其他 sink 类型选项。这些选项必须包含在双引号 ("...") 中，并且每个选项后面必须有分号 (;)。

sink 类型 File 的选项包括：

- maxfiles=n
- maxsize=n

在现有跟踪配置文件中，可看到 force 选项。此版本跟踪实用程序不支持此选项，因此它对跟踪机制无影响。配置跟踪机制时，可忽略此选项。

maxfiles

maxfiles 选项后跟 1 到 100 之间的整数值。此选项可用于指定要保留的历史跟踪日志文件数。每次应用程序开始跟踪文件时，将通过对文件名称添加 “.001” 来重命名文件，对以前的文件（如果有）进行备份。如果已经有 “.001” 文件，那么它将重命名为 “.002”，依此类推。如果当前日志文件达到最大大小，则也执行此相同的备份方案。

maxsize

maxsize 选项后跟 0 到 1000 之间的整数值或浮点值，它以兆字节 (MB) 为单位指定用于每个文件的最大磁盘空间。

如果写入文件的最后一个跟踪输出块使文件大于指定的最大值，那么下一次输出时将备份并关闭当前的输出文件，然后创建新的输出文件。 0 值是特殊情况，它对文件数无限制，直到用完磁盘空间为止。

可使用 ovtrcmon 或 ovtrcgui 工具查看接收跟踪输出的目标文件的内容。用标准文本编辑器打开文件时，格式不太整齐。

语法：

```
SINK: File "<文件名>" "maxfiles=[1..100];maxsize=[0..1000];"
```

示例：

```
SINK: File "C:\\TEMP\\Output.trc" "maxfiles=10;maxsize=100;"
```

TRACE

TRACE 行必须以 TRACE 开头，后跟冒号 (:) 和一个空格 ( )。该行上的参数必须用空格分隔。

第一个参数是跟踪组件名称，必须包含在双引号 ("...") 中。可指定多个组件。

第二个参数是跟踪类别名称，也必须包含在双引号 ("...") 中。可指定多个组件。如果在该代码中使用一个标准类别，它会映射到此处指定的字符串值。有关标准类别常量到字符串值的确切映射，请参见相应语言的文档 (C++、 Java)。

语法：

```
TRACE: "<组件名称>" "<类别名称>" <关键字列表>
```

示例：

```
TRACE: "database" "Parms" Error Info Warn
TRACE: "xpl.io" "Trace" Info
```

可使用“*”作为组件名称和/或类别名称。当以应用程序直接从文件读取其配置信息的这种模式使用应用程序时，这很有用。

应用程序尝试确定 component A 和 category B 的设置时，先检查配置是否包含该对的显式跟踪定义。如果有跟踪定义，则使用这些设置。如果没有，则检查是否有 component A 和 category * 的配置。如果有，则使用这些设置。如果没有，则检查是否有 component * 和 category * 的配置。如果有，则使用那些设置。如果没有，则不激活跟踪。

其余的参数是关键字选项的可变列表。列表中至少必须有关键字 Error、Info 或 Warn 中的一个。支持的关键字有：

表 12　　跟踪关键字

关键字	描述
Error	启用标记为错误的跟踪。
Warn	启用标记为警告的跟踪。
Info	启用标记为信息的跟踪。
Support	启用正常跟踪。跟踪输出包括信息、警告和错误消息。建议使用此选项排除故障。可以长时间启用跟踪，因为使用此选项捕获跟踪输出的开销最少。

跟踪配置文件示例

```
TCF Version 3.2
APP: "dbmanager"
SINK: File "C:\\TEMP\\Output.trc" "maxfiles=10;maxsize=100;"
TRACE: "DbManager" "Parms" Error Info Warn Developer
```

创建配置文件

如果要在不使用配置文件的情况下启用跟踪机制，请跳过这一部分并转到第 402 页的用命令行工具启用跟踪并查看跟踪消息。

可以用命令行工具 ovtrccfg、文本编辑器或 ovtrcgui 实用程序（仅适用于 Windows 节点）创建跟踪配置文件。

使用命令行工具

运行以下命令生成跟踪配置文件：

ovtrccfg -app <*应用程序名称*> **[-cm** <*组件名称*>**]** **[-sink** <*文件名*>**]** **-gc** <*配置文件名*>

该命令用您要跟踪的应用程序和组件的详细信息创建配置文件。

使用文本编辑器

如果要用文本编辑器手动创建配置文件，请执行以下步骤：

1 用文本编辑器创建新文件。

2 按以下格式在开头指定配置文件的版本号：

```
TCF Version < 版本号 >
```

例如：

```
TCF Version 1.0
```

3 按以下格式指定要跟踪的应用程序：

```
APP < 应用程序名称 >
```

例如：

```
APP "coda"
```

4 按以下格式指定存储跟踪文件的目标位置：

```
SINK: File "< 文件名 >" "maxfiles=< 最大文件数 >;maxsize=< 最大大小 >"
```

跟踪文件名必须具有扩展名 trc。务必按以下格式指定跟踪输出文件的完整路径：

```
< 驱动器 >:\\< 目录 >\\< 文件名 >
```

例如：

```
SINK: File "C:\\TEMP\\Output.trc" "maxfiles=10;maxsize=100;"
```

如果没有为 SINK 参数指定任何值，跟踪机制会开始将跟踪输出文件放到跟踪的应用程序的主目录中。

5 按以下格式指定要跟踪的组件和类别：

```
TRACE: "< 组件名称 >" "< 类别名称 >" "< 关键字列表 >"
```

如果要跟踪多个组件或类别，则使用换行符添加多个 TRACE 语句。

例如：

```
TRACE: "bbc.cb" "Parms" Error Info Warn
```

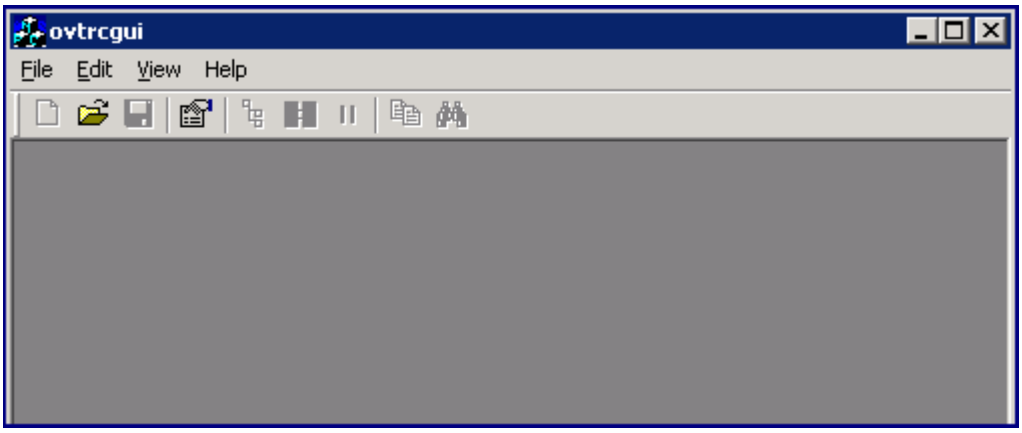
```
TRACE: "bbc.https.server" "Trace" Info
```

6 用 tcf 扩展名保存文件。

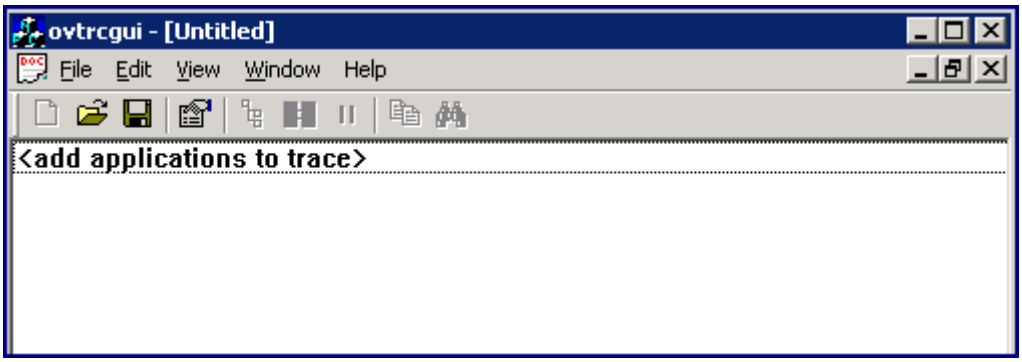
使用跟踪 GUI

在 Windows 节点上，可以用跟踪 GUI （ovtrcgui 实用程序）创建跟踪配置文件。要使用此实用程序创建跟踪配置文件，请执行以下步骤：

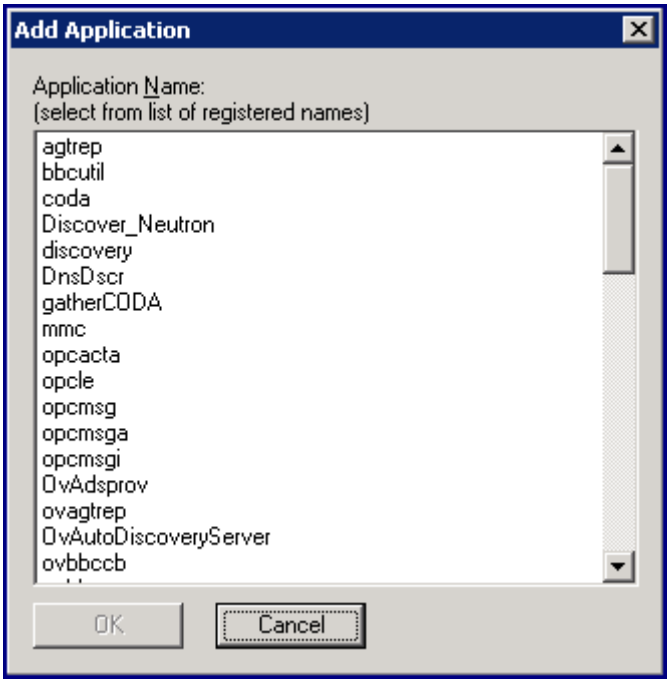
1 从 *%OvInstallDir%*\support 目录运行 ovtrcgui.exe 文件。将打开 ovtrcgui 窗口。



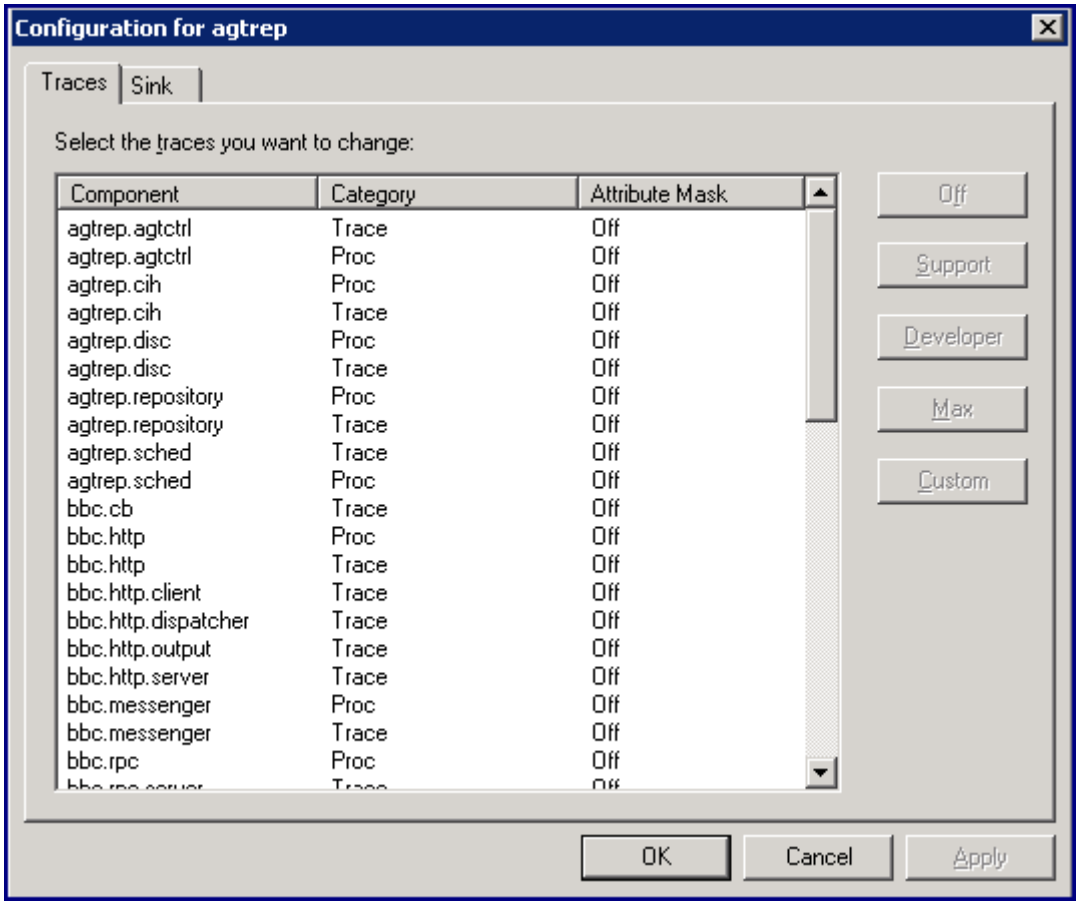
2 在 ovtrcgui 窗口中，单击**文件 (File) → 新建 (New) → 跟踪配置 (Trace Configuration)**。将打开新的跟踪配置编辑器。



3 在 ovtrcgui 窗口中，单击**编辑 (Edit) → 添加应用程序 (Add Application)**。或者右键单击编辑器，然后单击**添加应用程序 (Add Application)**。将打开 “添加应用程序 (Add Application)” 窗口。



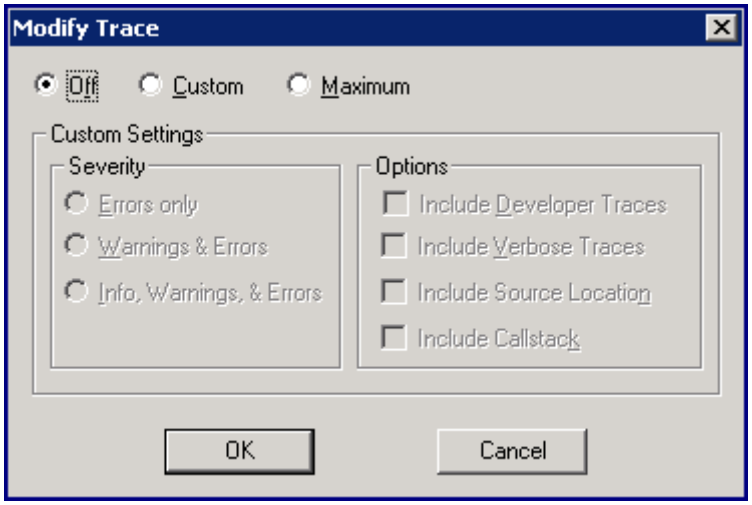
4 选择要跟踪的应用程序，然后单击**确定 (OK)**。将打开 “< *应用程序* > 的配置 (Configuration for <*application*>)” 窗口。



“< *应用程序* > 的配置 (Configuration for <*application*>)”窗口的“跟踪 (Traces)”选项卡列出了所选应用程序的所有组件和类别。默认情况下，所有组件和类别的跟踪都设置为 `Off`。

5 在“跟踪 (Traces)”选项卡中，单击某个组件和类别对，然后单击以下某个按钮：

- **Support**：单击它将收集标记为信息通知的跟踪消息。
- **Developer**：单击它将收集标记为信息通知的跟踪消息及所有 Developer 跟踪。
- **最大 (Max)**：单击它设置跟踪的最大级别。
- **自定义 (Custom)**：单击“自定义 (Custom)”时，将打开“修改跟踪 (Modify Trace)”窗口。



在“修改跟踪 (Modify Trace)”窗口中选择自定义选项，然后选择跟踪级别和所选的选项，再单击**确定 (OK)**。

在“< *应用程序* > 的配置 (Configuration for <*application*>)”窗口中，可单击**关闭 (Off)** 禁用组件类别对的跟踪。

6 单击**确定 (OK)**。

7 转到 Sink 选项卡。

8 在“文件名 (File Name)”文本框中指定跟踪输出文件的名称。文件扩展名必须是 `.trc`。

指定 `.trc` 文件的完整路径。

9 从下拉列表指定历史文件数 （请参见第 396 页的 maxfiles）。

10 从下拉列表指定最大文件大小 （请参见第 396 页的 maxsize）。

11 单击**应用 (Apply)**。

12 单击**确定 (OK)**。

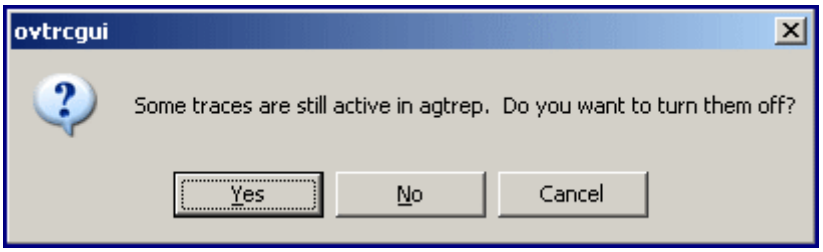
单击**确定 (OK)** 时， `ovtrcgui` 实用程序将启用跟踪机制。

13 单击**文件 (File)** → **保存 (Save)**。将打开 “另存为 (Save As)”对话框。

14 在“另存为 (Save As)”对话框中，浏览到合适的位置，在“文件名 (File Name)”文本框中用 `.tcf` 扩展名指定跟踪配置文件名，然后单击**保存 (Save)**。

ovtrcgui 实用程序用指定名称将新的跟踪配置文件保存到指定位置，并根据文件中指定的配置启用跟踪机制。可以用 ovtrcgui 实用程序打开跟踪配置文件，并添加新配置详细信息。

15 如果尝试关闭跟踪配置编辑器或 ovtrcgui 窗口，将显示以下消息：



16 如果单击**否 (No)**，跟踪机制将继续跟踪系统上配置的应用程序。如果单击**是 (Yes)**，ovtrcgui 实用程序将立即禁用跟踪机制。

用命令行工具启用跟踪并查看跟踪消息

下面概括的过程涵盖了启用跟踪所需的常规步骤顺序。要启用跟踪机制，请执行以下步骤：

1 用 ovtrccfg 发出跟踪配置请求。

ovtrccfg -cf < *配置文件名* >

其中， < *配置文件名* > 是在第 397 页的创建配置文件中创建的跟踪配置文件的名称。

如果不想使用跟踪配置文件，可以用以下命令启用跟踪：

ovtrccfg -app < *应用程序* > **[-cm** < *组件* >**]**

2 如果配置静态跟踪机制，则启动要跟踪的应用程序。

3 运行再现要跟踪的问题所需的应用程序特定命令。所需行为再现后，就可以停止跟踪。

4 用 ovtrcmon 发出跟踪监视请求。

要监视跟踪消息，请使用其他 ovtrcmon 命令选项运行以下某个命令或类似命令：

- 监视来自 /opt/OV/bin/trace1.trc 的跟踪消息，并以文本格式将跟踪消息定向到文件：

  ovtrcmon -fromfile /opt/OV/bin/trace1.trc -tofile /tmp/traceout.txt

- 以详细格式查看来自 /opt/OV/bin/trace1.trc 的跟踪消息：

  ovtrcmon -fromfile /opt/OV/bin/trace1.trc -verbose

- 以详细格式查看来自 /opt/OV/bin/trace1.trc 的跟踪消息并将跟踪消息定向到文件：

  ovtrcmon -fromfile /opt/OV/bin/trace1.trc -short > /tmp/traces.trc

5 要使用 ovtrccfg 停止或禁用跟踪，请运行以下命令：

ovtrccfg -off

6 收集跟踪配置文件和跟踪输出文件。评估跟踪消息或将文件打包传输到 HP Software 在线支持进行评估。系统上可能有多个版本的跟踪输出文件。Maxfiles 选项允许跟踪机制生成多个跟踪输出文件。这些文件具有扩展名 .trc 和后缀 n（n 是 1 到 99999 之间的整数）。

用跟踪 GUI 启用跟踪并查看跟踪消息

在 Windows 节点上，可以用 ovtrcgui 实用程序配置跟踪并查看跟踪消息。

启用跟踪机制

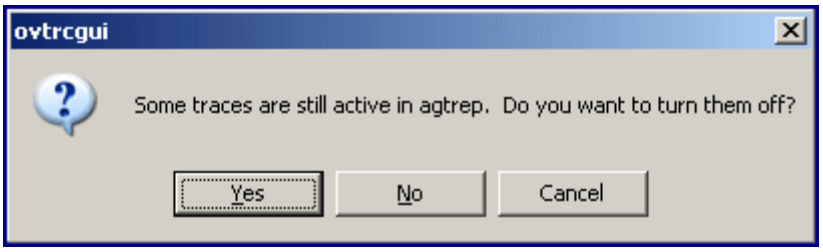
要用 ovtrcgui 实用程序启用跟踪机制，而不使用跟踪配置文件，请执行以下步骤：

1 执行第 399 页的使用跟踪 GUI 中的第 399 页的步骤 1 到第 401 页的步骤 6 的操作。

2 关闭跟踪配置编辑器。

3 提示保存对 Untitled 的更改时，单击**否 (No)**。

将出现以下消息：



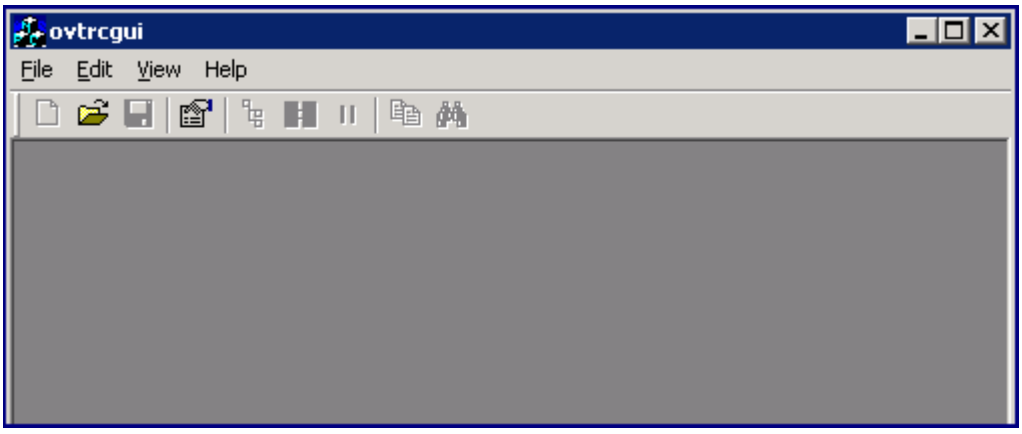
4 单击**否 (No)**。如果单击**是 (Yes)**， ovtrcgui 实用程序将立即禁用跟踪机制。

要用 ovtrcgui 实用程序启用跟踪机制并使用跟踪配置文件，请转到本地系统上跟踪配置文件所在的位置，然后双击跟踪配置文件。或者打开 ovtrcgui 实用程序，单击**文件 (File)** → **打开 (Open)**，选择跟踪配置文件，然后单击**打开 (Open)**。

查看跟踪消息

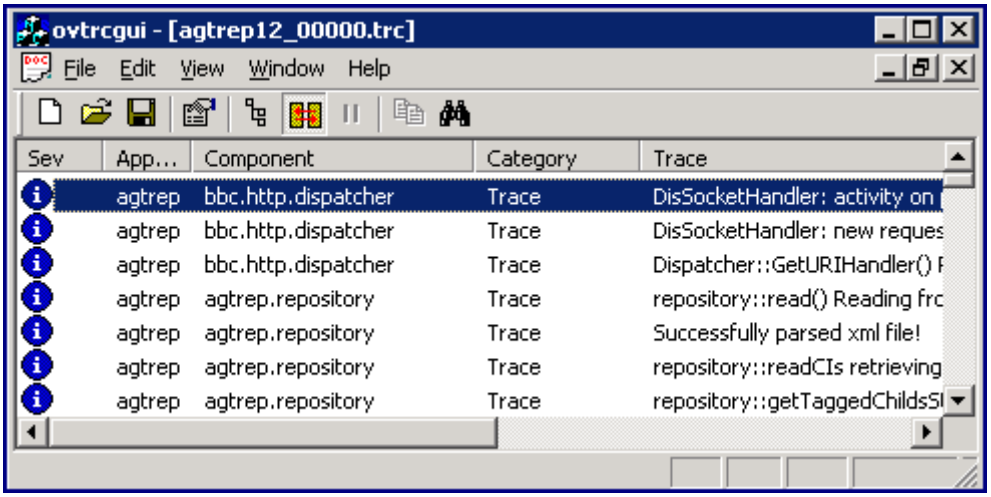
要用 ovtrcgui 实用程序查看跟踪输出文件，请执行以下步骤：

1 从 *%OvInstallDir%*\support 目录运行 ovtrcgui.exe 文件。将打开 ovtrcgui 窗口。



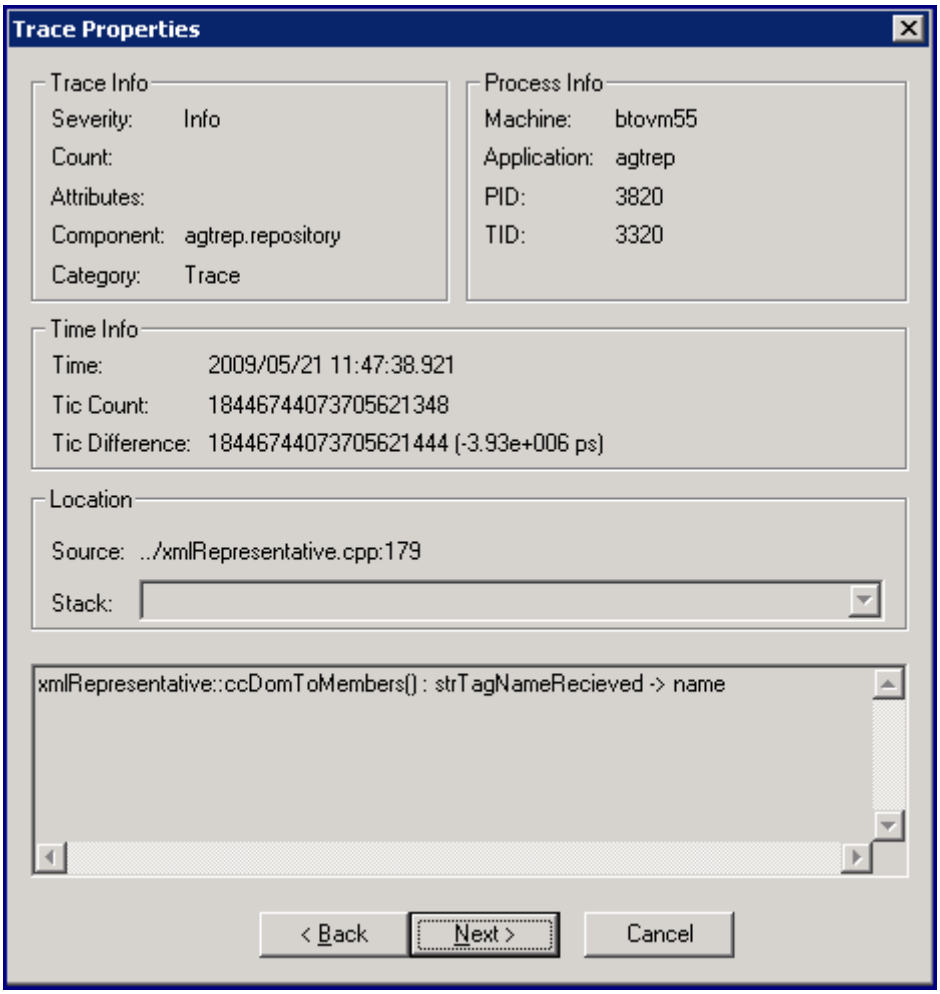
2 单击**文件 (File) → 打开 (Open)**。将打开 “打开 (Open)” 对话框。

3 导航到放置跟踪输出文件的位置，选择 .trc 文件，然后单击**打开 (Open)**。 ovtrcgui 实用程序将显示 .trc 文件的内容。



.trc 文件中的每一新行都表示一个新的跟踪消息。

4 双击跟踪消息查看详细信息。将打开 “跟踪属性 (Trace Properties)” 窗口。



“跟踪属性 (Trace Properties)” 窗口显示以下详细信息：

- 跟踪信息 (Trace Info)：
  - *严重性 (Severity)*：跟踪消息的严重性。
  - *计数 (Count)*：消息的序列号。
  - *属性 (Attributes)*：跟踪消息的属性。
  - *组件 (Component)*：发出跟踪消息的组件的名称。
  - *类别 (Category)*：跟踪的应用程序分配的任意名称。
- 进程信息 (Process Info)：
  - *计算机 (Machine)*：节点的主机名。
  - *应用程序 (Application)*：跟踪的应用程序的名称。
  - *PID*：跟踪的应用程序的进程 ID。
  - *TID*：跟踪的应用程序的线程 ID。
- 时间信息 (Time Info)：
  - *时间 (Time)*：跟踪消息的当地对应时间和日期。
  - *Tic 计数 (Tic count)*：高解析度的经过时间。

— *Tic 差 (Tic difference)：*

- 位置 (Location)
  - — *源 (Source)：*生成跟踪的源的行号和文件名。
  - — *堆栈 (Stack)：*跟踪的应用程序中的调用堆栈的描述。

5 单击**下一步 (Next)** 查看下一条跟踪消息。

6 查看完所有跟踪消息之后，单击**取消 (Cancel)**。

使用跟踪列表视图

默认情况下， `ovtrcgui` 实用程序在跟踪列表视图中显示跟踪文件的跟踪消息。跟踪列表视图以表格格式显示跟踪消息。

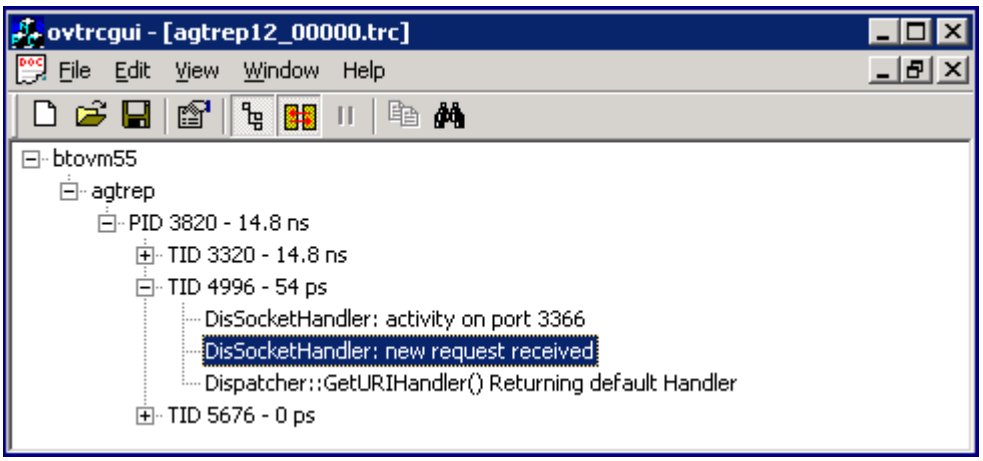
跟踪列表视图使用以下列显示每条跟踪消息：

表 13　　跟踪列表视图

列	描述	
严重性 (Severity)	表示跟踪消息的严重性。该视图用以下图标显示消息的严重性：	
	• Info	
	• Warning	
	• Error	
应用程序 (Application)	显示跟踪的应用程序的名称。	
组件 (Component)	显示生成跟踪消息的被跟踪应用程序之组件的名称。	
类别 (Category)	显示跟踪消息的类别。	
跟踪 (Trace)	显示跟踪消息文本。	

使用过程树视图

您可以在过程树视图中以结构化格式查看跟踪消息。过程树视图根据进程 ID 和线程 ID 对消息排序，并以树视图形式呈现数据。

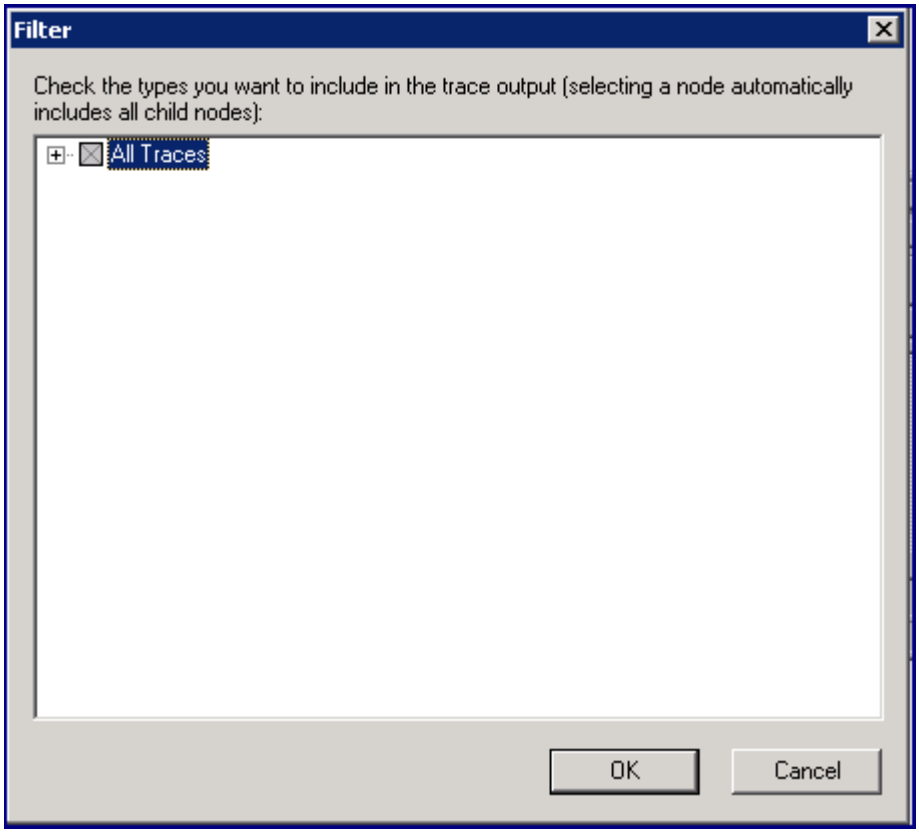


可展开进程 ID 和线程 ID 查看跟踪消息。要回到跟踪列表视图，请单击  。

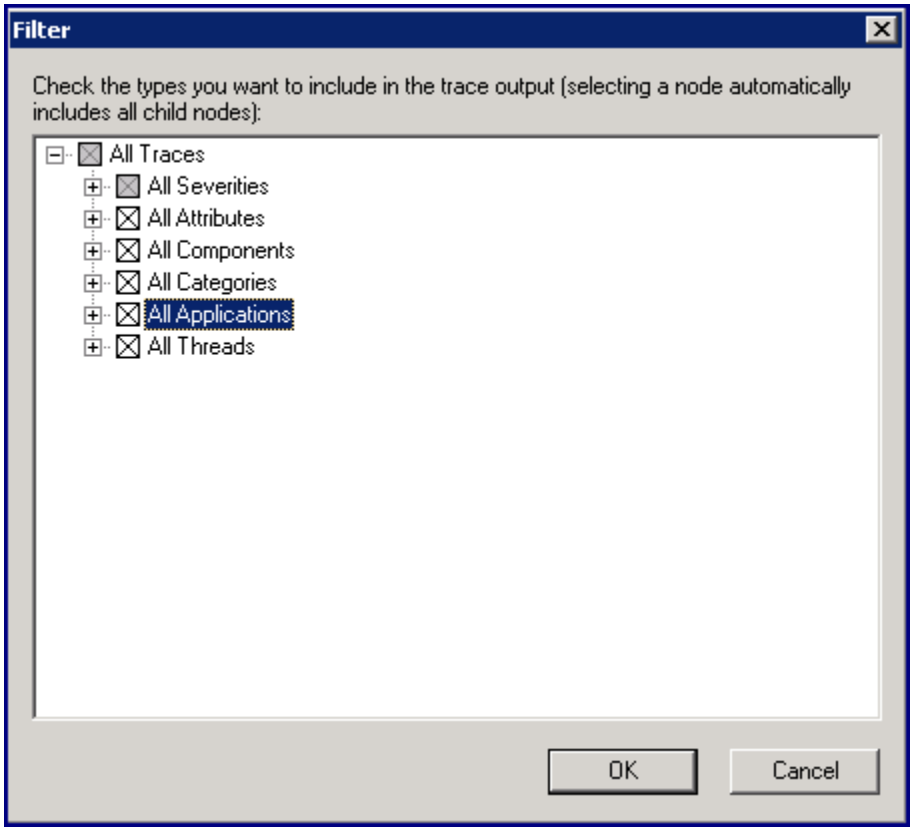
过滤跟踪

ovtrcgui 实用程序显示根据跟踪配置文件中设置的配置而记录到跟踪输出文件的所有跟踪消息。您可以过滤可用消息，在 ovtrcgui 控制台中只显示您选择的消息。要过滤可用跟踪消息，请执行以下步骤：

1 在 ovtrcgui 控制台中，单击**视图 (View)** → **过滤器 (Filter)**。将打开 “过滤器 (Filter)” 对话框。



2 展开**所有跟踪 (All Traces)**。该对话框以树的形式列出所有过滤参数。



3 展开参数可进行过滤跟踪消息的选择。

4 单击**确定 (OK)**。可以在 ovtrcgui 控制台中只看到过滤后的消息。

27 故障排除

本部分描述使用 HP Operations Agent 11.00 时遇到的常见问题的解决方案或变通方法。本部分涵盖的内容包括：

- 操作监视组件
- 性能收集组件
- RTMA

操作监视组件

- *问题：*在 Windows Server 2008 节点上， opcmsga 进程无法工作，且 ovc 命令显示 opcmsga 进程的状态为 aborted。

  解决方案：

  运行以下命令，将 OPC_RPC_ONLY 变量设置为 TRUE：

  ovconfchg -ns eaagt -set OPC_RPC_ONLY TRUE

- *问题：*在 Windows 节点上，策略中的 Perl 脚本不能工作。

  *原因：*策略中可用的 Perl 脚本要求 PATH 配置变量包括 Perl（HP Operations Agent 附带）所在的目录。

  解决方案：

  a 运行以下命令，将 PATH 配置变量设置为 Perl 目录：

  ovconfchg -ns ctrl.env -set PATH "%ovinstalldir%nonOV\perl\a\bin"

  b 运行以下命令，重新启动代理程序：

  — **ovc -kill**

  — **ovc -start**

- *问题：*通过 ovconfchg 命令更改变量值之后，更改不生效。

  原因 1：

  变量需要重新启动代理程序。

  解决方案 1：

  运行以下命令，重新启动代理程序：

  a **ovc -kill**

  b **ovc -start**

  原因 2：

在节点上部署的 ConfigFile 策略将变量设置为特定值。

解决方案：

如果部署的 ConfigFile 策略包含将配置变量设置为特定值的命令，通过 ovconfchg 命令所做的更改不会生效。必须从节点删除 ConfigFile 策略，或修改策略以包含将变量设置为所需值的命令。

原因 3：

节点上的配置文件或作业文件覆盖了更改。

解决方案：

在节点上打开配置文件或作业文件，确保它们不包括相互冲突的变量设置。

- *问题：*更改配置变量 SNMP_SESSION_MODE 的值之后，ovc 显示 opctrapi 进程的状态为 Aborted。

  原因：

  更改配置变量 SNMP_SESSION_MODE 的值之后，HP Operations Agent 会尝试重新启动 opctrapi。有时重新启动 opctrapi 的过程会失败。

  解决方案：

  运行以下命令，重新启动 opctrapi：

  ovc -start opctrapi

性能收集组件

- *问题：*在 HP-UX 11.11 系统上，status.midaemon 文件中出现以下错误：

  mi_shared - MI initialization failed (status 28)

  *原因：*midaemon 二进制文件页面太大。

  *解决方案：*要解决这个问题，请执行以下步骤：

  a 以根用户身份登录到系统。

  b 运行以下命令停止 HP Operations Agent：

  /opt/OV/bin/opcagt -stop

  c 运行以下命令对 midaemon 进行备份：

  cp /opt/perf/bin/midaemon /opt/perf/bin/midaemon.backup

  d 运行以下命令将 midaemon 二进制文件的页面大小减小到 4K：

  chatr +pi 4K /opt/perf/bin/midaemon

  e 运行以下命令启动 HP Operations Agent：

  /opt/OV/bin/opcagt -start

- 安装 HP Operations Agent 之后，如果启用跟踪机制，System.txt 文件中出现以下错误消息：

Scope data source initialization failed

*解决方案：*忽略此错误。

RTMA

- *问题：*在 vSphere Management Assistant (vMA) 节点上， rtmd 进程无法工作，且 ovc 命令显示 rtmd 进程的状态为 aborted。

  *原因：*rtmd 进程无法将系统的主机名解析为 IP 地址。

  解决方案：

  a 以具有根特权的身份登录到节点。

  b 用文本编辑器从 /etc 目录打开 hosts 文件。

  c 找到字词 localhost 所在的行。

  d 从该行的开头删除 # 字符。

  e 保存文件。

  f 运行以下命令，启动所有进程：

  ovc -restart

- *问题：*在 HP Performance Manager 的诊断视图下无法访问数据。

  *原因：*rtmd 进程未运行。

  *解决方案：*要检查 rtmd 进程是否在 HP Operations Agent 节点上运行，请运行 **ovc -status rtmd**。要启动 rtmd 进程，请运行 **ovc -start rtmd**。

- *问题：*在 HP-UX 11.11 系统上， status.perfd 文件中出现以下错误：

  mi_shared - MI initialization failed (status 28)

  *原因：*perfd 二进制文件页面太大。

  *解决方案：*要解决这个问题，请执行以下步骤：

  a 以根用户身份登录到系统。

  b 运行以下命令停止 HP Operations Agent：

  /opt/OV/bin/opcagt -stop

  c 运行以下命令对 perfd 进行备份：

  cp /opt/perf/bin/perfd /opt/perf/bin/perfd.backup

  d 运行以下命令将 perfd 二进制文件的页面大小减小到 4K：

  chatr +pi 4K /opt/perf/bin/perfd

  e 运行以下命令启动 HP Operations Agent：

  /opt/OV/bin/opcagt -start

索引

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z