Hewlett Packard Enterprise

HPE Operations Agent

软件版本：12.01
对于 Windows®、HP-UX、Linux、Solaris 和 AIX 操作系统

用户指南

文档发布日期：2017 年 8 月
软件发布日期：2017 年 8 月

法律声明

担保

Hewlett Packard Enterprise Development Company, L.P 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。HPE 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。

受限权利声明

机密计算机软件。必须拥有 HPE 授予的有效许可证，方可拥有、使用或复制本软件。按照 FAR 12.211 和 12.212，并根据供应商的标准商业许可的规定，商业计算机软件、计算机软件文档与商品技术数据授权给美国政府使用。

版权声明

商标声明

Adobe® 是 Adobe Systems Incorporated 的商标。

Microsoft® 和 Windows® 是 Microsoft 公司集团在美国的注册商标。

UNIX® 是 The Open Group 的注册商标。

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，用于指示软件版本。
- 文档发布日期，该日期将在每次更新文档时更改。
- 软件发布日期，用于指示该版本软件的发布日期。

要检查是否有最新的更新，或者验证是否正在使用最新版本的文档，请访问：**https://softwaresupport.hpe.com**

需要注册 HPE Passport 才能登录此站点。要注册 HPE Passport ID，请访问：**https://hpp12.passport.hpe.com/hppcf/createuser.do**

或单击 HPE 软件支持页面顶部的 **Register** 链接。

此外，如果订阅了相应的产品支持服务，则还会收到更新的版本或新版本。有关详细信息，请与您的 HPE 销售代表联系。

支持

请访问 HPE 软件联机支持网站：**https://softwaresupport.hpe.com**

此网站提供了联系信息，以及有关 HPE 软件提供的产品、服务和支持的详细信息。

HPE 软件联机支持提供客户自助解决功能。通过该联机支持，可快速高效地访问用于管理业务的各种交互式技术支持工具。作为尊贵的支持客户，您可以通过该支持网站获得下列支持：

- 搜索感兴趣的知识文档
- 提交并跟踪支持案例和改进请求
- 下载软件修补程序
- 管理支持合同
- 查找 HPE 支持联系人
- 查看有关可用服务的信息
- 参与其他软件客户的讨论
- 研究和注册软件培训

大多数提供支持的区域都要求您注册为 HPE Passport 用户再登录，很多区域还要求用户提供支持合同。要注册 HPE Passport ID，请访问：

https://hpp12.passport.hpe.com/hppcf/createuser.do

要查找有关访问级别的详细信息，请访问：

https://softwaresupport.hpe.com/web/softwaresupport/access-levels

HPE Software Solutions Now 可访问 HPSW 解决方案和集成门户网站。此网站将帮助您寻找可满足您业务需求的 HPE 产品解决方案，包括 HPE 产品之间的集成的完整列表以及 ITIL 流程的列表。此网站的 URL 为 **https://softwaresupport.hpe.com/**

目录

第 I 部分: 简介

HPE Operations Agent 通过收集指示系统必需元素的运行状况、性能、可用性和资源利用率的各种度量来帮助您监视系统。

通过其嵌入式数据收集器 **oacore**，HPE Operations Agent 会连续收集系统上的性能和运行状况数据，并将收集的数据存储到度量数据存储区中。

将 HPE Operations Agent 与 HPOM、Smart Plug-in (SPI) 和 HPE Operations Manager i (HPE OMi) 结合使用时，可以添加监视业务应用程序、基础结构(系统资源)以及监视的系统上运行的应用程序工作负载的功能。

以下功能可增强 HPE Operations Agent 的数据收集和监视功能。

功能	描述
parmfile	**oacore** 数据收集器的数据收集机制由 **parm** 文件中的设置控制。基于 **parm** 文件中定义的类，**oacore**数据收集器收集代表系统全面运行状况和性能的大型数据集。
Utility 和 Extract	可以使用诸如 Utility 和 Extract 等工具查看存储在度量数据存储区中的特定信息。
自定义数据记录	可以使用自定义数据记录 API 或 DSI 将自定义数据记录到度量数据存储区中。
基线	可以使用基线的流程计算并提供参考值以分析性能趋势并动态设置最佳阈值。
实时度量访问	实时度量访问 (RTMA) 组件提供对系统性能度量的实时访问权限。
alarmdef 文件	可以在 **alarmdef** 文件中定义警报。由于数据由 **oacore** 或其他收集器记录，因此会将该数据与 `alarmdef` 文件中的警报定义进行比较。监视的度量满足或超过定义的条件时，将触发警报或操作。
事务跟踪	可以跟踪事务在应用程序中的进度。事务跟踪提供从事务开始到结束经过的时间的客户端视图，帮助您管理服务级别协议 (SLA)，并且在服务级别目标 (SLO) 超过 **alarmdef** 文件中定义的条件时生成警报。

本文档中使用的约定

本文档中使用以下约定：

约定	描述
`<OvInstallDir>` HPE Operations Agent 的安装目录。	**<OvInstallDir>** 在本文档中用于表示以下位置： • 在 Windows 上：`%ovinstalldir%` • 在 HP-UX/Linux/Solaris 上：`/opt/OV/` • 在 AIX 上：`/usr/lpp/OV/`
`<OvDataDir>` HPE Operations Agent 配置和运行时数据文件的目录。	**<OvDataDir>** 在本文档中用于表示以下位置： • 在 Windows 上：`%ovdatadir%` • 在 HP-UX/Linux/Solaris 上：`/var/opt/OV/` • 在 AIX 上：`/var/opt/OV/`
`<OvInstallBinDir>` bin 目录包含 HPE Operations Agent 的所有二进制文件(可执行文件)。	**<OvInstallBinDir>** 在本文档中用于表示以下位置： • 在 Windows x64 上：`%ovinstalldir%\bin\win64\` • 在 Windows x32 上：`%ovinstalldir%\bin\win32\` • 在 Windows x86 上：`%ovinstalldir%\bin\` • 在 HP-UX/Linux/Solaris 上：`/opt/OV/bin/` • 在 AIX 上：`/usr/lpp/OV/bin/`

第 II 部分: 配置

在基于 HPOM 的管理环境中安装 HPE Operations Agent 后，可以从中央控制台监视和管理网络环境中部署的系统和应用程序。在节点上部署 HPOM 策略后，可以使用操作监视组件的不同组件。配置变量可用于配置 HPE Operations Agent 的默认行为。可以使用 ovconfchg 命令将所需值分配给这些变量。可以配置监视代理程序、SNMP 陷阱拦截器、消息风暴抑制、备份服务器、RTMA 组件和安全组件。

第 1 章: 使用 HPE 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监视。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区中可用的数据。
- **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：

- 对于 HPOM on Windows：请参见 HPOM for Windows 联机帮助中的“事件策略编辑器 (Event Policy Editors)”部分。
- HPOM on UNIX/Linux：请参见《HPOM for UNIX 9.10 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：

   在 Windows 上：

   ```
   %ovinstalldir%bin
   ```

   在 HP-UX/Linux/Solaris 上：

```
/opt/OV/bin
```

在 AIX 上：

```
/usr/lpp/OV/bin
```

4. 运行以下命令：

- 使用非默认的公共字符串：

  ```
  ovconfchg -ns eaagt SNMP_COMMUNITY <community_string>
  ```

  在此实例中，<community_string> 是您选择的非默认公共字符串。

- 使用不同的公共字符串：

  ```
  ovconfchg -ns eaagt SNMP_COMMUNITY_LIST <list_of_community_strings>
  ```

  在此实例中，<list_of_community_strings> 是您选择的公共字符串的逗号分隔列表。HPE 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”
  ```

  HPE Operations Agent 首先尝试建立与节点的 SNMP 会话，并尝试使用公共字符串 C1 针对 OID 执行 SNMP Get 操作。如果操作不成功，则 HPE Operations Agent 使用公共字符串 C2 执行相同的操作，以此类推。

  备注: 如果 HPE Operations Agent 无法使用所有通过 SNMP_COMMUNITY_LIST 指定的公共字符串，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。

持续保存监视的对象

可以将 HPE Operations Agent 配置为定期存储监视的对象和会话变量的值。 存储监视的对象和会话变量的值可确保值被保留，并且在中断或出现故障时可用。

OPC_MON_SAVE_STATE 变量允许您将代理程序配置为保留监视的对象和会话变量的值。

要确保代理程序配置为定期存储监视的对象和会话变量的值，请执行以下步骤：

1. 以根身份或管理员身份登录节点。
2. 运行以下命令：

   ```
   ovconfchg -ns eaagt -set OPC_MON_SAVE_STATE TRUE
   ```

   代理程序开始保留监视的对象和会话变量的值。

安装 HPE Operations Agent 12.01 会在以下方面影响 OPC_MON_SAVE_STATE 变量：

- 如果您在安装 HPE Operations Agent 12.01 之前没有将该变量设置为任何值，则 OPC_MON_SAVE_STATE 变量假设为值 FALSE。
- 如果您在安装 HPE Operations Agent 12.01 之前使用 ovconfchg 命令为该变量设置了值(TRUE 或 FALSE)，则配置的值在安装过程完成后保持有效。

例如，如果在安装版本 12.01 之前已使用命令 `ovconfchg -ns eaagt -set OPC_MON_SAVE_STATE TRUE`，则在安装 HPE Operations Agent 12.01 之后将保留相同值(TRUE)。

增强安全参数以执行 SNMPv3 GET

在 HPE Operations Agent 节点上，从 HPOM 管理服务器部署测量阈值策略后，监视代理程序组件 (**opcmona**) 将监视 MIB 对象。

要查询 MIB 数据，**opcmona** 在指定对象 ID (OID) 上执行 **SNMPv3 GET**。启用 **opcmona** 执行 **SNMPv3Get** 还需要其他安全参数。下表列出了这些参数：

参数	描述
SNMPV3_ENGINEID	指定用于唯一识别 SNMP 实体的 SNMP 引擎 ID。
SNMPV3_USER	指定管理员创建的 SNMPv3 用户名。
SNMPV3_AUTHTYPE	指定用于加密密码的协议。 您可以使用消息摘要算法 5 (MD5) 或安全哈希算法 (SHA) 协议加密密码。
SNMPV3_ AUTHPASSPHRASE	指定使用 **opcpwcrpt** 实用程序加密的密码。 **备注:** 有关详细信息，请参见增强安全参数以执行 SNMPv3 GET (第 20 页)。
SNMPV3_ENCRYPTTYPE	指定用于加密协议数据单元 (PDU) 的协议。 您可以使用数据加密标准 (DES) 或高级加密标准 (AES) 协议加密 PDU。
SNMPV3_ ENCRYPTPASSPHRASE	指定 DES 和 AES 用于加密 PDU 的密钥。 **备注:** 管理员创建的加密密钥使用 **opcpwcrpt** 实用程序进行加密。 有关详细信息，请参见增强安全参数以执行 SNMPv3 GET (第 20 页)。

启用 opcmona 执行 SNMPv3Get

在下列情景之一中可启用 **opcmona** 执行 **SNMPv3Get**:

- 当使用节点信息策略配置 opctrapi 时
- 当未使用节点信息策略配置 opctrapi 时

当使用节点信息策略配置 opctrapi 时

如果使用节点信息策略配置 **opctrapi**，节点上将提供 SNMP 实体的引擎 ID。有关详细信息，请参见使用节点信息策略配置 opctrapi。在测量阈值策略中添加引擎 ID 以启用 **opcmona** 执行 **SNMPv3Get**。

备注: 可以运行 `ovconfget eaagt` 命令检索引擎 ID。您将获得以下输出:

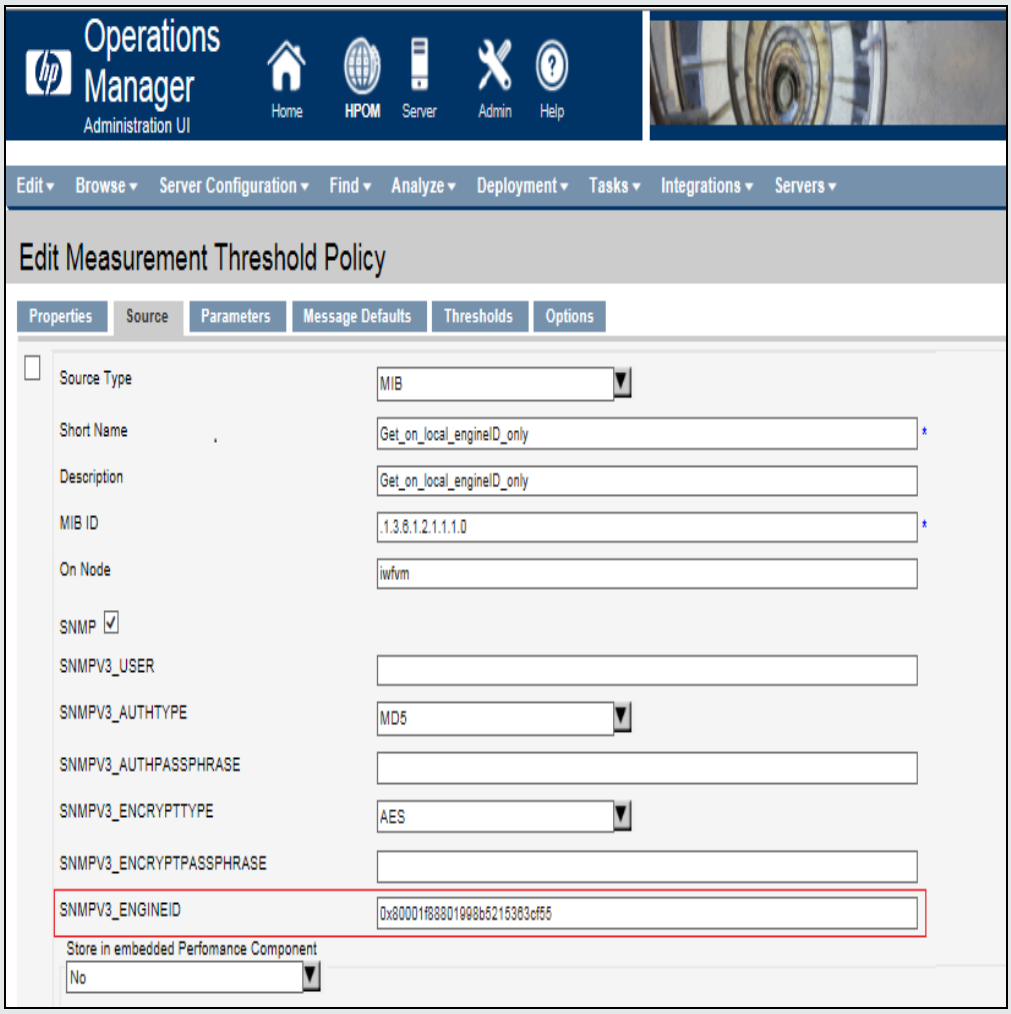
```
OPC_INSTALLATION_TIME=Wed Jan 7 22:42:20 IST 2015
OPC_NODENAME=iwfvm05835.hpswlabs.hp.com
SNMP_V3_
USERSDATA=demosnmpv3user:MD5:7*=*C61Ntd=@*Eb#@*E0##D:DES:7*=*C61Ntd=@*Eb
#@*E0##D:0x80001f88801998b5215363cf55
```

在此实例中，引擎 ID 是 0x80001f88801998b5215363cf55。

示例

我们假设配置 **opctrapi** 使用的引擎 ID 是 0x80001f88801998b5215363cf55。

如下所示在测量阈值策略中添加引擎 ID:



将 SNMP 实体的引擎 ID 添加到测量阈值策略后，**opcmona** 便可以在 SNMP 实体的 OID 上执行 **SNMPv3Get** 了。

当未使用节点信息策略配置 opctrapi 时

如果未使用节点信息策略配置 **opctrapi**，必须将以下内容添加到测量阈值策略：

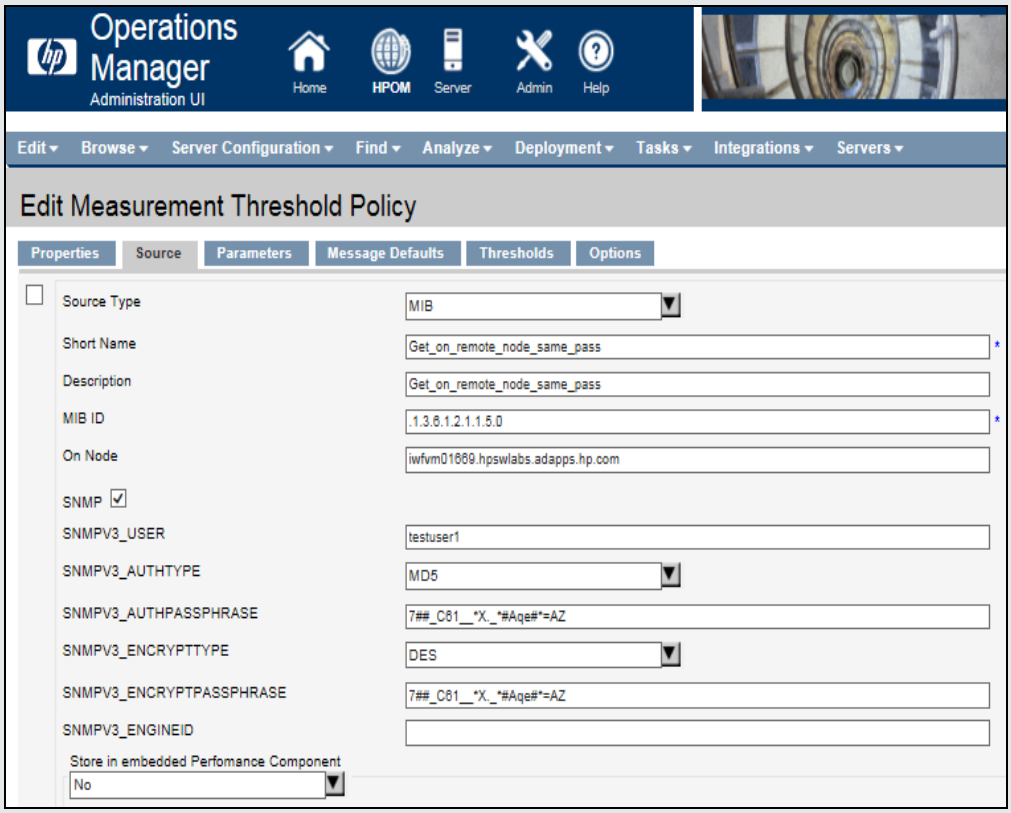
SNMPV3_USER `<user name>`
SNMPV3_AUTHTYPE `<authentication method>`
SNMPV3_AUTHPASSPHRASE `<authentication password>`
SNMPV3_ENCRYPTTYPE `<encryption method>`
SNMPV3_ENCRYPTPASSPHRASE `<encryption key>`

备注:

SNMPV3_AUTHPASSPHRASE 和 SNMPV3_ENCRYPTPASSPHRASE 将自动加密。

如果加密密码和身份验证密码相同，则可以将 SNMPV3_ENCRYPTPASSPHRASE 变量值留空。

例如：



将 SNMP 实体参数添加到测量阈值策略后，**opcmona** 便可以在 SNMP 实体的 OID 上执行 SNMPv3Get 了。

使用 SourceEX API 将安全参数添加到策略

引入了新接口 **SourceEx_SNMPV3** 以支持用于监视 SNMPv3 MIB 的多个安全参数。SourceEx_SNMPV3 API 允许您使用 Perl 脚本将安全参数添加到测量阈值策略。

备注: 无法在测量阈值策略的 vbscript 中使用 SourceEx_SNMPV3 API。

在以下场景中，使用以下语法将安全参数添加到测量阈值策略：

1. 使用节点信息策略配置 **opctrapi** 时：

```
Policy-> SourceEx_SNMPV3 ("SNMP\\<object id>[\\<hostname>]
","<Engine ID>");
```

备注: 仅指定引擎 ID 时，**opcmona** 将从为 **opctrapi** 部署的节点信息策略中获取其他安全参数以接收 SNMPv3 陷阱。有关详细信息，请参见使用节点信息策略配置 opctrapi。

例如：

```
ADVMONITOR "TestMIBviaPerl_V3"
DESCRIPTION "<initial policy version>"
SCRIPTTYPE "Perl"
INSTANCEMODE  SAME
MAXTHRESHOLD
SEPARATORS "                "
SEVERITY Unknown
EXTERNAL "SNMPPolicy"
DESCRIPTION ""
MSGCONDITIONS
DESCRIPTION "New threshold level"
CONDITION_ID "c5e0a2e4-a401-4186-876f-8b20b1ffcd8b"
CONDITION
THRESHOLD
SCRIPT "
use warnings;
my $MetricEx = $Policy->SourceEx_SNMPV3
(\"SNMP\\\\\\\\.1.3.6.1.2.1.1.1.0\\\\\\\\1x.18x.28.7x
\",'0x8000000001020304');
```

在此实例中：

参数	值
<object id>	\\.1.3.6.1.2.1.1.1.0
<hostname>	\\1x.18x.28.7x
<Engine ID>	0x8000000001020304

2. 未使用节点信息策略配置 **opctrapi** 时：

```
$Policy-> SourceEx_SNMPV3 ("SNMP\\<object id>[\\<hostname>]",
"<user name>","<authentication method>","<authentication
password>","<encryption method>","<encryption key>");
```

例如：

```
ADVMONITOR "TestMIBviaPerl_V3"
DESCRIPTION "<initial policy version>"
```

```
SCRIPTTYPE "Perl"
INSTANCEMODE  SAME
MAXTHRESHOLD
SEPARATORS "                "
SEVERITY Unknown
EXTERNAL "SNMPPolicy"
DESCRIPTION ""
MSGCONDITIONS
DESCRIPTION "New threshold level"
CONDITION_ID "c5e0a2e4-a401-4186-876f-8b20b1ffcd8b"
CONDITION
THRESHOLD
SCRIPT "
use warnings;
my $MetricEx = $Policy->SourceEx_SNMPV3
(\"SNMP\\\\\\\\\.1.3.6.1.2.1.1.1.0\\\\\\\\\1x.18x.28.7x\",
\"TestUser\",\"MD5\",'7*=*C61Ntd=@*Eb#@*E0##D',\"DES\",'7*=*C61Ntd=@*Eb
#@*E0##D');
```

在此实例中：

参数	值
<object id>	\\.1.3.6.1.2.1.1.1.0
<hostname>	\\1x.18x.28.7x
<user name>	TestUser
<authentication method>	MD5
<authentication password>	'7*=*C61Ntd=@*Eb#@*E0##D'
<encryption method>	DES
<encryption key>	'7*=*C61Ntd=@*Eb#@*E0##D'

注： 确保密码用单引号括起来。

远程配置性能收集组件

可以从管理服务器远程对受管节点执行某些配置任务。除了在每个节点上本地执行性能收集组件的配置任务外，还可以从 HP Operations Manager 控制台使用一组特殊的策略和工具来配置并使用性能收集组件的多个节点。

只有在 HP Operations Manager for Windows 或 HP Operations Manager on UNIX/Linux 管理服务器上安装了 HPE Operations Agent 部署包时，此功能才可用。

开始之前

在开始从 HP Operations Manager 控制台远程配置和控制性能收集组件之前，必须将 HPE Operations Agent 检测组中的检测文件部署到正在运行代理程序的节点上。

要从 HP Operations Manager for Windows 控制台部署检测，请执行以下步骤：

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署检测，不要在虚拟节点上部署

1. 在控制台树中，右键单击运行代理程序的节点或节点组，然后单击**所有任务 > 部署辅助工具**。将打开“部署辅助工具”对话框。
2. 在“部署辅助工具 (Deploy Instrumentation)”对话框中，单击 HPE Operations Agent，然后单击**确定**。开始在节点上部署必要的检测文件。

要从 HP Operations Manager on UNIX/Linux 控制台部署检测，请执行以下步骤：

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署检测，不要在虚拟节点上部署

1. 登录到管理 UI。
2. 单击**部署 > 部署配置**。
3. 在“分发参数”部分选择**规范**，然后单击**请选择**。将打开**选择器**弹出框。
4. 在**选择器**弹出框中，选择运行代理程序的节点。
5. 选择**强制更新**选项覆盖旧的检测文件。
   在从旧版本的代理程序升级的节点上选择此选项。
6. 单击**分发**。

部署 OA-PerfCollComp-opcmsg 策略

当性能收集组件生成警报时，OA-PerfCollComp-opcmsg 策略将警报消息发送到 HPOM 消息浏览器。该策略位于 **HP Operations Agent > 性能收集组件 > 消息拦截器 (Message Interceptor)** 策略组中。在部署性能收集组件的其他策略之前，先在节点上部署此策略。

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署策略，不要在虚拟节点上部署。

配置性能收集组件

HPE Operations Agent 的性能收集组件的行为取决于以下文件中指定的配置设置：

- 收集参数文件 (**parm**)
- 警报定义文件 (**alarmdef**)

有关收集参数和警报定义文件的详细信息，请参见《HPE Operations Agent 概念指南》中的“性能收集组件”部分。

配置 parm 文件

parm 文件定义 **oacore** 收集器的数据收集机制。HPE Operations Agent 在每个节点上部署 **parm** 文件，该文件位于以下路径：

在 HP-UX、Solaris、AIX 和 Linux 上：**/var/opt/perf**

在 Windows 上：%ovdatadir%

您可以修改 **parm** 文件中指定的设置，以自定义数据收集机制。但是，如果用 HPE Operations Agent 管理很多节点，在每个节点上修改 **parm** 文件的每个单独副本就变得有些困难。

借助 HP Operations Manager 控制台，可以从管理服务器将修改后的 **parm** 文件集中部署到多个节点。

从 HP Operations Manager for Windows

HP Operations Manager for Windows 控制台为您提供 ConfigFile 策略，这些策略帮助您从中央管理服务器跨多个节点部署对 **parm** 文件的任何更改。不同 ConfigFile 策略可用于不同的节点操作系统。

要通过编辑 **parm** 文件修改收集机制，请执行以下步骤：

1. 标识希望采用修改后的收集机制的节点。
2. 在控制台树中，单击**策略管理 > 策略组 (Policy groups) > HPE Operations Agent > 性能收集组件 (Performance Collection Component) > 收集配置 (Collection configuration)**。用于配置 **parm** 文件的 ConfigFile 策略显示在详细信息窗格中。
3. 为希望采用修改后的收集机制的平台双击 ConfigFile 策略(例如：对于 HP-UX，双击 **parm** 文件)。将打开“<platform> 的 **parm** 文件 (parm file for <platform>)”对话框。
4. 在“数据 (Data)”选项卡中修改设置。有关 **parm** 文件中的配置参数的更多详细信息，请参见《HPE Operations Agent 用户指南》中的“parm 文件参数”部分。
5. 单击**保存并关闭**。在详细信息窗格中，策略的版本增加了 .1。

6. 在您选择的节点上部署更新后的策略。

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署策略，不要在虚拟节点上部署

从 HP Operations Manager on UNIX/Linux 9.10

HP Operations Manager on UNIX/Linux 9.10 控制台为您提供 ConfigFile 策略，这些策略帮助您从中央管理服务器跨多个节点部署对 **parm** 文件的任何更改。不同 ConfigFile 策略可用于不同的节点操作系统。

要从 HP Operations Manager for UNIX 9.10 控制台通过编辑 **parm** 文件修改收集机制，请执行以下步骤：

1. 标识希望采用修改后的收集机制的节点。
2. 在控制台中，单击**浏览 > 所有策略组**。页面上将显示所有可用策略组的列表。
3. 单击 **H**。将显示 HPE Operations Agent 策略组。
4. 依次单击 **HP Operations Agent**、**性能收集组件 (Performance Collection Component)** 和**收集配置 (Collection Configuration)**。将显示 **parm** 文件的可用 ConfigFile 策略的列表。
5. 为希望采用修改后的收集机制的平台单击 ConfigFile 策略。将显示策略“OA_<platform>ParmPolicy”页面。
6. 单击  ，然后单击**编辑 (原始模式)**。将显示“编辑配置文件策略...(Edit Config File policy...)”页面。
7. 在“内容”选项卡中修改设置

   有关 **parm** 文件中的配置参数的更多详细信息，请参见《HPE Operations Agent 用户指南》中的“parm 文件参数”部分。
8. 单击**保存**。
9. 在您选择的节点上部署更新后的策略。

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署策略，不要在虚拟节点上部署

配置 alarmdef 文件

警报定义文件 (**alarmdef**) 为性能子代理程序提供默认警报生成流程规范。HPE Operations Agent 在每个节点上部署 **alarmdef** 文件，该文件位于以下路径：

在 HP-UX、Solaris、AIX 和 Linux 上： `/var/opt/perf/`

在 Windows 上： `%ovdatadir%`

您可以修改 **alarmdef** 文件中的默认设置，以自定义警报生成机制。可以使用 HP Operations Manager 控制台在多个节点上集中分发修改后的 **alarmdef** 文件。

从 HP Operations Manager for Windows

HP Operations Manager for Windows 控制台为您提供 ConfigFile 策略，这些策略帮助您从中央管理服务器跨多个节点部署对 **alarmdef** 文件的任何更改。不同 ConfigFile 策略可用于不同的节点操作系统。

要通过编辑 **alarmdef** 文件修改收集机制，请执行以下步骤：

标识希望采用修改后的收集机制的节点。

1. 在控制台树中，单击**策略管理 > 策略组 (Policy groups) > HP Operations Agent > 性能收集组件 (Performance Collection Component) > 警报定义 (Alarm definition)**。用于配置 **alarmdef** 文件的 ConfigFile 策略显示在详细信息窗格中。
2. 为希望采用修改后的收集机制的平台双击 ConfigFile 策略(例如：对于 HP-UX，双击 alarmdef 文件)。将打开“<platform> 的 alarmdef 文件 (Alarmdef file for <platform>)”对话框。
3. 在“数据 (Data)”选项卡中修改设置。有关 **alarmdef** 文件中的配置参数的更多详细信息，请参见《HPE Operations Agent 用户指南》中的“alarmdef 文件参数”部分。
4. 单击**保存并关闭**。在详细信息窗格中，策略的版本增加了 .1。
5. 在您选择的节点上部署更新后的策略。

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署策略，不要在虚拟节点上部署

从 HP Operations Manager on UNIX/Linux 9.10

HP Operations Manager on UNIX/Linux 9.10 控制台为您提供 ConfigFile 策略，这些策略帮助您从中央管理服务器跨多个节点部署对 **alarmdef** 文件的任何更改。不同 ConfigFile 策略可用于不同的节点操作系统。

要从 HP Operations Manager for UNIX 9.10 控制台通过编辑 **alarmdef** 文件修改收集机制，请执行以下步骤：

1. 标识希望采用修改后的警报机制的节点。
2. 在控制台中，单击**浏览 > 所有策略组**。页面上将显示所有可用策略组的列表。
3. 单击 **H**。将显示 HP Operations Agent 策略组。

4. 依次单击 **HP Operations Agent**、**性能收集组件 (Performance Collection Component)** 和**警报定义 (Alarm Definition)**。将显示 **alarmdef** 文件的可用 ConfigFile 策略的列表。
5. 为希望采用修改后的收集机制的平台单击 ConfigFile 策略。将显示策略“OA_<platform>AlarmdefPolicy”页面。
6. 单击  ，然后单击**编辑 (原始模式)**。将显示“编辑配置文件策略...(Edit Config File policy...)”页面。
7. 在“内容 (Content)”选项卡中修改设置。有关 **alarmdef** 文件中的配置参数的更多详细信息，请参见《HPE Operations Agent 用户指南》中的“alarmdef 文件参数”部分。
8. 单击**保存**。
9. 在您选择的节点上部署更新后的策略。

备注: 如果监视群集节点，请确保在构成群集的所有节点上部署策略，不要在虚拟节点上部署

远程使用 HPE Operations Agent

可以用 HP Operations Manager 控制台启动、停止、监视和查看 HPE Operations Agent 的详细信息。可以从 HP Operations Manager 控制台使用不同工具管理 HPE Operations Agent 的操作。必须在部署代理程序的节点上启动这些工具。运行工具的结果显示在以下部分中：

HP Operations Manager for Windows

“工具状态 (Tool Status)”窗口中的“工具输出 (Tool Output)”部分

HP Operations Manager on UNIX/Linux

Java GUI(HP Operations Manager for UNIX 操作 UI)中的“应用程序输出”窗口

可以从 HP Operations Manager 控制台使用以下工具：

工具	描述
启动代理程序	在受管节点上启动 HPE Operations Agent。
停止代理程序	在受管节点上停止 HPE Operations Agent。
重新启动代理程序	在受管节点上重新启动 HPE Operations Agent。
查看状态	查看受管节点上 HPE Operations Agent 进程、服务和守护进程的状态。

查看版本信息	查看受管节点上 HPE Operations Agent 的版本。
刷新警报服务	刷新性能收集组件的警报服务。
扫描性能组件的日志文件	扫描节点上的 scope 收集器使用的日志文件。
检查性能组件的参数文件语法	帮助您检查受管节点中参数文件的语法。
检查性能组件的 Alarmdef 文件语法	帮助您检查受管节点中 alarmdef 文件的语法。
查看策略部署后操作的状态	帮助您检查节点上 parm 或 alarmdef 策略的部署状态。在启动此工具时，确保将 parm 或 alarmdef(根据需要)指定为工具参数。 可以在使用 HP Operations Manager for Windows 时，在“编辑参数 (Edit Parameters)”窗口中的“参数”框中设置工具参数。 使用 HP Operations Manager on UNIX/Linux 时，打开该工具的“编辑工具状态 (Edit Tool Status)”页面，转到“OVO 工具 (OVO Tool)”选项卡，然后在“参数”框中指定工具参数
设置 Realtime 永久许可证	设置 HP Ops OS Inst to Realtime Inst LTU 的永久许可证。
设置 Glance 永久许可证	设置 Glance Software LTU 的永久许可证。
获取许可证状态	显示节点上 LTU 的状态。

配置 SNMP 陷阱拦截器

默认情况下，SNMP 陷阱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，然后可根据配置生成相应的事件。

备注: SNMP 陷阱拦截器 (`opctrapi`) 未设置 MIB 文本部分的格式。例如，消息文本将 MIB 显示为 `.1.3.6.1.4` 而非 `iso.identified-organization.dod.internet.private`。

可通过配置以下属性修改 SNMP 陷阱拦截器的默认行为：

- SNMP_TRAP_PORT：默认端口是 **162**。可以将此值修改为 HPE 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将所有可用 SNMP 陷阱转发至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 SNMP 陷阱拦截器将 HPE 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和要将 SNMP 陷阱转发至的目标计算机的公共字符串。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 SNMP 陷阱拦截器将转发 OID 以 1.2.3 开头的所有 SNMP 陷阱。默认情况下，允许 SNMP 陷阱拦截器转发陷阱时，会转发所有可用陷阱。

  备注: 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 SNMP 陷阱拦截器的默认行为，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 <port_number>
     ```

     必须为 <port_number> 指定整数值。确保指定的 <port_number> 可用。
   - 要允许 SNMP 陷阱拦截器将 SNMP 陷阱转发到远程计算机，请运行以下命令：

     ```
     ovconfchg -ns eaagt setSNMP_TRAP_FORWARD_ENABLE TRUE
     ```
   - 如果允许 SNMP 陷阱拦截器将 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
     "<machinename>:<port>"
     ```

     <machinename> 是要将 SNMP 陷阱转发至的计算机的完全限定域名，<port> 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。
   - 如果要只将节点上选定的 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 Filter>"
     ```

<OID Filter> 是使用通配符的 OID。SNMP 陷阱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 SNMPv3 陷阱的 SNMP 陷阱拦截器

在网络环境中，请务必确保陷阱发送设备与 HPE Operations Agent 之间的安全通信。简单网络管理协议 v3 (SNMPv3) 通过对用户进行身份验证并对通过网络发送的数据包进行加密，实现发送陷阱的设备的安全访问。

在早期版本的 HPE Operations Agent 中，**opctrapi** 进程配置为拦截 SNMPv1 和 SNMPv2 陷阱。在 HPE Operations Agent 12.01 中，**opctrapi** 还可以拦截 SNMPv3 陷阱和通知消息。

配置以下变量以允许 **opctrapi** 拦截 SNMPv3 陷阱：

备注: opctrapi 进程仅在 NETSNMP 模式中拦截 SNMPv3 陷阱。

- SNMP_V3_ENABLE - 此变量是必需的，必须在 `eaagt` 命名空间中进行设置。默认情况下，此变量设置为 TRUE。

  要禁止 opctrapi 拦截 SNMPV3 陷阱，请将 SNMP_V3_ENABLE 设置为 FALSE。如果将变量设置为 FALSE，**opctrapi** 将仅拦截 SNMPv1 和 SNMPv2 陷阱。
- SNMP_V3_USERSDATA - 此变量是必需的，必须在 `eaagt` 命名空间中进行设置。使用此变量可配置用户。

  SNMP_V3_USERSDATA = `<Parameters for User 1>;<Parameters for User 2>;<Parameters for User n>`

  在此实例中，`<Parameters for User>` 包含以下内容：

  `<user name>:<authentication method>:<authentication password>:<encryption method>:<encryption key>:<Engine ID>`

参数	描述
`<user name>`	指定管理员创建的 SNMPv3 用户名。
`<authentication method>`	指定用于加密密码的协议。 您可以使用消息摘要算法 5 (MD5) 或安全哈希算法 (SHA) 协议加密密码。
`<authentication password>`	指定使用 **opcpwcrpt** 实用程序加密的密码。 **备注:** 有关详细信息，请参见使用 opcpwcrpt 实用程序加密密码。
`<encryption method>`	指定用于加密协议数据单元 (PDU) 的协议。 您可以使用数据加密标准 (DES) 或高级加密标准

	(AES) 协议加密 PDU。
<encryption key>	指定 DES 和 AES 用于加密 PDU 的密钥。 **备注:** 管理员创建的加密密钥使用 **opcpwcrpt** 实用程序进行加密。 有关详细信息，请参见使用 opcpwcrpt 实用程序加密密码。
<Engine ID>	指定用于唯一标识 SNMP 实体的 SNMPEngineID。

配置 opctrapi 以拦截 SNMPv3 陷阱

可以使用以下某种方法配置 opctrapi 以拦截 SNMPv3 陷阱：

- 使用节点信息策略
- 使用 XPL 变量

使用节点信息策略

执行以下步骤：

1. 登录到 HPOM 服务器。
2. 将以下变量添加到节点信息策略：
   a. 要启用 SNMPv3，请将 SNMP_V3_ENABLE 变量设置为 True
      `SNMP_V3_ENABLE = TRUE`
   b. 要配置用户，请运行以下命令：
      `SNMP_V3_USERSDATA =` <Parameters for User 1>;<Parameters for User 2>;<Parameters for User n>
3. 在节点上部署节点信息策略。opctrapi 自动重新启动。

下面是节点信息策略中用于配置 **opctrapi** 以拦截 SNMPv3 陷阱的代码段示例：

```
;XPL config
[eaagt]
SNMP_V3_ENABLE=TRUE
SNMP_V3_USERSDATA=<Parameters for User 1>;<Parameters for User
2>;<Parameters for User n>
```

备注: 使用节点信息策略类型，可以修改运行 HPE Operations Agent 的受管节点上的配置信息。

使用 XPL 变量

执行以下步骤：

1. 登录到 HPE Operations Agent 节点。
2. 要启用 SNMPv3 陷阱拦截，请运行以下命令：

   ```
   ovconfchg -ns eaagt -set SNMP_V3_ENABLE TRUE
   ```
3. 要设置用户，请运行以下命令：

   ```
   ovconfchg -ns eaagt -set SNMP_V3_USERSDATA <Parameters for User 1>; <Parameters for User 2>;<Parameters for User n>
   ```

   opctrapi 自动重新启动。

例如：

```
ovconfchg -ns eaagt -set SNMP_V3_ENABLE TRUE
ovconfchg -ns eaagt -set SNMP_V3_USERSDATA "snmpv3User1:SHA:7*=*C61Ntd=@*Eb#@*E0##D:DES:7*=*C61Ntd=@*Eb#@*E0##D:0x8000000001020304;snmpv3User2:SHA:8*=*D61Ntd=@*Eb#@*E0##D:DES:8*=*D61Ntd=@*Eb#@*E0##D:0x8000000001030404;"
```

在此实例中：

参数	值	值
`<user name>`	snmpv3User1	snmpv3User2
`<authentication method>`	SHA	SHA
`<authentication password>`	7*=*C61Ntd=@*Eb#@*E0##D	8*=*D61Ntd=@*Eb#@*E0##D
`<encryption method>`	DES	DES
`<encryption key>`	7*=*C61Ntd=@*Eb#@*E0##D	8*=*D61Ntd=@*Eb#@*E0##D
`<Engine ID>`	0x8000000001020304	0x8000000001030404

注：各参数用冒号 (:) 分隔，各用户用分号 (;) 分隔。

使用 **opcpwcrpt** 实用程序加密密码

1. 以具有管理员特权的身份登录到 HPOM 服务器。
2. 打开命令提示符。
3. 转到以下位置：

在 Windows 上：

```
"%OvBinDir%\OpC\install"
```

在 Linux 上：

```
/opt/OV/bin/OpC/install
```

4. 运行以下命令对密码进行加密：

```
# opcpwcrpt <your password>
```

输出字符串是加密的密码。使用加密的密码正确替换 `<authentication password>` 或 `<encryption key>`。

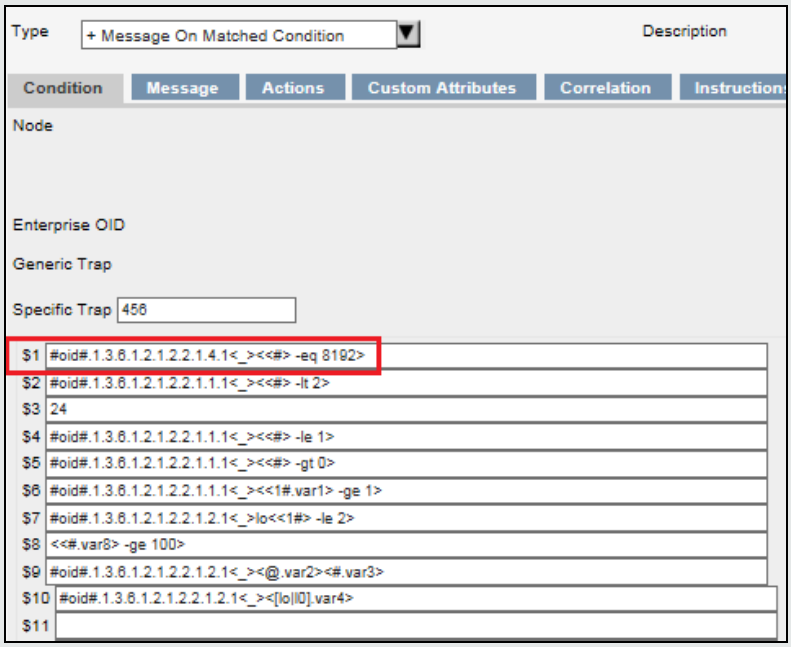
增强 SNMP 陷阱拦截器以基于 Varbind 的对象 ID 拦截陷阱

SNMP 陷阱拦截器 **opctrapi** 增强为基于 varbind 的对象 ID (OID) 以及位置解析 SNMP 陷阱。

要启用 **opctrapi** 以基于对象 ID 拦截 SNMP 陷阱，可以创建或修改 SNMP 陷阱拦截器策略并添加(或修改)如以下示例所示的内容：

```
<keyword><object id><separator><pattern string>
```

例如：



在屏幕捕获中标记的实例中：

`<keyword>` 始终为“#oid#”

`<object id>` 为 .1.3.6.1.2.1.2.2.1.4.1

<separator> 为 <_>

<pattern string> 为 <<#> -eq 8192>

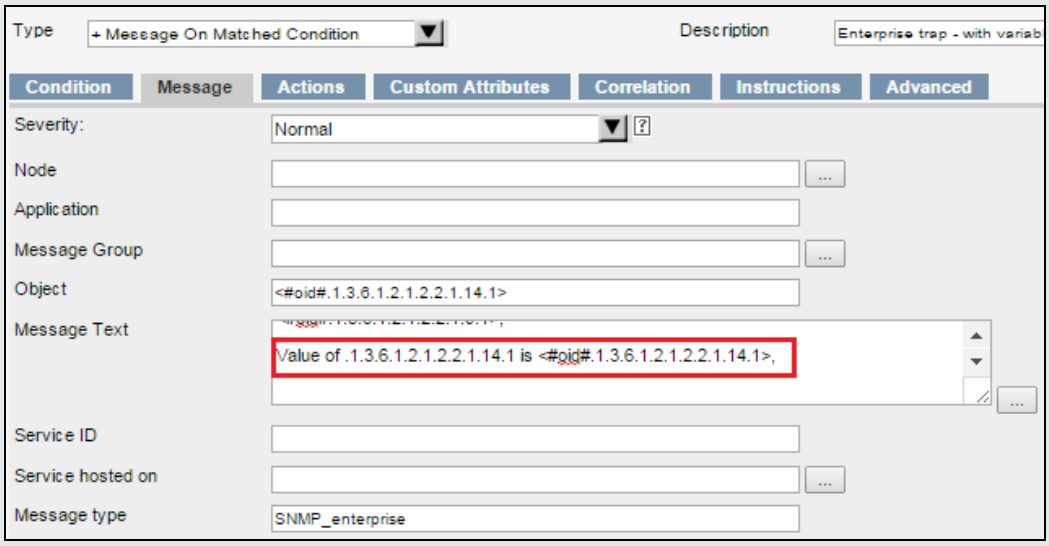
值为 8192

匹配 varbind 时，**opctrapi** 会检查字段是否以关键字 **#oid#** 开头。如果字段以关键字 **#oid#** 开头，陷阱中的 varbind OID 将与字段中提到的模式字符串进行比较，并且条件已经过验证。

备注: 在 SNMP 陷阱拦截器策略中，也可以将 OID 用于：

- 消息属性(消息文本、消息组、服务名称、消息对象、消息关键字、对象、应用程序)
- 自定义消息属性 (CMA)

在以下示例中，OID 用于消息文本：



允许 opctrapi 使用 NETSNMP 日志记录

opctrapi 进程可将 SNMPv3 陷阱的跟踪消息记录在 <%ovdatadir%>/tmp/opc/trace 文件中。这些消息未提供足够的信息，无法解决有关与传入 SNMPv3 陷阱关联的安全参数的问题。

要解决此问题，**opctrapi** 进程已增强为使用 NETSNMP 日志记录。必须在 eaagt 命名空间中设置以下跟踪变量以允许 **opctrapi** 使用 NETSNMP 日志记录：

- OPC_TRACE - 默认情况下，此变量设置为 FALSE。将此变量设置为 TRUE 可启用跟踪。

- OPC_TRC_PROCS - 此变量可列出必须跟踪的所有进程的名称。进程名称必须用逗号分隔。
- OPC_DBG_PROCS - 此变量可列出必须调试的所有被跟踪进程的名称。进程名称必须用逗号分隔。

例如：

OPC_TRACE=TRUE

OPC_TRC_PROCS=opctrapi

OPC_DBG_PROCS=opctrapi

设置这些变量后；将在 <%datadir%>/tmp/OpC 中创建新的日志文件 netsnmp.log。netsnmp.log 文件包含可用于解决问题的有关错误和传入陷阱的全面信息。

例如：

可以在 netsnmp.log 文件中检查以下跟踪条件：

- 未知用户错误 - 如果未配置 SNMPv3 用户且已收到 SNMPv3 陷阱，则会记录此错误。
- 身份验证失败错误 - 如果 SNMPv3 密码或身份验证方法不正确，则会记录此错误。
- 解密错误 - 如果 SNMPv3 加密方法或密钥不正确，则会记录此错误。

备注: netsnmp.log 文件未定期滚动更新。即使重新启用跟踪，跟踪消息也会添加到相同的文件。

将 HPE Operations Agent 与 NNMi 集成

HP Network Node Manager i Software (NNMi) 是网络管理软件，它针对网络元素发现和状态使用现有的 SNMP 代理程序。与 HPE Operations Agent 集成可帮助您监视 NNMi 转发的陷阱，以及查看 HPOM 控制台上丰富的 SNMP 陷阱。

HPE Operations Agent 与 NNMi 集成在以下情况下可用：

- Northbound 接口 - NNMi Northbound 接口用于将 NNMi 事件转发到可接收 SNMPv2c 陷阱的任何应用程序。SNMP 事件使用 Northbound 接口从 NNMi 发送到 SNMP 陷阱拦截器 **(opctrapi)**。要确保陷阱发送设备与接收应用程序中的事件正确关联，必须使用 varbind 自定义这些陷阱的规则。通过使用包含 **opctrapi** 规则以拦截陷阱(由 NNMi Northbound 接口转发)的策略完成集成。**opctrapi** 还基于严重性、节点名称等在策略中定义的各种配置设置传出消息的各种参数。有关使用 Northbound 接口的详细信息，请参见《NNMi 部署参考》

(版本 9.22)。

- 事件转发 - NNMi SNMP 陷阱转发机制在将每个 SNMP 陷阱转发到陷阱目的地之前先扩展陷阱。有两种事件转发机制：
  - 原始陷阱转发 - 不添加任何其他 varbind。
  - 默认陷阱转发 - NNMi 添加 varbind 识别原始源对象。**opctrapi** 使用此机制增强 SNMP 陷阱。

有关默认陷阱转发的详细信息，请参见 NNMi 联机帮助中的“配置陷阱转发”。

NNMi 通过 NNMi 事件转发或 Northbound 接口转发事件，并将 varbind 添加到原始事件。这些 varbind 包含其他陷阱信息。要使用 HPE Operations Agent 处理和使用此信息，必须将 SNMP 拦截器 **(opctrapi)** 配置为具有以下增强功能：

- 将 SNMP 陷阱可用的严重性关联到 HPOM 消息。这仅适用于 Northbound 接口。有关详细信息，请参见与 NNMi Northbound 接口集成以关联消息严重性。
- 从 NNMi 事件创建 HPOM 自定义消息属性 (CMA) - 这仅适用于 Northbound 接口。有关详细信息，请参见集成 NNMi Northbound 接口以在 HPOM 控制台上添加 CMA。
- 获取生成 SNMP 陷阱的源节点的名称。这仅适用于默认陷阱转发。有关详细信息，请参见与 NNMi 陷阱转发接口集成以分配 SNMP 陷阱的源节点。

与 NNMi Northbound 接口集成以关联消息严重性

HPE Operations Agent 使用由 NNMi 工具 `nnmopcexport.ovpl` 生成的策略。此工具将设置 SNMP 陷阱的严重性。如果 NNMi 工具生成的严重性为“正常”、“严重”、“警告”、“重大”或“轻微”，则消息严重性将显示为“正常”、“严重”、“警告”、“重大”或“轻微”。

仅当该工具指示 HPOM 策略生成的消息严重性为**未知**时，才将 **opctrapi** 配置为使用陷阱中可用的 varbind 以获取严重性值。此配置有助于将通过 SNMP 陷阱获取的严重性与 HPOM 消息严重性相关联。

将 SNMP 陷阱拦截器配置为增强消息严重性

要基于 SNMP 陷阱中可用的严重级别设置消息严重性，可以配置 SNMP 陷阱拦截器 **(opctrapi)**。从 SNMP 陷阱的特定 OID 读取严重性，然后将相同严重级别与 HPOM 消息关联。此配置步骤可帮助 opctrapi 使用 SNMP 陷阱中可用的严重级别。

备注: 该配置适用于 HPOM 中设置的消息严重性为“未知”的规则。

可通过配置以下属性修改 **opctrapi** 的默认行为：

- OPC_SNMP_SET_SEVERITY：默认情况下，此属性设置为 **FALSE**。通过将此属性设置为 **TRUE**，可以启用 SNMP 陷阱拦截器以读取具有特定 varbind OID **(.1.3.6.1.4.1.11.2.17.19.2.2.12)** 的 SNMP 陷阱并设置消息的严重性。如果此默认 OID 值在 SNMP 陷阱中不可用，则消息严重性将保持为“未知”。
- OPC_SNMP_OVERRIDE_SEVERITY_OID：可以设置新的 OID 值。新值将指定消息的严重性。

要配置 SNMP 拦截器，请执行以下步骤：

1. 以具有必要特权的身份登录到节点。
2. 在命令提示符中，
   - 要启用 SNMP 拦截器以读取 SNMP 陷阱中定义的严重级别，请运以下命令：

     ```
     ovconfchg -ns eaagt.integration.nnm -set OPC_SNMP_SET_SEVERITY
     TRUE
     ```

   - 要设置新的 OID 值，请运行以下命令：

     ```
     ovconfchg -ns eaagt.integration.nnm -set OPC_SNMP_OVERRIDE_
     SEVERITY_OID<OID>
     ```

     在上述命令中，<OID> 为对象标识符。此处，OID 用于派生 SNMP 陷阱的严重级别。

     默认情况下，严重级别源自默认 OID **“.1.3.6.1.4.1.11.2.17.19.2.2.12”**。可以通过运行以下命令更改默认 OID：

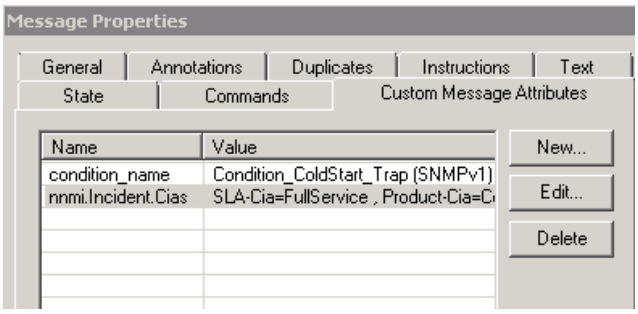
     ```
     ovconfchg -ns eaagt.integration.nnm -set OPC_SNMP_OVERRIDE_
     SEVERITY_OID<.1.3.6.1.4.1.11.2.17.19.2.2.22>
     ```

     此后，消息的严重性将基于新的 OID 值 <.1.3.6.1.4.1.11.2.17.19.2.2.22>。

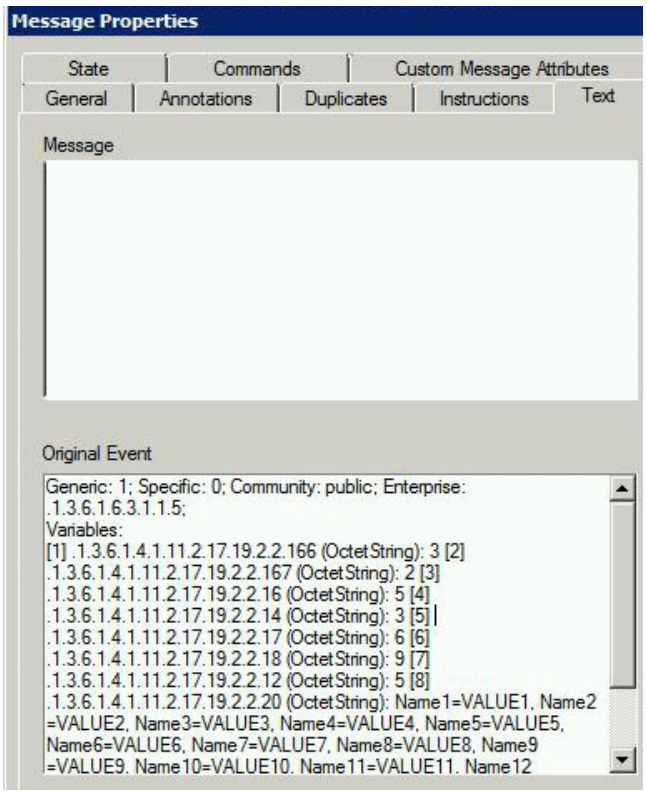
集成 NNMi Northbound 接口以在 HPOM 控制台上添加 CMA

在 NNMi Northbound 集成中转发 SNMP 陷阱时，此陷阱包含表示自定义突发事件属性 (CIA) 的 varbind。如果在 HPOM 策略中配置了 CMA[1]，NNMi CIA 将显示在 HPOM 控制台上的单个自定义消息属性 (CMA) 中。列表显示了 CMA 名称和值对。这是默认行为。CMA 选项卡中的消息将显示为：

[1]自定义消息属性 (CMA) 可以是对您有意义的任何信息，而且您可以将多个 CMA 附加到一条消息中。



如果未在 HPOM 策略中配置 CMA，NNMi CIA 消息将显示为文本消息：



配置 **opctrapi** 后，SNMP 陷阱拦截器应可以访问 CIA 并将每个 CIA 作为 OM 消息中的自定义消息属性 (CMA)

将 SNMP 陷阱拦截器配置为从 HPOM 控制台上的 NNMi CIA 创建 CMA

可以通过配置以下属性在 CMA 属性选项卡中配置 opctrapi 并将 NNMi CIA 显示为 CMA 名称和值对：

- OPC_SPLIT_NNM_CUSTOM_ATTR：默认情况下，此属性设置为 **FALSE**。通过将此属性设置为 **TRUE**，varbind **(.1.3.6.1.4.1.11.2.17.19.2.2.20)** 中的所有 NNMi CIA 值将显示为单个 CMA。

- OPC_SPLIT_NNM_CUSTOM_ATTR_MAX：此变量是**可选的**。仅当 OPC_SPLIT_NNM_CUSTOM_ATTR 变量设置为 TRUE 时，才会启用此属性。该变量定义了可由 HPOM 消息读取和解析的 NNMi 自定义属性数量。默认情况下，该值设置为 20。这表示只有 20 个 CMA 会单独显示相应的值，而剩余的 NNMi CIA 会显示在单个 CMA 中。 可以根据需要指定值。

执行以下步骤来修改 **opctrapi** 的默认行为，以便从 NNMi CIA 创建单独的 CMA：

1. 以具有必要特权的身份登录到节点。
2. 在命令提示符中，
   - 要启用 **opctrapi** 以从 varbind (**.1.3.6.1.4.1.11.2.17.19.2.2.20**) 中显示的列表读取并创建单独的 CMA，请运行以下命令：

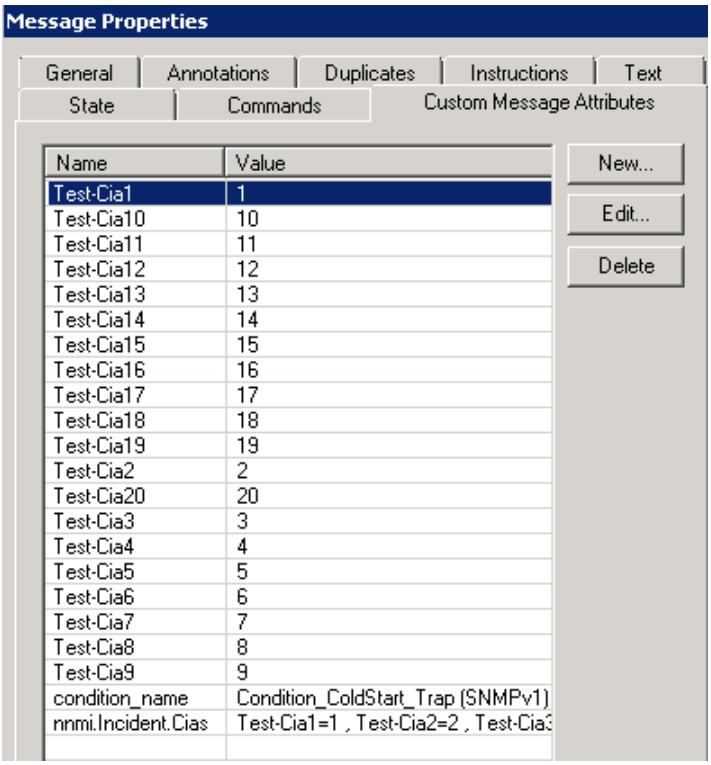
     ```
     ovconfchg -ns eaagt.integration.nnm -set OPC_SPLIT_NNM_CUSTOM_ATTR TRUE
     ```

   - 要设置该变量的值以读取消息并创建单独的 CMA 名称和值对，请运行以下命令：

     ```
     ovconfchg -ns eaagt.integration.nnm -set OPC_SPLIT_NNM_CUSTOM_ATTR_MAX<Value>
     ```

     在上述命令中，<Value> 为整数。默认情况下，该值为 20。

     配置后，CMA 选项卡中的消息将显示名称和值属性。此外，NNMi CIA 的未截断消息 `nnmi.Incident.Cias` 也会显示在列表中。



与 NNMi 陷阱转发接口集成以分配 SNMP 陷阱的源节点

在以前的版本中，只要 NNMi 将 SNMP v2c 陷阱转发到节点(Operations Agent 位于此节点上)，HPOM 消息中的源节点名称就会显示为 NNMi 的节点名称。

HPE Operations Agent 与 NNMi 转发接口相集成。NNMi 转发接口会添加 varbind 以标识生成陷阱的源节点。

可以配置 **opctrapi** 以获取生成陷阱的源节点。生成 HPOM 消息后，SNMP 陷阱拦截器 (**opctrapi**) 应将源节点名称设置为生成陷阱的节点名称，而非 NNMi 的节点名称。

配置 SNMP 陷阱拦截器以获取源节点名称

可以配置以下属性：

OPC_NODENAME_FROM_NNM_FRWD_TRAP：

默认情况下，此属性设置为 **FALSE**。通过将此属性设置为 **TRUE**，可以为 varbind“.1.3.6.1.4.1.11.2.17.2.19.1.1.2.0”和 .1.3.6.1.4.1.11.2.17.2.19.1.1.3.0 搜索所有陷阱。**Opctrapi** 使用以下 varbind：

- (.1.3.6.1.4.1.11.2.17.2.19.1.1.2.0) - 用于标识 IP 地址类型。
- (.1.3.6.1.4.1.11.2.17.2.19.1.1.3.0) - 用于获取 IP 地址以将节点设置为源节点。

要将 opctrapi 配置为在 NNMi 转发 SNMPv2 事件时分配源节点名称，请执行以下步骤：

1. 以具有必要特权的身份登录到节点。
2. 在命令提示符中，运行以下命令使 opctrapi 能够读取 varbind 并分配生成陷阱的节点：

   ```
   ovconfchg -ns eaagt.integration.nnm -set OPC_NODENAME_FROM_NNM_
   FRWD_TRAP TRUE
   ```

   备注: 如果未设置变量或设置为 False，则 **opctrapi** 将不会获取生成陷阱的源(节点名称)。

配置消息风暴抑制

当在较短时间间隔内有超乎寻常的大量新消息到达管理服务器并涌向活动消息浏览器时，会出现消息风暴。此现象会造成管理服务器中断。

HPE Operations Agent 可以在受管节点上检测和抑制消息风暴。配置以下变量：

- OPC_MSG_STORM_DETECTION - 此变量是**必需的**。默认情况下，此属性设置为 **FALSE**。将此属性设置为 **TRUE**，可以启用消息风暴检测。

- OPC_MSG_STORM_DETECTION_CATEGORY - 此变量是**必需的**。仅当 OPC_MSG_STORM_DETECTION 变量设置为 TRUE 时，才启用此属性。您可以为 POLICY、MSGGROUP、APPLICATION、OBJECT 或 SEVERITY 等任何一个消息属性设置此变量。
- OPC_MSG_STORM_RATE - 此变量是**必需的**。此变量定义以下参数：
  - 阈值 - 定义传入消息的限制。当传入的消息数超过定义的限制时，将检测到消息风暴条件。
  - 时间 - 对传入消息计数以检测消息风暴条件的时间间隔。
  - 重置 - 定义消息数低于定义的值时的限制。此参数用于检测解析消息风暴条件的时间。
- OPC_SEND_INTERNAL_MSG_ON_MSGSTORM - 此变量是**可选的**。定义是发送还是停止内部消息。默认情况下，此值设置为 TRUE。
- OPC_SUPPRESS_MSG_ON_MSG_STORM - 此变量是**可选的**。定义是发送还是抑制消息。默认的行为是，如果满足阈值条件，将检测到消息风暴条件，阈值之外的所有消息将受到抑制。默认情况下，此值为 TRUE。
- OPC_MSG_STORM_TRACE_SUPPRESSED_MSGS - 此变量是**可选的**。定义是否仅在 <OPC_SUPPRESS_MSG_ON_MSG_STORM> 设置为 TRUE 时将消息记录到日志文件中。默认情况下，此值为 FALSE。

当消息代理程序检测到消息风暴时或当消息风暴得到解析时，消息将记录到日志文件 (System.txt) 中：

在 Windows 上：

```
%OvDataDir%\log
```

在 HP-UX/Linux/Solaris 上：

```
/var/opt/OV/log
```

在 `eaagt.msgstorm` 命名空间可以使用这些参数。

在受管节点上检测消息风暴的优势在于：

- 无需任何 ECS 电路
- 消息风暴在源节点上识别
- 配置步骤简单，可以对各种消息属性进行配置。

备注: 在受管节点上检测和抑制消息风暴的示例。

通过将 OPC_MSG_STORM_DETECTION 参数设置为 TRUE 可以检测风暴条件。

启用消息风暴条件后，将参数 -

OPC_MSG_STORM_DETECTION_CATEGORY 定义为 POLICY。部署的策略有 Opclepolicy 和 Opcmsgipolicy。

要设置 OPC_MSG_STORM_RATE 参数，您可以计算传入消息率以设置参数。请参见配置消息风暴抑制 (第 43 页)。可以根据消息率设置值。

将 OPC_MSG_STORM_RATE 参数值设置为“阈值”为 100，“时间”为 20，“重置值”为 50。

Opclepolicy 发送 50 条消息，Opcmsgipolicy 发送 101 条消息。因为 Opclepolicy 发送的 101 条消息超过阈值 (100)，所以检测到风暴。

当参数 OPC_SEND_INTERNAL_MSG_ON_MSGSTORM 检测并解析消息风暴时，系统会向您发送通知。默认情况下，此值设置为 TRUE。

默认情况下，超过 100 的消息计数将受到抑制或被忽略。可通过将参数 (OPC_SUPPRESS_MSG_ON_MSG_STORM) 设置为 FALSE 来获取受到抑制的消息。

可通过设置参数 (OPC_MSG_STORM_TRACE_SUPPRESSED_MSGS) 在日志文件中记录受抑制的消息。

配置消息风暴检测和抑制

备注: 只要更改了任何配置参数，就请确保重新启动消息代理程序。如果消息代理程序重新启动，则与消息风暴检测关联的内容将重置。

要配置消息风暴检测或抑制，请执行以下步骤：

1. 以具有必要特权的身份登录到节点，也可以远程配置。
2. 要启用消息风暴检测，请运行以下命令：

   ```
   ovconfchg -ns eaagt.msgstorm -set OPC_MSG_STORM_DETECTION TRUE
   ```

   默认值设置为 FALSE。
3. 要设置消息的类别，请运行以下命令：

   ```
   ovconfchg -ns eaagt.msgstorm -set OPC_MSG_STORM_DETECTION_
   CATEGORY<CATEGORY>
   ```

   备注: 这是必需的变量。

   类别必须定义为以下值之一：
   - POLICY
   - GROUP
   - APPLICATION

- OBJECT
- SEVERITY

备注: 只能定义其中一个可用选项。不可使用各种值的组合。

4. 运行以下命令以设置消息速率变量：

   ovconfchg -ns eaagt.msgstorm -set OPC_MSG_STORM_RATE<ThresholdValue:Time:ResetValue>

   备注: 这些变量是必需的。

   这些变量的定义方式如下：

   - ThresholdValue - 设置一个数字值。如果设置的时间间隔内的消息数超过此阈值，则此条件称为消息风暴条件。在此条件下，消息代理程序将抑制消息，直至满足重置条件。检查此阈值，确定设置的类别中所有可用组的消息风暴状态。可以检查消息速率来设置此参数。请参见配置消息风暴检测和抑制 (第 45 页)

     示例：类别设置为 SEVERITY。在此消息严重性类别中，存在五种组，例如：严重、重大、轻微、警告和正常。消息按严重性分组并根据配置的阈值进行检查。

     将 ThresholdValue 设置为 100。收到的“严重”、“重大”、“轻微”、“警告”和“正常”严重性的消息数分别为 80、60、40、50 和 110。在此仅检测到正常状态的消息风暴，因为消息数 (110) 大于设置的阈值 (100)。剩余 10 个正常严重性的消息将被抑制。

   - Time - 设置时间间隔(以秒为单位)。记录传入消息计数的时间间隔。建议值低于 900 秒。
   - ResetValue - 设置一个必须小于等于阈值的数字值。如果满足以下条件，则重置消息风暴检测：
     - 按设置的时间间隔检查消息速率。
     - 消息速率应小于重置值。

     设置消息风暴条件参数的示例。

     set OPC_MSG_STORM_DETECTION=TRUE

     set OPC_MSG_STORM_DETECTION_CATEGORY=SEVERITY

     set OPC_MSG_STORM_RATE=100:60:45

     其中，ThresholdValue 为 100，Time 为 60，ResetValue 为 45。

检测消息风暴后，按设置的定期时间间隔(60 秒)检查传入消息数。在设置的时间间隔内，如果消息数小于重置值 (45)，则消息代理程序 (opcmsga) 将停止忽略消息，消息风暴状态结束。

备注: 步骤 5、6 和 7 是可选步骤。

5. 要接收和停止内部消息，请运行以下命令：

   ovconfchg -ns eaagt.msgstorm -set OPC_SEND_INTERNAL_MSG_ON_MSGSTORM<Value>

   <Value> 必须定义为以下值之一：

   - TRUE - 每当检测到消息风暴状态和每当此状态得到解决时，生成内部消息。默认值为 TRUE。
   - FALSE - 对于消息风暴状态出现和此状态得到解决，不生成内部消息。

6. 在消息风暴条件下，运行以下命令以抑制或接收消息：

   ovconfchg -ns eaagt.msgstorm -set OPC_SUPPRESS_MSG_ON_MSG_STORM<Value>

   值必须定义为以下值之一：

   - TRUE - 消息代理程序将抑制消息风暴条件下的消息。默认值为 TRUE。
   - FALSE - 消息代理程序将不抑制消息风暴条件下的消息。

7. 如果 OPC_SUPPRESS_MSG_ON_MSG_STORM 变量设置为 FALSE，**跳过此步骤**。

   在消息风暴条件下，运行以下命令以记录抑制的消息：

   ovconfchg -ns eaagt.msgstorm -set OPC_MSG_STORM_TRACE_SUPPRESSED_MSGS <Value>

   值必须定义为以下值之一：

   - TRUE - 消息代理程序将在日志文件中保存所有抑制的消息。
   - FALSE - 消息代理程序将不保存抑制的消息。默认情况下，此值为 FALSE。

   以下目录中的 (msgsuppress.log <process id of the process>) 内包含抑制的消息：

   在 Windows 上：

   %OvDataDir%\tmp\OpC

   在 HP-UX/Linux/Solaris 上：

   /var/opt/OV/tmp/OpC

8. 运行以下命令以重新启动消息代理程序：

```
ovc -restart opcmsga
```

检查消息速率

消息速率用于测量通过系统传递的消息的平均速率。检查消息平均速率后，可以配置理想的阈值限制，然后重置限制以检测并抑制消息风暴条件。只能在配置消息风暴检测后才能检查消息速率。

要检查特定时间的消息速率，请运行以下命令：

在 Windows 上：

```
%OVInstallDir%/lbin/eaagt/opcmsga -message_rate
```

在 HP-UX/Linux/Solaris 上：

```
/opt/OV/lbin/eaagt/opcmsga -message_rate
```

在 AIX 上：

```
/usr/lpp/OV/lbin/eaagt/opcmsga -message_rate
```

配置备份服务器

消息代理程序将消息发送到主服务器，即 HP Operations Manager (HPOM)。如果您的环境中有多个 HPOM 服务器，您还可以另配置一个服务器来充当环境中的备份服务器。在环境中配置备份服务器之后，如果到主服务器的通信链接关闭，无法将消息发送到主服务器，则消息代理程序会将消息发送到备份服务器。

您还可以配置备份服务器实现主服务器与备份服务器之间的消息同步的负载分布(例如，在 manager-of-manager (MoM) 场景中)。有关 MoM 场景的详细信息，请参见《HP Operations Manager for UNIX 概念指南 (HP Operations Manager for UNIX Concepts Guide)》或 HP Operations Manager 联机帮助。备份服务器可以是环境中的另一个 HPOM 服务器。可以将一个或多个服务器配置为备份服务器。

可以通过配置以下变量的值，使 HPE Operations Agent 将消息发送到备份服务器：

- OPC_BACKUP_MGRS - 可以指定一个或一列要配置为备份服务器的服务器。列表中的值可用逗号或分号分隔。
- OPC_BACKUP_MGRS_FAILOVER_ONLY - 当此变量的值设置为 **TRUE** 时，只有在主服务器关闭时，消息代理程序才会将消息转发到备份服务器。而当此值设置为 **FALSE** 时，消息发送到备份服务器，而不管主服务器的状态。此变量的默认值是 **FALSE**。

如果您已在初始化期间配置了备份服务器的列表，则消息代理程序将创建备份服务器的列表，然后检查为 OPC_BACKUP_MGRS_FAILOVER_ONLY 变量设置的值。消息到达 `msgagtdf` 文件，而且 OPC_BACKUP_MGRS_FAILOVER_ONLY 变量的值为：

- **FALSE** - 消息代理程序将消息依次发送到主服务器和备份服务器。消息成功发送到这两个服务器后，从 `msgagtdf` 文件中删除消息条目。
- **TRUE** - 消息代理程序将消息发送到主服务器。如果消息发送失败，则消息代理程序尝试将消息转发到备份服务器。如果列出了多个备份服务器且消息至少传送到了其中一个备份服务器，则从 `msgagtdf` 文件中删除消息条目。消息代理程序不会尝试将消息重新发送到其他备份服务器，主服务器再次启动时，消息代理程序也不会将消息发送到该服务器。

注：当 OPC_BACKUP_MGRS_FAILOVER_ONLY 设置为 **TRUE** 时，如果在启动本地自动操作的消息发送到主服务器后主服务器关闭，则消息代理程序会将操作响应发送到备份管理器。但备份管理器会丢弃该响应，因为备份服务器没有启动了操作的原始消息。

先决条件：

- 必须在 HPE Operations Agent 节点上安装备份服务器的可信证书。
- 必须在备份服务器上安装主服务器的可信证书。
- HPE Operations Agent 节点必须添加到备份服务器。

要设置变量的值，请执行以下步骤：

1. 以根身份或管理员身份登录节点。
2. 运行以下命令：
   - 要设置备份服务器，请运行以下命令：

     `ovconfchg -ns eaagt -set OPC_BACKUP_MGRS` <servername>

     <servername> 是备份服务器的名称。例如 abc.ind.hp.com。

     要配置备份服务器的列表，请使用以下任一命令：
     - `ovconfchg -ns eaagt -set OPC_BACKUP_MGRS` <servername1>,<servername2>,...,<servernameN>
     - `ovconfchg -ns eaagt -set OPC_BACKUP_MGRS` <servername1>;<servername2>;...;<servernameN>

       备份服务器的顺序取决于服务器名称的指定顺序。在前面的示例中，<servername1> 充当第一个备份服务器，<servername2> 充当第二个备份服务器。
   - 要修改 OPC_BACKUP_MGRS_FAILOVER_ONLY 变量的值，请运行以下命令：

```
ovconfchg -ns eaagt -set OPC_BACKUP_MGRS_FAILOVER_ONLY TRUE
```

默认值是 **FALSE**。

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 HPE 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。

检查 Perfd 进程的许可证

只有在使用 HP Ops OS Inst to Realtime Inst LTU、Glance Pak Software LTU 或 Glance Software LTU 后，才可以使用此组件。有关详细信息，请参见《HPE Operations Agent 许可证指南》。

perfd 进程仅在验证许可证已启用后才启动数据收集。如果许可证未启用，该进程处于空闲状态。所有活动记录在日志文件中。

修改设置

您可以从 **perfd.ini** 文件修改 **perfd** 进程的配置设置，该文件可从节点的以下目录中找到：

在 Windows 上：

```
%ovdatadir%
```

在 HP-UX/Linux/Solaris 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
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	20

(续)

参数	描述	默认值
	status-perfd.<port> 位于节点上的以下目录中： **在 Windows 上：** %ovdatadir% **在 HP-UX/Linux/Solaris 上：** /var/opt/perf	
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
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** 将拒绝所有连接请求。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 IPv6 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。 **备注:** 如果明确要求 **perfd** 仅接受 IPv4 连接，则在 **perfd.ini** 文件中将 IPv4 值设置为 true。	False

(续)

参数	描述	默认值
logsize	此参数将指定一个文件大小，超过此大小 **perfd** 将滚动更新其日志或跟踪文件。如果指定的大小小于 4096 字节，则忽略此参数。	1048576 字节
add	使用此参数可以指定必须收集数据的度量类的列表(以逗号分隔)。当 lightweight 参数设置为 True 时，使用此参数添加单独的度量类非常有用。	
exclude	使用此参数可以指定不得收集数据的度量类的列表(以逗号分隔)。当 lightweight 参数设置为 False 时，此参数用于排除单独的度量类。	

要更改默认设置，请执行以下步骤：

1. 在节点上，使用文本编辑器打开 **perfd.ini** 文件。
2. 修改设置。
3. 保存文件。
4. 重新启动 HPE Operations Agent 使更改生效。

通过 IPv6 连接监视 HPE Operations Agent

HPE Operations Agent 12.01 安装完成后，可确定支持的 IP 配置并与对应的 IP 地址绑定。IPv6 连接不需要其他配置。对于双堆栈节点，该节点可确定受支持服务器的 IP 地址。

服务器	双堆栈节点
IPv4	使用 IPv4
IPv6	使用 IPv6
双堆栈	使用 IPv6

备注: 如果服务器和节点都是双堆栈，则优先使用 IPv6 通信。

限制访问

可以将 HPE Operations Agent 配置为阻止其他系统使用 RTMA 组件访问本地系统的实时性能数据。

cpsh、**padv** 和 **mpadv** 实用程序可用于远程访问代理程序节点的实时性能数据。可以使用 **cpsh** 实用程序查看运行 **perfd** 进程的任何远程系统上的实时性能数据。

可以使用 **padv** 或 **mpadv** 实用程序在运行 **perfd** 进程的任何远程系统上运行 **adviser** 脚本。

这些实用程序依赖 **perfd** 进程访问实时性能数据。

要通过撤消其他系统对 perfd 进程的访问权来阻止其访问本地系统的实时性能数据，请执行以下步骤：

备注: 以下过程不会阻止 HP Performance Manager 的诊断视图访问系统中的实时性能数据。

1.以管理员或根用户身份登录到系统。转到以下目录：

在 Windows 上：

```
%OvDataDir%\
```

在 HP-UX/Linux/Solaris 上：

```
/var/opt/perf/
```

2.环境中的所有其他系统(其中 HPE Operations Agent 可用)现在无法访问此系统中的实时性能数据。

3.在此位置中创建一个空文件，然后将该文件另存为 **authip**。

4.要使系统能够访问此系统中的实时性能数据，请用文本编辑器打开 **authip** 文件，添加系统的 FQDN 或 IP 地址，然后保存该文件。可以指定多个 FQDN 或 IP 地址(条目必须用新行分隔)。

5.重新启动 **perfd** 以启用更改。

authip 文件中指定的系统现在可以访问此系统中的实时性能数据。

6.要允许访问所有系统，请删除 **authip** 文件。

配置代理程序用户

安装 HPE 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 HPE Operations Agent 配置为

以特权少于根用户或本地系统用户的非默认用户运行。

只能以非根/非本地系统用户运行操作监视组件，并以默认根/本地系统用户运行剩余组件。

基于使用的用户帐户，可以配置代理程序操作的以下模式：

- **非特权**：HPE Operations Agent 的所有组件均以特权少于根或本地系统用户的非默认用户帐户运行。

  备注: 无法在 HP-UX 上以非特权模式运行 HPE Operations Agent。

  不得在 HPE Operations Agent 和 NNMi 共存于系统中时以非特权用户模式运行 HPE Operations Agent。

- **根**：HPE Operations Agent 的所有组件均以默认用户(根或本地系统)运行。这是代理程序操作的默认模式。
- **混合**：性能收集组件以根用户运行，而操作监视组件以本地用户帐户运行。

  备注: 在混合模式下运行时，建议根或特权用户启动代理程序进程。

将代理程序配置为在安装 HP Performance Manager 的系统上以非默认用户运行时，HP Performance Manager 的 `OvTomCatB` 服务开始以非默认代理程序用户运行。

在受 HPOM 管理的环境中，也可以将代理程序配置为通过未运行该代理程序的用户执行自动命令或操作员触发的命令。

使用非默认用户的要求

要使用的非默认代理程序用户必须满足以下要求：

- **仅适用于 Windows 的要求：**
  - 用户必须对注册表项 `HKEY_LOCAL_MACHINE/Software/Hewlett-Packard/OpenView` 具有完全控制权
  - 用户必须对注册表项 `HKEY_LOCAL_MACHINE/Software/Microsoft/WindowsNT/CurrentVersion/Perflib` 具有读取访问权限
  - 用户必须具有以下权限：
    - 作为服务登录 - 启动和停止性能收集组件服务。
    - 管理审核和安全日志 - 用户可以在事件查看器的安全日志中查看已审核的事件。具有此特权的用户可以查看并清除安全日志。
    - 充当操作系统的一部分 - 控制代理程序进程
    - 替换进程级别的令牌 - 以非根用户身份创建代理程序进程。

- 要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。

  备注: 在 Windows 上，代理程序用户无法读取日志文件，因为该用户没有权限。

- 要使用自动命令、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。

使用非默认用户的限制

要使用的非默认(非根或非特权)代理程序用户具有以下限制：

- HP-UX 平台不支持非特权用户模式。
- HPOM 管理服务器不支持非特权用户模式，但 HP Operations Manager for Unix、Linux 和 Solaris 自版本 9.20 起支持混合用户模式。
- AIX WPAR 不支持非特权用户和混合模式。
- 无法收集 Xen、KVM、VMware vSphere、Hyper-V 和其他虚拟化域的 BYLS 度量数据。
- 默认情况下，具有非特权和混合用户模式的代理程序用户将没有读取受监视日志文件的权限。
- 默认情况下，具有非特权和混合用户模式的代理程序用户将没有使用自动命令、操作员触发的命令、工具或计划任务启动程序的权限。
- 如果具有非特权和混合模式的代理程序用户没有管理权限，HP Operations Smart Plug-in 可能需要额外的配置或用户权限。
- HPE Operations Agent 无法收集由非特权用户拥有的所有进程实例中以 `PROC_REGION_*` 或 `PROC_FILE_*` 开头的度量。此外，使用更高特权运行的进程(如 **ovbbccb** 和 `sshd`)将在非特权模式下可用或不可用，具体取决于操作系统。
- 在 Windows 上，对于由除代理程序用户以外的用户拥有的进程，度量 `PROC_USER_NAME` 显示为 `Unknown`。
- 在 AIX 上，将代理程序配置为使用非默认用户后，可能会在命令行控制台(或 `/var/opt/OV/log` 目录下的 `oainstall.log` 文件)中显示以下错误消息：

  ```
  Product activation failure.Refer to the log file for moredetails.
  ```

  忽略此错误。
- 在 Solaris 上，HPE Operations Agent 获取最多 80 个字符的进程详细信息，这是对 Solaris 的限制。**opcmona** 会读取 `/proc/pid/psinfo` 文件并将结果存储在结构中。如果稍后需要扩展信息，则系统将读取 `/proc/pid/as` 文件。如果代理程序以非根用户身份运行，并且没有打开 `/proc/pid/as` 文件的权限，则会将进程详细信息与 `psinfo` 中可用的限制信息进行比较。

备注: 要在安装期间将代理程序配置为以非默认用户身份运行，请参见《HPE Operations Agent 和 HPE Operations 基础结构 SPI 安装指南 (HPE Operations Agent and HPE Operations Smart Plug-ins for Infrastructure Installation Guide)》。

安装后配置代理程序用户

如果无法在安装时将 HPE Operations Agent 配置为使用非默认用户，可以在安装后使用 `oainstall` 脚本(位于代理程序节点上)的 `-configure` 选项或 `ovswitchuser` 命令完成此配置。

在 Windows 上更改默认用户

如果无法在安装时将代理程序配置为使用非默认用户运行(请参见 Configure the Agent User During Installation)，建议采用非活动模式安装 HPE Operations Agent。有关详细信息，请参见《HPE Operations Agent 和 HPE Operations 基础结构 SPI 安装指南 (HPE Operations Agent and HPE Operations Smart Plug-ins for Infrastructure Installation Guide)》。

可以使用以下方法之一将代理程序配置为使用非默认用户：

- 使用配置文件
- 使用 ovswitchuser 命令

备用方法：使用 ovswitchuser 命令

备注: 确保在开始使用 `ovswitchuser.vbs` 命令之前停止所有操作代理程序进程。

如果不想使用配置文件，请执行以下步骤：

备注: 建议配置过程使用配置文件。

1. 停止代理程序：

   在 Windows 64 位节点上：

   ```
   %ovinstalldir%bin\win64\opcagt -kill
   ```

   在其他 Windows 节点上：

   ```
   %ovinstalldir%bin\opcagt -kill
   ```

2. 运行以下命令，将代理程序配置为使用非默认用户运行：

   ```
   cscript "%ovinstalldir%bin\ovswitchuser.vbs" -
   existinguser<DOMAIN\USER>-existinggroup<GROUP>-
   passwd<PASSWORD>
   ```

在此实例中：

<DOMAIN\USER> 是域和用户名。

<GROUP> 是用户所属的组的名称，例如 AgentGroup。

<PASSWORD> 是用户密码。

备注: 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。

3. 运行以下命令对非默认用户设置必要的权限。

   `cscript %ovinstalldir%lbin\xpl\ovsetscmpermissions.vbs -user`<User_Name> `-f`

   在此实例中，<User_Name> 是非默认用户的名称。

4. 运行以下任一命令：

   在非本地系统帐户下运行此代理程序的所有组件：

   在 Windows 64 位节点上：

   `%OvInstallDir%bin\win64\ovconfchg -ns eaagt -set MODE NPU`

   在所有其他 Windows 节点上：

   `%OvInstallDir%bin\ovconfchg -ns eaagt -set MODE NPU`

   仅以非本地系统帐户运行 操作监视组件：

   在 Windows 64 位节点上：

   `%OvInstallDir%bin\win64\ovconfchg -ns eaagt -set MODE MIXED`

   在所有其他 Windows 节点上：

   `%OvInstallDir%bin\ovconfchg -ns eaagt -set MODE MIXED`

5. 运行以下命令：

   在 Windows 64 位节点上：

   `%OvInstallDir%bin\win64\ovconfchg -ns ctrl.sudo -set OV_SUDO_USER`<User_Name>

   `%OvInstallDir%bin\win64\ovconfchg -ns ctrl.sudo -set OV_SUDO_GROUP`<Group_Name>

   `%OvInstallDir%bin\win64\ovconfchg -ns eaagt -set OPC_PROC_ALWAYS_INTERACTIVE NEVER`

   在所有其他 Windows 节点上：

   `%OvInstallDir%bin\ovconfchg -ns ctrl.sudo -set OV_SUDO_USER`<User_Name>

   `%OvInstallDir%bin\ovconfchg -ns ctrl.sudo -set OV_SUDO_GROUP`<Group_Name>

```
%OvInstallDir%bin\ovconfchg -ns eaagt -set OPC_PROC_ALWAYS_
INTERACTIVE NEVER
```

在此实例中，<User_Name> 是非默认用户的名称；<Group_Name> 是该非默认用户所属的组。

6. 如果选择了非特权操纵模式，则运行以下命令：

   在 Windows 64 位节点上：

   ```
   %OvInstallDir%bin\win64\ovconfchg -ns eaagt -set OPC_RPC_ONLY TRUE
   ```

   在所有其他 Windows 节点上：

   ```
   %OvInstallDir%bin\ovconfchg -ns eaagt -set OPC_RPC_ONLY TRUE
   ```

7. 如果选择了混合操作模式，则运行以下命令：

   在 Windows 64 位节点上：

   ```
   %OvInstallDir%bin\win64\ovconfchg -ns eaagt -set NPU_TASK_SET
   EVENT_ACTION
   ```

   在所有其他 Windows 节点上：

   ```
   %OvInstallDir%bin\ovconfchg -ns eaagt -set NPU_TASK_SET EVENT_
   ACTION
   ```

8. 运行以下命令：

   备注: 这是 UNIX/Linux 节点的要求，在 Windows 节点上不强制要求此步骤。但是，建议在 Windows 节点上也完成此步骤。

   在 Windows 64 位节点上：

   ```
   %OvInstallDir%bin\win64\ovconfchg -ns eaagt -set SNMP_TRAP_PORT
   <SNMP_Port>
   %OvInstallDir%bin\win64\ovconfchg -ns bbc.cb -set SERVER_
   PORT<Comm_Port>
   ```

   在所有其他 Windows 节点上：

   ```
   %OvInstallDir%bin\ovconfchg -ns eaagt -set SNMP_TRAP_PORT <SNMP_
   Port>
   %OvInstallDir%bin\ovconfchg -ns bbc.cb -set SERVER_PORT<Comm_
   Port>
   ```

   在此实例中，

   <Comm_Port> 是您选择的通信端口号。如果将 SERVER_PORT 设置为 383，则不需要其他配置。如果未将端口值设置为 383，请确保在 HPOM 管理服务器上配置此端口。

   <SNMP_Port> 是 HPE Operations Agent 接收 SNMP 陷阱的端口。

   这些端口必须高于 1024。

9. 重新启动代理程序：

在 Windows 64 位节点上：

```
%ovinstalldir%bin\win64\opcagt -start
```

在所有其他 Windows 节点上：

```
%ovinstalldir%bin\opcagt -start
```

在 UNIX/Linux 上更改默认用户

如果无法在安装时将代理程序配置为使用非默认用户运行(请参见 Configure the Agent User During Installation)，建议采用非活动模式安装 HPE Operations Agent。有关详细信息，请参见《HPE Operations Agent 和 HPE Operations 基础结构 SPI 安装指南 (HPE Operations Agent and HPE Operations Smart Plug-ins for Infrastructure Installation Guide)》。

可以使用以下方法之一将代理程序配置为使用非默认用户：

- 使用配置文件
- 使用 ovswitchuser 命令

使用配置文件

要使用配置文件更改默认代理程序用户，请执行以下步骤：

1. 在要安装代理程序的系统上，创建一个新文件并使用文本编辑器打开该文件。
2. 输入以下语句之一，指定代理程序的操作模式：

   要以非本地系统帐户运行代理程序，请输入：

   set eaagt:MODE=NPU

   要以非本地系统帐户仅运行操作监视组件，请输入：

   set eaagt:MODE=MIXED
3. 如果已选择非特权模式(即，如果已在上一步中输入 `set eaagt:MODE=NPU`)，请输入以下语句：

   set eaagt:OPC_RPC_ONLY=TRUE
4. 如果已选择非特权模式(即，如果已在上一步中输入 `set eaagt:MODE=NPU`)，请输入以下语句：

   set eaagt:SNMP_TRAP_PORT=<SNMP_port_number>

   set bbc.cb:SERVER_PORT=<Comm_port_number>

   备注: 这是 UNIX/Linux 节点所必需的，因为 UNIX/Linux 上的非根用户无权访问低于 1024 的端口。

   在此实例中，

   <Comm_Port> 是您选择的通信端口号。

<SNMP_Port> 是 HPE Operations Agent 接收 SNMP 陷阱的端口。

这些端口必须高于 1024。

5. 如果已选择混合模式(即，如果已在步骤 2 中输入 `set eaagt:MODE=MIXED`)，请输入以下语句：

   set eaagt:NPU_TASK_SET=EVENT_ACTION

6. 输入以下语句：

   set ctrl.sudo:OV_SUDO_USER=<User_Name>

   set ctrl.sudo:OV_SUDO_GROUP=<Group_Name>

   在此实例中，<User_Name> 是非默认用户的名称；<Group_Name> 是该非默认用户所属的组。

7. 将文件保存到系统上的本地目录中。
8. 将代理程序重新配置为以配置文件中指定的用户运行：
   a. 转到节点上的以下位置：

      `/opt/OV/bin/OpC/install`

   b. 运行以下命令：

      `./oainstall.sh -a -configure -agent_profile`<path>/<profile_file>

      在此实例中，<profile_file> 是配置文件的名称；<path> 是配置文件的完整路径。

备用方法：使用 ovswitchuser 命令

备注: 使用 **ovswitchuser.sh** 命令启动之前，请确保停止所有操作代理程序进程。

如果不想使用配置文件，请执行以下步骤：

备注: 使用配置文件配置是建议的配置步骤。

1. 转到以下目录：

   在 HP-UX/Linux/Solaris 上：

   `/opt/OV/bin`

   在 AIX 上：

   `/usr/lpp/OV/bin`

2. 运行以下命令停止代理程序：

   `./opcagt -kill`

3. 运行以下命令，将代理程序配置为使用非默认用户运行：

   `./ovswitchuser.sh -existinguser`<User_Name>`-existinggroup`<Group_Name>

在此实例中：

<User_Name> 是运行代理程序的用户的名称。

<Group_Name> 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。

备注: 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. 运行以下任一命令：

   以非根用户的身份运行代理程序的所有组件：

   ```
   ./ovconfchg -ns eaagt -set MODE NPU
   ```

   使用非根用户仅运行操作监视组件：

   ```
   ./ovconfchg -ns eaagt -set MODE MIXED
   ```

5. 运行以下命令：

   备注: 这是 UNIX/Linux 节点所必需的，因为 UNIX/Linux 上的非根用户无权访问低于 1024 的端口。

   ```
   ./ovconfchg -ns eaagt -set SNMP_TRAP_PORT <SNMP_Port>
   ./ovconfchg -ns bbc.cb -set SERVER_PORT<Comm_Port>
   ```

   在此实例中，

   <Comm_Port> 是您选择的通信端口号。如果将 SERVER_PORT 设置为 383，则不需要其他配置。如果未将端口值设置为 383，请确保在 HPOM 管理服务器上配置此端口。

   <SNMP_Port> 是 HPE Operations Agent 接收 SNMP 陷阱的端口。

   这些端口必须高于 1024。

6. 运行以下命令启动代理程序：

   ```
   ./opcagt -start
   ```

将代理程序配置为以非根用户身份运行后，**System.txt** 文件中可能显示以下错误消息：

```
ovbbccb (22461/1):(bbc-188) Cannot change the root directory for the
current process.
```

忽略此错误。

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 HPE Operations Agent 配置为以其他用户帐户启动命令。可以通过设置节点上的 eaagt 命名空间中的 OVO_STD_USER 参数来完成此配置。可使用以下方式配置此参数：

- 在 HPE 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 <user>/|<encrypted password> 的格式指定 OVO_STD_USER 的值

  用用户名替换 <user>。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。

  用命令 **opcpwcrpt** <password> 的输出替换 <encrypted password>。可在管理服务器上在命令提示符处启动该命令。

  在 Windows 上：

  作为 $OVO_STD_USER 运行

  在 HP-UX/Linux/Solaris 上：

  $OVO_STD_USER 后跟操作

  备注: OVO_STD_USER 变量当前仅对 Windows 平台可用。

当您配置或启动工具时，也可以使用 OVO_STD_USER。指定用户名 $OVO_STD_USER，将密码留空。

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。

备注: 如果代理程序无法以 OVO_STD_USER 启动命令或工具，代理程序可以使用代理程序当前运行所用的用户帐户启动命令或工具。例如，如果您指定了错误的用户名或密码，则可能发生这种情况。

配置安全组件的非对称密钥

RSA 证书和非对称加密用于在节点之间以及节点和管理服务器之间 SSL 握手时进行安全通信。

安装 HPE Operations Agent 时：

1. 证书客户端组件 (OvSecCc) 会将配置变量 ASYMMETRIC_KEY_LENGTH 设置为 2048(默认值)
2. 证书客户端组件 (OvSecCm) 将创建用于通信的 2048 位 RSA 密钥对。此 RSA

密钥对用于安全通信。

3. 创建的 CA 证书带有 <CA_ovcoreid_keylength> 别名。

   示例：CA_8cd78962-ab51-755c-1279-85f5ba286e97_2048

   2048 是默认密钥长度。

要提高加密强度，必须增加 RSA 密钥的长度。使用 ASYMMETRIC_KEY_LENGTH 配置变量在 sec.cm 命名空间中设置 RSA 密钥的长度。

按照以下步骤设置 ASYMMETRIC_KEY_LENGTH 配置变量并在管理服务器和受管节点上应用配置更改：

在管理服务器上

执行以下步骤在管理服务器上更新配置变量：

1. 使用以下命令更新配置变量 ASYMMETRIC_KEY_LENGTH：

   ```
   ovconfchg –ns sec.cm -set ASYMMETRIC_KEY_LENGTH <RSA Encryption algorithm supported key length>
   ```

2. 转到管理服务器上的以下位置：

   在 Windows 上：

   ```
   %ovinstalldir%\lbin\seccs\install\
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   %ovinstalldir%/lbin/seccs/install/
   ```

3. 运行 MigrateAsymKey 工具：

   在 Windows 上：

   ```
   cscript MigrateAsymKey.vbs -createCAcert
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   ./MigrateAsymKey.sh -createCAcert
   ```

   MigrateAsymKey：

   - 创建新的 CA 密钥对，
   - 根据在配置变量 ASYMMETRIC_KEY_LENGTH 中设置的新密钥长度值创建新 CA 证书。
   - 在管理服务器上更新本地代理程序和所有其他 OV 资源组的受信任证书，
   - 为本地代理程序和所有 OV 资源组创建新证书。

   备注: 在升级证书服务器组件过程中，带有 -createCAcert 选项的迁移工具将在安装之后运行。-createCAcert 用于创建与设置为变量 ASYMMETRIC_

KEY_LENGTH 的新配置值对应的 CA 证书，并更新针对服务器和所有 OV 资源组上的代理程序创建的 CA 证书。因此，不要运行带有 -createCAcert 选项的迁移工具，除非 ASYMMETRIC_KEY_LENGTH 配置变量进行了任何配置变更。

4. 要在所有受管节点上更新受信任证书，请运行以下命令：

   ```
   ovcert –updatetrusted
   ```

   备注: 在运行带有 -createNodecert 选项的 MigrateAsymKey 工具之前，必须先在所有受管节点上运行 ovcert –updatetrusted 命令。

5. 要为管理服务器上的代理程序和服务器上的所有 OV 资源组创建新节点证书，请运行以下命令：

   在 Windows 上：

   ```
   cscript MigrateAsymKey.vbs -createNodecert
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   ./MigrateAsymKey.sh -createNodecert
   ```

   此命令将为管理服务器上的 Operations Agent 和带有新 RSA 密钥对的所有 OV 资源组创建新节点证书。

备注: 可通过运行以下命令验证更新的密钥长度：`ovcert –certinfo <certificate_alias>`。密钥长度使用 ASYMMETRIC_KEY_LENGTH 进行更新。

在代理程序节点上

要在代理程序节点(受管节点)上应用配置变更，请执行以下步骤：

1. 使用以下命令更新配置变量 ASYMMETRIC_KEY_LENGTH：

   ```
   ovconfchg –ns sec.cm -set ASYMMETRIC_KEY_LENGTH <RSA Encryption algorithm supported key length>
   ```

2. 要删除代理程序上现有的节点证书，请运行以下命令：

   ```
   ovcert –remove<certificate alias>
   ovcert –remove <CA certificate alias>
   ```

3. 要从管理服务器请求新节点证书，请运行以下命令：

   ```
   ovcert -certreq
   ```

配置安全组件的对称密钥

安装 HPE Operations Agent 时，安全组件 OvSecore 和 OvJSecCore 将继续针对加密和解密使用旧的默认算法值。从版本 11.10 开始，OvSecCore 和 OvJSecCore 将包含在适用于 Windows 的 HPSharedComponent 包中。

受支持的对称算法如下所示：

- Blowfish
- DES
- DES3
- AES128
- AES192
- AES256

要更改默认对称密钥算法，请使用以下配置变量：

- **DEF_SYM_KEY_ALGO** - 此变量用于为加密设置默认对称密钥算法。在 `sec.core` 命名空间中设置此配置变量。受支持的算法值为：
  - eBlowfish
  - eDES
  - eDES3
  - eAES128
  - eAES192
  - eAES256
  - eDefault – 将 AES128 用作默认算法。

  备注: 要启用 FIPS 兼容，请使用以下与 FIPS 兼容的对称算法之一：
  - 3DES
  - AES128
  - AES198
  - AES256

- **ENABLE_DEF_SYM_KEY_ALGO** - 将此变量设置为 TRUE 可允许使用设置为 DEF_SYM_KEY_ALGO 的默认对称密钥算法。

  备注: 如果未设置，则 MigrateSymKey 工具会将 ENABLE_DEF_SYM_KEY_ALGO 配置变量设置为 TRUE。

按照以下步骤更新运行有 HPE Operations Agent 的管理服务器和节点上的配置设置：

1. 运行以下命令将配置变量 DEF_SYM_KEY_ALGO 设置为任何支持的算法：

   ```
   ovconfchg -ns sec.core -set DEF_SYM_KEY_ALGO <Supported Algorithm>
   ```

   备注: 如果未设置 DEF_SYM_KEY_ALGO 变量且 ENABLE_DEF_SYM_KEY_ALGO 变量设置为 TRUE，则 eAES128 将用作默认算法。

2. 转到安装有代理程序的管理服务器或节点上的以下位置：

   在 Windows 上：

   ```
   %ovinstalldir%\bin\secco\
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   %ovinstalldir%/lbin/secco/
   ```

3. 运行 MigrateSymKey 工具：

   在 Windows 上：

   ```
   MigrateSymKey -sym_key_algo [eBlowfish | eDES | eDES3 | eAES128 | eAES192 | eAES256]
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   ./MigrateSymKey.sh -sym_key_algo [eBlowfish | eDES | eDES3 | eAES128 | eAES192 | eAES256]
   ```

   MigrateSymKey 工具将 ENABLE_DEF_SYM_KEY_ALGO 配置参数设置为 TRUE，并根据设置为配置变量 DEF_SYM_KEY_ALGO 的算法迁移密钥库内容。

用法：

```
MigrateSymKey
        -sym_key_algo [eBlowfish | eDES | eDES3 | eAES128 | eAES192 | eAES256]
        -help
```

`-sym_key_algo`：

- 如果未设置，则将“ENABLE_DEF_SYM_KEY_ALGO”更新为 TRUE。
- 如果已指定对称密钥算法，则使用指定值更新 DEF_SYM_KEY_ALGO。
- 如果未指定算法，则 eAES128 将用作对称密钥算法。

使用哈希算法配置安全组件

HPE Operations Agent 12.01 及更高版本支持可配置的哈希算法以执行哈希运算。**HASH_ALGO** 是提供用于设置哈希算法的配置变量。此配置变量位于 **sec.core** 命令空间下。

HASH_ALGO 支持以下哈希算法：

- eMD5
- eSHA1
- eSHA224
- eSHA256
- eSHA384
- eSHA512
- eDefault

备注: 要启用 FIPS 兼容，请使用以下与 FIPS 兼容的哈希算法之一：

- SHA1
- SHA224
- SHA256
- SHA384
- SHA512

使用以下命令将配置变量 HASH_ALGO 设置为任何支持的算法：

`ovconfchg -ns sec.core -set HASH_ALGO` <Supported Algorithm>

示例： `ovconfchg -ns sec.core -set HASH_ALGO=eSHA224`

如果 HASH_ALGO 变量设置为 eDefault(或其他任何不受支持的值)，则 sec.core 使用 eSHA256 作为 HASH_ALGO 的值。

哈希还用于保护 HPE Operations Agent 密钥库的安全。要启用 **HASH_ALGO** 中指定的算法以保护密钥库的安全，必须使用 MigrateSymKey 工具。MigrateSymKey 工具会在 `sec.core` 命名空间中将 **HASH_ALGO_AS_SEED** 配置变量设置为 TRUE，并迁移密钥库内容。

备注:

- 在将 HASH_ALGO_AS_SEED 配置变量设置为 TRUE 之前，确保 OvSecCore 和 OvJSecCore 组件的版本是 12.00 或更高版本。
- 将 HASH_ALGO_AS_SEED 设置为 TRUE 后，只能使用 MigrateSymKey 工具更新 HASH_ALGO 值。请勿使用 ovconfchg 设置 HASH_ALGO_AS_SEED。
- 将 HASH_ALGO_AS_SEED 设置为 TRUE 时，管理服务器和代理程序节点必须使用相同的 HASH_ALGO 值才能使 ovcert -certreq 和 ovcm -grant 正常运行。
- 请勿在设置 HASH_ALGO_AS_SEED 值后进行恢复。

按照以下步骤使用 MigrateSymKey 工具：

1. 转到安装有 HPE Operations Agent 的管理服务器或节点上的以下位置：

   在 Windows 上：

   `%ovinstalldir%\lbin\secco\`

   在 HP-UX/Linux/Solaris 上：

   `/opt/OV/lbin/secco/`

   在 AIX 上：

   `/usr/lpp/OV/lbin/secco/`
2. 运行迁移工具：

   `MigrateSymKey -hash_algo [eMD5 | eSHA1 | eSHA224 | eSHA256 | eSHA384 | eSHA512]`

运行工具后，密钥库内容将根据指定的算法值迁移，并且 HASH_ALGO_AS_SEED 会设置为 TRUE。

用法：

```
MigrateSymKey

 -hash_algo [eMD5 | eSHA1 | eSHA224 | eSHA256 | eSHA384 | eSHA512]

 -help
```

- `-hash_algo` 会进行以下更新：
  - 如果未设置，则将“HASH_ALGO_AS_SEED”更新为 TRUE。
  - 将“HASH_ALGO”配置变量更新为指定的算法。
  - 如果未指定算法，则将使用 eSHA256 进行更新。

验证

- 如果 MigrateSymKey 工具成功完成操作，则会打印以下消息：`"Migration successful“`。
- 运行 MigrateSymKey 工具后，验证 HASH_ALGO_AS_SEED 是否已设置为 TRUE，以及 HASH_ALGO 是否已设置为指定的算法。

配置与 FIPS 兼容的 HPE Operations Agent

联邦信息处理标准 (FIPS) 是一种安全标准，可用于授权加密模块，从而保护硬件和软件产品中未分类、有价值且敏感的信息。为了跨网络环境强制实施安全通信并管理加密模块的所有要求，HPE Operations Agent 已增强为使其与 FIPS 兼容。

使 HPE Operations Agent 与 FIPS 兼容的先决条件

1. 确保已安装 HPE Operations Agent 版本 12.00 或更高版本。
2. 确保 RSA 密钥的长度大于或等于 2048 位。
   要设置或更改 RSA 密钥的长度，请参见配置安全组件的非对称密钥。
3. 确保使用与 FIPS 兼容的对称算法进行加密和解密。默认情况下，AES128 用作与 FIPS 兼容的对称算法。要更改默认对称算法，请参见配置安全组件的对称密钥。
4. 确保使用与 FIPS 兼容的哈希算法进行加密和解密。要更改默认哈希算法，请参见使用哈希算法配置安全组件。

使用 FIPS_tool 启用 FIPS 兼容

使用 OvSecCore 组件提供的 **FIPS_tool** 启用 FIPS 兼容。FIPS_tool 可在以下位置中找到：

在 Windows 上

```
%ovinstalldir%\lbin\secco\
```

在 Linux 上

```
/opt/OV/lbin/secco/
```

用法：

```
FIPS_tool
        -enable_FIPS [-Java_Home <jre_dir_path>]
        -disable_FIPS [-Java_Home <jre_dir_path>]
        -help
```

按照以下步骤使用 FIPS_tool 启用 FIPS 模式：

1. 转到安装有 HPE Operations Agent 的管理服务器或节点上的以下位置：
   在 Windows 上
   `%ovinstalldir%\bin\secco\`
   在 UNIX 上
   `/opt/OV/lbin/secco/`
   在 AIX 上
   `/usr/lpp/OV/lbin/secco`
2. 运行 FIPS_tool 以启用 FIPS 模式：
   `FIPS_tool -enable_FIPS [-Java_Home <jre_dir_path>]`
   ENABLE_FIPS_MODE 配置变量设置为 TRUE。

在此实例中，-Java_Home 指定 jre 目录路径。

例如：

```
FIPS_tool -enable_FIPS -Java_Home /opt/OV/nonOV/jre/b
FIPS_tool -enable_FIPS -Java_Home /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.51-2.4.5.5.el7.x86_64/jre
```

备注: 运行 FIPS_tool 以禁用 FIPS 模式：

```
FIPS_tool -disable_FIPS [-Java_Home <jre_dir_path>]
```

例如：

```
FIPS_tool -disable_FIPS -Java_Home /opt/OV/nonOV/jre/b
FIPS_tool -disable_FIPS -Java_Home /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.51-2.4.5.5.el7.x86_64/jre
```

如果运行 FIPS_tool 以禁用 FIPS 模式，则将禁用 FIPS 模式，但不会恢复在 FIPS 模式下所做的配置更改。

FIPS 模式下的配置设置

```
[sec.core]
DEF_SYM_KEY_ALGO=< FIPS compliant symmetric algorithm>
ENABLE_DEF_SYM_KEY_ALGO=TRUE
ENABLE_FIPS_MODE=TRUE
HASH_ALGO=< FIPS compliant hash algorithm>
HASH_ALGO_AS_SEED=TRUE
[sec.cm]
ASYMMETRIC_KEY_LENGTH=< FIPS compliant RSA key size>
[eaagt]
CRYPT_USING_LCORE=TRUE
```

例如：

```
[sec.core]
DEF_SYM_KEY_ALGO=eAES128
ENABLE_DEF_SYM_KEY_ALGO=TRUE
ENABLE_FIPS_MODE=TRUE
HASH_ALGO=eSHA256
HASH_ALGO_AS_SEED=TRUE
[sec.cm]
ASYMMETRIC_KEY_LENGTH=2048
[eaagt]
CRYPT_USING_LCORE=TRUE
```

备注:

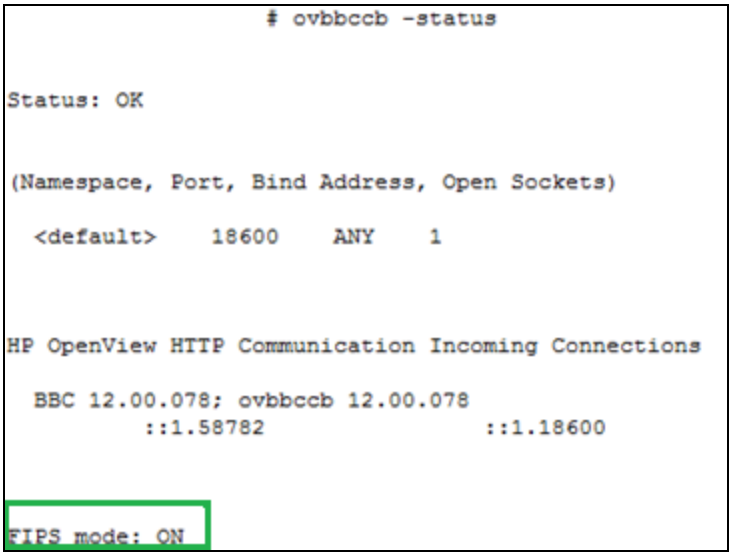
如果在 HPE Operations Agent 节点上启用 FIPS 模式，则必须在管理服务器上启用 FIPS 模式，然后从管理服务器重新部署所有策略。

验证 HPE Operations Agent 是否在 FIPS 模式下运行

运行以下命令：

```
ovbbccb -status
```

如果 HPE Operations Agent 在 FIPS 模式下运行，则在命令输出中，FIPS 模式处于打开状态。



疑难解答

Java 服务器与 Java 客户端之间的 SSL 通信在 FIPS 模式下失败。

Java 服务器与 Java 客户端之间的 SSL 通信在 FIPS 模式下失败，因为“Netscape 注释 (Netscape Comment)”字段被视为“Netscape 证书类型 (Netscape CertType)”字段。

要解决此问题，“Netscape 注释 (HP OpenView 证书) (Netscape Comment (HP OpenView Certificate))”字段将从 HPE Operations Agent 的当前版本 (12.01) 中的 OvSecCm 删除，并重新颁发证书。

备注: 运行以下命令检查证书详细信息：

openssl x509 –in <cert.pem> -text –noout

无法识别的 SSL 消息，纯文本连接。

出现此问题的原因是在启用 FIPS 时在 SSL 握手期间由 RSA Bsafe jar 导致的延迟。

要解决此问题，请在管理服务器上设置以下配置以延长 SSL 握手超时：

```
[bbc.http]

SSL_HANDSHAKE_TIMEOUT=60000
```

IO 异常：(sec.core-114) SSL 连接错误(找不到 ecdh 参数)。

仅当启用 FIPS 时，才会出现此问题。FIPS 模式下不支持 ECDHE 算法。如果在 SSL 通信期间，默认选择 ECDHE 算法，则在启用 FIPS 后将遇到此错误。

要解决此问题，请在 `java.security` 文件中禁用 ECDHE 算法：

```
jdk.tls.disabledAlgorithms= ECDHE
```

监视 Windows 事件日志

HPE Operations Agent 的日志文件封装器组件支持 Windows 事件日志的事件转发功能。即，可以读取从不同计算机转发的事件，并且该功能允许您监视 Windows 事件日志。Windows 事件日志策略帮助您将代理程序配置为监视所选的 Windows 事件日志。

以下版本的 Windows 提供事件日志的新类别 - 应用程序和服务日志：

- Windows Vista
- Windows Server 2008
- Windows Server 2008 R2
- Windows 7

可以使用具有适当配置的 Windows 事件日志策略的 HPE Operations Agent 监视这些应用程序和服务日志。

HPE Operations Agent 无法监视以下类型的事件日志：

- 源自远程系统的事件(通过使用 Windows 的事件订阅功能收集)
- 保存的事件日志

HPE Operations Agent 可以监视具有以下事件级别的事件：

- 错误
- 信息
- 警告
- LOG_ALWAYS

- 详细
- 审核失败
- 审核成功

下表显示了如何在消息浏览器中显示事件日志字段。

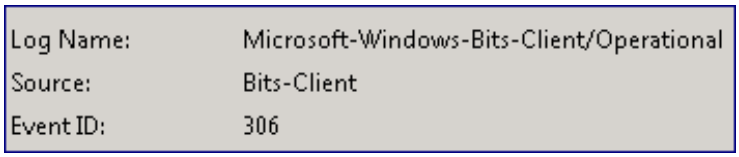
表 1：事件日志与消息浏览器字段的关联

事件日志字段	消息浏览器字段	注释
日期	日期	在受管节点上创建事件的日期。
时间	时间	在受管节点上创建事件的时间。
事件 ID	消息文本	将在任何其他消息文本之前显示事件 ID。
源	应用程序	无
类型 错误 错误 信息 警告 log_always 详细 审核失败 审核成功	严重性 严重 错误 正常 警告 正常 正常 警告 正常	将事件日志类型严重性映射到 Operations Manager 消息严重性。
类别	对象	无
描述	消息文本	所有其他消息文本(在事件 ID 之后)。
用户	未映射	未映射
计算机	节点	节点在管理服务器中的名称。
	消息组	空

从 HPOM for Windows 监视应用程序和服务事件日志

要创建用于监视应用程序和服务日志的 Windows 事件日志策略，请执行以下步骤：

1. 登录到 Windows 事件日志所在的 Windows 节点。
2. 打开事件查看器窗口。
3. 在控制台树中，选择事件。在详细信息窗格中，将显示事件日志的名称(在“日志名称 (Log Name)”字段旁边)。

   记下显示在详细信息窗格中的日志文件的名称。
4. 打开 HPOM for Windows 控制台。
5. 在控制台树的“按类型分组的代理程序策略 (Agent Policies Grouped by Type)”下，右键单击 **Windows 事件日志**，然后单击**新建 > 策略**。

   Windows 事件日志策略的策略编辑器将打开。
6. 在“源”选项卡的“事件日志名称”字段中，输入 Windows 事件日志的名称(在步骤 3 中记下的名称)。

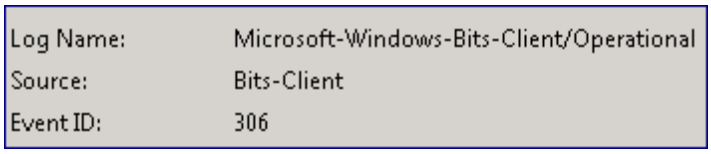
   

7. 按照 HPOM for Windows 联机帮助中的指示指定策略中的其他详细信息。
8. 保存策略。
9. 在 Windows 节点上部署策略。

从 HPOM on UNIX/Linux 9.xx 监视应用程序和服务事件日志

要创建用于监视应用程序和服务日志的 Windows 事件日志策略，请执行以下步骤：

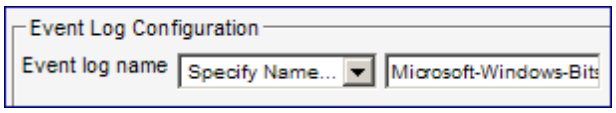
1. 登录到 Windows 事件日志所在的 Windows 节点。
2. 打开事件查看器窗口。
3. 在控制台树中，选择事件。在详细信息窗格中，将显示事件日志的名称(在“日志名称 (Log Name)”字段旁边)。



记下显示在详细信息窗格中的日志文件的名称。

4. 登录到 HPOM for UNIX 管理 UI。
5. 单击 **OMU**。
6. 单击**浏览 > 所有策略类型**。
7. 单击“Windows 事件日志”。将打开策略类型 **Windows_Event_Log** 页面。
8. 单击  ，然后单击**新建策略**。将打开“添加 Windows_Event_Log 策略”页面。

   在“源”选项卡的“事件日志名称”字段中，选择“指定名称”。将显示新的文本框。在该文本框中输入 Windows 事件日志(已在步骤 3 中记下)的名称。



9. 按照 HPOM for UNIX 联机帮助中的指示指定策略中的其他详细信息。
10. 保存策略。
11. 在 Windows 节点上部署策略。

第 2 章: 适用于 RTMA 组件的 adviser

只有在启用 Glance Pak Software LTU 或 Glance Pak Software LTU 后，才可以使用 adviser 功能。

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 联机帮助。

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。

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，HPE 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 HP-UX/Linux/Solaris 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\<os_type>\adv
```

在 HP-UX/Linux/Solaris 上：

```
/var/opt/perf/perfd/os/<os_type>/adv
```

在此实例中，<os_type> 指定要运行脚本的节点上的操作系统。

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 HP-UX/Linux/Solaris 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 <script_name>-n<system_name>
   ```

   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。

   提示: 在远程系统上使用脚本时，请确保远程系统上正在运行 perfd 进程。您可以阻止其他系统运行本地系统上的 adviser 脚本。有关详细信息，请参见限制访问。

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 HP-UX/Linux/Solaris 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 <system_list_text_file> -s <script_name>

   adviser 脚本开始在 <system_list_text_file> 文件中指定的系统上运行，并根据脚本文件的配置显示结果。

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 HP-UX/Linux/Solaris 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：

[any text or characters]

或

// [any text or characters]

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。

```
item1 {>, <, >=, <=, ==, !=} item2 [OR item3 \
                {>, <, >=, <=, ==, !=} item4]
```

或：

```
item1 {>, <, >=, <=, ==, !=} item2 [AND item3 \
                {>, <, >=, <=, ==, !=} item4]
```

(“==”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 度量测量每秒单位数
- 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 = ""
```

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 condition [FOR duration {SECONDS, MINUTES, INTERVALS}]
 [condition [FOR duration {SECONDS, MINUTES, INTERVALS}] ] ...
[START statement]
[REPEAT [EVERY duration [SECONDS, MINUTES, INTERVAL, INTERVALS]]
 statement]
[END statement]
```

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。

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 or CRITICAL), (YELLOW or WARNING), RESET] ALERT printlist
```

RED 和 YELLOW 分别与 CRITICAL 和 WARNING 同义。

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
```

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。

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。

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 condition THEN statement [ELSE statement]
```

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。

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 语句会终止 LOOP 语句。

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。

PRINT 语句

使用 PRINT 语句将您所收集的数据打印到标准输出(padv 命令控制台)。您可以使用 PRINT 语句记录度量或算出的变量。

语法：

```
PRINT printlist
```

```
PRINT Example
```

```
PRINT "The Application OTHER has a total CPU of ",
 other:app_cpu_total_util, "%"
```

开始执行此语句时，将类似以下内容的消息打印到 padv 命令控制台：

> **注：** The Application OTHER has a total CPU of 89%.

SYMPTOM 语句

语法：

```
SYMPTOM variable [TYPE = {CPU, DISK, MEMORY, NETWORK}]

RULE measurement {>, <, >=, <=, ==, !=} value PROB probability

[RULE measurement {>, <, >=, <=, ==, !=} value PROB probability]

 .

 .

 .
```

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。

`variable` 指的是此症状的名称。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。

measurement 和 value 之间的条件为 **True** 的所有概率的总和就是症状发生的概率。

第 3 章: 性能警报

可以使用性能收集组件定义警报。当 **oacore** 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。

由于数据由 **oacore** 或其他收集器记录，因此会将该数据与 `alarmdef` 文件中的警报定义进行比较。如果 **oacore** 或 DSI 度量满足或超过定义的条件，则会触发警报或操作。

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数据，并报告结果。

有关为 DSI 度量定义警报的详细信息，请参见“使用记录到度量数据存储区中的 DSI 数据”章节中的定义 DSI 度量的警报。

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)。

但是，如果数据不记录到数据库文件(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。有关详细信息，请参见阈值。

警报定义中定义的操作可以包括：

- 使用操作系统命令执行的本地操作
- 发送到 Network Node Manager (NNM) 和 HPOM 的消息

警报生成器

性能收集组件的警报生成器组件处理本地系统上的 `alarmdef` 文件和可用的系统性能数据，然后在必要时生成警报。警报生成器由以下组件构成：

- 警报生成器服务器 (`perfalarm`)
- 警报生成器数据库 (`agdb`)

警报生成器服务器扫描 alarmdef 文件中的信息，并根据 alarmdef 文件中的配置信息将警报发送到目标。agdb 数据库包括 perfalarm 组件针对特定事件转发 SNMP 陷阱的目标系统的列表。您可以修改 perfalarm 组件的默认行为，也可以通过 agsysdb 实用程序访问 agdb 数据库中的可用数据。

运行以下命令查看发送警报通知的目标系统列表：

```
agsysdb -l
```

启用 perfalarm

默认情况下，警报生成器服务器 (perfalarm) 已禁用。使用下列方法之一启用 perfalarm：

- **安装 HPE Operations Agent 之前：**
  a. 在配置文件中，将变量 ENABLE_PERFALARM 设置为 **True**：

     ```
     set nonXPL.config:ENABLE_PERFALARM=TRUE
     ```

  b. 运行以下命令以使用配置文件安装 HPE Operations Agent：

     在 Windows 上：

     cscript oainstall.vbs -i -a -agent_profile <path>\<profile_file> -s <server_ip_address>

     在 HP-UX/Linux/Solaris 上：

     ./oainstall.sh -i -a -agent_profile <path>/<profile_file> -s <server_ip_address>

     在此实例中：

     <profile_file> 是配置文件的名称。

     <path> 是配置文件的完整路径。

     <server_ip_address> 是管理服务器的 IP 地址或主机名。

- **安装 HPE Operations Agent 之后：**
  a. 运行以下命令启用 perfalarm：

     在 Windows 上：

     ```
     copy "%ovinstalldir%newconfig\alarmdef.mwc" "%ovdatadir%"

     "%ovinstalldir%bin\ovpacmd" start alarm
     ```

     在 HP-UX/Linux/Solaris 上：

     ```
     cp/opt/perf/newconfig/alarmdef /var/opt/perf/

     /opt/perf/bin/ovpa start alarm
     ```

     在 AIX 上：

     ```
     cp/usr/lpp/perf/newconfig/alarmdef /var/opt/perf/
     ```

```
/usr/lpp/perf/bin/ovpa start alarm
```

b. 转到节点上的以下位置：

在 64 位 Windows 上：

```
%ovinstalldir%bin\win64\OpC\install
```

在其他 Windows 上：

```
%ovinstalldir%bin\OpC\install
```

在 HP-UX/Linux/Solaris 上：

```
/opt/OV/bin/OpC/install
```

在 AIX 上：

```
/usr/lpp/OV/bin/OpC/install
```

c. 运行以下命令重新配置 HPE Operations Agent：

在 Windows 上：

```
cscript oainstall.vbs -a -configure -agent_profile<path>/<profile_file>
```

在 HP-UX/Linux/Solaris 上：

```
./oainstall.sh -a -configure -agent_profile<path>/<profile_file>
```

在此实例中：

<profile_file> 是配置文件的名称。

<path> 是配置文件的完整路径。

备注: 如果从 HPE Operations Agent 11.xx 升级到 12.00，perfalarm 将像之前定义的那样继续运行。

将 SNMP 陷阱发送到 Network Node Manager

要将 SNMP 陷阱发送到 Network Node Manager，必须在性能收集组件中使用以下命令将系统名称添加到 agdb：

```
agsysdb -add systemname
```

生成警报时，会将 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 systemname
```

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。

将消息发送到 HPOM

默认情况下，警报生成器不执行 EXEC 语句中的任何警报所定义的本地操作，而只是将消息发送到 HPOM 的事件浏览器。

可以使用以下命令更改默认值，停止将信息发送到 HPOM：

```
agsysdb -ovo OFF
```

表 9：将信息发送到 HPOM 和执行本地操作的设置

HPOM 标记	操作监视组件正在运行	操作监视组件未运行
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。

执行本地操作

默认情况下，性能收集组件不运行在 EXEC 语句中指定的本地命令。

可以按如下方式更改默认值以启用本地操作：

```
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

本地操作标记	操作监视组件正在运行	操作监视组件未运行
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。

表 10：将信息发送到 HPOM 和执行本地操作的设置(续)

本地操作标记	操作监视组件正在运行	操作监视组件未运行
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
```

将出现以下信息：

```
PA alarming status:

OVO messages : on      Last Error : none

Exec Actions : on

Analysis system:<hostname>, Key=<ip address>

PerfView : no Last Error :<error number>

SNMP : yes Last Error :<error number>
```

为警报分析历史数据

可以使用 utility 程序的 analyze 命令查找数据存储区中的警报条件(请参见第 6 章，Utility 命令)。这与前面说明的实时警报的处理不同，因为您要将数据存储区中的历史数据与 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。如果在足够长的时间内满足警报条件，以至于在第一个警报结束前生成另一个警报，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
警告： 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 HPOM 或将 SNMP 陷阱发送到 NNM
- EXEC - 执行操作系统命令
- PRINT - 当使用 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，请参见警报语法参考)。

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  condition [[AND,OR]condition]
  FOR duration [SECONDS, MINUTES]

  [TYPE="string"]
  [SERVICE="string"]
  [SEVERITY=integer]
  [START action]
  [REPEAT EVERY duration [SECONDS, MINUTES] action]
  [END action]

[RED, CRITICAL, ORANGE, MAJOR, YELLOW, MINOR, CYAN, WARNING,
  GREEN, NORMAL, RESET] ALERT message

EXEC "UNIX command"

PRINT message
IF condition
   THEN action
   [ELSE action]

{APPLICATION, PROCESS, DISK, LVOLUME, TRANSACTION, NETIF, CPU,
   FILESYSTEM} LOOP action
```

```
INCLUDE "filename"

USE "data source name"

[VAR] name =  value

ALIAS name = "replaced-name"

SYMPTOM  variable [ TYPE = {CPU, DISK, MEMORY, NETWORK}]
  RULE condition  PROB probability
  [RULE condition PROB probability]
    .
    .
```

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

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，请参见性能收集组件 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
```

对于多实例数据类型(如 application、process、disk、netif、transaction、lvolume、cpu 和 filesystem)，除非使用 LOOP 语句，否则必须将度量与数据类型名称关联。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。

备注: 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 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(其中 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
```

备注: 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。Width 指定字段应格式化为多宽，decimals 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：

```
metric names [|[-]width[|decimals]]
```

```
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 condition [[AND,OR]condition]
  FOR duration{SECONDS, MINUTES}
  [TYPE="string"]
  [SERVICE="string"]
  [SEVERITY=integer]
  [START action]
  [REPEAT EVERY duration {SECONDS, MINUTES} action]
  [END action]
```

在此实例中：

- 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。如果正在使用 ServiceNavigator，则可以将性能收集组件警报与 ServiceNavigator 中定义的服务链接。有关详细信息，请参见《HP Operations ServiceNavigator 概念和配置指南 (HP Operations ServiceNavigator Concepts and Configuration Guide)》 。

SERVICE="Service_id"

- SEVERITY 是介于 0 到 32767 之间的整数。
- START、REPEAT 和 END 是用于指定在满足、再次满足或停止警报条件时所采取操作的关键字。ALARM 语句中应至少有 START、REPEAT 或 END 之一。每个关键字后面都会跟着一项操作。
- action 指示 ALARM START、REPEAT 或 END 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。

- Conditions 定义为两个项之间的比较。

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

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

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
```

- 还可以在子条件之间不指定“OR”和“AND”运算符来使用复合条件。例如：

```
ALARM gbl_cpu_total_util > 90
gbl_cpu_sys_mode_util > 50 for 5 minutes
```

两个条件均为 **True** 时生成警报。

FOR 为条件必须保持 **True** 以触发警报指定 SECONDS 或 MINUTES 单位的持续时间。

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。

不要设置比收集间隔短的持续时间。因为在完成下一个收集循环之前，度量值不会更改，设置比收集间隔短的持续时间意味着要对生成重复警报进行多余的处理。

在 parm 文件中，collectioninterval 参数下的以下参数控制着收集间隔：

- global：此参数指示除进程度量外，所有记录到 oacore 和 dsi 数据库文件的度量的收集间隔(以秒为单位)。默认情况下，此值设置为 5 分钟。
- process：此参数指示进程度量的收集间隔(以秒为单位)。

  有关 collectioninterval 参数的详细信息，请参见配置数据记录间隔。
- REPEAT EVERYnSECONDS, MINUTES 指定警报重复前的时间段。

  在此实例中，**n** 表示持续时间。例如，REPEAT EVERY5SECONDS, MINUTES

如何使用

警报循环开始于所有用 **AND** 连接的警报条件或某个用 **OR** 连接的警报条件为 **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)。

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 HPOM，将 SNMP 陷阱发送到 NNM。VAR 语句

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 示例将测试度量 gbl_swap_space_util 是否大于 90。根据对警报生成器的配置，ALERT 可通过 SNMP 陷阱发送到 NNM，也可以作为消息发送到 Operations Manager。

REPEAT 将每隔 15 分钟检查一次 gbl_swap_space_util 条件。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 NNM 或将消息发送到 HPOM。EXEC 可以触发将邮件消息作为节点上的本地操作发送，具体取决于警报生成器的配置。

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 示例将测试度量 gbl_cpu_total_util 和 gbl_cpu_sys_mode_util。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 警报。

```
ALARM gbl_cpu_total_util > 85
  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."
```

下一个 ALARM 示例将测试度量 gbl_cpu_total_util 和 gbl_cpu_sys_mode_util。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 > 85
  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 gbl_cpu_total_util > 85 THEN    PROCESS LOOP...
```

不同间隔不能按您的预期同步，因此结果是无效的。

备注: 对于 GlancePlus，请在进程循环中使用进程度量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，而不是等到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 message
```

- RED 与 CRITICAL 同义，ORANGE 与 MAJOR 同义，YELLOW 与 MINOR 同义，CYAN 与 WARNING 同义，GREEN 与 NORMAL 同义。这些关键字将警报符号转化为与警报条件关联的颜色。
- RESET - 当 ALERT 条件结束时，发送 RESET ALERT 和消息。如果尚未在警报定义中定义 reset，ALARM 条件结束时只发送 RESET ALERT，而不发送消息。
- message - 用于创建消息的字符串和数字值的组合。使用参数 [|[-]width [|decimals]] 设置数字值的格式。Width 指定字段应格式化为多宽，decimals 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 NNM 或将消息发送到 HPOM，具体取决于警报生成器的配置。对于发送到 HPOM 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性为严重的警报。

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 "system command"
```

system command - 是要在本地系统上执行的命令。

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 HPOM，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 语句将执行 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"
```

备注: 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。

有关详细信息，请参见 PRINT 语句 (第 85 页)

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 condition THEN action [ELSE action]
```

Condition - 条件定义为两个项之间的比较。

```
item1 {>, <, >=, <=, ==, !=}item2
   [AND, OR[item3 {>, <, >=, <=, ==, !=}item4]]
```

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。

action - 任何操作或设置一个变量。(ALARM 在这种情况下无效。)

如何使用

IF 语句测试 condition。如果 condition 为 true，执行 THEN 后面的 action。如果 condition 为 false，则 action 取决于可选的 ELSE 子句。如果已指定 ELSE 子句，执行子句之后的 action；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 NNM 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 gbl_cpu_total_util > 85 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 action。

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。
- action - PRINT、EXEC、ALERT，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"filename"

其中 filename 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 alarmdef 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。

备注: 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"datasourcename"

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有序搜索列表。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，请参见本章前面的度量名称。

perfalarm 的度量名称和数据源匹配的默认行为是先在 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。当遇到“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] name = value

name - 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。

value - 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
```

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 = ""
```

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 name ="replaced-name"
```

- name - 名称必须以字母开头，可包括字母、数字和下划线字符。
- replaced-name - 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。

有关详细信息，请参见SYMPTOM 语句 (第 85 页)

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 NNM 中创建严重性为严重的警报。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 NNM，或将消息发送到 HPOM(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

必须在警报定义文件 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
   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。

第 4 章: 安全环境中的 HPE Operations Agent

HPOM 与 HPE Operations Agent 一起帮助您从中央控制台监视和管理网络环境中部署的系统和应用程序。在基于 HPOM 的管理环境中，在您感兴趣的系统上安装 HPE Operations Agent 之后，您便可开始监视这些系统。使用从 HPOM 控制台部署到代理程序节点上的策略，可以启用代理程序的不同监视功能。

HPE Operations Agent 在分布式环境中的主要责任是：

- **监视数据**
  HPE Operations Agent 可将特定度量的值与预设值进行比较，并根据其配置执行必要的操作。从 HPOM 控制台部署到节点的策略对于简化 HPE Operations Agent 的监视功能尤为重要。
- **收集和存储数据**
  您可计划 HPE Operations Agent 的性能数据收集器在监视的系统上收集和记录您感兴趣的数据。可通过安装 SPI 来添加其他收集功能，并将 SPI 收集到的数据记录到代理程序数据存储区。

策略

要使用代理程序，必须从 HPOM 控制台将配置详细信息和规范的集合(称为“策略”)部署到受管节点上。根据部署的策略类型，会启用 HPE Operations Agent 的不同组件。策略可向代理程序提供以下详细信息：

- **监视源详细信息**
  - 要监视的对象
  - 对象的监视轮询间隔
  - 监视的对象的阈值
  - 根据设置的阈值分析数据的规则和条件
- **事件详细信息**
  您可使用策略将 HPE Operations Agent 配置为，在监视的对象违反阈值规则时生成含有消息、说明和严重性标记的事件。这些事件将以消息的形式转发到 HPOM 控制台。您可将代理程序设置为根据这些事件执行某种操作。
- **数据收集详细信息**
  如果要监视外部程序收集到的数据，可计划 HPE Operations Agent 将这些数据记录到其数据存储区。

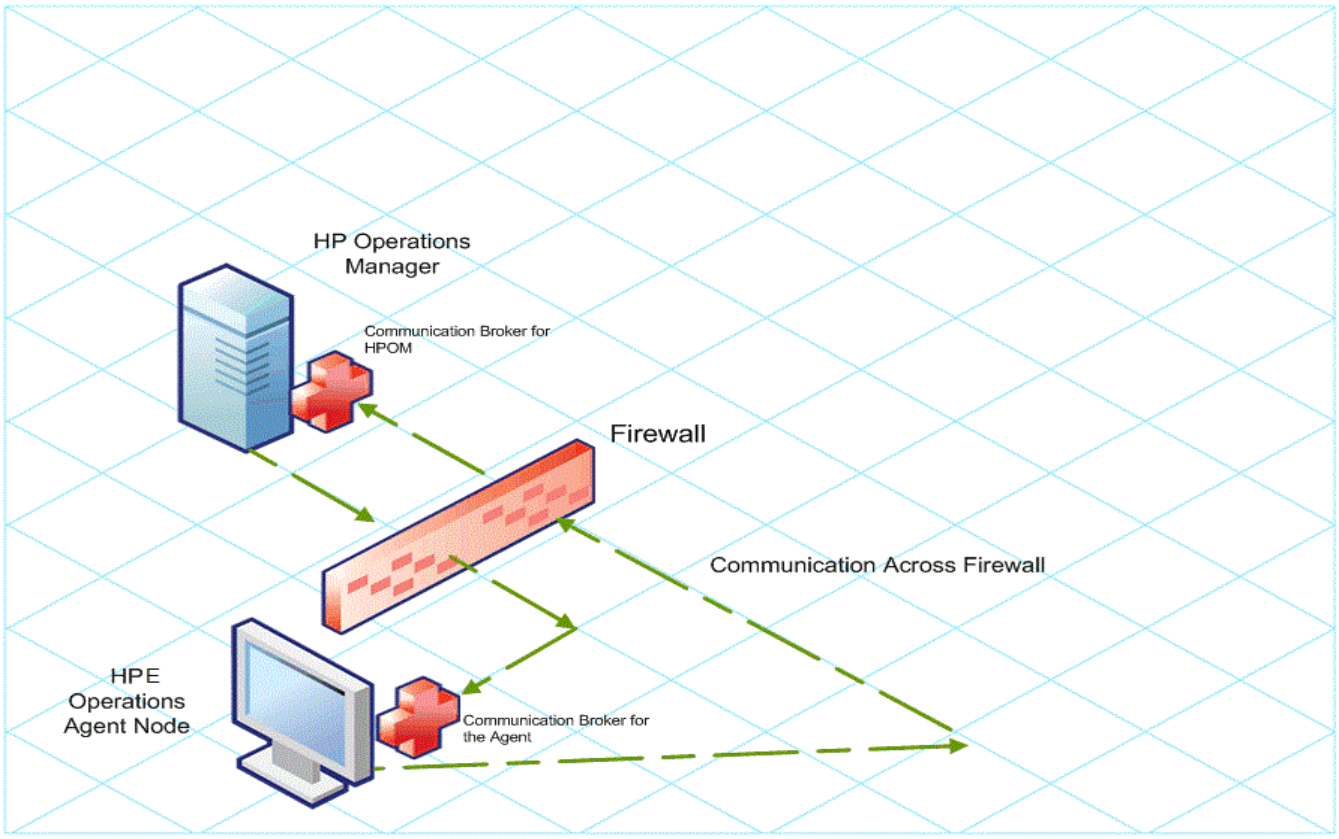
HTTPS 通信模式

HPE Operations Agent 节点可使用 HTTPS 通信模式轻松地相互通信以及与其他行业标准产品通信。

HTTPS 通信的优点

- **通过防火墙通信**
  在 HTTPS 协议的帮助下，HPE Operations Agent 节点可与跨防火墙可用的其他系统通信。可在使用 HTTP 代理和防火墙构建的安全环境中部署 HPE Operations Agent。

  下图演示了如何使用 HTTPS 通信跨越防火墙。

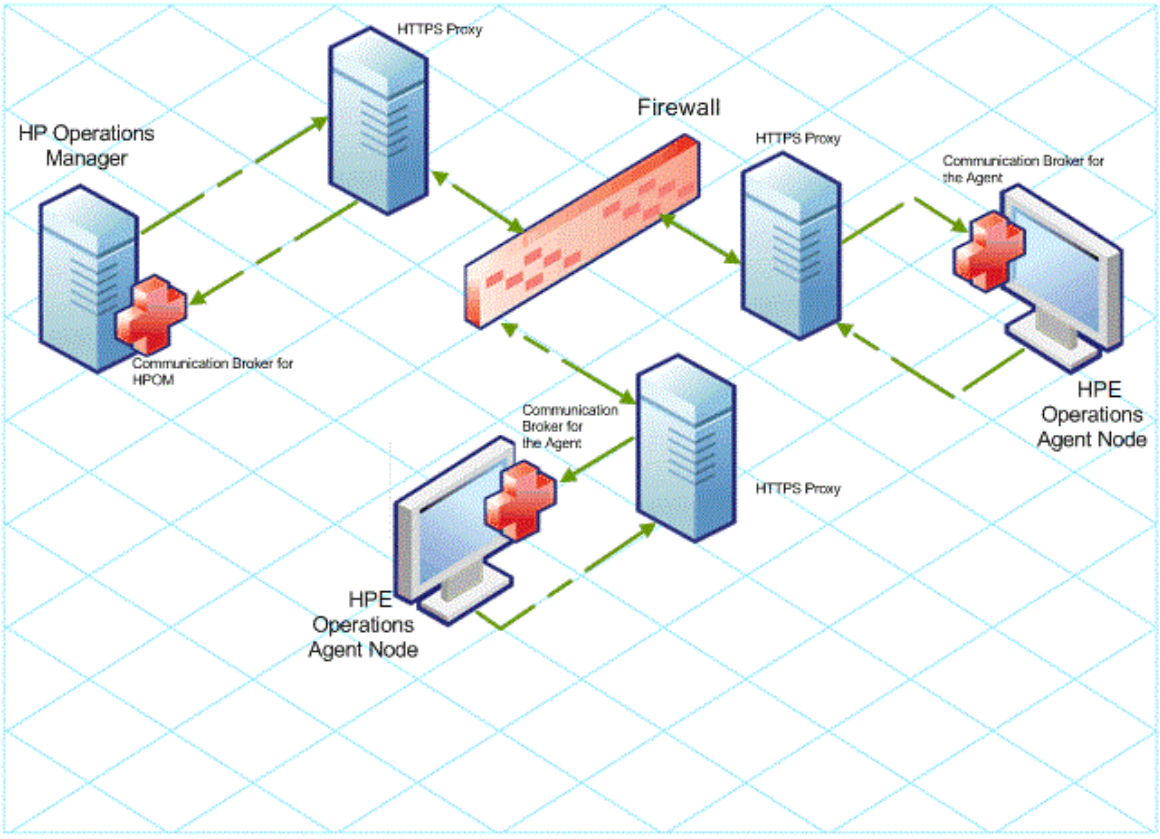


- **高级安全性**
  HPE Operations Agent 产品使用安全套接字层 (SSL) 限制和控制用户访问。在 SSL 的帮助下，HPE Operations Agent 产品将压缩并加密其与其他系统的通信中涉及的所有数据。

  此外，所有远程消息均通过通信中介器组件到达，从而提供了 HPE Operations Agent 节点的单一端口入口。

从 HPE Operations Agent 节点中，如果您想要发送消息、文件或对象，则可配置一个或多个标准 HTTP 代理以跨越防火墙或到达远程系统。

下图演示了如何使用外部 HTTPS 代理跨越防火墙：



- **开放标准**
  HPE Operations Agent 的 HTTPS 通信是基于行业标准 HTTP 1.1 协议和 SSL 套接字构建的。HPE Operations Agent 符合 HTTP、SSL 和 SOAP 等开放标准，从而使您能够最大程度地利用当前的 HTTP 基础结构。
- **可扩展性**
  HPE Operations Agent 的 HTTPS 通信设计为，不管环境规模以及发送和接收的数据量如何，都能正常执行。可根据您组织的要求配置 HPE Operations Agent 的 HTTPS 通信。

通信中介器

“通信中介器”组件为 HPE Operations Agent 节点提供单一端口解决方案。在典型的部署场景中，可向 HPE Operations Agent 节点注册多个服务器以进行数据通信。HPE Operations Agent 产品通过通信中介器定向对该节点上所有已注册服务器的请求。通信中介器按 HTTP 代理转发 HTTP 请求的方式，以透明方式将请求转发到已注册的服务器。通信中介器的默认端口是 383。您可将 HPE Operations Agent 产品配置为对通信中介器使用其他端口。

为提高 UNIX 系统上的安全性，通信中介器用 chroot 启动。chroot 通过使指定的路径充当根目录，限制部分文件系统对通信中介器进程可见，从而减少未经授权的访问。

通信中介器在 UNIX 系统上作为守护进程运行，在 Windows 系统上作为服务运行。

通信中介器最少使用一个端口来接受节点的传入数据。该端口与唯一节点标识符 (OVCoreID) 关联来标识节点。您可将通信中介器配置为对高可用性节点使用多个端口。

防火墙场景

防火墙可保护网络中的系统免遭外部攻击。防火墙通常将 Internet 与专用的 Intranet 分隔开来。您可实施多级别防火墙，以限制从敏感度较低的环境访问受信任度更高的环境。

防火墙将网络环境分成两个基本区域：**受信任区域**和**隔离区域 (DMZ)**(如 Internet)。防火墙配置可确保从 DMZ 到受信任区域的数据传输受限制或受控制。根据配置，防火墙可允许**双向通信**或**仅出站通信**。

如果在您的环境中将防火墙配置为允许双向通信，则网络将允许在某些限制下跨防火墙进行双向 HTTPS 通信。您可在此环境中配置防火墙设置以使用以下配置选项：

- 代理：如果您的网络仅允许某些代理系统通过防火墙进行连接，则可以通过这些代理重定向 HPE Operations Agent 通信。
- 本地端口：如果您的网络仅允许从某些本地端口进行出站连接，则可以将 HPE Operations Agent 配置为使用特定本地端口。
- 通信中介器端口：如果您的网络仅允许入站连接到某些目标端口，但不是 383 端口，则可以配置备用通信中介器端口。

如果您环境中的防火墙允许仅出站通信，您可为 HPE Operations Agent 产品配置一个**反向通道代理 (RCP)**。为 HPE Operations Agent 节点配置的 RCP 像 HTTP 代理一样工作，支持您将数据从 DMZ 传输到受信任(安全)区域。RCP 将建立到通信中介器的通信通道，而不是直接与 HP 软件系统通信。通信中介器将验证源自 DMZ 的信息，然后将经验证的信息传输到受信任(安全)区域中存在的 HPE Operations Agent 节点。

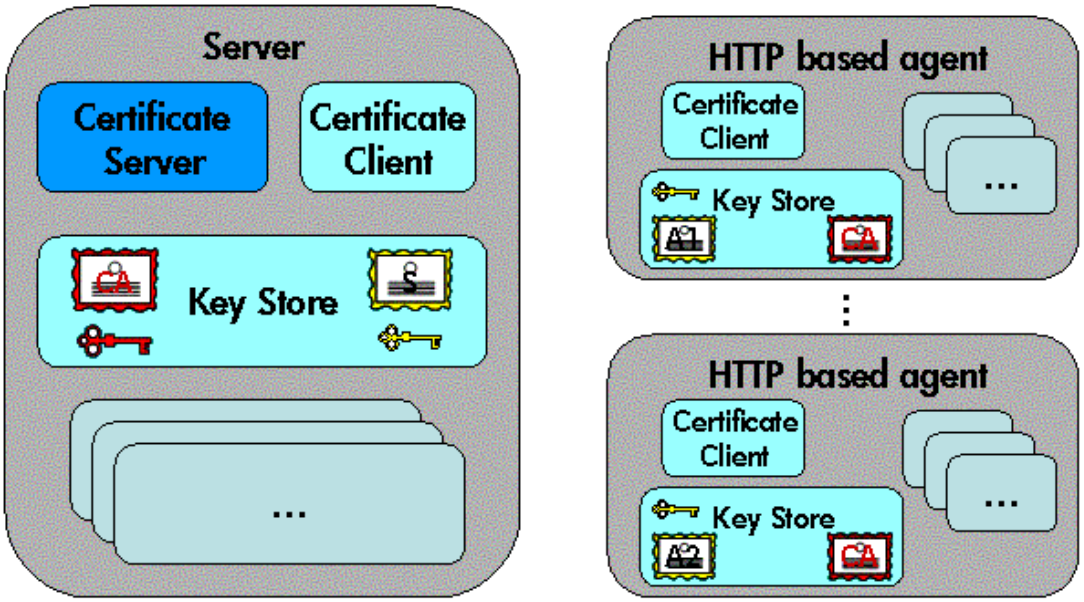
基于 HTTPS 的安全组件

要与其他 HPE Operations Agent 节点或 HPOM 服务器通信，HPE Operations Agent 节点必须具有一个有效的行业标准 X509 证书。各节点在交换由 1024 位密钥签名的证书之后，相互进行通信。交换证书有助于节点标识受管环境中的其他节点或服务器。

负责创建和管理证书的主要组件是：

- 证书服务器(驻留在 HPOM 服务器上)
- HPE Operations Agent 密钥库
- HPE Operations Agent 证书客户端

下图演示了这些组件:

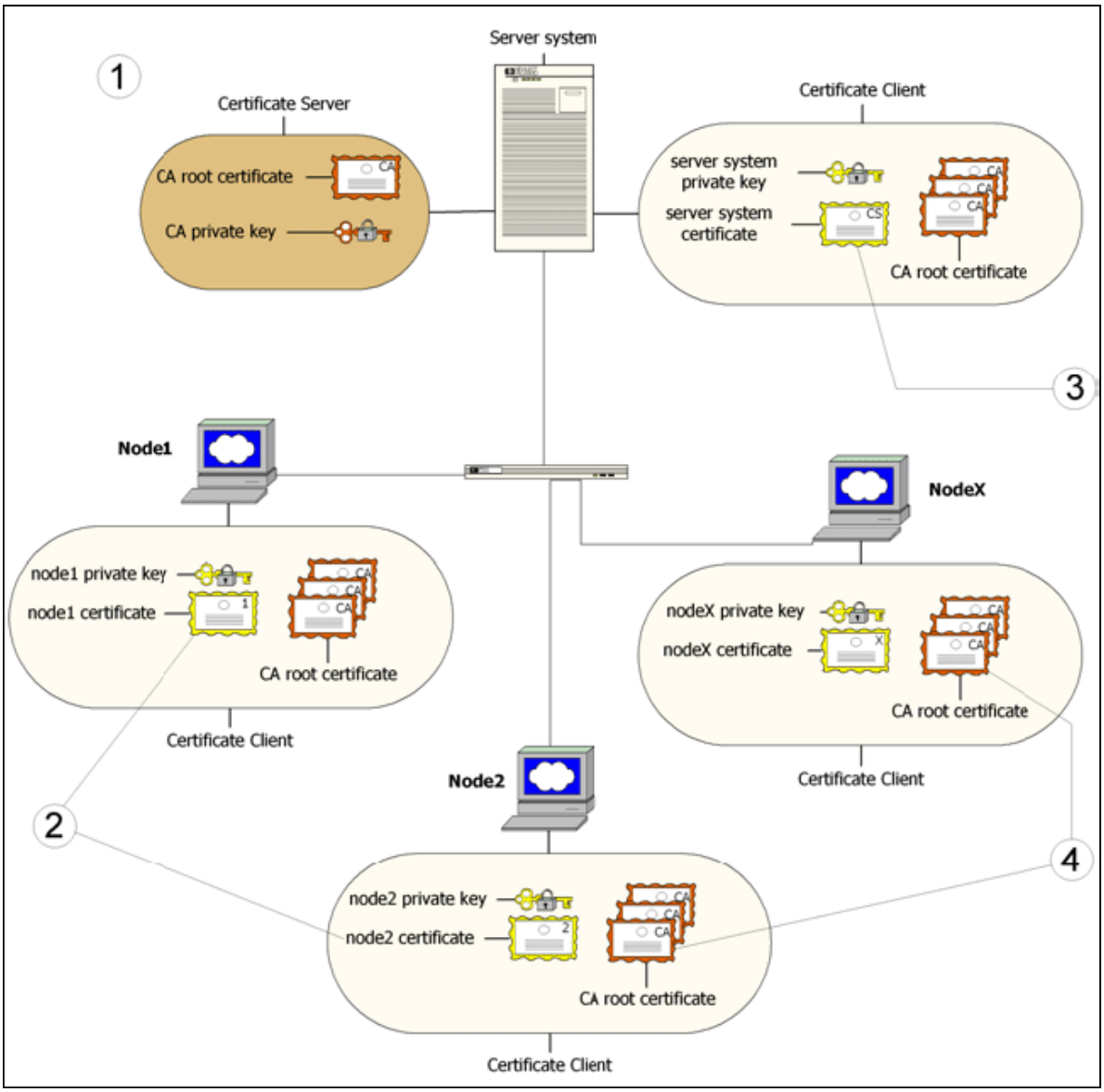


对于托管 HPE Operations Agent 的每个系统，将为参数 OvCoreId 分配唯一标识符值，该参数是在相应系统上安装 HPE Operations Agent 期间创建的。

备注: 代理程序节点的 OvCoreId 参数不发生更改，即使是在更改系统的主机名或 IP 地址之后也是如此。

对于每个代理程序节点，OvCoreId 用作唯一标识符且包含在对应的节点证书中。安装期间会为 OvCoreId 分配值。

下图演示了 HPE Operations Agent 部署中的经验证的通信环境:



1. 服务器系统托管证书服务器，而证书服务器包含必需的证书颁发机构 (CA) 功能。
2. 每个系统都有一个已由证书服务器使用证书颁发机构私钥签名的证书。
3. 服务器系统还需要一个证书来管理其身份。
4. 每个系统都具有一个受信任的根证书列表，其中必须至少包含一个证书。受信任的根 (CA) 证书用于验证通信伙伴的身份；如果可使用受信任的证书列表验证提供的证书，则信任通信伙伴。

   当证书客户端由多个 HPOM 服务器管理时，需要受信任的根证书列表。

证书

HPE Operations Agent 使用以下两种类型的证书：

- 根证书
- 节点证书

根证书是自签名证书，它包含证书服务器的证书颁发机构的标识。属于根证书的私钥存储在证书服务器系统上，且受保护以免于未经授权的访问。证书颁发机构使用其根证书对所有证书进行数字签名。

受管环境中的每个代理程序节点都会接收一个由证书服务器颁发的节点证书。颁发证书时，在代理程序节点上运行的证书客户端会将对应的私钥存储在文件系统中。

备注: 节点证书包含唯一的节点标识符 `OvCoreId`。`OvCoreId` 的示例如下：
`d498f286-aa97-4a31-b5c3-806e384fcf6e`

每个节点都可通过其节点证书进行安全验证。节点证书可由环境中使用根证书验证签名的所有其他节点进行验证。节点证书可用于在两个使用客户端和服务器身份验证的 HTTPS 节点之间建立基于 SSL 的连接，且可配置为对所有通信进行加密。

证书客户端提供的 `ovcert` 工具会列出密钥库的内容或显示有关已安装证书的信息。

HPE Operations Agent 证书服务器

证书服务器负责以下工作：

- 创建和安装自签名的根证书。
- 从文件系统导入自签名的根证书。
- 存储根证书的私钥。
- 批准或拒绝证书申请。
- 创建新的证书及对应的私钥，或创建手动证书安装的安装密钥。
- 为客户端提供服务以自动检索受信任的根证书。

证书颁发机构

备注: 每个 HPOM 服务器自动配置为证书颁发机构。每个代理程序节点的默认证书服务器便是与该节点关联的 HPOM 服务器。

证书颁发机构是证书服务器的一部分，是证书管理的信任中心。由此证书颁发机构签名的证书将被视为有效证书，因此可信。证书颁发机构必须托管于高度安全的位置。默认情况下，它安装在托管 HPOM 的系统上。

由于证书颁发机构是信任根，因此它处理的是自签名的根证书。此根证书及对应的私钥在文件系统上创建并存储，且受一定保护以允许证书颁发机构进行处理。初始化之后，证书颁发机构将使用其根证书对批准的证书申请进行签名。

证书客户端

证书客户端在每个代理程序系统上运行。

证书客户端的运行方式如下：

- 证书客户端检查节点是否具有有效的证书。
- 如果节点没有证书，则证书客户端将生成新的公钥和私钥对，并根据节点的唯一标识(OvCoreId 值)创建证书申请。证书客户端将证书申请与其他节点属性一起发送到证书服务器，然后证书客户端等待响应。
- 其他节点属性(例如，节点的 DNS 名称和 IP 地址)有助于证书服务器标识申请源。
- 当证书服务器颁发新证书时，证书客户端会将该证书安装在节点上。证书客户端可确保所有基于 HTTPS 的通信均使用该证书。
- 如果申请未成功处理，则会记录描述性错误并设置关联的状态。

此外，证书客户端还将执行以下任务：

- 证书客户端与证书服务器联系以更新服务器的受信任根证书。
- 它支持从文件系统导入节点证书及对应的私钥。
- 它支持导入受信任的根证书。
- 它提供状态信息。 状态包括：正常、有效证书、无证书、已申请证书和证书申请被拒绝。

根证书更新和部署

可能需要更新一个或多个节点的受信任的根证书，例如，在托管若干证书服务器的环境中。

可采用安全方式向证书客户端提供所有当前受信任的根证书。这样通常足以提供证书颁发机构的根证书。但是，可能需要向选定的证书客户端另外部署一个或多个根证书，例如，当环境中存在多个证书颁发机构时。

使用证书客户端，可通过命令行工具 ovcert 申请证书服务器以更新受信任的根证书。

第 III 部分: 使用 HPE Operations Agent 性能收集组件

HPE Operations Agent 通过收集指示系统必需元素的运行状况、性能、资源利用率和可用性的各种度量来帮助您监视系统。

oacore 数据收集器的数据收集机制由 **parm** 文件中的设置控制，对于 Windows，该文件位于 **%ovdatadir%** 目录中，对于 UNIX 或 Linux，该文件位于 `/var/opt/perf` 目录中。基于 **parm** 文件中定义的类，**oacore** 数据收集器收集系统范围的资源利用率信息，如进程数据、不同设备的性能数据、事务数据和逻辑系统数据。

由 **oacore** 收集的数据存储在度量数据存储区中。度量数据存储区是基于关系数据库管理系统 (RDBMS) 的数据存储区。对于记录到度量数据存储区中的每个数据类，将创建一个数据库文件。数据库文件可在 `datadir/databases/oa` 中找到。

可以使用诸如 **Utility** 和 **Extract** 等工具查看存储在度量数据存储区中的特定信息。也可以使用 `utility` 程序的 `analyze` 命令对照警报定义分析历史数据，并报告结果。扫描报告列出了 **parm** 文件的配置设置、每个应用程序的名称和定义、应用程序中的添加/删除或更改以及有关磁盘空间可用性的信息。

您可以使用基线的流程来基于存储在度量数据存储区中的历史数据计算和提供参考值。基线数据提供参考值来分析性能趋势并动态设置最佳阈值来分析资源利用率的模式。

第 5 章: 管理数据收集

使用其嵌入式数据收集器 **oacore**，HPE Operations Agent 可持续收集系统中的性能和运行状况数据，并将所收集的数据存储在度量数据存储区中。可以使用 extract 和 utility 等工具查看并分析记录的数据。可以将 HPE Operations Agent 与 HP Performance Manager 或 HP Reporter 等数据分析工具集成，以图形和报告的形式分析数据。

配置参数文件 **parm** 文件可用于配置**oacore**数据收集器的默认数据记录机制。可以使用 **parm** 文件控制数据记录间隔和所记录的数据的类型。

使用度量数据存储区

使用 HPE Operations Agent 版本 12.01，度量数据存储区会替换基于日志文件的数据存储区。多个数据存储区(如 CODA、Scope 和 DSI 日志文件)已合并到基于关系数据库管理系统 (RDBMS) 的数据存储区中。

将 HPE Operations Agent 11.xx 升级到 12.00 后，来自 HPE Operations Agent 11.xx 的数据(存储在 CODA 数据库文件、Scope 日志文件和 DSI 日志文件中)以只读模式保留，并保存在以下位置：

```
/var/opt/perf/datafiles/
```

可以通过诸如 ovcodautil、extract、utility 等实用程序或诸如 HP Performance Manager 和 HP Reporter 等报告工具访问旧数据。

备注:

- 来自 HPE Operations Agent 11.xx 的旧数据不会迁移到 度量数据存储区。
- 在 HP-UX AR 系统上，如果将 HPE Operations Agent 11.xx 升级到 12.xx，则来自 HPE Operations Agent 11.xx 的数据将保存在以下位置：

  ```
  /var/opt/perf/datafiles/
  ```

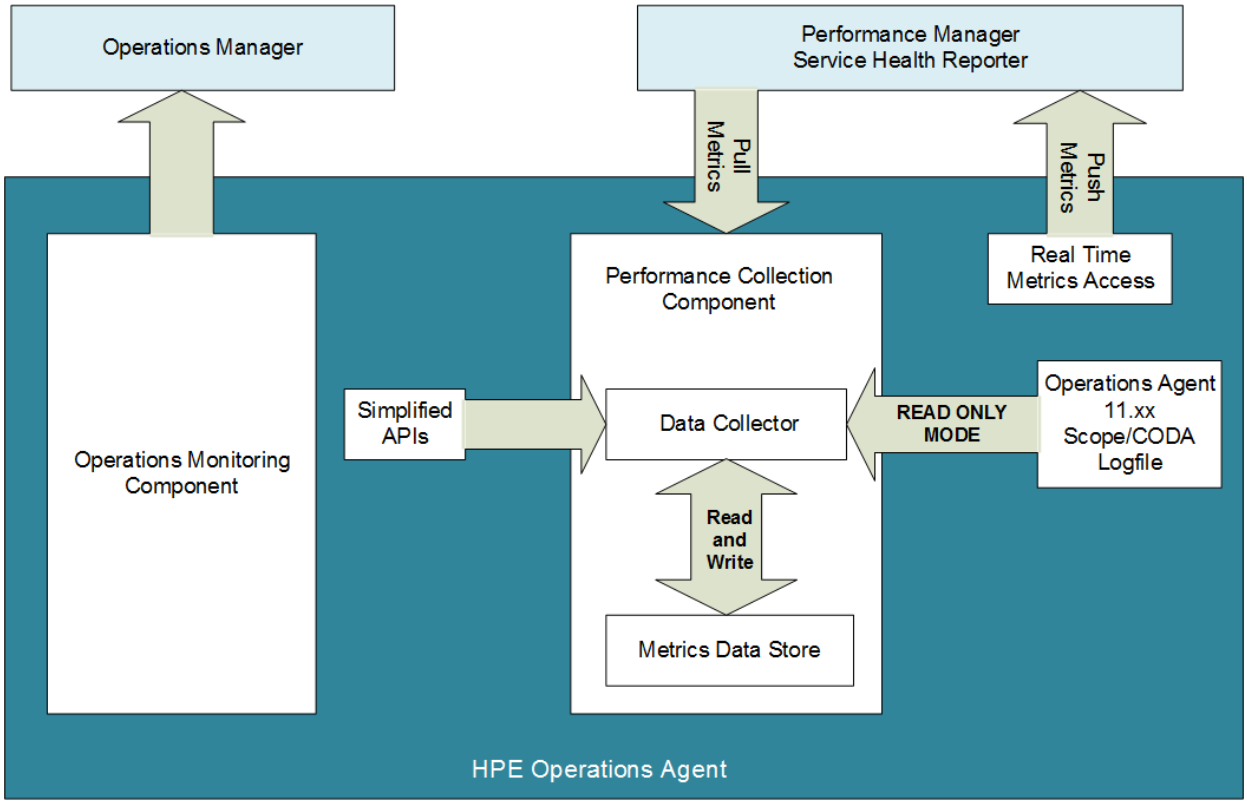
  可以使用诸如 ovcodautil、extract、utility 等工具或通过诸如 HP Performance Manager 和 HP Reporter 等报告工具访问此数据。
- 在 HP-UX IA 系统上，如果将 HPE Operations Agent 11.xx 升级到 12.xx，则来自 HPE Operations Agent 11.xx 的数据将保存在以下位置：

  ```
  /var/opt/OV/tmp/BackUp/
  ```

  无法使用诸如 ovcodautil、extract、utility 等工具或通过诸如 HP Performance Manager 和 HP Reporter 等报告工具访问此数据。
- 将 HPE Operations Agent 11.xx 升级到 12.xx 后，无法使用诸如 HP

Performance Manager 或 HP Reporter 等工具访问来自 HP Operations Agent 11.xx 的逻辑系统数据(保存在日志文件中)。

下图概述了 HPE Operations Agent 的新体系结构：



尽管数据存储和数据收集机制已更改，但策略间的阈值比较过程仍保持未变。

收集数据

使用当前版本的 HPE Operations Agent，可将 **CODA** 和 **Scope** 进程(UNIX 和 Linux 节点上为 **scopeux**，Windows 节点上为 **scopent**)合并到名为 **oacore** 的单个进程中。

oacore 数据收集器收集大量系统性能度量，以便您详细了解系统运行状况和性能。数据收集器捕获以下信息：

- 系统范围的资源利用率信息
- 进程数据
- 不同设备的性能数据
- 事务数据
- 逻辑系统数据

oacore 数据收集器收集基于 **parm** 文件中指定的数据类的数据。可在 **parm** 文件中定义的默认性能度量类为 global、application、process、disk device、lvolume、transaction、configuration、netif、CPU、filesystem 和 host bus adapter。

oacore 数据收集器收集的数据存储在特定于类的数据库文件中。对于记录到度量数据存储区中的每个数据类，将创建一个数据库文件。数据库文件可在 **datadir/databases/oa** 中找到。

验证 oacore 进程的状态

如果已从 HPOM 部署 HPE Operations Agent，则会自动启动 **oacore** 进程。如果已在节点上安装 HPE Operations Agent，则仅当节点上提供合适的许可证时才会启动 **oacore** 进程。有关许可证的详细信息，请参见《HPE Operations Agent 许可证指南》。

oacore 进程的状态将记录到位于以下位置的 `System.txt` 文件中：

在 Windows 上：

```
%ovdatadir%\log\System.txt
```

在 Linux 上：

```
/var/opt/OV/log
```

每次启动、停止 **oacore** 数据收集器或生成警告或错误时，会在此文件中追加新信息。

安装 HPE Operations Agent 后，执行以下步骤以检查 **oacore** 进程的状态：

1. 登录到 HPE Operations Agent 节点。
2. 运行以下命令：

   在 Windows X64 系统上：

   ```
   "%ovinstalldir%bin\win64\ovc"
   ```

   在 Windows X86 系统上：

   ```
   "%ovinstalldir%bin\win32\ovc"
   ```

   在 AIX 上：

   ```
   /usr/lpp/OV/bin/ovc
   ```

   在 HP-UX/Linux/Solaris/ 上：

   ```
   /opt/OV/bin/ovc
   ```

如果 **oacore** 进程正在运行，您将获得以下输出：

```
oacore      Operations Agent Core              AGENT,OA     (25357)
Running
```

启动 oacore 进程

如果 **oacore** 进程未在运行，请执行以下步骤启动 **oacore** 进程：

1. 登录到节点。
2. 转到以下目录：

   在 Windows X64 系统上：

   `%ovinstalldir%bin\win64\`

   在 Windows X86 系统上：

   `%ovinstalldir%bin`

   在 AIX 上：

   `/usr/lpp/OV/bin`

   在 HP-UX/Linux/Solaris 上：

   `/opt/OV/bin`
3. 运行以下命令启动进程：

   `ovc -start oacore`

注：要停止 **oacore** 进程，请运行以下命令：

`ovc -stop oacore`

备注: oacore 在超时之前很早就完成了数据访问请求。数据访问请求以到达的顺序完成。

如果请求大型数据(例如：2 百万个进程记录)，则 **oacore** 可能会需要比平常更长的时间来完成请求。当 **oacore** 正在处理大型数据访问请求时，其他请求(如 dsilog、extract)可能会超时。如果出现超时错误，则必须重新请求。

验证数据记录

oacore 数据收集器所收集的度量数据记录在度量数据存储区中。按照以下步骤验证数据记录：

1. 登录到节点。
2. 运行以下命令：

   在 Windows X64 系统上：

   `%ovinstalldir%bin\win64\ovcodautil -obj`

   在 Windows X86 系统上：

   `%ovinstalldir%bin\ovcodautil -obj`

   在 AIX 上：

```
/usr/lpp/OV/bin/ovcodautil -obj
```

在 HP-UX/Linux/Solaris 上：

```
/opt/OV/bin/ovcodautil -obj
```

如果正在记录数据，则将显示所有记录的度量的列表。

控制数据库文件所用的磁盘空间

性能收集组件具有自动管理数据库文件的功能。

控制数据库文件用来存储默认性能度量类的磁盘空间

存储默认性能度量类的数据库文件的大小取决于在 **parm** 文件中指定的最大大小。如果 **parm** 文件中的大小规范发生更改，**oacore** 仅在启动过程中才会检测到这种更改。

当达到 **parm** 文件中指定的最大大小时，会滚动更新类的数据库文件。在滚动更新期间，删除最旧的 20% 数据。如果 **parm** 文件中未指定大小，则在达到 1 GB 的最大大小时滚动更新数据库文件。

备注: 数据库文件的大小不能设置为低于 1 MB。

更改 parm 文件中指定的最大大小

对于记录到数据存储区的每一类数据，会创建 5 个数据库文件。这些数据库文件的大小取决于 **parm** 文件中指定的最大大小。

备注: 对于记录到数据存储区中的每个数据类，将创建一个数据库文件。当第一个数据库文件的大小接近 **parm** 文件中指定的最大大小的 20% 时，创建第二个数据库文件。此过程继续进行，直至创建了 5 个数据库文件。

如果 **parm** 文件中的大小规范发生更改，**oacore** 仅在启动过程中才会检测到这种更改。新大小仅对新的数据库文件有效。最终，随着在滚动更新过程中删除较旧的数据库文件，创建大小约为 **parm** 文件中所指定最大大小的 20% 的新数据库文件。

备注: 总大小可能不同于(可能大于或小于)配置的大小，因为新大小仅对新的数据库文件有效。随着在滚动更新过程中逐渐删除较旧的数据库文件，总大小将匹配 **parm** 文件中指定的大小。

例如：

如果在 **parm** 文件中将一类数据的最大大小指定为 10 MB，则创建 5 个文件，每个文件的大小为 2 MB。

在滚动更新期间，当达到 **parm** 文件中指定的最大大小 (10 MB) 时，删除最旧的数据库文件(大小为 2 MB)。

场景 1 - 将最大大小从 10 MB 减少为 5 MB：

必须重新启动性能收集组件以使更改生效。

所有现有数据库文件的大小继续为 2 MB。

最终，随着在滚动更新过程中删除较旧的数据库文件，创建最大大小为 1 MB 的新数据库文件。

删除所有旧文件(重新启动前存在)之后，将会创建 5 个数据库文件，且每个文件的最大大小为 1 MB。由此达到 **parm** 文件中指定的 5 MB 的最大大小。

注：总大小可能超过配置的大小，因为新大小仅对新的数据库文件有效。随着在滚动更新过程中逐渐删除较旧的数据库文件，总大小将匹配 **parm** 文件中指定的大小。

场景 2 - 将最大大小从 10 MB 增加为 15 MB：

必须重新启动性能收集组件以使更改生效。

所有现有数据库文件的大小继续为 2 MB。

最终，随着在滚动更新过程中删除较旧的数据库文件，创建最大大小为 3 MB 的新数据库文件。

删除所有旧文件(重新启动前存在)之后，将会创建 5 个数据库文件，且每个文件的最大大小为 3 MB。由此达到 **parm** 文件中指定的 15 MB 的最大大小。

注：总大小可能小于配置的大小，因为新大小仅对新文件有效。随着在滚动更新过程中逐渐删除较旧的文件，总大小将匹配 **parm** 文件中指定的大小。

控制存储自定义数据的数据库文件所用的磁盘空间

默认情况下，存储自定义数据的数据库文件的最大大小设置为 1 GB。无法配置此大小。

对于记录到数据存储区中的每个数据类，将创建一个数据库文件。由于第一个数据库文件大小达到约等于 1 GB 的 20% (200 MB)，因此将创建第二个数据库文件。继续此过程，直到创建 5 个数据库文件，每个文件大小为 200 MB。达到最大大小 1 GB 时，将滚动更新这些数据库文件。在滚动更新期间，将删除最旧数据中的 20%(最旧的数据库文件将永久删除)。

备注:

- 在 UNIX 系统上，**oacore** 进程需要足够的资源，如打开文件描述符。在大多数 UNIX 系统上，此限制配置为 256 (ulimit –n)。对于 **oacore** 进程，此限制足以在正常场景中运行。在配置为记录其他自定义类的系统上，必须成比例地配置打开文件描述符。

  例如：
  如果大约具有 20 个受管类，则建议将打开文件描述符配置为 512。
  如果大约具有 50 个受管类，则建议将打开文件描述符配置为 1024。

  配置打开文件描述符后，必须重新启动 **oacore** 进程。
- 在 Solaris 平台上，命令 `ovc -start oacore` 将启动 `/opt/perf/bin/runoacore` 脚本。在启动 **oacore** 进程之前，此脚本会将打开文件描述符的限制设置为 4096。

停止和重新启动数据收集

oacore 进程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)。(有关事务跟踪的详细信息，请参见什么是事务跟踪？)
- 正在更改 **parm** 文件并要使更改生效。对 **parm** 文件进行的更改仅在重新启动 **oacore** 进程后才生效。
- 正在关闭系统。
- 正在添加硬件或修改配置更改。进行的更改仅在重新启动 **oacore** 进程后才生效。

停止数据收集

ovpa 脚本的 stop 选项确保当 **oacore** 和其他性能收集组件进程停止时数据不会丢失。

要手动停止数据收集，请运行以下命令：

在 Windows 上：

```
%ovinstalldir%bin\ovpacmd stop
```

在 HP-UX/Linux/Solaris 上：

```
/opt/perf/bin/ovpa -stop
```

在 AIX 上：

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

备注：oacore 不记录 NFS 数据，但是可以通过本地文件系统上的 GlancePlus 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。

要在系统关闭或停止 **oacore** 和其他性能收集组件进程后启动它们，请使用 `<InstallDir>/ovpa start`。此处 InstallDir 是安装性能收集组件的目录。

要在 **oacore** 和其他进程正在运行时重新启动它们，请使用 `<InstallDir>/ovpa restart`。此处 InstallDir 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 **oacore**时，性能收集组件将继续使用在停止 **oacore** 进程之前使用的相同数据库文件。新记录追加到现有文件的末尾。

备注: ttd.conf 配置文件中的 SEM_KEY_PATH 条目用于为 UNIX 平台上的 ttd 和 midaemon 进程中使用的信号量生成 IPC 密钥。使用的默认值为 /var/opt/perf/datafiles。如果 midaemon 或 ttd 由于 sem id 冲突而无法响应，您可以更改 SEM_KEY_PATH 的值。

夏令时

在夏令时期间，如果系统时间前进一小时，数据记录会继续，不作任何更改。如果系统时间延后一小时，则这一小时内不会记录任何数据，直到系统时间与上次记录数据的时间戳同步为止。

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

如果系统时间前进一小时，数据记录会继续，不作任何更改。如果系统时间延后一小时，则这一小时内 **oacore** 停止记录，直到系统时间与上次记录数据的时间戳同步为止。

备注: 系统时间必须与数据收集时间同步，才能避免收集中出现任何缺口。

使用 parm 文件

parm 文件是包含 **oacore** 数据收集器如何记录特定性能数据的说明的文本文件。

在未安装以前版本的 HPE Operations Agent 的节点上安装 HPE Operations Agent 12.00 时，**parm** 文件位于两个不同的目录中。从早期版本的 HPE Operations Agent 升级到当前版本过程中，将升级位于其中一个目录中的 **parm** 文件。有关详细信息，请参见以下场景中 **parm** 文件的可用性：

- 安装 HPE Operations Agent 12.xx
- 升级到 HPE Operations Agent 12.xx

安装 HPE Operations Agent 12.xx

在未安装以前版本的 HPE Operations Agent 的节点上安装 HPE Operations Agent 12.xx 时，默认 **parm** 文件位于两个不同的目录中：

在 Windows 上：

```
%ovinstalldir%\newconfig
```

```
%ovdatadir%
```

备注: 在 Windows 上，**parm** 文件带有 .mwc (**parm.mwc**) 扩展名。

在 HP-UX/Linux/Solaris 上：

```
/opt/perf/newconfig
```

```
/var/opt/perf
```

在 AIX 上：

```
/usr/lpp/perf/newconfig
```

```
/var/opt/perf
```

oacore 数据收集器的数据收集机制受 %ovdatadir% (Windows) 和 **/var/opt/perf** (UNIX 或 Linux)目录下的 **parm** 文件的控制。

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

升级到 HPE Operations Agent 12.xx

在 Windows 上

在节点上升级 HPE Operations Agent 时，升级过程会更新 **newconfig** 目录下的 **parm** 文件的副本。驻留在 %ovdatadir% 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。此方法使您在升级后也能保留配置的数据收集机制。

本产品升级后，可以将 **parm** 文件的现有配置设置与 **newconfig** 目录下的 **parm** 文件的新版本进行比较，然后进行必需更改。有关修改 **parm** 文件的详细信息，请参见修改 parm 文件 (第 130 页)

在 Unix/Linux 上

在所有 UNIX/Linux 系统上，m4 宏处理器实用程序用于预处理 **parm** 文件。性能收集组件会预处理位于 **/var/opt/perf** 目录中的 **parm** 文件以创建属于文本文件的运行时 **parm** 文件。

可以使用以下命令生成运行时 **parm** 文件：

m4 -DPARMOS=<Operatingsystem>/var/opt/perf/parm > parm.m4

为 <Operatingsystem> 设置相应的值。

备注: <Operatingsystem> 的值区分大小写。要获取操作系统的相应值，请运行以下命令：

`uname`

例如：

如果在 Linux 操作系统上运行 `uname` 命令，将生成以下输出：

Linux

预先存在的 **parm** 文件(不带 m4 实用程序)适用于早期版本的 HPE Operations Agent。

系统在安装时检查是否存在 m4。如果 m4 不存在，请确保在系统上安装 m4。

有关详细信息，请参见《HPE Operations Agent 和 HPE Operations 基础结构 SPI 安装指南 (HPE Operations Agent and HPE Operations Smart Plug-in for Infrastructure Installation Guide)》中的“安装 HPE Operations Agent 的先决条件 (Prerequisites for Installing HPE Operations Agent)”。

备注: 可以将自定义应用程序定义添加到外部文件，并使用命令 #include (var/opt/perf/parm.apps) 将其包括在 **parm** 文件中。有关如何定义应用程序的详细信息，请参见应用程序定义参数 (第 150 页)

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` 参数必须遵循它们所定义的 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/Solaris 上：

```
/opt/perf/bin/ovpa -restart
```

备注: 可使用命令 `/opt/perf/bin/ovpa -restart scope` 重新启动性能收集组件。从早期版本升级到 HPE Operations Agent 12.xx 之后，保留此命令仅用于向后兼容性。

在 AIX 上：

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

备注: 可使用命令 `/usr/lpp/perf/bin/ovpa -restart scope` 重新启动性能收集组件。从早期版本升级到 HPE Operations Agent 12.xx 之后，保留此命令仅用于向后兼容性。

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 stop RTMA

%ovinstalldir%bin\ovpacmd start RTMA
```

在 HP-UX/Linux/Solaris 上：

```
/opt/perf/bin/pctl restart
```

在 AIX 上：

```
/usr/lpp/perf/bin/pctl restart
```

parm 文件参数

数据收集器 **oacore** 由收集参数文件 (**parm**) 中的特定参数控制，执行以下操作：

- 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。

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

备注: 仅当从系统启动、添加或删除设备时才会收集系统配置参数。

下表中列出的 **parm** 文件参数由 oacore 使用。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，请参见参数描述和应用程序定义参数。

oacore 使用的 parm 文件参数

参数	值或选项
参数描述	
id	系统 ID
log	global application [=all] - 可使 **oacore** 在每个间隔将所有应用程序 (无论应用程序是否是活动的) 记录到数据存储区中。 application [=prm](仅限 HP-UX) - 可使 **oacore** 将活动 Process Resource Manager (PRM) 组记录到数据存储区中。 process device=disk,lvm,cpu,filesystem,all (lvm 仅限 HP-UX 和 Linux) transaction=correlator, resource (resource 仅限 HP-UX)

oacore 使用的 parm 文件参数(续)

参数	值或选项
	备注: 从早期版本升级到 HPE Operations Agent 12.00 后，仅记录高级别事务类度量。 logicalsystem (对于 Solaris，仅 Solaris 10 操作环境或更高版本支持逻辑系统) 在 AIX 上，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
subprocinterval	以秒为单位的值(除 HP-UX 外)
javaarg	True|False
procthreshold (与 threshold 相同)	cpu=percent disk=rate(在非 Linux 或 Windows 上) memory=nn(值以 MB 为单位) IO=rate(值以 KB/秒为单位) nonew nokilled shortlived
appthreshold	cpu=percent
diskthreshold	util=rate
bynetifthreshold	iorate=rate

oacore 使用的 parm 文件参数(续)

参数	值或选项
fsthreshold	util=rate
lvthreshold	iorate=rate
bycputhreshold	cpu=percent
fstypes	要只包括特定文件系统来记录其数据，请使用语法 <file_system1>, <file_system2>, <file_system3>, ... 要排除文件系统，请使用语法 !<file_system>。
wait	cpu=percent(仅限 HP-UX) disk=percent(仅限 HP-UX) mem=percent(仅限 HP-UX) sem=percent(仅限 HP-UX) lan=percent(仅限 HP-UX)
size	大小(值以 MB 为单位) process=nn(最大值为 4096) 以下类的最大值为 2048： global=nn application=nn device=nn transaction=nn logicalsystem=nn
collectioninterval	process=ss(值以秒为单位) global=ss
gapapp	空白 unassignedprocesses existingapplicationname other
Flush	ss(值以秒为单位) 0(禁用数据刷新)
project_app **注：**仅限 Solaris	true false (仅限 Solaris 10 及更高版本)
proccmd	0(禁用进程命令记录)

oacore 使用的 parm 文件参数(续)

参数	值或选项
	1(启用进程命令记录) 记录的进程命令的最大长度始终为 4096。
proclist **注：** 仅限 Solaris	all(全局区域中的性能收集组件监视在全局和非全局区域中运行的所有进程) local(全局区域中的性能收集组件仅监视在全局区域中运行的进程) 此参数在非全局区域中不起作用。
appproc **注：** 仅限 Solaris	all(将性能收集组件配置为使用属于全局和非全局区域的应用程序的进程计算 APP_ 度量) local(将性能收集组件配置为仅使用属于全局区域的应用程序的进程计算 APP_ 度量) 此参数在非全局区域中不起作用。
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) **注：** 此参数在 HP-UX 上不起作用。必须在 HP-UX 上运行 **midaemon -ignore_mt** 命令以在上面的模式之间切换。有关详细信息，请参见记录使用基于内核的规范化算出的度量。
cachemem	f 或 free(当计算 GBL_MEM_UTIL 度量的值时，HPE Operations Agent 不包括缓冲区缓存大小) u 或 user(当计算 GBL_MEM_UTIL 度量的值时，HPE Operations Agent 包括缓冲区缓存大小)
应用程序定义参数	
application	应用程序名称
file	文件名 [, ...]
argv1	第一个命令参数 [,]

oacore 使用的 parm 文件参数(续)

参数	值或选项
cmd	命令行正则表达式
user	用户登录名称 [,]
group	组名 [,]
或	
priority	低值 - 高值 (范围因平台而异)

参数描述

以下是各个 **parm** 文件参数的描述：

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。此标识符包括在数据存储区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 **oacore** 要收集的数据类型。

- **log global** 可使 **oacore** 将全局记录记录到数据存储区中。在系统上，必须具有可查看和分析性能数据的全局数据记录。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** 可使 **oacore** 在数据存储区中中记录活动应用程序记录。默认情况下，**oacore** 只记录间隔期间有活动进程的应用程序。

  parm 文件中的参数 `log application=all` 可使 **oacore** 在每个间隔将所有应用程序(无论应用程序是否是活动的)记录到数据存储区中。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)。

  仅可以在 HP-UX 上，指定 `log application=prm` 参数使 **oacore** 将活动 Process Resource Manager (PRM) 组记录到数据存储区中。如果指定此参数，则 **oacore** 将不会记录 **parm** 文件中列出的用户定义的应用程序集。此外，收集的所有应用程序度量将反映 PRM 上下文，且将按 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** 可使 **oacore** 将有关感兴趣的进程的信息记录到数据存储区中。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 可使 **oacore** 将有关单个磁盘、逻辑卷(仅限 HP-UX 和 Linux)、CPU 和文件系统的信息记录到数据存储区中。

  备注: 仅在 HP-UX 或 Linux 操作系统上使用 `lvm`。

  默认情况下，仅记录在间隔期间生成 I/O 的磁盘、逻辑卷和接口。无论选定的记录设备选项是什么，都始终记录 netif(本地 LAN 设备)记录和磁盘记录。

  备注: 默认情况下，即使 **parm** 文件中未指定，也会记录 netif 记录。

  例如:

  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, netif
  ```

  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 允许 **oacore** 在每个间隔将所有磁盘、逻辑卷、CPU 和网络接口设备记录到数据存储区，无论设备是否处于活动状态。

- **log transaction** 可使 **oacore** 将 ARM 事务记录到数据存储区中。要使 **oacore** 收集数据，您的系统上必须运行已通过应用程序响应测量 (ARM) API 检测的进程。

  log transaction 参数的默认值为 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。

  备注: 当升级到 HPE Operations Agent 12.xx 后，记录事务支持 **oacore** 仅记录高级事务类度量。

- **log logicalsystems** 允许 **oacore** 将有关逻辑系统的信息记录到数据存储区。逻辑系统数据按 **parm** 文件中指定的间隔定期汇总。

  备注: Windows 和 Linux 平台不支持 BYLS 度量。

  HPE Operations Agent 12.xx 不会从 Xen、KVM、VMware vSphere、Hyper-V 和其他虚拟化域收集 BYLS 度量。

  在 AIX 6.1 TL2 上，可以使用 **parm** 文件中的 logicalsystems 参数来配置 LPAR 和 WPAR 的 BYLS 日志记录。有关详细信息，请参见oacore 使用的 parm 文件参数 (第 132 页)。

备注: 如果您指定的 log 参数不包含选项，则 **oacore** 仅记录全局和进程数据。

阈值

threshold 参数使 **oacore** 仅将严重信息记录到数据存储区中，过滤掉不必要不严重的系统详细信息。

某个数据类的特定实例超过指定阈值时，**oacore** 将记录该实例的记录。可为阈值指定更低的值使 **oacore** 记录更多的数据，也可为阈值指定更高的值使 **oacore** 记录更少的数据。

以下参数指定各种度量类的阈值：

- 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。 值 percent 是表示 CPU 总体使用情况的实数。 **例如：** 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`	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。

进程数据的 procthreshold 选项 (续)

	备注: 此选项在 Linux 和 Windows 上不可用。 **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。
IO	设置 IO 阈值，进程必须超过该值才能成为“感兴趣的”进程并被记录。 此值以 KB 为单位。 **例如:** IO=100 表示进程 I/O 率 (PROC_IO_BYTE_RATE) 超过每秒 100 KB 时记录进程。
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进程。
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已终止进程。
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 **oacore** 在 **parm** 文件中找到阈值 shortlived，只要删除 nonew 和 nokilled 选项，它就将记录 **shortlived** 进程，而不管 CPU 或磁盘的阈值如何。 默认情况下，不记录任何 shortlived 进程。

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。
- 虚拟内存集大小在 900 MB 以上的进程。
- 平均物理磁盘 I/O 速率大于每秒 5 KB 的进程。

appthreshold

appthreshold 参数用于指定 APPLICATION 数据类(APP_CPU_TOTAL_UTIL 度量)的阈值。阈值条件基于应用程序必须超过才能在数据存储区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 **oacore** 记录 CPU 利用率超过 0% 的应用程序。

diskthreshold

diskthreshold 参数指定 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 **oacore** 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 **oacore** 记录每秒传输超过 60 个数据包的网络接口的详细信息。如果不指定此参数的值或注释掉此参数，**oacore** 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使用的磁盘空间百分比(FS_SPACE_UTIL 度量)。

parm 文件中的默认设置使 **oacore** 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 **oacore** 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 数据类阈值条件基于 CPU 繁忙时间百分比 (BYCPU_CPU_TOTAL_UTIL)。

parm 文件中的默认设置使 **oacore** 记录繁忙时间超过 90% 的 CPU 的详细信息。

subprocinterval

subprocinterval 参数(如果指定)覆盖 **oacore** 用于对进程数据进行采样的默认值。

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，**oacore** 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，请参见配置数据记录间隔 (第 156 页)。

在一些具有数千个活动线程或进程的系统上，应当将 subprocinterval 设置为更长以降低总体 **oacore** 开销。在具有大量您要记录的短期进程的其他系统上，可以考虑将 subprocinterval 设置为较低，但是在此情况下应当严密监视对 **oacore** 开销的影响。此设置的值必须是 **parm** 文件中指定的进程记录间隔的乘数。

备注: 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，则名为 gapapp 的应用程序将添加到应用程序列表。
- 如果 gapapp = UnassignedProcesses，则名为 UnassignedProcesses 的应用程序将添加到应用程序列表。
- 如果 gapapp = ExistingApplicationName(或)gapapp = other，则与全局值的差异将添加到指定的应用程序而不会单独记录，并且一个新条目会添加到应用程序列表。

fstypes

fstypes 参数可用于监视系统上的特定文件系统。默认情况下，**oacore** 收集器会记录附加到系统的所有文件系统的数据。使用 fstypes 参数，可以仅针对特

定文件系统启用数据收集。

在 **parm** 文件中，必须将 fstypes 参数设置为要监视的文件系统的名称。可以指定由逗号分隔的多个文件系统名称。

此参数的语法为：

```
fstypes=<file_system1>,<file_system2>, ...
```

<file_system1> 和 <file_system2> 是文件系统类型。

对于 HP-UX、Linux 和 AIX，**/etc/fstab** 文件中列出了所有可用文件系统以及文件系统类型。

对于 AIX，**/etc/filesystems** 文件中列出了所有可用文件系统以及文件系统类型。

对于 Solaris，**/etc/vfstab** 文件中列出了所有可用文件系统以及文件系统类型。

要查明 Windows 上的文件系统类型，应在 Windows 资源管理器中右键单击磁盘驱动器，然后单击**属性**。为“文件系统”显示的值是可用 fstypes 参数指定的文件系统类型。

可以将此参数设置为包含列表或排除列表。以逗号分隔的文件系统类型的列表表示包含列表，使代理程序能够仅监视指定文件系统的数据。

在配置 fstypes 参数时，使用操作系统命令所返回的文件系统名称。您也可以将此参数设置为排除列表，即以！字符开头的文件系统类型的逗号分隔列表。在列表的开头指定！字符可确保代理程序不监视属于该列表的任何文件系统。

示例 1：

```
fstypes = tmpfs, mvfs, nfs
```

示例 2：

```
fstypes = !tmpfs, mvfs
```

示例 3：

```
fstypes =
```

或

```
fstypes = *
```

指定 * 或 blank 字符可确保 HPE Operations Agent 监视所有可用的文件系统。

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 的间隔百分比大于为 wait 参数指定的值时，则在数据存储区中记录该进程的详细信息。有关值和选项，请参见 oacore 使用的 parm 文件参数 (第 132 页)。

例如：

如果进程记录间隔定义为 60 秒且 CPU 的 `wait` 参数设置为 50%，则在数据存储区中捕获所有等待 CPU 时间大于等于 30 秒的进程。

Size

`size` 参数用于设置数据库文件的最大大小(以 MB 为单位)。**oacore** 收集器在初始化时读取这些规范。

达到 **parm** 文件中指定的最大大小时，存储在默认性能度量类中的数据库文件将进行转存。如果未在 **parm** 文件中指定大小，达到最大大小 1 GB 时，将滚动更新这些数据库文件。

默认情况下，存储自定义数据的数据库文件的最大大小设置为 1 GB。达到最大大小 1 GB 时，将滚动更新这些数据库文件。

有关控制数据库文件使用的磁盘空间的详细信息，请参见控制数据库文件所用的磁盘空间。

javaarg

`javaarg` 参数是仅影响 **proc_proc_argv1** 度量的值的标记。此参数默认设置为 true。

`javaarg` 为 true 时，使用类或 jar 名称记录 proc_proc_argv1 度量。此设置对定义特定于 Java 的应用程序很有用。如果某个进程的命令字符串中存在类名称，可以使用“argv1 = 应用程序限定符”按类名称定义应用程序。

当 `javaarg` 设置为 false 时或如果未在 **parm** 文件中定义该参数，则使用该进程的命令字符串中的第一个参数的值来记录 proc_proc_argv1 度量。

例如：

执行以下进程时：

```
java -XX:MaxPermSize=128m -XX:+CMSClassUnloadingEnabled -Xms8000m -
Xmx8000m -Dserver_port=1099 -jar ./ApacheJMeter.jar
```

- 如果 javaarg 设置为 **True**，则 proc_proc_argv1 的值将为 -jar ./ApacheJMeter.jar
- 如果 javaarg 设置为 **False**，则 proc_proc_argv1 的值将为 -XX:MaxPermSize=128m

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 间隔的默认值为 3600 秒。

您可以通过将值指定为 0(零)来禁用 flush 参数。如果 flush 参数设置为 0，**oacore** 将不会记录违反 **parm** 文件中指定的阈值的应用程序和设备数据。

project_app

如果将 project_app 参数设置为 True，则性能收集组件将每个 Solaris 项目视为应用程序(并将项目 ID 视为应用程序 ID)。要忽略 Solaris 项目，请将此参数设置为 False。

备注: 仅 Solaris 10 及更高版本支持 project_app 参数。

proclist

只能在 Solaris 全局区域中使用此参数；对在非全局区域中运行的 HPE Operations Agent 没有任何影响。

在全局区域中，如果将此参数设置为 all，则性能收集组件监视所有全局和非全局区域进程。要仅监视属于全局区域的进程，请将此参数设置为 local。

appproc

appproc 参数仅在 Solaris 上可用。只能在全局区域中使用此参数；对在非全局区域中运行的 HPE Operations Agent 没有任何影响。

在全局区域中，如果将此参数设置为 **all**，则在计算所有 APP_ 度量的值时，性能收集组件 将包括针对全局和非全局区域应用程序的进程。要仅包括全局区域应用程序来计算 APP_ 度量，请将此参数设置为 **local**。

proccmd

proccmd 参数允许将进程命令记录到数据存储区。默认情况下，此参数值设置为 0，表示禁止记录进程命令。要允许记录进程命令，请将此参数值设置为 1。

注：当此参数值大于等于 1 时，启用 proccmd 参数记录。记录的进程命令长度始终为 4096，与此参数中指定的值无关。

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。

在 Linux 计算机上，此参数默认设置为 false。

此参数在 HP-UX 上不起作用。必须在 HP-UX 上运行 `midaemon -ignore_mt` 命令以在模式之间切换。有关详细信息，请参见记录使用基于内核的规范化算出的度量 (第 162 页)。

当在 HP-UX 系统上将 `ignore_mt` 标记作为命令行参数设置为 `midaemon` 时，仅影响很少的度量。有关受影响的度量列表的详细信息，请参见parm 文件 (第 328 页)。

如果系统上已禁用多线程性能，则性能收集组件将忽略此参数。因此，GBL_IGNORE_MT 度量的值记录为 **True**。

备注: 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。

在 HP-UX 系统上将 ignore_mt 标记设置为 midaemon 时影响的度量

全局度量类	
GBL_CPU_TOTAL_UTIL	GBL_CPU_CSWITCH_UTIL_CUM
GBL_CPU_TOTAL_UTIL_CUM	GBL_CPU_CSWITCH_UTIL_HIGH
GBL_CPU_TOTAL_UTIL_HIGH	GBL_CPU_CSWITCH_TIME
GBL_CPU_TOTAL_TIME	GBL_CPU_CSWITCH_TIME_CUM
GBL_CPU_TOTAL_TIME_CUM	GBL_CPU_INTERRUPT_UTIL
GBL_CPU_SYS_MODE_UTIL	GBL_CPU_INTERRUPT_UTIL_CUM
GBL_CPU_SYS_MODE_UTIL_CUM	GBL_CPU_INTERRUPT_UTIL_HIGH
GBL_CPU_SYS_MODE_TIME	GBL_CPU_INTERRUPT_TIME
GBL_CPU_SYS_MODE_TIME_CUM	GBL_CPU_INTERRUPT_TIME_CUM
GBL_CPU_TRAP_TIME	GBL_CPU_VFAULT_TIME

GBL_CPU_TRAP_TIME_CUM	GBL_CPU_VFAULT_TIME_CUM
GBL_CPU_TRAP_UTIL	GBL_CPU_VFAULT_UTIL
GBL_CPU_TRAP_UTIL_CUM	GBL_CPU_VFAULT_UTIL_CUM
GBL_CPU_TRAP_UTIL_HIGH	GBL_CPU_VFAULT_UTIL_HIGH
GBL_CPU_USER_MODE_TIME	GBL_CPU_IDLE_UTIL
GBL_CPU_USER_MODE_TIME_CUM	GBL_CPU_IDLE_UTIL_CUM
GBL_CPU_USER_MODE_UTIL	GBL_CPU_IDLE_UTIL_HIGH
GBL_CPU_USER_MODE_UTIL_CUM	GBL_CPU_IDLE_TIME
GBL_CPU_NICE_UTIL	GBL_CPU_IDLE_TIME_CUM
GBL_CPU_NICE_UTIL_CUM	GBL_CPU_NORMAL_UTIL
GBL_CPU_NICE_UTIL_HIGH	GBL_CPU_NORMAL_UTIL_CUM
GBL_CPU_NICE_TIME	GBL_CPU_NORMAL_UTIL_HIGH
GBL_CPU_NICE_TIME_CUM	GBL_CPU_NORMAL_TIME
GBL_CPU_NNICE_UTIL	GBL_CPU_NORMAL_TIME_CUM
GBL_CPU_NNICE_UTIL_CUM	GBL_CPU_SYSCALL_UTIL
GBL_CPU_NNICE_UTIL_HIGH	GBL_CPU_SYSCALL_UTIL_CUM
GBL_CPU_NNICE_TIME	GBL_CPU_SYSCALL_UTIL_HIGH
GBL_CPU_NNICE_TIME_CUM	GBL_CPU_SYSCALL_TIME
GBL_CPU_REALTIME_UTIL	GBL_CPU_SYSCALL_TIME_CUM
GBL_CPU_REALTIME_UTIL_CUM	GBL_CPU_WAIT_UTIL
GBL_CPU_REALTIME_UTIL_HIGH	GBL_CPU_WAIT_TIME
GBL_CPU_REALTIME_TIME	GBL_CPU_WAIT_TIME_CUM
GBL_CPU_REALTIME_TIME_CUM	GBL_CPU_WAIT_UTIL_CUM
GBL_CPU_CSWITCH_UTIL	GBL_CPU_WAIT_UTIL_HIGH
应用程序度量类	

APP_PRM_CPU_TOTAL_UTIL_CUM	APP_CPU_NORMAL_UTIL
APP_CPU_NNICE_UTIL	APP_CPU_USER_MODE_UTIL
APP_CPU_NNICE_TIME	APP_CPU_TOTAL_TIME
APP_CPU_TOTAL_UTIL	APP_CPU_SYS_MODE_TIME
APP_CPU_TOTAL_UTIL_CUM	APP_CPU_NICE_TIME
APP_CPU_SYS_MODE_UTIL	APP_CPU_REALTIME_TIME
APP_CPU_NICE_UTIL	APP_CPU_NORMAL_TIME
APP_CPU_REALTIME_UTIL	APP_CPU_USER_MODE_TIME
PROC 度量类	
PROC_CPU_TOTAL_UTIL	PROC_CPU_REALTIME_UTIL
PROC_CPU_TOTAL_UTIL_CUM	PROC_CPU_REALTIME_UTIL_CUM
PROC_CPU_TOTAL_TIME	PROC_CPU_REALTIME_TIME
PROC_CPU_TOTAL_TIME_CUM	PROC_CPU_REALTIME_TIME_CUM
PROC_CPU_SYS_MODE_UTIL	PROC_CPU_CSWITCH_UTIL
PROC_CPU_SYS_MODE_UTIL_CUM	PROC_CPU_CSWITCH_UTIL_CUM
PROC_CPU_SYS_MODE_TIME	PROC_CPU_CSWITCH_TIME
PROC_CPU_SYS_MODE_TIME_CUM	PROC_CPU_CSWITCH_TIME_CUM
PROC_CPU_USER_MODE_UTIL	PROC_CPU_INTERRUPT_UTIL
PROC_CPU_USER_MODE_UTIL_CUM	PROC_CPU_INTERRUPT_UTIL_CUM
PROC_CPU_USER_MODE_TIME	PROC_CPU_INTERRUPT_TIME
PROC_CPU_USER_MODE_TIME_CUM	PROC_CPU_INTERRUPT_TIME_CUM
PROC_CPU_NICE_UTIL	PROC_CPU_NORMAL_UTIL
PROC_CPU_NICE_UTIL_CUM	PROC_CPU_NORMAL_UTIL_CUM
PROC_CPU_NICE_TIME	PROC_CPU_NORMAL_TIME
PROC_CPU_NICE_TIME_CUM	PROC_CPU_NORMAL_TIME_CUM

PROC_CPU_NNICE_UTIL	PROC_CPU_SYSCALL_UTIL
PROC_CPU_NNICE_UTIL_CUM	PROC_CPU_SYSCALL_UTIL_CUM
PROC_CPU_NNICE_TIME	PROC_CPU_SYSCALL_TIME
PROC_CPU_NNICE_TIME_CUM	PROC_CPU_SYSCALL_TIME_CUM
PROC_CPU_ALIVE_TOTAL_UTIL	
PROC_CPU_ALIVE_USER_MODE_ UTIL	
PROC_CPU_ALIVE_SYS_MODE_UTIL	
BYCPU 度量类	
BYCPU_CPU_TOTAL_UTIL	BYCPU_CPU_INTERRUPT_TIME
BYCPU_CPU_TOTAL_UTIL_CUM	BYCPU_CPU_INTERRUPT_TIME_ CUM
BYCPU_CPU_TRAP_TIME	BYCPU_CPU_INTERRUPT_UTIL
BYCPU_CPU_TRAP_TIME_CUM	BYCPU_CPU_INTERRUPT_UTIL_ CUM
BYCPU_CPU_TRAP_UTIL	BYCPU_CPU_CSWITCH_TIME
BYCPU_CPU_TRAP_UTIL_CUM	BYCPU_CPU_CSWITCH_TIME_CUM
BYCPU_CPU_USER_MODE_TIME	BYCPU_CPU_CSWITCH_UTIL
BYCPU_CPU_USER_MODE_TIME_ CUM	BYCPU_CPU_CSWITCH_UTIL_CUM
BYCPU_CPU_USER_MODE_UTIL	BYCPU_CPU_VFAULT_TIME
BYCPU_CPU_USER_MODE_UTIL_ CUM	BYCPU_CPU_VFAULT_TIME_CUM
BYCPU_CPU_NICE_TIME	BYCPU_CPU_VFAULT_UTIL
BYCPU_CPU_NICE_TIME_CUM	BYCPU_CPU_VFAULT_UTIL_CUM
BYCPU_CPU_NICE_UTIL	BYCPU_CPU_REALTIME_TIME
BYCPU_CPU_NICE_UTIL_CUM	BYCPU_CPU_REALTIME_TIME_CUM

BYCPU_CPU_NNICE_TIME	BYCPU_CPU_REALTIME_UTIL
BYCPU_CPU_NNICE_TIME_CUM	BYCPU_CPU_REALTIME_UTIL_CUM
BYCPU_CPU_NNICE_UTIL	BYCPU_CPU_NORMAL_TIME
BYCPU_CPU_NNICE_UTIL_CUM	BYCPU_CPU_NORMAL_TIME_CUM
BYCPU_CPU_TOTAL_TIME	BYCPU_CPU_NORMAL_UTIL
BYCPU_CPU_TOTAL_TIME_CUM	BYCPU_CPU_NORMAL_UTIL_CUM
BYCPU_CPU_SYS_MODE_TIME	BYCPU_CPU_SYSCALL_TIME
BYCPU_CPU_SYS_MODE_TIME_CUM	BYCPU_CPU_SYSCALL_TIME_CUM
BYCPU_CPU_SYS_MODE_UTIL	BYCPU_CPU_SYSCALL_UTIL
BYCPU_CPU_SYS_MODE_UTIL_CUM	BYCPU_CPU_SYSCALL_UTIL_CUM
SYSCALL 度量类	
SYSCALL_CPU_TOTAL_TIME	SYSCALL_CPU_TOTAL_TIME_CUM

cachemem

使用 **parm** 文件中的 `cachemem` 参数，可以将代理程序配置为在报告总内存利用率数据时包括缓冲区缓存大小。

可以将 `cachemem` 参数设置为以下某个值：

1. **将 cachemem 参数设置为 free (f)。**
   默认情况下，该参数设置为 free (f)。以下度量值受影响：
   GBL_MEM_UTIL - 不包括缓冲区缓存大小。
   GBL_MEM_FREE_UTIL - 包括缓冲区缓存大小。
2. **将 cachemem 参数设置为 user (u)。**
   以下度量值受影响：
   GBL_MEM_UTIL - 包括缓冲区缓存大小。
   GBL_MEM_FREE_UTIL - 不包括缓冲区缓存大小。

备注：

在 AIX 计算机上，应当将 cachemem 参数设置为 **free (f)** 以匹配 topas 命令。

在 Solaris 计算机上，cachemem 参数仅适用于 ZFS 的自适应替换缓存 (ARC)。

cachemem 参数默认设置为 **user (u)**。可以在 parm 文件中将此参数设置为 **free (f)**，然后重新启动数据收集器。在 parm 文件中将 cachemem 参数设置为 **free** 时，GBL_MEM_UTIL 不包括 ZFS ARC 缓存。

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。

备注: 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：

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。

备注: 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。

应用程序名称

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。

指定应用程序时，以下规则或约定适用：

- 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** 文件中多次使用同一应用程序名称。
- 等号 (=) 是应用程序关键字和应用程序名称之间的可选符号。
- 应用程序名称必须在引用它的任意 file、user、group、cmd、argv1 和 or 参数组合之前，所有此类参数都针对最后的应用程序工作负载定义来应用。
- 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。如果文件列表字符数大于 170，请在另一个 file、user、group、cmd 或 argv1 语句后的下一行继续扩展列表。
- 最多可以定义 998 个应用程序。

- 最多可以将 4096 个 file、user、group、argv1 和 cmd 规范用于合并的应用程序。
- 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 语句不捕获的所有进程。

例如:

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

application xyz
file xyz*,startxyz
```

其中:

xyz* 只算作一条规范,即使它可以与多个程序文件匹配。

备注: 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** 目录还包含您可以根据需要在 **parm** 文件中复制和修改的其他示例(这些示例位于名为 **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。如果找不到 application 语句,则生成错误。

文件名可以是以下任意一项:

- **在 UNIX/Linux 上:**
  - 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。
- **在 Windows 上:**
  - 单个程序文件,比如 winword。
  - 一组程序文件(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所有文件规范最多为 1000 条中只算作一条规范。

备注: 对于 Windows,为 parm 文件中的应用程序定义可执行文件时,不需要

文件扩展名。例如，您可以定义 parm 文件中的 winword，而无需 .exe 扩展名。

file 参数中的名称长度限制为 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。将文件规范与设置为进程的 argv[0] 值(通常是其基本名称)的特定度量 PROC_PROC_NAME 进行比较。

例如：

在 UNIX/Linux 上：

```
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
```

在 Windows 上：

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

如果未指定 file 参数，则所有满足其他参数的程序都将适用。

备注: 星号 (*) 是除 cmd 限定符之外 **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(javaarg 为 **True** 时以及它是 Java 进程的类或 jar 名称时除外)。

argv1 参数使用 **parm** 参数所用的相同模式匹配语法，例如 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：

以下应用程序定义存储命令行中的第一个参数为 -title、-fn 或 -display 的所有进程：

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

以下应用程序定义存储特定 Java 应用程序(当 javaarg=true 时)：

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

以下示例演示 argv1 参数如何与 file 参数合并：

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

cmd

cmd 参数通过命令字符串指定要包括在应用程序中的进程。该命令字符串包含所执行的程序及其参数。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。此参数每行只能有一个选择，但是可以有多行。

以下示例演示 cmd 参数的用法：

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

用户

user 参数指定哪些用户(登录名称)属于应用程序。

格式为：

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

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
```

如果在不指定域名的情况下指定用户名，则使用本地系统的所有用户名：

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。

如果未指定 user 参数，则所有满足其他参数的程序都将适用。

user 参数中的名称长度限制为 15 个字符。

Group

group 参数指定属于应用程序的用户组名称。

例如：

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

如果未指定 group 参数，则所有满足其他参数的程序都将适用。

group 参数中的名称长度限制为 15 个字符。

或

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。or 参数分隔的参数被视为独立定义。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
或
user mark
file vi, store, dmp
```

它定义由用户 Julie 运行的任何程序和其他程序 vi、store、dmp(如果它们由用户 Mark 执行)组成的 (Prog_Dev_Group2) 应用程序。

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
```

进程的优先级范围从 -511 到 255，具体取决于运行 HPE Operations Agent 的平台。可在进程生命周期中更改优先级。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。

备注: 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*
或
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

oacore 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。可以使用 **parm** 文件中的 collection interval 参数覆盖此默认间隔。

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(参见 subprocinterval)的倍数，且它必须等分为全局收集间隔。
- 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)。

为框架配置数据收集

您可以通过只在一个 LPAR 节点上安装 HPE Operations Agent 来从单个框架上可用的所有 AIX LPAR 中收集性能数据。也可以将代理程序配置为从所有监视的 LPAR 所在的 AIX 框架中收集性能数据。

如果启用对框架的监视功能，可以获得以下优势：

- 收集配置信息：
  - 监视的 LPAR 所在的框架的名称和 UUID
  - 框架的型号、序列号和类型
  - 框架的 CPU 配置和内存容量
- 借助其他信息，可以分析框架的资源利用率
- 使用数据分析工具(如 HP Performance Manager)分析框架和所有 LPAR 的 CPU 占用率

任务 1：配置无密码 SSH 访问

在硬件管理控制台 (HMC) 系统上，可以在 LPAR 节点(安装了代理程序的节点)和 HMC 系统之间配置无密码 SSH 访问。为用户配置对无密码 SSH 的访问权限后，可以在 HMC 系统上使用 SSH 协议从 LPAR 节点远程运行命令，而无需提供密进行身份验证。

执行以下步骤：

配置无密码 SSH 访问

在 LPAR 节点上

1. 以根用户身份登录到 LPAR 节点。
2. 运行以下命令创建目录:

   ```
   # mkdir .ssh
   ```

   备注: 可以运行以下命令删除 .ssh 文件 `# rm -rf .ssh`
3. 运行以下命令生成 RSA 公钥/私钥对:

   ```
   # ssh-keygen -t rsa
   ```
4. 指定一个文件以保存该密钥:

   ```
   //.ssh/<filename>_rsa
   ```

   私钥将保存在 `//.ssh/ <filename>_rsa` 中，公钥将保存在 `//.ssh/ <filename>_rsa.pub` 中。
5. 运行以下命令查看密钥:

   ```
   # cat ./ssh
   ```

   ```
   # cat .ssh/<filename>_rsa.pub
   ```

将显示生成的密钥。

在 HMC 计算机上

1. 使用用户凭据登录到 HMC 计算机。
2. 运行以下命令设置无密码访问:

   ```
   mkauthkeys -a 'ssh-rsa<key>’
   ```

   在此实例中，<key> 是步骤 3 中生成的密钥。

   备注: 整个密钥必须列在一行中。

验证无密码 SSH 访问

执行以下步骤以验证是否已为用户创建无密码 SSH 访问:

1. 以根用户身份登录到 LPAR 节点。
2. 运行以下命令:

   ```
   ssh hmcuser@hmchost.example.domain.com lssyscfg r sys
   ```

如果该命令在不提示输入密码的情况下显示框架列表，则用户已正确配置，并且可以监视框架利用率。

任务 2：在 HMC 系统上启用框架利用率监视

1. 以根用户身份登录到 HMC 系统。
2. 运行以下命令:

```
chlparutil -r config -s 60
```

任务 3：配置 HPE Operations Agent

1. 登录到 LPAR 节点。
2. 转到 `/var/opt/perf` 目录。
3. 创建一个新文件。
4. 使用文本编辑器打开该文件，然后向该文件添加以下内容：

   <hmcusername>@<hmc_fqdn>

   在此实例中：

   <hmcusername> 是在任务 1：配置无密码 SSH 访问 (第 156 页)中授予了 HMC 系统无密码 SSH 访问权限的用户。

   <hmc_fqdn> 是 HMC 主机的完全限定域名。

   例如：

   ```
   hmcusername@hmchost.example.domain.com
   ```

5. 在 `/var/opt/perf` 目录中将该文件另存为 `hmc`。
6. 在 `parm` 文件中配置 `logicalsystem` 参数并启动收集进程。有关配置 `parm` 文件的详细信息，请参见使用 parm 文件 (第 129 页)。
7. 重新启动性能收集组件。

   HPE Operations Agent 将每隔五分钟收集一次特定于框架的性能数据。

在 Linux 上为 GlancePlus 启用全局度量和进程系统调用度量

在 Linux 计算机上，可以为 GlancePlus 启用 GBL_SYSCALL、SYSCALL 和 PROCSYSCALL 性能度量值的收集。开始配置度量收集前，确保停止在 Linux 计算机上使用 FTRACE 的所有进程。

备注： RHEL/OEL/CentOS 6.1 及以上版本、SLES 11 SP1 及以上版本、Ubuntu 11.10 及以上版本、Debian 6.0 及以上版本均支持这些度量。

可以按照以下方法之一为 GlancePlus 启用度量收集：

- 使用 `init_ftrace.sh` 配置度量收集
- 使用手动步骤配置度量收集

使用 init_ftrace.sh 配置度量收集

要在 Linux 计算机上使用 `init_ftrace.sh` 配置度量收集，请执行以下步骤：

1. 以具有必要特权的身份登录到节点。
2. 要在 Linux 计算机上停止所有性能收集组件进程，请运行以下命令：

   ```
   /opt/perf/bin/ovpa stop all
   ```
3. 在 Linux 计算机上停止所有 midaemon 进程。要检查 midaemon 进程，请运行以下命令：

   ```
   ps -ef I grep midaemon
   ```

   如果 midaemon 进程正在 Linux 计算机上运行，请运行以下命令：

   ```
   killall midaemon
   ```

   这会停止所有 midaemon 进程。
4. 要配置度量收集，请运行以下命令：

   ```
   /opt/perf/bin/init_ftrace.sh
   ```

   该命令会在 Linux 计算机上装入 debugfs 并启用 FTRACE。如果 FTRACE 已与任何其他应用程序正在您的计算机上运行，则消息显示为：

   Do you want to reset the FTRACE interface for use with midaemon?(Y/N).

   可以选择以下选项之一：

   Y – 在 Linux 计算机上重置 FTRACE 界面。

   N – 不会配置度量收集。无法在 GlancePlus 中查看度量。
5. 要启动所有性能收集组件进程以及 midaemon 进程和度量收集，请运行以下命令：

   ```
   /opt/perf/bin/ovpa start all
   ```

使用 GlancePlus 中的以下屏幕检查性能度量：

Y - 检查 SYSCALL 度量

L - 检查 PROCSYSCALL 度量

备注: GlancePlus 不在“系统调用 (System Call)”屏幕中时，即使已装入 debugfs 且已启用 FTRACE，也不会影响系统的性能。要从 **/sys/kernel/debug** 卸载 debugfs，可以运行命令 `umount /sys/kernel/debug`。

使用手动步骤配置度量收集

要在 Linux 计算机上手动配置度量收集，请执行以下步骤：

1. 以具有必要特权的身份登录到节点。
2. 要在 Linux 计算机上停止所有性能收集组件进程，请运行以下命令：

   ```
   /opt/perf/bin/ovpa stop all
   ```
3. 在 Linux 计算机上停止所有 midaemon 进程。要检查 midaemon 进程，请运行以下命令：

```
ps -ef I grep midaemon
```

如果 midaemon 进程正在 Linux 计算机上运行，请运行以下命令：

```
killall midaemon
```

这会停止所有 midaemon 进程。

4. 手动在 Linux 计算机上安装 debugfs

备注: 如果 Linux 计算机上已安装 debugfs，则跳过此步骤。要检查是否已安装 debugfs，请运行以下命令：

```
cat /proc/mounts |grep debugfs
```

命令生成以下任一输出：

nodev /sys/kernel/debug debugfs rw,relatime 0 0 - 计算机上已安装 debugfs。

nil - 计算机上未安装 debugfs。

要手动安装 debugfs，请执行以下步骤：

i. 运行以下命令：

```
mount -t debugfs nodev /sys/kernel/debug
```

输出将显示为

nil - 计算机上已成功安装 debugfs。

如果计算机上未成功安装 debugfs，则显示失败消息。

5. 手动在 Linux 计算机上启用 FTRACE

备注: 如果 Linux 计算机上已启用 FTRACE，则跳过此步骤。要检查是否已启用 FTRACE，请运行以下命令：

```
cat /proc/sys/kernel/ftrace_enabled
```

命令生成以下任一输出：

1 - 计算机上已启用 FTRACE。

0 - 计算机上未启用 FTRACE。

要手动启用 FTRACE，请执行以下步骤：

i. 运行以下命令：

```
echo "1" >/proc/sys/kernel/ftrace_enabled
```

输出将显示为

nil - 计算机上已成功启用 FTRACE

如果计算机上未成功启用 FTRACE，则显示失败消息。

6. 要启动所有性能收集组件进程以及 midaemon 进程和度量收集，请运行以下

命令：

```
/opt/perf/bin/ovpa start all
```

使用 GlancePlus 中的以下屏幕检查性能度量：

Y - 检查 SYSCALL 度量

L - 检查 PROCSYSCALL 度量

备注: GlancePlus 不在“系统调用 (System Call)”屏幕中时，即使已装入 debugfs 且已启用 FTRACE，也不会影响系统的性能。要从 **/sys/kernel/debug** 卸载 debugfs，可以运行命令 `umount /sys/kernel/debug`。

疑难解答

本部分描述配置度量收集时遇到的常见问题的解决方法或变通方法。

问题：

如果发生 midaemon 进程错误，请参见 /var/opt/perf/status.mi 了解详情。如果 midaemon 用完了其共享内存段的空间，则可能会遇到 midaemon 进程错误问题。

解决方案：

要避免发生此问题，必须清除不想要的共享内存。要清除共享内存，请执行以下步骤：

1. 运行以下命令：

   ```
   ipcs -m | grep 0x0c6629c9
   ```

   此命令将生成输出，其中在第二个数据字段中带有变量值。要清除共享内存，请运行以下命令：

   ```
   ipcrm –m <field_value>
   ```

   示例

   以下示例显示如何清除共享内存：

   ```
   ipcs -m | grep 0x0c6629c9
   output= 0x0c6629c9 18841617 root 640 8704448 7
   ipcrm –m 18841617
   ```

2. 要启动 midaemon，请运行以下命令：

   ```
   /opt/perf/bin/ovpa restart
   ```

在启用了超线程或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。

此版本的 HPE Operations Agent 引入了一个新的配置参数 `ignore_mt`，可用来配置性能收集组件以记录已使用**基于内核的规范化**计算的 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。

   备注: 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：

   ```
   /opt/perf/bin/midaemon -ignore_mt
   ```
5. 运行以下命令，启动 HPE Operations Agent：

   ```
   /opt/OV/bin/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)。

如果重新启动 HPE Operations Agent，性能收集组件将使用基于线程的规范化重新开始记录 CPU 数据，并且必须使用上述步骤再次配置性能收集组件。要使代理程序能够始终使用基于内核的规范化，请执行以下步骤：

1. 在代理程序节点上，转到以下位置：

   /var/opt/perf
2. 用文本编辑器打开以下文件：

   vppa.env
3. 将 MIPARMS 参数设置为 ignore_mt。
4. 保存文件。
5. 运行以下命令，重新启动代理程序：

   ```
   /opt/OV/bin/opcagt –start
   ```

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 HPE Operations Agent：

   在 Windows 上：

   ```
   %ovinstalldir%bin\opcagt -start
   ```

   在 Linux 和 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。

第 6 章: 使用 utility 程序

Utility 程序是用于管理和报告有关收集参数 (parm) 文件和警报定义 (alarmdef) 文件的信息的工具。可以使用 utility 工具执行以下任务：

- 扫描 HPE Operations Agent 数据存储区并生成显示如下信息的报告：
  - 覆盖的日期和时间
  - 收集参数 **(parm)** 文件中应用程序和进程设置的影响
- 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
- 对照警报定义处理存储在数据存储区中的数据以检测历史数据中的警报条件

运行 utility 程序

HPE Operations Agent 12.xx 仅支持以命令行模式运行 utility 程序。命令选项及其关联参数将通过命令行界面传递到 utility 程序。命令行界面适用于允许 shell 脚本轻松调用 utility 程序并允许输入和输出进行重定向的所有平台。

下表中列出了命令行选项和参数：

表 1：命令行参数

命令选项	参数	描述
-f	filename	指定输出文件名。 **例如：** 要将所有 **utility** 报告的输出文件指定为 utilrept，请运行以下命令： utility -f utilrept -d -xs 有关详细信息，请参见文件名。
-D		启用 analyze、scan 和 parm 文件检查的详细信息。 有关详细信息，请参见

命令选项	参数	描述
		detail 命令。
-d		禁用 analyze 和 parm 文件检查的详细信息。 有关详细信息，请参见 detail 命令。
-T		生成简洁的输出报告格式。
-v		在命令行中执行命令时，显示这些命令。
-xp	parmfile	检查 parm 文件的语法。 **例如：** 要检查 parm 文件的语法，请运行以下命令： utility -xp 有关详细信息，请参见 parmfile 命令。
-xc	alarmdef	检查语法并设置 -xa (analyze 命令)要使用的 alarmdef 文件名。 **例如：** 有关详细信息，请参见 checkdef 命令。
-xa		对照 alarmdef 文件分析数据存储区中存储的数据。 **例如：** 要查看警报事件以及警报摘要，请运行以下命令： utility -xa -D 有关详细信息，请参见 analyze 命令。

命令选项	参数	描述
-xs	datastore	扫描数据存储区并生成报告。 **例如：** 要扫描数据，请运行以下命令： utility -D -xs 有关详细信息，请参见 scan 命令。
-? 或 ?		显示命令行语法。

实用程序扫描报告

utility 程序的 scan 命令读取数据存储区并编写有关其内容的报告。该实用程序扫描报告为您提供有关每个数据类所使用的磁盘空间的信息。

例如：

如果运行命令 utility -xs，将生成实用程序扫描报告或类摘要报告，如下图中所示：

```
                    # utility -xs
      CLASSNAME                          RECORDS     STARTTIME           ENDTIME           HH:MM:SS
-------------------------------------------------------------------------------------------------------
SCOPE::CORE                                    0 0000/00/00 00:00:00  0000/00/00 00:00:00  00:00:00
SCOPE::FILESYSTEM                         209150 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::CPU                                 41830 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::GLOBAL                              20915 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::APPLICATION                         45116 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::PROCESS                            106857 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::DISK                                44536 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::NETIF                               41830 2015/07/13 15:13:43  2015/07/21 16:01:00  192:47:17
SCOPE::LVOLUME                                 0 0000/00/00 00:00:00  0000/00/00 00:00:00  00:00:00
SCOPE::TRANSACTION                             0 0000/00/00 00:00:00  0000/00/00 00:00:00  00:00:00
SISPI::LOGINS                               1690 2015/07/14 16:00:02  2015/07/21 16:00:02  168:00:00
SISPI::GLOBAL                                169 2015/07/14 16:00:03  2015/07/21 16:00:03  168:00:00
```

如果运行命令 utility -xs -D，您将看到以下输出：

扫描报告(版本 11.xx 和 12.xx)之间的比较将在“将 HPE Operations Agent 12.xx 与早期版本做比较”章节的实用程序扫描报告部分中列出。

utility 命令

本章详细描述 **utility** 程序的各个命令。

analyze

使用 analyze 命令基于警报定义 (**alarmdef**) 文件中的警报定义分析数据存储区中的数据，并报告生成的警报状态和活动。在发出 analyze 命令之前，可以运行 checkdef 命令检查警报定义语法。

命令行模式的默认警报定义文件是 **/var/opt/perf/alarmdef**。

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生成过多或过少的警报。

可以选择将 analyze 与 start、stop 和 detail 命令一同运行，以自定义分析过程。

如果只要查看警报摘要报告，请发出 detail off 命令。 在命令行模式下，detail off 是默认命令。要查看警报事件以及警报摘要，请指定 detail on (-D) 命令。

当 analyze 命令正在执行时，将列出警报事件，如 alarm start、end 和 repeat status。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。

例如

要查看性能警报摘要报告，请运行以下命令：

```
utility -xa
```

将看到以下输出：

```
Performance Alarm Summary:

Alarm   Count   Minutes
    1       3        25
    3       4        20
    4       16      125

Analysis coverage using "/var/opt/perf/alarmdef":

Start:05/05/2014 00:00:00    Stop:05/05/2014 16:55:00

Total time analyzed:Days:0  Hours:16  Minutes:55
```

要查看警报事件以及警报摘要，请运行以下命令：

```
utility -xa -D
```

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。

例如

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 -xc
```

确定警报定义正确后，可以使用 analyze 命令对照数据存储区中的数据处理警报定义。

detail

使用 detail 命令可控制 analyze、parmfile 和 scan 报告中打印内容的详细程度。在命令行模式中，默认值为 detail off。

参数

detail	on	-D	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	-d	在 **parm** 文件报告中，不打印应用程序定义。 在 scan 报告中，不打印收集器收集时间、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。

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

help

使用 help 命令访问 **Utility** 程序的联机帮助。

例如

要查看 Utility 程序的命令行参数，请运行以下命令：

```
utility -?
```

filename

使用 filename 命令指定所有 **utility** 报告的输出文件。报告的内容取决于在 filename 命令之后发出的其他命令。

例如

```
utility -f <file name> -d -xs
```

要将所有 **utility** 报告的输出文件指定为 utilrept，请运行以下命令：

```
utility -f utilrept -d -xs
```

filename 命令中的 <file name> 参数必须表示您具有写访问权限的有效文件名。输出已附加到现有文件末尾。如果文件不存在，则创建一个文件。

要确定当前输出文件，请运行无任何参数的 filename 命令。

如果输出文件不是 standard output，大多数命令将显示在输入它们时的输出文件。

备注: 在 AIX 系统上，无法使用带 -f 选项的 utility -xs 命令指定输出文件。

parmfile

使用 parmfile 命令查看并检查用于收集数据的 **parm** 文件设置的语法。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。

示例

要检查 **parm** 文件的语法，请运行以下命令：

```
utility -xp
```

将看到以下输出：

```
33 file  names used to define applications
1 user  names used to define applications
0 group names used to define applications

Parm File:"/var/opt/perf/parm" had 0 Warnings.
```

parmfile 命令检查当前 **parm** 文件的语法，并报告任何警告或错误。

要查看记录参数设置的列表，请运行以下命令：

```
utility -xp -v
```

scan

使用 scan 命令读取数据存储区并编写有关其内容的报告。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。

有关详细信息，请参见 detail 和 list。

例如：要扫描数据，请运行以下命令：

```
utility -D -xs
```

第 7 章: 使用 extract 程序

Extract 程序用于检索和分析 HPE Operations Agent 的数据存储区中记录的历史数据。extract 程序执行 export 函数。该程序从数据存储区读取数据，并以 ASCII 格式将结果导出到输出文件。

备注: oacore 进程必须处于运行状态，extract 程序才能正常运行。

使用命令行界面运行 extract 程序

HPE Operations Agent 12.xx 仅支持使用命令行模式运行 extract 程序。要从数据存储区提取数据，请使用命令行模式运行 extract 程序。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。它允许 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。

备注: 从 HPE Operations Agent 版本 11.xx 升级到 12.xx 期间，存储在 CODA 数据库文件、scope 日志文件和 DSI 日志文件中的旧数据以只读模式保留。extract 程序既可以从旧数据存储区又可以从新数据存储区读取数据。

下表列出了命令行选项和参数：

表 3：命令行参数

命令选项	参数		描述
-b	date	time	指定 export 函数的开始日期和时间。 语法： mm/dd/yyyy hh:mm:ss
-B		UNIX start time	以 UNIX 格式指定 export 函数的开始时间。
-e	date	time	指定 export 函数的结束日期和时间。 语法： mm/dd/yyyy hh:mm:ss
-E		UNIX stop time	以 UNIX 格式指定 export 函数的结束时间。
-l	logfile		指定 DSI 数据的输入日志文件。

表 3：命令行参数(续)

命令选项	参数		描述
-C	classname		指定自描述 (DSI) 数据进行导出操作。
gapcdznituyhx GADZNITUYHX			指定要导出的数据类型。 g = 全局详细信息 a = 应用程序详细信息 p = 进程详细信息 c = 配置详细信息 d = 磁盘设备详细信息 z = lvolume 详细信息 n = netif 详细信息 i = 逻辑系统详细信息 **备注:** Windows 和 Linux 平台不支持逻辑系统详细信息。 t = 事务详细信息 u = CPU 详细信息 y = 文件系统详细信息 h = 主机总线适配器 (HBA) 详细信息 x = 核心详细信息 G = 全局摘要 A = 应用程序摘要 D = 磁盘设备摘要 Z =lvolume 摘要 N = netif 摘要

表 3：命令行参数(续)

命令选项	参数		描述
			I = 逻辑系统摘要 **备注:** Windows 和 Linux 平台不支持逻辑系统摘要。 T = 事务摘要 U = CPU 摘要 Y = 文件系统摘要 H = HBA 摘要 X = 核心摘要
-r	export template file		指定 export 函数的导出模板文件。 **备注:** 如果导出模板文件的路径包含空格，请确保将其置于双引号中。
-f	filename		指定输出文件名。
-xp	xopt		将数据导出为 ASCII 格式文件。
-? 或 ?			显示命令行语法。

使用 Export 函数

export 命令会读取数据库中存储的数据，并将结果导出到输出文件。输出文件能以多种方式使用，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？

执行以下步骤：

1. 指定类类型或数据类型。请参见默认性能度量类 (第 174 页)
   例如： -a(应用程序类)， -g(全局类)
2. 指定要导出数据的输出文件。

   备注:

如果未指定输出文件的名称，则导出的数据将写入到默认输出文件中。默认输出文件取决于导出的数据类。

例如： 要导出详细信息全局类数据，将使用默认输出文件 `xfrdGLOBAL.asc`。请参见表 5：默认输出文件 (第 176 页)

3. 运行 export 命令。
4. 查看导出的数据。请参见导出文件的输出 (第 179 页)

备注: 导出模板文件 `reptfile` 和 `reptall` 随性能收集组件一起提供。这些文件位于 `/var/opt/perf/` 目录中。`Reptfile` 具有已取消注释的预定义度量集。默认情况下，该文件用于执行常见导出任务。在 `reptall` 文件中，所有度量将进行注释，并用于导出特定度量。要自定义导出模板文件，请参见生成自定义的输出文件。

例如：

要导出自 2014 年 7 月 26 起的 CPU 详细信息数据，请输入以下命令：

```
extract -u -b 7/26/14 -xp
```

其中

`-u`	指定 CPU 详细信息。
`-b`	指定 export 函数的开始日期。
`-xp`	将数据导出到外部格式文件。

因为没有指定导出模板文件，将使用默认导出模板文件 `reptfile`。

默认性能度量类

可导出以下数据类：

表 4：类类型

类类型	汇总间隔
`global`	5、15、30、60、180、360、720、1440、10080 分钟。
`application`	5、15、30、60、180、360、720、1440、10080 分钟。
`process`	未汇总此度量类
`disk device`	5、15、30、60、180、360、720、1440、10080 分

表 4：类类型 (续)

	钟。
lvolume	5、15、30、60、180、360、720、1440、10080 分钟。
transaction	5、15、30、60、180、360、720、1440、10080 分钟。
configuration	5、15、30、60、180、360、720、1440、10080 分钟。
netif	5、15、30、60、180、360、720、1440、10080 分钟。
cpu	5、15、30、60、180、360、720、1440、10080 分钟。
filesystem	5、15、30、60、180、360、720、1440、10080 分钟。
host bus adapter	5、15、30、60、180、360、720、1440、10080 分钟。
core	5、15、30、60、180、360、720、1440、10080 分钟。

备注: 如果还提到未在此表中列出的任何其他汇总间隔，则 extract 程序会汇总到最近的汇总间隔。

输出文件

在发出 export 命令之前，可以使用默认输出文件导出数据或指定输出文件。

- 如果在发出 export 命令之前指定输出文件，则所有数据将导出到此单个文件中

  例如：

  要将详细信息全局数据导出到名为 myout 的输出文件中，请运行以下命令：

  ```
  extract -g -f myout -xp
  ```

  其中

-g	指定全局详细信息。
-f	指定输出文件名。

myout	是输出文件的名称。
-xp	导出数据。

- 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将分成不同的默认输出文件。

  例如：

  ```
  extract -xp -g
  ```

  该导出命令会导致数据的全局类导出到 xfrdGLOBAL.asc 文件中。

下表列出了所有默认输出文件：

表 5：默认输出文件

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

表 5：默认输出文件(续)

xfrsTRANSACTION.asc	事务摘要数据文件
xfrdCONFIGURATION.asc	配置数据文件
xfrdHBA.asc	主机总线适配器 (HBA) 详细信息数据文件
xfrsHBA.asc	HBA 摘要数据文件
xfrdCORE.asc	核心详细信息数据文件
xfrsCORE.asc	核心摘要数据文件

其中 ext= asc (ASCII)。

备注: 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。

默认文件名根据数据类型名称创建。根据数据不同，前缀为 xfrd 或 xfrs。xfrd 用于详细数据，xfrs 用于摘要数据。

例如，classname = ACCTG_INFO 的 **export** 文件名有：

xfrdACCTG_INFO.asc	ACCT_INFO 的详细 ASCII 数据
xfrsACCTG_INFO.asc	ACCT_INFO 的摘要 ASCII 数据

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：

```
HEADINGS    [ON]
            [OFF]
SEPARATOR= "|"
SUMMARY=VALUE
OUTPUT=Filename
DATA TYPE DATATYPE
METRICS
```

例如：

```
HEADINGS ON
SEPARATOR="|"
SUMMARY=60

************************************************************************
**
** The rest of the file consists of specifics about the metrics to be
** exported for a single data type. Each exportable class may have one
** set of definitions in which the following lines appear.
**
** DATA TYPE specifies the type (class) of data. (required)
**
** OUTPUT    specifies the name of the output file where the exported data is
**           to be written.  OUTPUT may be specified for each class named.
**           (optional)
**
**
**  Individual metric names which belong to the class. Metrics will be exported
**  in the order listed whenever possible. In order to be exported, metrics
**  must not be commented out (that is, they must not contain an asterisk before
**  the metric name).
**
** This sample report was generated with a section for each data type which
** is available in the currently open log file. Following the DATA TYPE
** line, each metric which is available for export for this data type is
** listed, but commented out. To select metrics for your report, delete
** the asterisk (*) in the first column.
**
************************************************************************
************************************************************************

************************************************************************
DATA TYPE GLOBAL

**............................. Global Record Identification Metrics

* GBL_PROC_SAMPLE
* GBL_SYSTEM_UPTIME_HOURS
* GBL_SYSTEM_UPTIME_SECONDS
* GBL_STATTIME
* GBL_INTERVAL
* GBL_CSWITCH_RATE
* GBL_INTERRUPT_RATE
* GBL_INTERRUPT
```

参数

headings	指定是否在 export 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。 如果指定了headings on，ASCII 格式会将每个写入度量列的导出标题和列标题置于第一条数据记录之前。
separator	指定 ASCII 格式的数据中每个字段之间打印的字符。

	默认的分隔符是竖线。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	指定要汇总的数据的每个摘要间隔的分钟数。 默认间隔是 5、15、30、60、180、360、720、1440 和 10080 分钟。如果提到其他汇总间隔，extract 程序会汇总到最近的汇总间隔。
output	指定导出的数据将写入的输出文件的名称。可通过将输出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，为导出的每一类或数据类型进行指定。可以为输出指定任何有效的文件名。
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。
metrics	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。

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。

导出文件的输出

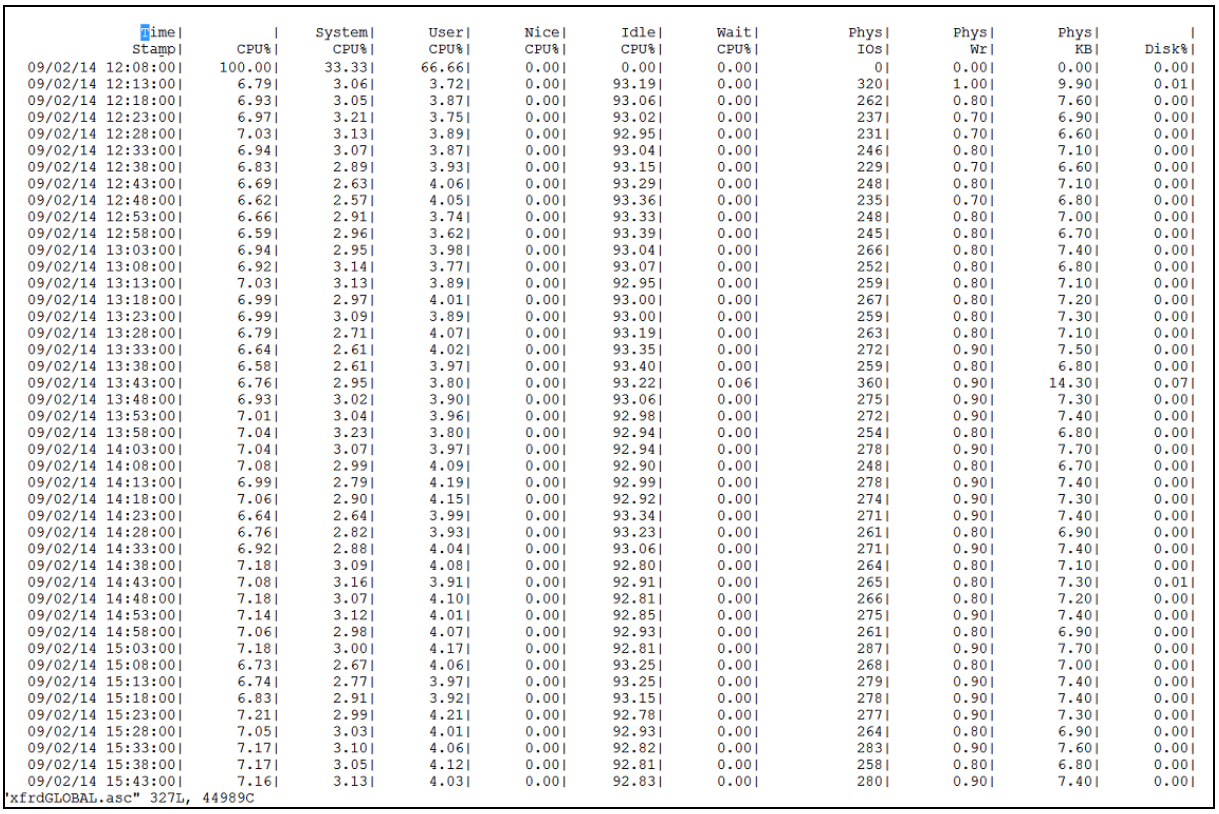
每个导出文件的内容包括：

名称(application、netif、lvolume 或 transaction)	如果指定了 headings on。
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。

例如：

如果导出全局类数据，则可以看到以下输出：



Time Stamp	CPU%	System CPU%	User CPU%	Nice CPU%	Idle CPU%	Wait CPU%	Phys IOs	Phys Wr	Phys KB	Disk%
09/02/14 12:08:00	100.00	33.33	66.66	0.00	0.00	0.00	0	0.00	0.00	0.00
09/02/14 12:13:00	6.79	3.06	3.72	0.00	93.19	0.00	320	1.00	9.90	0.01
09/02/14 12:18:00	6.93	3.05	3.87	0.00	93.06	0.00	262	0.80	7.60	0.00
09/02/14 12:23:00	6.97	3.21	3.75	0.00	93.02	0.00	237	0.70	6.90	0.00
09/02/14 12:28:00	7.03	3.13	3.89	0.00	92.95	0.00	231	0.70	6.60	0.00
09/02/14 12:33:00	6.94	3.07	3.87	0.00	93.04	0.00	246	0.80	7.10	0.00
09/02/14 12:38:00	6.83	2.89	3.93	0.00	93.15	0.00	229	0.70	6.60	0.00
09/02/14 12:43:00	6.69	2.63	4.06	0.00	93.29	0.00	248	0.80	7.10	0.00
09/02/14 12:48:00	6.62	2.57	4.05	0.00	93.36	0.00	235	0.70	6.80	0.00
09/02/14 12:53:00	6.66	2.91	3.74	0.00	93.33	0.00	248	0.80	7.00	0.00
09/02/14 12:58:00	6.59	2.96	3.62	0.00	93.39	0.00	245	0.80	6.70	0.00
09/02/14 13:03:00	6.94	2.95	3.98	0.00	93.04	0.00	266	0.80	7.40	0.00
09/02/14 13:08:00	6.92	3.14	3.77	0.00	93.07	0.00	252	0.80	6.80	0.00
09/02/14 13:13:00	7.03	3.13	3.89	0.00	92.95	0.00	259	0.80	7.10	0.00
09/02/14 13:18:00	6.99	2.97	4.01	0.00	93.00	0.00	267	0.80	7.20	0.00
09/02/14 13:23:00	6.99	3.09	3.89	0.00	93.00	0.00	259	0.80	7.30	0.00
09/02/14 13:28:00	6.79	2.71	4.07	0.00	93.19	0.00	263	0.80	7.10	0.00
09/02/14 13:33:00	6.64	2.61	4.02	0.00	93.35	0.00	272	0.90	7.50	0.00
09/02/14 13:38:00	6.58	2.61	3.97	0.00	93.40	0.00	259	0.80	6.80	0.00
09/02/14 13:43:00	6.76	2.95	3.80	0.00	93.22	0.06	360	0.90	14.30	0.07
09/02/14 13:48:00	6.93	3.02	3.90	0.00	93.06	0.00	275	0.90	7.30	0.00
09/02/14 13:53:00	7.01	3.04	3.96	0.00	92.98	0.00	272	0.90	7.40	0.00
09/02/14 13:58:00	7.04	3.23	3.80	0.00	92.94	0.00	254	0.80	6.80	0.00
09/02/14 14:03:00	7.04	3.07	3.97	0.00	92.94	0.00	278	0.90	7.70	0.00
09/02/14 14:08:00	7.08	2.99	4.09	0.00	92.90	0.00	248	0.80	6.70	0.00
09/02/14 14:13:00	6.99	2.79	4.19	0.00	92.99	0.00	278	0.90	7.40	0.00
09/02/14 14:18:00	7.06	2.90	4.15	0.00	92.92	0.00	274	0.90	7.30	0.00
09/02/14 14:23:00	6.64	2.64	3.99	0.00	93.34	0.00	271	0.90	7.40	0.00
09/02/14 14:28:00	6.76	2.82	3.93	0.00	93.23	0.00	261	0.80	6.90	0.00
09/02/14 14:33:00	6.92	2.88	4.04	0.00	93.06	0.00	271	0.90	7.40	0.00
09/02/14 14:38:00	7.18	3.09	4.08	0.00	92.80	0.00	264	0.80	7.10	0.00
09/02/14 14:43:00	7.08	3.16	3.91	0.00	92.91	0.00	265	0.80	7.30	0.01
09/02/14 14:48:00	7.18	3.07	4.10	0.00	92.81	0.00	266	0.80	7.20	0.00
09/02/14 14:53:00	7.14	3.12	4.01	0.00	92.85	0.00	275	0.90	7.40	0.00
09/02/14 14:58:00	7.06	2.98	4.07	0.00	92.93	0.00	261	0.80	6.90	0.00
09/02/14 15:03:00	7.18	3.00	4.17	0.00	92.81	0.00	287	0.90	7.70	0.00
09/02/14 15:08:00	6.73	2.67	4.06	0.00	93.25	0.00	268	0.80	7.00	0.00
09/02/14 15:13:00	6.74	2.77	3.97	0.00	93.25	0.00	279	0.90	7.40	0.00
09/02/14 15:18:00	6.83	2.91	3.92	0.00	93.15	0.00	278	0.90	7.40	0.00
09/02/14 15:23:00	7.21	2.99	4.21	0.00	92.78	0.00	277	0.90	7.30	0.00
09/02/14 15:28:00	7.05	3.03	4.01	0.00	92.93	0.00	264	0.80	6.90	0.00
09/02/14 15:33:00	7.17	3.10	4.06	0.00	92.82	0.00	283	0.90	7.60	0.00
09/02/14 15:38:00	7.17	3.05	4.12	0.00	92.81	0.00	258	0.80	6.80	0.00
09/02/14 15:43:00	7.16	3.13	4.03	0.00	92.83	0.00	280	0.90	7.40	0.00

"xfrdGLOBAL.asc" 327L, 44989C

备注：

即使未在 reptfile 或 reptall 中选择 **gbl_statdate** 和 **gbl_stattime** 度量，导出的文件也始终包含 24 小时制的时间戳。

使用 HPE Operations Agent 12.01 时，显示在 extract 输出中的度量标头与使用实时工具(例如 perfd 和 Glance)显示的标头一致。

生成自定义的输出文件

性能收集组件提供导出模板文件 `reptall`。此文件位于 `/var/opt/perf` 目录中。可以按如下所述使用 `reptall` 自定义输出文件：

自定义导出文件：

要自定义导出文件，请执行以下步骤：

1. 在导出模板文件中，取消注释要导出的度量。
2. 保存并导出文件。

ASCII 格式备注

ASCII(或文本)格式最适合将文件复制到打印机或终端。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 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。

备注: 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。

提示: 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。

extract 命令

本章描述 **extract** 程序的命令和示例。

表 6：extract 命令

命令选项	描述
application	使用 application 选项可指定要导出的应用程序数据的类型。 **例如：** 要导出详细的应用程序数据，请运行以下命令： `extract -a -r /var/opt/perf/myrept -xp` 由于没有指定输出文件，应用程序数据将导出到 xfrdAPPLICATION.asc。 该输出文件包含 myrept 导出模板文件中指定的所有应用程序度量。
cpu	使用 cpu 选项可指定 CPU 的汇总级别。

表 6：extract 命令(续)

	例如： 要导出从 2014 年 7 月 26 日开始收集的 CPU 详细数据，请运行以下命令： `extract -u -b 7/26/14 -xp` 在此示例中，由于没有指定输出文件，CPU 数据将导出到 `xfrdCPU.asc`。 由于没有指定导出模板文件，将使用默认导出模板文件 reptfile。reptfile 中指定的所有 CPU 度量均包括在输出文件中。
disk	使用 `disk` 选项可指定要导出的磁盘设备数据的类型。 **例如：** 要导出从 2014 年 7 月 5 日开始收集的磁盘详细数据，请运行以下命令： `extract -d -b 7/5/14 -xp` 在此示例中，由于没有指定输出文件，磁盘详细数据将导出到 `xfrdDISK.asc`。 由于没有指定导出模板文件，将使用默认导出模板文件 reptfile。reptfile 中指定的所有磁盘度量均包括在输出文件中。
export	使用 `export` 选项可开始将数据复制到输出文件的过程。 **例如：** 要导出全局详细数据，请运行以下命令： `extract -xp -g` 数据的全局类将导出到 `xfrdGLOBAL.asc` 文件中。
filesystem	使用 `filesystem` 选项可指定要导出的文件系统数据的汇总级别。 **例如：** 要导出从 2014 年 7 月 26 日开始收集的文件系统详细数据，请运行以下命令： `extract -y -b 7/26/14 -xp` 文件系统数据将导出到 `xfrdFILESYSTEM.asc`。

表 6：extract 命令(续)

	由于没有指定导出模板文件，将使用默认导出模板文件 reptfile。`reptfile` 中指定的所有文件系统度量均包括在输出文件中。
global	使用 `global` 选项可指定要导出的全局数据量。 **例如：** 要导出详细的全局数据，请运行以下命令： `extract -g -r /var/opt/perf/myrept -f myout -xp` 全局数据导出到名为 `myout` 的输出文件中。 该输出文件包含 myrept 导出模板文件中指定的所有全局度量。
help	使用 `help` 选项可访问联机帮助。
lvolume	使用 `lvolume` 选项可指定要导出的逻辑卷数据的类型。 **注：**仅 HP-UX 和 Linux 系统支持此选项。 **例如：** 要导出详细的逻辑卷数据，请运行以下命令： `extract -z -r /var/opt/perf/myrept -xp` 由于没有指定输出文件，逻辑卷数据将导出到 `xfrdVOLUME.asc`。 该输出文件包含 myrept 导出模板文件中指定的度量。
netif	使用 `netif` 选项可指定要导出的逻辑网络接口 (LAN) 数据的类型。 **例如：** 要导出详细的网络接口数据，请运行以下命令： `extract -n -r /var/opt/perf/myrept -xp` 由于没有指定输出文件，网络接口数据将导出到 `xfrdNETIF.asc`。 该输出文件包含 myrept 导出模板文件中指定的度量。
process	使用 `process` 选项可指定是否导出进程数据。 **例如：**

表 6：extract 命令(续)

	要导出详细的进程数据，请运行以下命令： `extract -p -r /var/opt/perf/myrept -xp` 由于没有指定输出文件，进程数据将导出到 `xfrdPROCESS.asc`。 该输出文件包含 myrept 导出模板文件中指定的所有进程度量。

start	使用 `start` 选项可设置 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。
	支持将 `First` 和 `last` 选项与开始 (`-b`) 或停止 (`-e`) 命令结合使用。
	例如：
	1. 要导出从 1999 年 6 月 5 日上午 8:00 开始的详细全局数据，请运行以下命令：
	`extract -g -b 06/05/99 8:00 -f myout -xp`
	全局数据导出到名为 `myout` 的输出文件中。
	由于没有指定导出模板文件，将使用默认导出模板文件 reptfile。reptfile 中指定的所有全局度量均包括在输出文件中。
	2. 要仅导出最晚的全局数据，请运行以下命令：
	`extract -g -b last-f myout -xp`
	3. 要将最早和最晚的全局数据导出到 `xfrdGLOBAL.asc` 文件中，请运行以下命令：
	`extract -g -b first -e last -xp`

stop	使用 `stop` 选项可在指定的日期和时间终止 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。 **例如：** 要导出从 2014 年 6 月 5 日上午 8:00 开始到 2014 年 6 月 5 日下午 5:00 结束的详细全局数据，请运行以下命令： `extract -g -b 6/5/14 8:00 -e 6/5/14 17:00  -f myout -xp` 全局数据导出到名为 `myout` 的输出文件中。 由于没有指定导出模板文件，将使用默认导出模板文件

表 6：extract 命令(续)

	reptfile。reptfile 中指定的所有全局度量均包括在输出文件中。
transaction	使用 `transaction` 选项可指定要导出的事务数据的类型。 **例如：** 要导出详细的事务数据，请运行以下命令： `extract -t -r /var/opt/perf/myrept -xp` 由于没有指定输出文件，事务数据将导出到 `xfrdTRANSACTION.asc`。 该输出文件包含 myrept 导出模板文件中指定的所有事务度量。
host bus adapter	使用 `host bus adapter` 选项可指定要导出的 HBA 数据量。 **例如：** 要导出从 2014 年 7 月 26 日开始收集的 HBA 详细数据，请运行以下命令： `extract -h -b 7/26/14 -xp` 在此示例中，由于没有指定输出文件，HBA 数据将导出到 `xfrdHBA.asc`。 **备注:** • 默认导出模板文件 reptfile 不包含 HBA 度量。要导出 HBA 数据，请手动添加 HBA 度量。 • Linux、Windows 和 HP-UX 系统支持此选项。
classname 和 logfile	使用 `classname` 选项可指定要导出的 DSI 数据类。 **语法** extract -xp -C <dsi class name> -l <log file path>
core	使用 `core` 选项可指定要导出的内核数据的类型。 **注：**Linux、Windows、HP-UX 和 Solaris 系统支持此选项。 **例如：**

表 6：extract 命令(续)

	要导出详细的应用程序数据，请运行以下命令： `extract -x -r /var/opt/perf/myrept -xp` 该输出文件包含 myrept 导出模板文件中指定的所有应用程序度量。

第 8 章: 使用 cpsh 程序

只有在启用 HP Ops OS Inst to Realtime Inst LTU 或 Glance Pak Software LTU 后，才可以使用 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. (以具有根特权或管理特权的身份)登录到安装了 HPE Operations Agent 的系统。
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 <system_name>
   ```

   其中 <system_name> 是远程系统的完全限定域名。

   或

   ```
   cpsh -n <ip_address>
   ```

   其中 <ip_address> 是远程系统的 IP 地址。

   注：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。可以阻止其他系统通过 cpsh 实用程序访问本地系统的性能数据。有关详细信息，请参见限制访问 (第 53 页)。

   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 <system_name> -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** <metric_class>。
2. 要针对给定类列出所有当前设置的度量，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** <metric_class>。
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：

   在 cpsh 提示符处，输入 **delete** <metric_name>。

   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：

   在 cpsh 提示符处，输入 **add** <metric_name>。

   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** <metric_class>。
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** <metric_class>。
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 类的度量，您可以从 cpsh 提示符查看汇总数据。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** <interval>。在此实例中，<interval> 是以秒为单位而指定的汇总间隔。<interval> 必须是 cpsh 连接到的 **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

启用阈值和过滤条件

您可以基于 Perfd 进程中的要求设置阈值和过滤选项，并查看满足已设置条件的数据。

设置以下选项：

Threshold - 此选项仅可用于 GLOBAL 类的度量。您可以指定全局阈值并查看在已设置全局阈值之上的数据。

Filter - 过滤选项可用于所有类的度量。

可以将条件设置为：

- **filter** <metric> <operator> <value>
- **threshold** <metric> <operator> <value>

也可以同时设置两个条件以仅查看满足已设置条件的数据。

示例 1：列出利用率超过 4% 的进程的数量。

使用过滤条件查看数据。

要查看过滤后的数据，请执行以下操作：

1. 打开 cpsh 提示符。
2. 在 cpsh 提示符处，输入 **class proc**
3. 输入 **filter proc_cpu_total_util > 4**
   cpsh 实用程序显示 CPU 利用率超过 4% 的进程。
4. 输入 **push**

结果显示为：

```
User   App     Interest        Process         ProcessName
Name   PID     ID Reason       Start Time      CPU Name
root  1028     4       05/07/2013 07:17:35     5.2 find
```

示例 2：列出利用率超过 4% 且阈值设置为 1% 的进程的数量。

依次设置阈值选项和过滤选项以查看数据。

要查看过滤后的数据，请执行以下操作：

1. 打开 cpsh 提示符。
2. 在 cpsh 提示符处，输入 **class proc**
3. 输入 **threshold proc_cpu_total_util > 1**
4. 输入 **filter proc_cpu_total_util > 3**
   cpsh 实用程序显示阈值设置为大于 1% 且 CPU 利用率大于 3% 的进程。
5. 输入 **push**

结果显示为：

```
User   App     Interest        Process         ProcessName
Name   PID     ID Reason       Start Time      CPU Name
```

```
    root  1028     4       05/07/2013 07:17:35     4.9 find
```

第 9 章: 在 Windows 上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。

备注: 在注册策略并开始收集之前，确保 **oacore** 进程正在运行。

生成性能计数器收集

要生成收集，请从**开始**菜单中选择 **ECB-ECM (扩展的收集生成器和管理器) (ECB-ECM (Extended Collection Builder and Manager))** 或使用命令提示符运行命令 `mwecbm` 以打开**扩展的收集生成器和管理器 (Extended Collection Builder and Manager)**。

将出现“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口，在左窗格中显示 Windows 对象的列表。有关生成收集的说明，请从“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口的“帮助”菜单中选择**帮助主题**。

生成 Windows 性能计数器收集后，使用**扩展的收集生成器 (Extended Collection Manager)** 窗格来注册、启动和停止新的和现有的收集。还可以使用**管理**选项来启动、停止或删除。

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底部的**扩展的收集生成器 (Extended Collection Manager)** 窗格。最初，不会出现任何收集，因为必须注册收集才能开始收集数据。

注册或存储您创建的收集后，“扩展的收集生成器 (Extended Collection Manager)”窗格将显示当前收集的列表。“扩展的收集生成器 (Extended Collection Manager)”窗格还显示每个收集的状态，并使您能够查看有关收集本身的信息(属性)。有关管理收集的说明，请从“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口的**帮助**菜单中选择**帮助主题**。

使用扩展的收集生成器和管理器的提示

<InstallDir>\paperdocs\ovpa\C\monxref.txt 文件包含 性能收集组件 度量到 Windows 性能计数器和命令的交叉引用。通过用于已由 性能收集组件 收集的度量的扩展的收集生成器和管理器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，默认度量名称将分配给 Windows 性能计数器，以供性能收集组件内部使用。通常，这些默认名称没有意义或难以解密。要使度量名称更有意义或将这些名称与源应用程序提供的度量名称匹配，请在“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口中将度量名称从左窗格拖到右窗格后，通过右键单击或双击度量名称来修改度量属性。(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。)

如果开始 61 个或以上的收集，则在收集超过 60 个时将进入错误状态。这可能会导致其他收集出现问题。

如果从配置有 **Wolfpack** 的系统中收集逻辑磁盘度量，则必须重新启动收集，才能收集在注册收集时不存在的任何新磁盘实例的数据。

必须在删除收集后重新启动性能收集组件，才能成功删除收集。如果性能收集组件未重新启动，则可能会在执行删除操作期间遇到错误。此错误通常表示某些文件未被成功删除。您可能需要手动删除在重新启动性能收集组件后仍保留的任何文件和目录。

扩展的收集生成器和管理器可能会报告具有缓存计数器的某些度量缺少值。当度量值溢出时，在某些情况下可能会出现此问题。还会将一条消息发送到 ECBM 状态文件。可以通过重新启动收集来解决此问题。

备注: ECBM 无法处理超过 2147483647 的度量值。

扩展的收集生成器和管理器联机帮助提供了扩展的收集生成器和管理器概念的说明以及有关创建和查看数据收集的说明。要查看联机帮助，请从桌面选择**开始 --> 所有程序 --> HPE Operations Agent --> 性能集合组件 --> ECB-ECM 联机帮助 (ECB-ECM Online Help)**。可以从性能收集组件主窗口的“代理”菜单中选择**扩展的收集 (Extended Collections)**，并从“扩展的收集生成器和管理器 (Extended Collection Builder and Manager)”窗口的“帮助”菜单中选择**帮助主题**。可通过在扩展的收集生成器和管理器中显示的对话框中选择**帮助**按钮获得联机帮助。

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：

\<InstallDir>\bin\mwcmcmd.exe

- 要显示各种选项，请输入以下命令：

\<InstallDir>\bin\mwcmcmd /?

- 要启动停止的收集，请输入以下命令：

mwcmcmd start <collection_name(s)>

- 要从变量实例策略开始新的收集，请输入以下命令：

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

以下选项可用：

-i <sampling_interval>- 更改采样间隔(秒)

-l <logfile_path_name> - 更改默认日志位置

-a <alarm_file> - 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 <collection_name(s)>

- 要注册策略文件，请输入以下命令：

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

仅当注册固定实例策略文件时，以下选项才可用：

-i <sampling_interval> - 更改采样间隔(秒)

-l <logfile_path_name> - 更改默认日志位置

-a <alarm_file> - 更改警报定义文件

- 要删除单个收集，请输入以下命令：

mwcmcmd delete <collection/policy_name> [option]

删除收集时，以下选项才可用：

-r - 重新启动 HPE 性能收集组件

备注: 删除策略或收集时，将删除所有关联的数据库文件。此外，进程 **oacore** 会在删除收集时停止，并在删除收集后自动启动。

- 要删除多个收集或策略，请输入以下命令：

mwcmcmd delete { <collection/policy_name(s)> | -c | -all }

-c - 删除所有收集

-a - 删除所有收集和策略

备注: 一次删除多个策略或收集时，将自动重新启动性能收集组件，并将删除所有关联的数据库文件。

- 要列出所有已注册的收集和策略，请输入以下命令：

mwcmcmd list

- 要列出收集或策略的属性，请输入以下命令：

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

- 要列出策略中的变量实例对象，请输入：

mwcmcmd objects <policy_name>

第 10 章: 基线概述

基线是基于存储在度量数据存储区中的历史数据[1]计算并提供参考值的过程。要计算特定时间段内的基线数据，将使用自过去几周的相应时间段内收集的度量数据。基线数据包括最小值、最大值、平均值和标准偏差值。基线数据在每小时结束时进行计算，并存储在度量数据存储区中。

基线数据用于：

- 提供参考值来监视每日性能
- 提供参考值来分析性能趋势
- 动态设置最佳阈值来分析资源利用率的模式

HPE Operations Agent 计算出的基线数据由 SI-AdaptiveThresholdingMonitor 策略用于监视性能和资源利用率。有关详细信息，请参见《HPE Operations 系统基础结构 SPI 用户指南》中的“自适应阈值策略”主题。

在 HPE Operations Agent 节点上配置基线

默认情况下，在 HPE Operations Agent 节点上不启用基线功能。要在 HPE Operations Agent 节点上配置基线功能，请执行以下步骤：

1. 更新 `baseline.cfg` 文件

   备注: `baseline.cfg` 是纯文本文件，用于定义要监视的度量。

   a. 登录到 HPE Operations Agent 节点。
   b. 转到以下目录：

      在 Windows 上：

      `%ovdatadir%`

      在 HP-UX/Linux/Solaris 上：

      `/var/opt/perf/`
   c. 更新 `baseline.cfg` 文件，并按照下列格式定义要监视的度量：

      `<Class>:<Metric>`

      在此实例中：

      ◦ `<Class>` 是度量类。
      ◦ `<Metric>` 是必须计算基线数据的度量。

      备注: 仅计算量表度量的基线数据。

例如：

Global:GBL_CPU_TOTAL_UTIL

在此实例中：

<Class>	Global
<Metric>	GBL_CPU_TOTAL_UTIL

2. 在 HPE Operations Agent 节点上启用基线功能
   a. 转到以下目录：

      在 Windows X64 上：

      `%ovinstalldir%bin\win64`

      在 Windows X86 上：

      `%ovinstalldir%bin`

      在 AIX 上：

      `/usr/lpp/OV/bin`

      在 HP-UX/Linux/Solaris 上：

      `/opt/OV/bin`

   b. 运行以下命令：

      `ovconfchg -ns oacore -set ENABLE_BASELINE TRUE`

      默认情况下，此值为 FALSE。

基线数据仅在每个小时结束时进行计算。如果要在启用基线后立即计算基线数据，必须重新启动 **oacore** 进程。运行以下命令重新启动 **oacore**：

```
ovc -restart oacore
```

备注: 还可以使用 XPL 配置启用基线。执行以下步骤：

1. 在 HPOM 控制台上，选择**策略管理 → 策略组 → 基础结构管理 (Infrastructure Management) → v12.0 → 设置和阈值 (Settings and Thresholds) → 代理程序设置 (Agent Settings) → OPC_PERL_INCLUDE_INSTR_DIR**
2. 将变量 ENABLE_BASELINE 设置为 TRUE，然后在所有所需节点上部署策略。

有关详细信息，请参见《HPE Operations 系统基础结构 SPI 用户指南》。

监视度量

备注: 计算基线数据后，请使用测量阈值策略监视所需度量。可以使用 SI-

ConfigureBaselining 和 SI-AdaptivethresholdingMonitor 之类的默认基础结构 SPI 策略监视基线度量。有关详细信息，请参见《HPE Operations 系统基础结构 SPI 用户指南》。

根据 baseline.cfg 文件中定义的类，在度量数据存储区中创建相应的基线类。

备注: 基线类以 _BASELINE 结尾。

例如：

如果在 baseline.cfg 文件中指定以下度量：

```
Disk:BYDSK_UTIL

Global:GBL_CPU_TOTAL_UTIL
```

将创建两个基线类：

```
DISK_BASELINE

GLOBAL_BASELINE
```

运行 utility -xs 命令查看有关基线类的信息：

```
# utility -xs

CLASSNAME              RECORDS   STARTTIME           ENDTIME             HH:MM:SS
-------------------------------------------------------------------------------------
SCOPE::DISK            295       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::NETIF           426       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::TRANSACTION     994       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::PROCESS         60262     2015/07/31 12:30:40 2015/07/31 14:52:00 2:21:20
SCOPE::CPU             284       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::CORE            0         0000/00/00 00:00:00 0000/00/00 00:00:00 00:00:00
SCOPE::APPLICATION     579       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::FILESYSTEM      4118      2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::GLOBAL          142       2015/07/31 12:30:40 2015/07/31 14:51:00 2:20:20
SCOPE::LVOLUME         0         0000/00/00 00:00:00 0000/00/00 00:00:00 00:00:00
SCOPE::DISK_BASELINE   6         2015/07/31 13:00:00 2015/07/31 14:00:00 1:00:00
SCOPE::GLOBAL_BASELINE 2         2015/07/31 13:00:00 2015/07/31 14:00:00 1:00:00
```

Baseline Classes

针对在 baseline.cfg 文件中指定的每个度量，将创建十六个基线度量。

例如：

如果在 baseline.cfg 文件中指定 GBL_CPU_TOTAL_UTIL，则将创建十六个全局基线度量。

备注: 可以运行 ovcodautil-obj 命令查看基线度量。

运行以下命令查看为 GLOBAL_BASELINE 类创建的基线度量:

```
ovcodautil -ds SCOPE -o GLOBAL_BASELINE -obj
```

```
GLOBAL_BASELINE        ATT   I64    INSTANCEID
GLOBAL_BASELINE        KEY   UTF8   GBL_SYSTEM_ID
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_MAX_BL
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_MIN_BL
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_AVG_BL
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_STDDEV_BL
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_STDDEV_BL_LASTHOUR
GLOBAL_BASELINE        GGE   I64    GBL_CPU_TOTAL_UTIL_BL_SAMPLESIZE
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_MAX_BL_LASTHOUR
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_MIN_BL_LASTHOUR
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_AVG_BL_LASTHOUR
GLOBAL_BASELINE        GGE   I64    GBL_CPU_TOTAL_UTIL_BL_SAMPLESIZE_LASTHOUR
GLOBAL_BASELINE        GGE   I64    GBL_CPU_TOTAL_UTIL_S0
GLOBAL_BASELINE        GGE   I64    GBL_CPU_TOTAL_UTIL_S0_LASTHOUR
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_S1
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_S1_LASTHOUR
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_S2
GLOBAL_BASELINE        GGE   R64    GBL_CPU_TOTAL_UTIL_S2_LASTHOUR
```

汇总类型为平均值 (Avg)、最小值 (Min)、最大值 (Max)、标准偏差 (STDDEV) 以及样本大小 (SAMPLESIZE)

以 _BL 结尾的基线度量包含历史数据(以及最后一小时的数据)。

以 _BL_LASTHOUR 结尾的基线度量仅包含最后一小时的数据。

可以创建测量阈值策略将最后一小时基线数据与历史数据进行比较。

例如: 可以使用测量阈值策略比较 GBL_CPU_TOTAL_UTIL_AVG_BL_LASTHOUR 和 GBL_CPU_TOTAL_UTIL_AVG_BL。

基线度量 S0、S1 和 S2 用于计算标准偏差。

备注: 要在 HPE Operations Agent 节点上停止基线功能，请运行以下命令:

```
ovconfchg -ns oacore -set ENABLE_BASELINE FALSE
```

如何在基线未正常运行时进行故障排除?

执行以下步骤:

1. **检查是否已启用基线。**
   a. 登录到 HPE Operations Agent 节点。
   b. 转到以下目录:
      在 32 位版本的 Windows 上:

"%ovinstalldir%/bin"

在 64 位版本的 Windows 上：

"%ovinstalldir%/bin/win64"

在 AIX 上：

/usr/lpp/OV/bin

在 HP-UX/Linux/Solaris 上：

/opt/OV/bin

c. 运行以下命令：

```
ovconfget oacore ENABLE_BASELINE
```

如果已启用基线，则输出为 **TRUE**。

2. **检查基线度量的 ovcodautil 输出。**

a. 转到以下目录：

在 32 位版本的 Windows 上：

"%ovinstalldir%/bin"

在 64 位版本的 Windows 上：

"%ovinstalldir%/bin/win64"

在 AIX 上：

/usr/lpp/OV/bin

在 HP-UX/Linux/Solaris 上：

/opt/OV/bin

b. 运行以下命令：

```
ovcodautil -ds SCOPE –o <CLASS>_BASELINE –rawonly
```

例如：

```
ovcodautil -ds SCOPE –o GLOBAL_BASELINE –rawonly
```

然后检查输出(如果已列出所需的基线度量)。

3. **检查基线配置文件中是否提到必须计算其基线数据的度量。**

a. 检查以下文件中是否采用正确格式提到度量：

在 Windows 上：

%ovdatadir%/baseline.cfg

在 HP-UX/Linux/Solaris/AIX 上：

/var/opt/perf/baseline.cfg

注：

必须计算其基线数据的度量必须采用以下格式之一：

```
[Baseline]
<Class>:<Metric>
```

例如：
Global:GBL_CPU_TOTAL_UTIL

4. **检查基线配置文件中提到的度量是否对正在使用的平台有效。**

备注: 仅计算量表度量的基线数据。

a. 转到以下目录：

在 32 位版本的 Windows 上：

```
"%ovinstalldir%/bin"
```

在 64 位版本的 Windows 上：

```
"%ovinstalldir%/bin/win64"
```

在 AIX 上：

```
/usr/lpp/OV/bin
```

在 HP-UX/Linux/Solaris 上：

```
/opt/OV/bin
```

b. 运行以下命令：

```
ovcodautil -ds SCOPE –o <Class> -m <Metrics> -rawonly
```

在此实例中：

<Class> 是度量类。

<Metrics> 是必须计算其基线数据的度量。

检查输出中是否已列出基线配置文件中提到的度量。

[1]历史数据是截止到前一小时收集的数据，并存储在度量数据存储区中。

第 11 章: 节点解析概述

一个节点可以有单个或多个网络接口。与每个网络接口关联的 IP 地址可能都有一个主机名。HPE Operations Agent 根据系统配置使用操作系统特定的 API 选择主机名(例如 gethostname /getaddrinfo /getnameinfo)。它会自动配置 OPC_NODENAME 参数，该参数提供 HPE Operations Agent 使用的本地主机名值。OPC_NODENAME 是一个内部参数，您无法配置。当节点具有多个 IP 地址或主机名时，可以通过将 OPC_IP_ADDRESS 设置为特定 IP 地址来覆盖默认配置。将与指定 IP 地址关联的主机名分配到 OPC_NODENAME。如果未将 IP 地址关联到主机名，则可以将 OPC_NAMESRV_LOCAL_NAME 配置为特定主机名。可以使用位于 `xpl.net` 命令空间下的 LOCAL_NODE_NAME 变量覆盖系统的本地节点名称。为 OPC_NODENAME、OPC_IP_ADDRESS 和 LOCAL_NODE_NAME 定义的值用于确定策略中的多个变量。消息变量如下：

- <$MSG_GEN_NODE>：返回发送消息的节点的 IP 地址。
- <$MSG_GEN_NODE_NAME>：返回发送消息的节点的主机名。
- <$MSG_NODE>：返回发生原始事件的节点的 IP 地址。
- <$MSG_NODE_NAME>：返回发生原始事件的节点的主机名。

备注: OPC_NODENAME、OPC_IP_ADDRESS 和 OPC_NAMESRV_LOCAL_NAME 参数可在 `eaagt` 命名空间下找到。

以下是一些示例：

示例 1：

以下是多网络接口环境中的 IP 地址和主机名：

```
IP1  abc.test.com  abc

IP2  xyz.test.com  xyz
```

其中 IP1 和 IP2 是两个不同的 IP 地址。

abc.test.com 和 xyz.test.com 是完全限定域名 (FQDN)。

abc 和 xyz 是主机名。

xyz 是本地系统的主机名。

OPC_NODENAME、<$MSG_GEN_NODE_NAME> 这两个参数会自动将 FQDN 值设置为 xyz.test.com，<$MSG_GEN_NODE> 参数会自动将 IP 地址设置为 IP2。这些配置是默认发生的，因为 xyz 是本地系统的主机名。

使用以下命令可选择配置其他 IP 地址(如 IP1)：

ovconfchg –ns eaagt –set OPC_IP_ADDRESS IP1

完成此配置后，参数 OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 会将 FQDN 值设置为 abc.test.com。<$MSG_GEN_NODE> 会将 IP 地址设置为 IP1。

示例 2：

以下是多网络接口环境中的 IP 地址和主机名：

```
IP1  xyz.test.com  xyz

IP2  xyz.test.com  xyz
```

其中 IP1 和 IP2 是两个不同的 IP 地址。

xyz.test.com 是完全限定域名 (FQDN)。

xyz 是本地系统的主机名。

<$MSG_GEN_NODE> 参数自动将 IP 地址设置为 IP1。此配置是默认发生的，因为 xyz 是本地系统的主机名，并且 IP1 是与 xyz 关联的第一个 IP 地址(按 IP 地址的顺序)。

使用以下命令可选择配置其他 IP 地址(如 IP2)：

ovconfchg –ns eaagt –set OPC_IP_ADDRESS IP2

完成此配置后，<$MSG_GEN_NODE> 会将 IP 地址设置为 IP2。

示例 3：

在多网络接口环境中，Windows IPv6 系统有四个已配置的 IP 地址，顺序如下：

```
IP1

IP2

IP3

IP4
```

这些 IP 地址按以下顺序与主机名关联；IP1 未与主机名关联。

```
IP3  xyz.test.com  xyz

IP2  xyz.test.com  xyz

IP4  xyz.test.com  xyz
```

默认情况下，<$MSG_GEN_NODE> 参数会自动将 IP 地址设置为 IP2，因为 IP2 是第一个配置的与本地系统名称关联的 IP 地址(按 IP 地址的顺序)。

使用以下命令可选择配置其他 IP 地址：

ovconfchg –ns eaagt –set OPC_IP_ADDRESS <IP_address>

完成此配置后，<$MSG_GEN_NODE> 将设置为特定 IP 地址。

示例 4：

以下是单个网络接口环境中的 IP 地址和主机名：

```
IP1  abc.test.com  abc

IP1  xyz.test.com  xyz
```

其中 IP1 是 IP 地址。

abc.test.com 和 xyz.test.com 是完全限定域名 (FQDN)。

abc 和 xyz 是主机名。

xyz 是本地系统的主机名。

默认情况下，OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 这两个参数会自动将 FQDN 值设置为 xyz.test.com，因为 xyz 是本地系统主机名。

使用以下命令可选择配置 IP 地址 IP1：

ovconfchg –ns eaagt –set OPC_IP_ADDRESS IP1

完成此配置后，OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 会将 FQDN 值设置为 abc.test.com。之所以这样配置，是因为 abc.test.com 是与 IP 地址 IP1 关联的第一个 FQDN。

示例 5：

以下是多网络接口环境中的 IP 地址：

```
IP1

IP2  xyz.test.com  xyz
```

其中 IP1 和 IP2 是两个不同的 IP 地址。

IP1 未与主机名关联。

xyz.test.com 是完全限定域名 (FQDN)。

xyz 是本地系统的主机名。

默认情况下，OPC_NODENAME、<$MSG_GEN_NODE_NAME> 这两个参数会自动将 FQDN 值设置为 xyz.test.com，<$MSG_GEN_NODE> 参数会自动将 IP 地址设置为 IP2。

使用以下命令可选择配置其他 IP 地址(如 IP1)：

ovconfchg –ns eaagt –set OPC_IP_ADDRESS IP1

完成此配置后，会将 OPC_IP_ADDRESS 设置为 IP1 并且不与主机名关联。可以使用以下命令配置 OPC_IP_ADDRESS 对应的主机名：

ovconfchg –ns eaagt –set OPC_NAMESRV_LOCAL_NAME<host_name>

其中 <host_name> 是类似 abc.test.com 的任何名称。运行这两个命令后，OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 这两个参数会将 FQDN 值设置为 abc.test.com。

注：在 NAT 环境中设置的 OPC_IP_ADDRESS 参数的行为与以上示例中所述相同。

示例 6：

以下是单个网络接口环境中的 IP 地址和主机名：

```
IP1  abc.test.com  abc xyz

IP1  xyz.test.com  xyz
```

其中 IP1 是 IP 地址

abc.test.com 和 xyz.test.com 是完全限定域名 (FQDN)。

abc 和 xyz 是主机名。

xyz 是本地系统的主机名。

默认情况下，OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 这两个参数会自动将 FQDN 值设置为 abc.test.com。如果外部事件来自名为 xyz 的节点，则 <$MSG_NODE_NAME> 参数会自动设置为 xyz。

您可以通过执行以下某个选项，将 OPC_NODENAME、<$MSG_GEN_NODE_NAME> 和 <$MSG_NODE_NAME> 配置为 FQDN 值 xyz.test.com：

- 交换主机文件中条目的顺序。
- 从第一个条目中删除别名 **xyz**。

示例 7：

以下是两个系统的 IP 地址和主机名：

```
IP1  abc.test.com  abc

IP2  xyz.test.com  xyz
```

其中 IP1 和 abc 是远程系统的 IP 地址和主机名。

IP2 和 xyz 是本地系统的 IP 地址和主机名。

abc.test.com 和 xyz.test.com 是完全限定域名 (FQDN)。

事件来自 IP1 时(例如，来自远程节点 abc 的陷阱)，默认发生以下配置：

- 参数 <$MSG_NODE> 设置为 IP1，因为 <$MSG_NODE> 返回发生原始事件的节点的 IP 地址。
- 参数 <$MSG_NODE_NAME> 设置为 abc，因为 <$MSG_NODE_NAME> 返回发生原始事件的节点的主机名。
- 参数 <$MSG_GEN_NODE> 设置为 IP2，因为 <$MSG_GEN_NODE> 返回发送消息的节点的 IP 地址。
- 参数 <$MSG_GEN_NODE_NAME> 设置为 xyz，因为 <$MSG_GEN_NODE_NAME> 返回发送消息的节点的主机名。

配置 LOCAL_NODE_NAME 变量后，HPE Operations Agent 会将 LOCAL_NODE_NAME 用作本地系统的主机名。

要配置其他本地主机名(如 xyz)，请使用以下命令：

```
ovconfchg -ns xpl.net -set LOCAL_NODE_NAME xyz
```

完成此配置后，参数 OPC_NODENAME 和 <$MSG_GEN_NODE_NAME> 会将 LOCAL_NODE_NAME 变量值设置为 xyz。

第 12 章：记录和跟踪

可使用记录和跟踪机制诊断和排除 HPE Operations Agent 中的问题。HPE 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 HPE 支持供进一步分析。

记录

HPE 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 HP-UX/Linux/Solaris 上：

```
/var/opt/OV/log
```

此外，HPE 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%\log\System.txt
- %ovdatadir%\status.perfalarm
- %ovdatadir%\status.ttd
- %ovdatadir%\status.mi
- %ovdatadir%\status.perfd-<port>
- %ovdatadir%\hpcs\hpcstrace.log

提示：在此实例中，<port> 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，请参见配置 RTMA 组件。

在 HP-UX/Linux/Solaris 上：

- /var/opt/OV/log/System.txt
- /var/opt/perf/status.perfalarm
- /var/opt/perf/status.ttd
- /var/opt/perf/status.mi

- /var/opt/perf/status.perfd
- /var/opt/OV/hpcs/hpcstrace.log

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 HPE 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 HP-UX/Linux/Solaris 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 HPE 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)。

下表中列出了跟踪的命令行选项和参数：

命令行选项

命令选项	描述
-host	指定要连接到的计算机的主机名。 语法：-host <host_name>

命令行选项(续)

命令选项	描述
-off	关闭所有跟踪。
-cf|-configuration	指定动态更改的配置文件的名称。 语法：-cf <config_file_name>
-vc|-viewconfig	显示所有应用程序的跟踪配置。
-app|-application	指定要配置的应用程序名称的列表。
-cm|-component	指定要配置的组件名称的列表。
-sink	指定跟踪输出文件名。 语法：-sink <filename>
-gc|-generate_configuration	指定要生成的配置文件的名称。 语法：-gc <file_name>
-rd|-resetdefault	将所有 .ini 文件重置为其默认内容。
-h|-help	显示命令行语法。
-version	显示工具版本号。

开始跟踪 HPE Operations Agent 进程之前，执行以下任务：

1. 标识应用程序
2. 设置跟踪类型
3. 可选。创建配置文件

标识应用程序

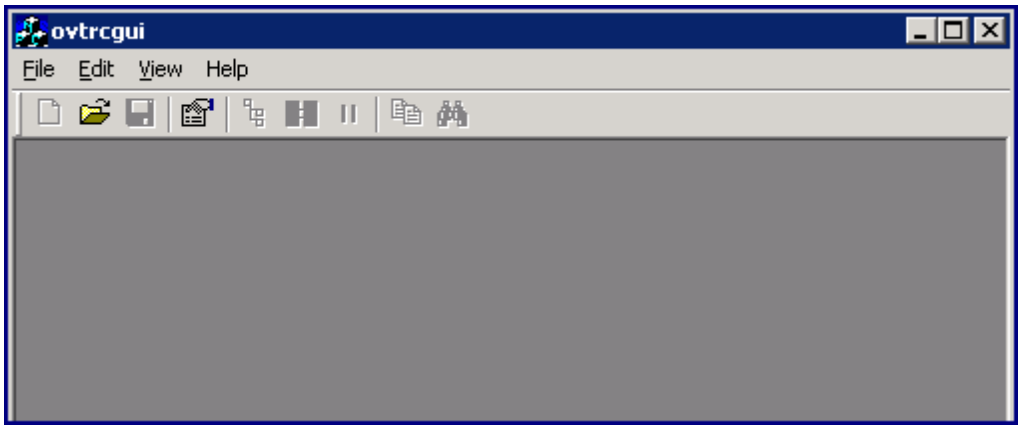
在受管系统上，标识要跟踪的 HPE 软件应用程序。使用 ovtrccfg -vc 选项查看所有启用了跟踪的应用程序的名称，以及为每个启用了跟踪的应用程序定义的组件和类别的名称。

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。

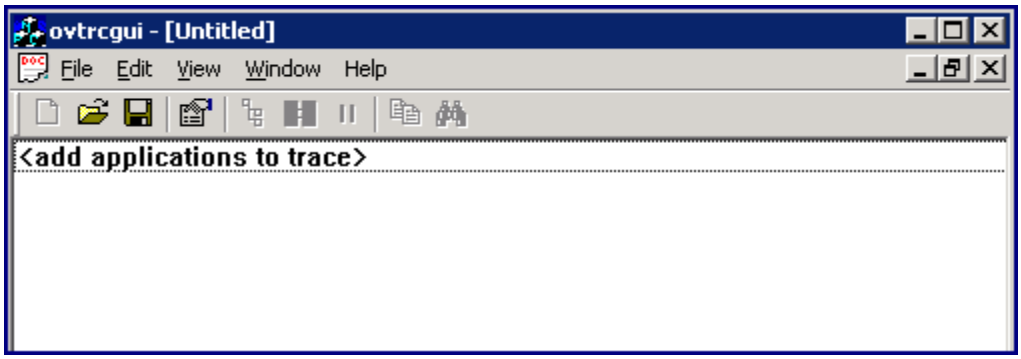
备注: 仅 Windows 系统支持 ovtrcgui 实用程序。

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：

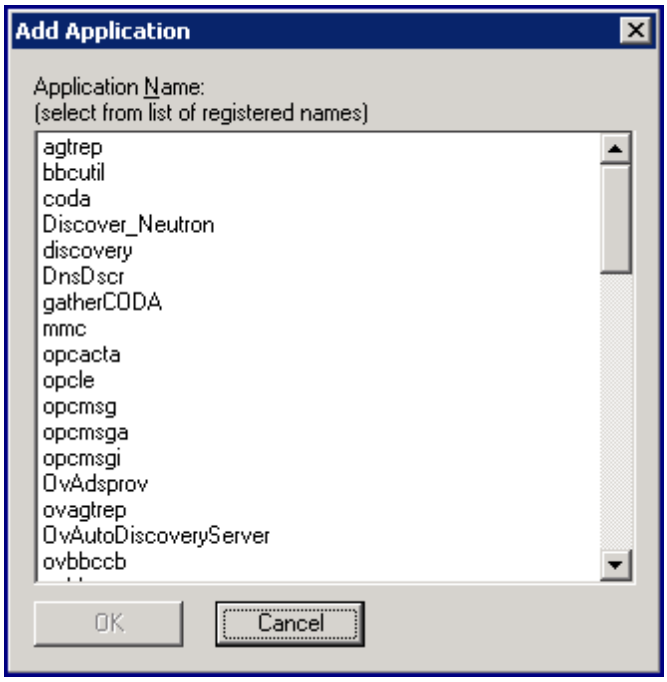
1. 从 %OvInstallDir%\support 目录运行 ovtrcgui.exe 文件。将打开 ovtrcgui 窗口。



2. 在 ovtrcgui 窗口中，单击**文件 → 新建 → 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 → 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. 转到位置 <data_dir>/conf/xpl/trc/
2. 查找 <application_name>.ini 文件。如果文件存在，则转到下面的步骤 3。
   如果 <application_name>.ini 文件不存在，请执行以下步骤：
   - 用文本编辑器创建新文件。
   - 按给定顺序将以下属性添加到该文件中：DoTrace、UpdateTemplate 和 DynamicTracing。

     提示: 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 <application_name>.ini 文件。

4. 要启用 static tracing，请确保 DoTrace 属性设置为 ON，DynamicTracing 属性设置为 OFF。
5. 要启用 dynamic tracing，请确保 DoTrace 和 DynamicTracing 属性设置为 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 <version_number>

APP:"<application_name>"

SINK:File "<file_name>" "maxfiles=[1..100];maxsize=[0..1000];"

TRACE:"<component_name>" "<category_name>" <keyword_list>
```

以下部分详细说明了每一行语法。

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下面的“用命令行工具启用跟踪并查看跟踪消息”部分。

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。

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. 使用 ovtrccfg 命令启用对配置文件中提及的特定应用程序的动态跟踪。
   `/opt/OV/support/ovtrccfg -cf <configuration_file_name>`
   其中 <configuration_file_name> 是上一节“创建配置文件”中创建的跟踪配置文件的名称。

   备注: 如果不想使用跟踪配置文件，可以用以下命令启用跟踪：

   `/opt/OV/support/ovtrccfg -app <application>[-cm <component>]`

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：
     ```
     /opt/OV/support/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：
     ```
     /opt/OV/support/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：
     ```
     /opt/OV/support/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：
   ```
   /opt/OV/support/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 HPE 软件联机支持进行评估。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。这些文件带有扩展名 .trc 和后缀 n(其中 n 是介于 1 和 99999 之间的整数)。

   例如：

   ```
   SINK:File "coda.trc" "force=0;maxfiles=10;maxsize=10;"
   root@/var/opt/OV/tmp/trc->lrt
   total 93092
   -rw-r----- 1 root bin 10475620 Aug 12 15:30 coda_00013.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00014.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00015.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00016.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00017.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00018.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00019.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:30 coda_00020.trc
   -rw-r----- 1 root bin 10475528 Aug 12 15:31 coda_00021.trc
   -rw-r----- 1 root bin  1027187 Aug 12 15:31 coda_00022.trc
   root@/var/opt/OV/tmp/trc->
   ```

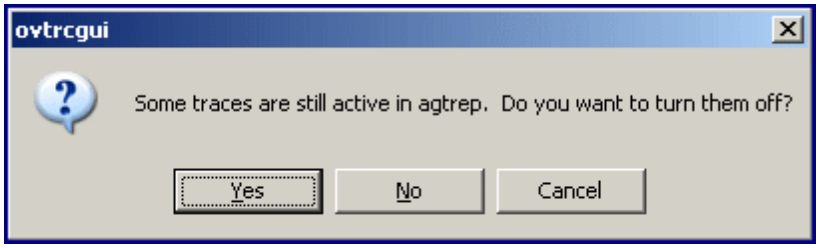
用跟踪 GUI 启用跟踪并查看跟踪消息

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. 执行使用跟踪 GUI (第 220 页)中的步骤 1 到步骤 6。
2. 关闭跟踪配置编辑器。
3. 提示保存对 Untitled 的更改时，单击**否**。
   将出现以下消息：



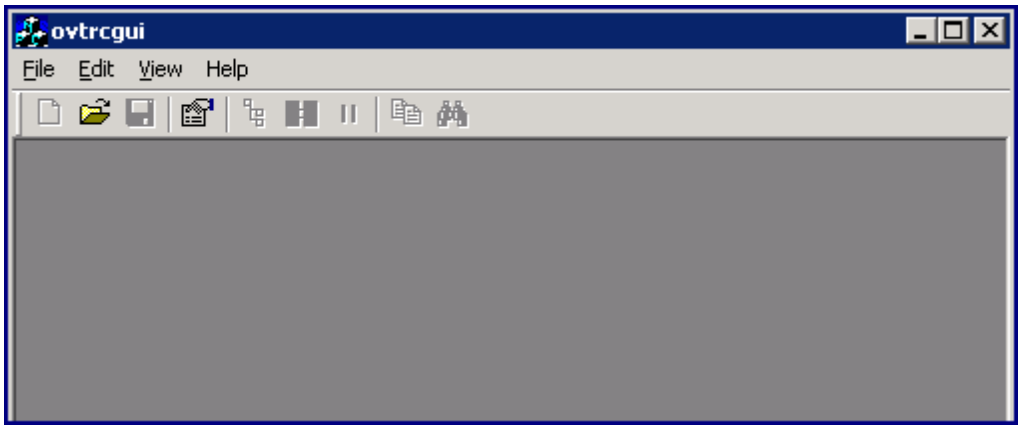
4. 单击**否**。如果单击**是**，ovtrcgui 实用程序将立即禁用跟踪机制。

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)**。

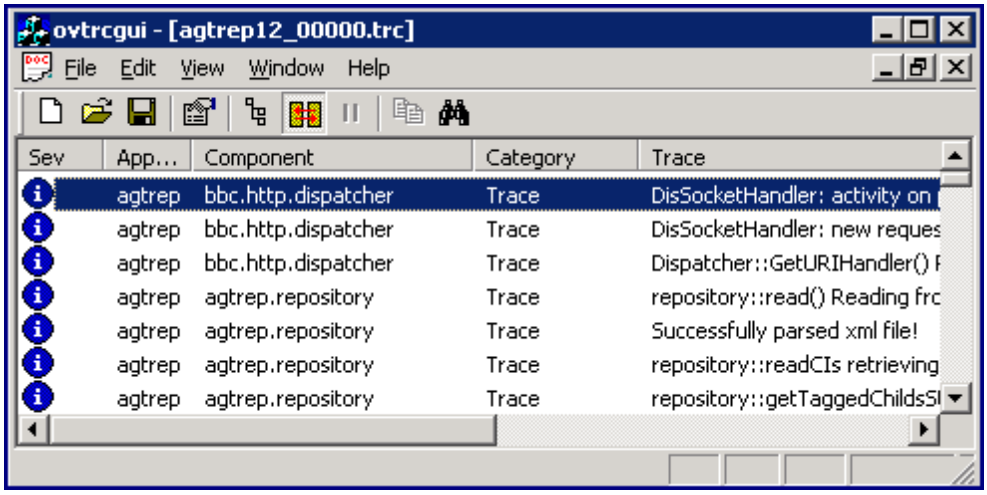
查看跟踪消息

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。

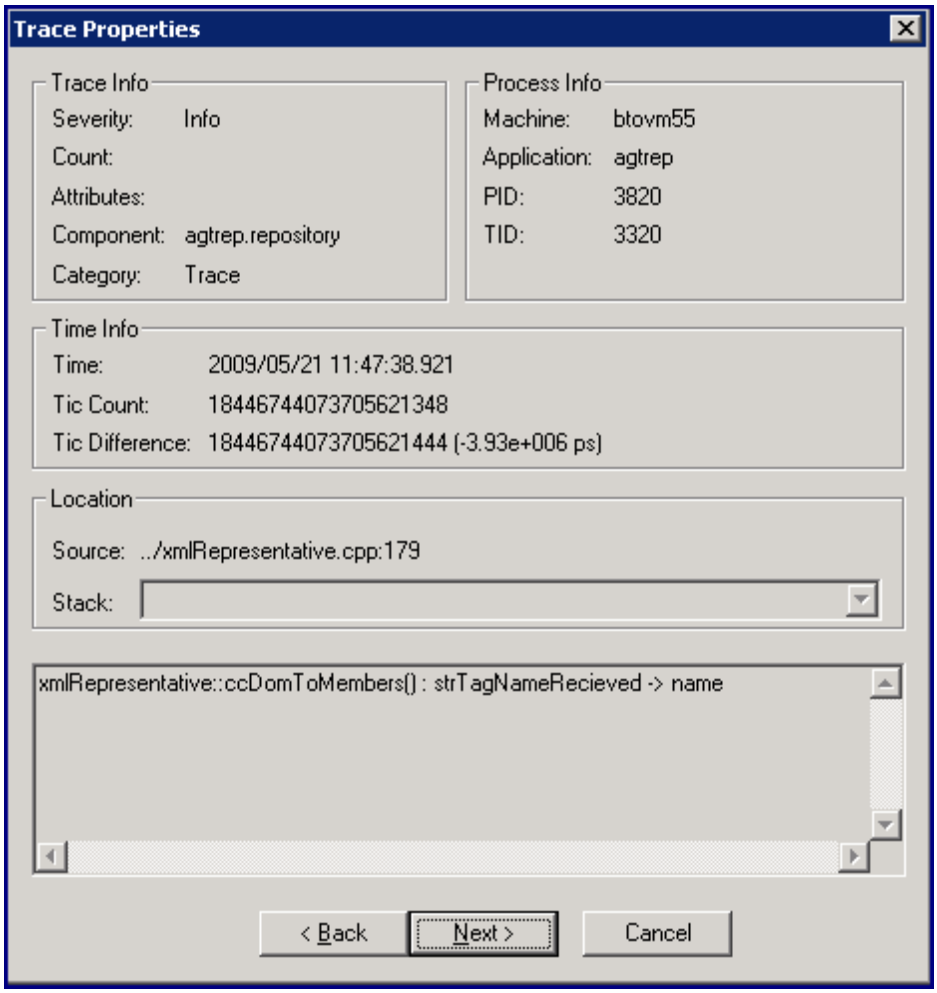


2. 单击**文件 (File)** → **打开 (Open)**。将打开“打开 (Open)”对话框。
3. 导航到放置跟踪输出文件的位置，选择 `.trc` 文件，然后单击**打开 (Open)**。ovtrcgui 实用程序显示了 `.trc` 文件的内容。



`.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：跟踪消息的严重性。
  - 计数：消息的序列号。
  - 属性：跟踪消息的属性。
  - 组件：发出跟踪消息的组件的名称。
  - 类别：跟踪的应用程序分配的任意名称。
- 进程信息 (Process Info)：
  - 计算机：节点的主机名。
  - 应用程序：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：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：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**下一个**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**。

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。

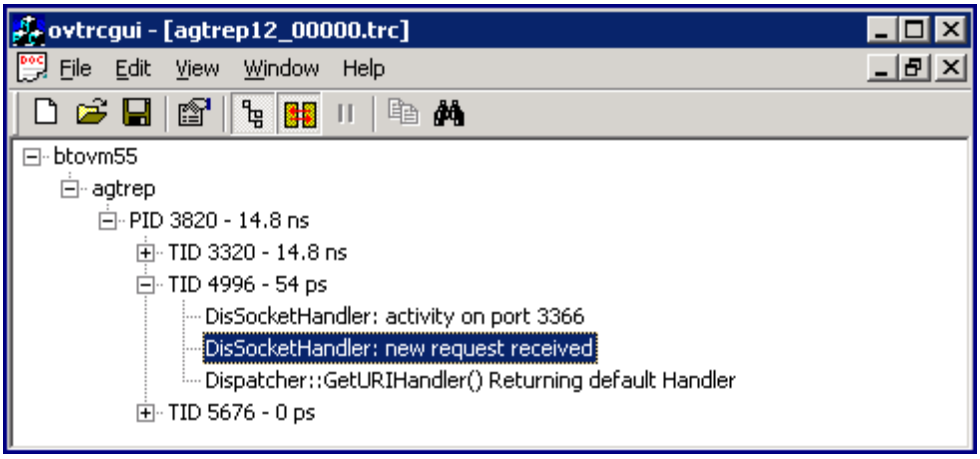
跟踪列表视图使用以下列显示每条跟踪消息：

表 13：跟踪列表视图

列	描述	
严重性	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：	
	信息	
	警告	
	错误	
应用程序	显示跟踪的应用程序的名称。	
组件	显示生成跟踪消息的被跟踪应用程序之组件的名称。	
类别	显示跟踪消息的类别。	
跟踪	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。

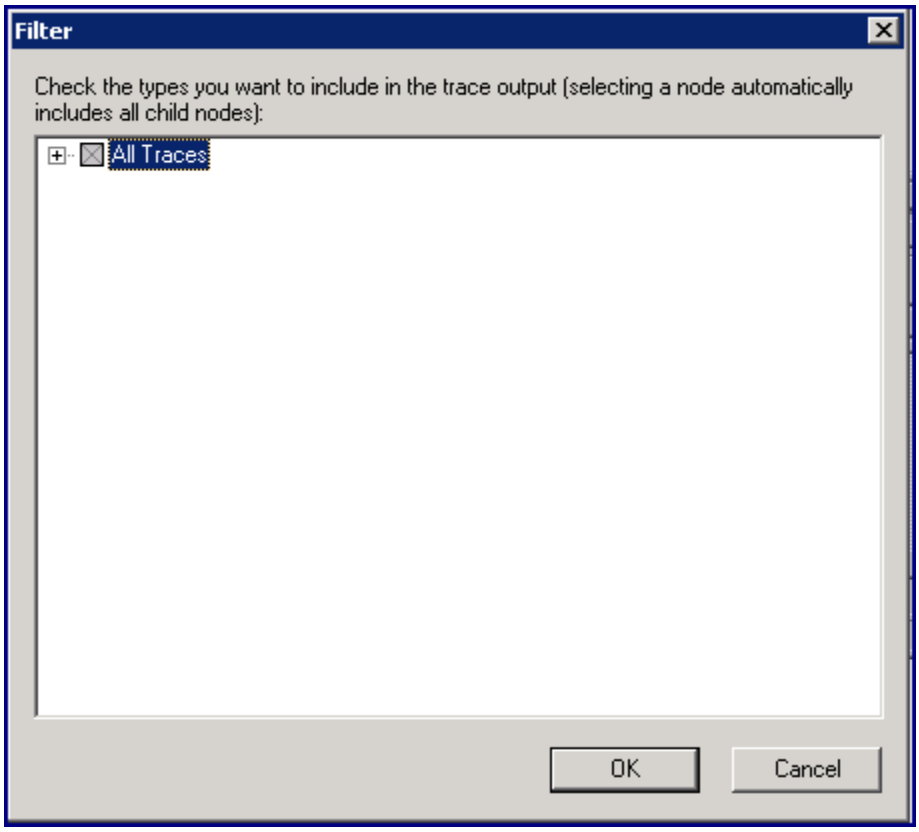


可展开进程 ID 和线程 ID 查看跟踪消息。要回到跟踪列表视图，请单击 。

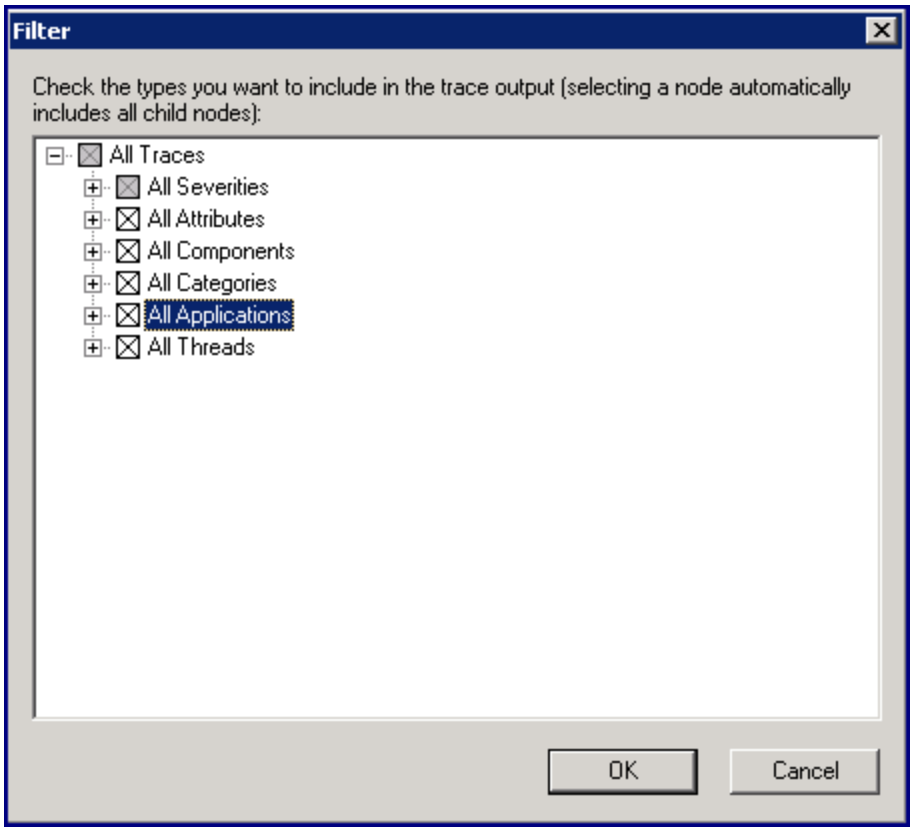
过滤跟踪

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 → 过滤 (Filter)**。将打开“过滤 (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。

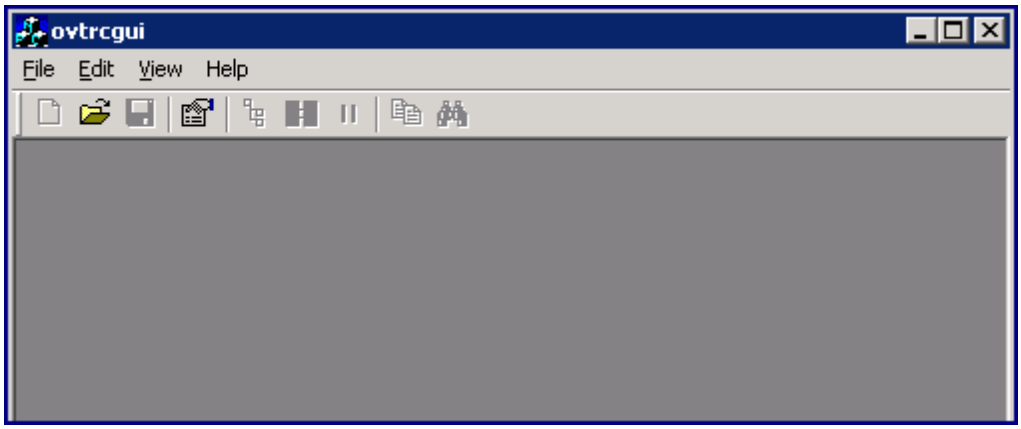


3. 展开参数可进行过滤跟踪消息的选择。
4. 单击**确定**。可以在 ovtrcgui 控制台中只看到过滤后的消息。

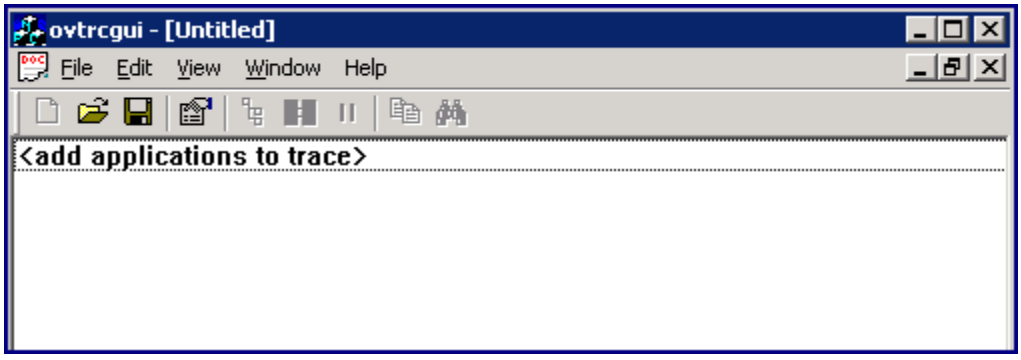
使用跟踪 GUI

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：

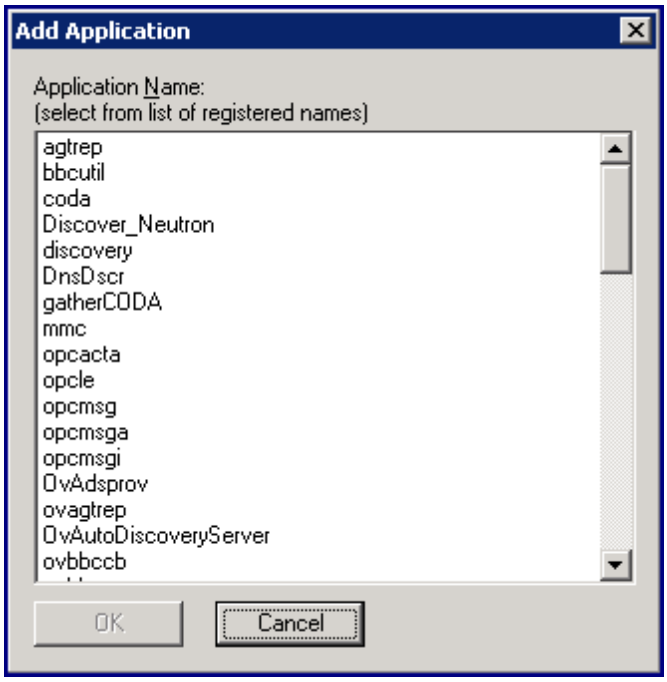
1. 从 %OvInstallDir%\support 目录运行 ovtrcgui.exe 文件。将打开 ovtrcgui 窗口。



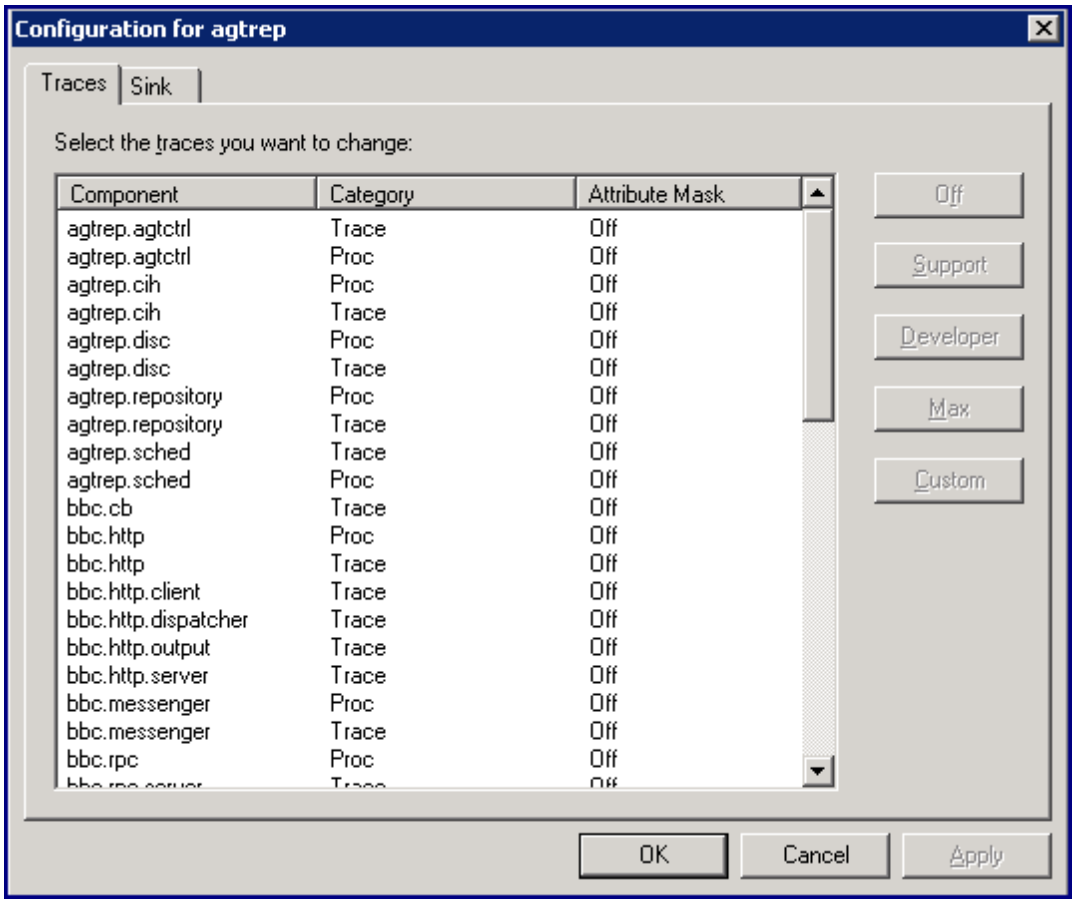
2. 在 ovtrcgui 窗口中，单击**文件 (File)** → **新建 (New)** → **跟踪配置 (Trace Configuration)**。将打开新的跟踪配置编辑器。



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。

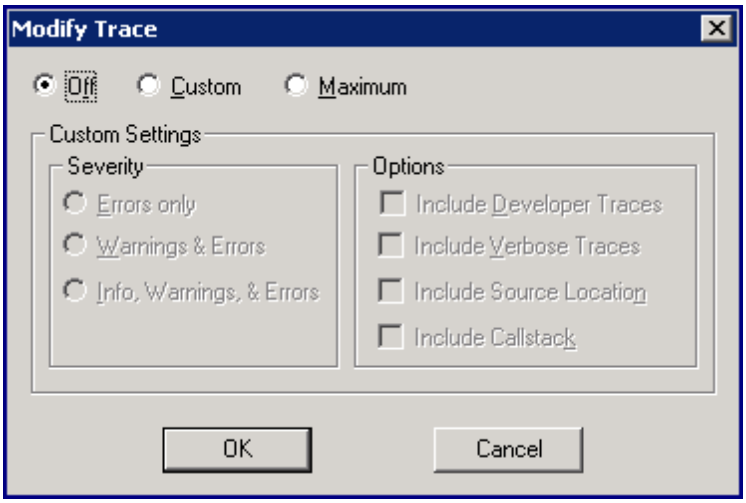


4. 选择要跟踪的应用程序，然后单击**确定**。“<application> 的配置 (Configuration for <application>)”窗口将打开。



“<application> 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关闭。

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：
   - **支持 (Support)**：单击它将收集标记为信息通知的跟踪消息。
   - **开发人员 (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)”，“修改跟踪 (Modify Trace)”窗口将打开。



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**。

提示: 在“<application> 的配置 (Configuration for <application>)”窗口中，可单击**关闭** 禁用组件类别对的跟踪。

6. 单击**确定**。
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`。

   提示: 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 maxfiles)。
10. 从下拉列表指定最大文件大小(请参见 maxsize)。
11. 单击**应用**。

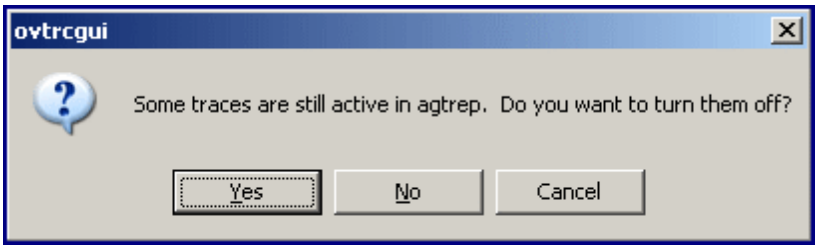
12. 单击**确定**。

    备注: 单击**确定**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。

    在“另存为”对话框中，浏览到合适的位置，在“文件名”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)**。

14. 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**，跟踪机制将继续跟踪系统上配置的应用程序。如果单击**是**，ovtrcgui 实用程序将立即禁用跟踪机制。

第 IV 部分: 记录自定义数据

HPE Operations Agent 可用于将自定义数据记录到度量数据存储区和默认性能度量类中。可以使用提交的 API 或 DSI 将自定义数据记录到度量数据存储区中。

使用基于 Perl 的可脚本化 API 可以简化将自定义数据记录到度量数据存储区中的流程。可以使用 API 记录多实例数据和 64 位数据类型。有关 API 的详细信息，请参见使用应用程序接口。

要使用 DSI 提交自定义数据，必须创建类规范文件，该文件定义用于记录数据并将数据记录到度量数据存储区中的模型。不能使用 DSI 记录 64 位数据类型和多实例数据。有关 DSI 的详细信息，请参见数据源集成概述。

备注: 建议使用 API 将自定义数据记录到度量数据存储区中。

第 13 章: 使用 Perl 应用程序编程接口提交自定义数据

自定义数据的记录过程已通过基于 Perl 的可脚本化 API 进行了简化。可使用 API 将自定义数据无缝地导入度量数据存储区中。使用 API，您可以将 64 位数据类型和多实例数据记录到数据存储区中。

备注: 仅当 **oacore** 进程运行时，可使用 API 将数据记录到度量数据存储区中。

HPE Operations Agent 12.xx 提供以下 API 用于将自定义数据记录到度量数据存储区中：

API 类型	**汇总类型**	**数据类型**	**如何选择 API 类型**
AddCounter	提供汇总间隔的最后一个值。	整型	使用计数度量将 CPU 数、物理 IO 数、分页计数、网络数据包计数等活动的累计计数提交到数据存储区。
AddGauge	提供汇总间隔内所有值的平均值。	整数、实数	使用量表度量将观察期的运行队列、用户数、文件系统空间利用率等即时值提交到数据存储区。
AddAttribute	提供不频繁更改的数据。所有汇总均不适用。 **备注:** 属性数据的历史记录不保存在数据存储区中。在数据存储区中，属性一旦更改，即被替换。	整数、实数、字符串	使用属性度量将操作系统名称、版本、发布、物理内存、CPU 时钟速度等静态定义或值提交到数据存储区。

备注:

- 请确保未提交使用不同 API 类型的度量。
  在以下实例中:

  ```
  AddCounter("Website:Global:UserLogins", "www.abcdefg.com", "Cumulative User Logins", 6000);
  AddGuage("Website:Global:UserLogins", "www.abcdefg.com", "Cumulative User Logins", 5000);
  ```

  当使用计数器和量表 API 类型提交“UserLogins”度量时，将得到错误结果。
- 提交度量后，将无法更改度量的类型。

例如: 将度量作为计数器提交后，无法将其更改为量表或属性。

文档使用以下术语:

- MetricObservation 对象包含记录到数据存储区的数据和收集时间戳。
- MetricObservationList 包含多个 MetricObservation 对象。
- OAAccess 是连接到数据存储区的客户端。
- SubmitObservations API 用于将 MetricObservationList 提交到度量数据存储区。
  SubmitObservations 将返回以下任一值:
  - 0 - MetricObservationList 已成功提交到度量数据存储区
  - -1 - MetricObservationList 提交失败
  - -2 - **oacore** 进程未运行

提交自定义数据

语法

```
<AddMethod>(<data source name>:<class name>:<metric name>, <Instance Identifier>, <Label of the metric>,<value of metrics>)
```

API 参数

下表列出了各种参数:

参数	描述
完全限定度量名称	指定完全限定度量名称。包括数据源名称、类名和度量名称。 <data source name>:<class name>:<metric name>

实例标识符	指定类的实例。可以是整数、实数或字符串值。
标签	指定度量的标签。
值	指定要提交的度量的值。 **备注:** 对于 AddGauge 方法，值可以是整数或实数值。 对于 AddAttribute 方法，值可以是整数、实数或字符串值。

例如：

AddGauge("MySQL:Global:MySQLMemUtil", "MyHost:3306", "MySQL Total CPU utilization percentage", 20.12);

其中：

AddMethod	AddGauge
Data source name	MySQL
Class name	Global
Metric name	MySQLMemUtil
Instance Identifier	MyHost:3306
Label of the metric	MySQL Total CPU utilization percentage
Value of metrics	20.12

如何将自定义数据提交到度量数据存储区中

要将数据提交到度量数据存储区中，请创建 MetricObservation 对象。将 MetricObservation 对象添加到 MetricObservationList 中。

备注: 可将任意数量的 MetricObservation 对象添加到 MetricObservationList 中。

使用 SubmitObservations API 将 MetricObservationList 提交到度量数据存储区中。

备注: 必须创建 OAAccess 对象以调用 SubmitObservations API。

下表列出了用于将数据提交到度量数据存储区中的步骤以及示例：

编号	用于将自定义数据提交到度量数据存储区中的步骤	示例
1.	创建 OAAccess。	$access = oaperlapi::OAAccess->new();
2.	创建 MetricObservationList 对象。	$mol = oaperlapi::MetricObservationList->new();
3.	使用时间戳创建 MetricObservation 对象。 **备注:** 可以使用不同时间戳创建多个 MetricObservation 对象。如果时间为零，则将采用当前时间。 对于您提交的每个数据类，请确保 MetricObservations 的时间戳采用递增顺序。	$interval = time; $mo = oaperlapi::MetricObservation->new($interval);
4.	使用 AddGauge 或 AddAttribute 等方法将值添加到 MetricObservation 对象中。	$mo->AddAttribute ("MySql:Global:MySQLVersion", "MyHost:3306", "MySQL version", "5.6.0"); $mo->AddGauge ("MySql:Global:MySQLMemUtil", "MyHost:3306", "MySQL Total CPU utilization percentage", 20.12);
5.	将 MetricObservation 对象添加到 MetricObservationList 对象中。	$mol->AddObservation($mo);
6.	使用 SubmitObservations API 将 MetricObservationList 提交到度量数据存储区中。	$access->SubmitObservations ($mol);

例如：

```
$access = oaperlapi::OAAccess->new();

$mol = oaperlapi::MetricObservationList->new();

$interval = time;
```

```
$mo = oaperlapi::MetricObservation->new($interval);

$mo->AddAttribute("MySql:Global:MySQLVersion", "MyHost:3306", "MySQL version", "5.6.0");

$mo->AddGauge("MySql:Global:MySQLMemUtil", "MyHost:3306", "MySQL Total CPU utilization percentage", 20.12);

$mol->AddObservation($mo);

$access->SubmitObservations($mol);
```

使用 API 将数据提交到数据存储区的用例

场景

John 是 www.abcdefg.com 网站的网络管理员。他想分析从美国访问其网站的用户数。John 还想监视在上午 10:00 到 11:00 之间所产生的购买数。

描述

John 想知道以下值：

- 一小时内从美国登录的用户数。
- 一小时内产生的购买数。

角色

John - 网络管理员

前提条件

监视用户登录数和购买数的持续时间为一小时。

时间段	累计用户登录数	位置	一小时内的购买数
10:00 – 10:30	6000	美国	400
10:30 – 11:00	8000	美国	600

备注：

确保路径前缀为 `<InstallDir>/nonOV/perl/a/bin`

使用以下命令运行下列脚本

在 Windows X64 上：

```
perl -I <Installdir>\support -I <Installdir>\nonOV\perl\a\lib\site_perl\5.16.0\MSWin32-AMD64-multi-thread <perl script>
```

在 Windows X86 上：

```
perl -I <Installdir>\support -I
<Installdir>\nonOV\perl\a\lib\site_perl\5.16.0\MSWin32-x86-multi-
thread <perl script>
```

在 HP-UX PA-RISC 上：

运行 perl 脚本时，需要将 LD_PRELOAD 设置为 liboaperlapi.sl 的路径

示例：

```
LD_PRELOAD=/opt/OV/nonOV/perl/a/lib/site_perl/5.16.0/PA-RISC2.0-
thread-multi/liboaperlapi.sl /opt/OV/nonOV/perl/a/bin/perl <perl
script>
```

在 UNIX 上：

```
perl <perl script>
```

要将数据提交到数据存储区，请运行以下 Perl 脚本：

```
use oaperlapi;

$now_string = localtime;
```

 # 步骤 1：创建访问对象

```
$access = oaperlapi::OAAccess->new();
```

 # 步骤 2：创建 MetricObservationList 对象

```
$molist = oaperlapi::MetricObservationList->new();
$obsTimestamp = 1415939400;
```

注：在此实例中，$obsTimestamp 表示在数据存储区记录第一条记录的时间。

 # 步骤 3：使用时间戳创建 MetricObservation 对象

```
$mgbl = oaperlapi::MetricObservation->new($obsTimestamp);
```

 # 步骤 4：向 MetricObservation 对象添加值

```
$mgbl->AddAttribute("Website:Global:Location", "www.abcdefg.com", "Location",
"USA");
$mgbl->AddCounter("Website:Global:UserLogins", "www.abcdefg.com", "Cumulative
User Logins", 6000);
$mgbl->AddGauge("Website:Global:Purchases", "www.abcdefg.com", "User Purchases
in an hour", 400);
```

 # 步骤 5：将 MetricObservation 对象添加到 MetricobservationList

```
$molist->AddObservation($mgbl);
# Go to the next Timestamp
$obsTimestamp += 1800;
```

 # 重复步骤 3、4 和 5，创建另一个观察，如下所示：

```
$mgbl = oaperlapi::MetricObservation->new($obsTimestamp);
$mgbl->AddCounter("Website:Global:UserLogins", "www.abcdefg.com", "Cumulative
User Logins", 8000);
$mgbl->AddGauge("Website:Global:Purchases", "www.abcdefg.com", "User Purchases
in an hour", 600);
$molist->AddObservation($mgbl);
```

 # 步骤 6：批量提交数据

```
$access->SubmitObservations($molist);
```

以下数据将记录到数据存储区中：

```
===
11/14/14 10:00:00|GLOBAL_ID                    |www.abcdefg.com|
11/14/14 10:00:00|LOCATION                      |USA            |
11/14/14 10:00:00|USERLOGINS                    |          6000|
11/14/14 10:00:00|PURCHASES                     |        400.00|
===
11/14/14 10:30:00|GLOBAL_ID                    |www.abcdefg.com|
11/14/14 10:30:00|LOCATION                      |USA            |
11/14/14 10:30:00|USERLOGINS                    |          8000|
11/14/14 10:30:00|PURCHASES                     |        600.00|
```

第 14 章: 数据源集成概述

数据源集成 (DSI) 功能可帮助您记录自定义数据，定义警报，以及从新的数据源访问性能收集组件的 **oacore** 数据收集器所记录的度量之外的其他度量。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。

您可以使用 HP Performance Manager 查看由 DSI 记录的数据，以及由 HPE Operations Agent 的数据收集器记录的标准性能度量。也可以使用 extract 程序导出使用 DSI 记录的数据，用于显示在电子表格或类似分析包中。

升级到 HPE Operations Agent 12.xx

通过 HPE Operations Agent 12.01，由 DSI 收集的数据存储在度量数据存储区中。对于记录到度量数据存储区中的每个 DSI 数据类，将创建一个数据库文件。但是，要保持向后兼容性，命令行将继续支持 logfile 参数。

```
sdlcomp <Class specification file> <log file name>
```

DSI 编译器 **sdlcomp** 不会基于日志文件名称创建文件；日志文件名称用作数据源名称。

例如：

如果“/tmp/test_log”或“C:\test_log”用作日志文件参数(在命令行中)，test_log 将用作数据源名称。

在从 HPE Operations Agent 11.xx 升级到 12.xx 之前，请确保 DSI 数据源已添加到位于以下位置的 datasources 文件中：

在 HP-UX/Linux/Solaris 上：

```
/var/opt/OV/conf/perf
```

在 Windows 上：

```
%ovdatadir%\conf\perf
```

将以下条目添加到 datasources 文件：

```
DATASOURCE=<Datasource Name>LOGFILE=<DSI Logfile Path>
```

仅当 DSI 数据源添加到 datasources 文件中时，才会在度量数据存储区中创建用于记录 DSI 数据的模型。

备注: HPE Operations Agent 12.xx 不支持的任何选项都将被忽略，并显示如以下示例所示的消息：

例如：

```
MAX INDEXES is obsoleted and will be ignored
INDEX BY is obsoleted and will be ignored
CAPACITY is obsoleted and will be ignored
ROLL BY is obsoleted and will be ignored
ACTION is obsoleted and will be ignored
PRECISION is obsoleted and will be ignored
```

在 HP Operations Agent 11.xx 中删除记录到 CODA 数据库的自定义数据源

从 HPE Operations Agent 11.xx 升级到 12.xx 后，较旧的文件中的自定义数据和模型以只读模式保留。如果运行 `sdlutil` (或 `ddfutil`)命令，则不会删除这些旧数据源。

在升级到 HPE Operations Agent 12.xx 之前或之后，可以删除旧数据源。升级到 HPE Operations Agent 12.xx 后，请按照以下步骤删除记录到 CODA 数据库的所有自定义数据源：

1. 停止 **oacore** 进程
2. 删除完整的 CODA 日志文件集
3. 启动 **oacore** 进程
4. 运行 `sdlutil` 命令完成清理。

将删除 `%OvDataDir%\datafiles\coda*` 中的所有 CODA 日志文件。

DSI 的工作原理

执行以下步骤将 DSI 数据记录到度量数据存储区中：

1. 创建模型以将 DSI 数据记录到度量数据存储区中。有关详细信息，请参见创建模型以将 DSI 度量记录到度量数据存储区中。
   a. 创建类规范文件。有关详细信息，请参见创建类规范文件 (第 236 页)
   b. 运行以下命令以使用 DSI 编译器 **sdlcomp** 来编译类规范文件：

      ```
      sdlcomp <class specification file> <log file name>
      ```

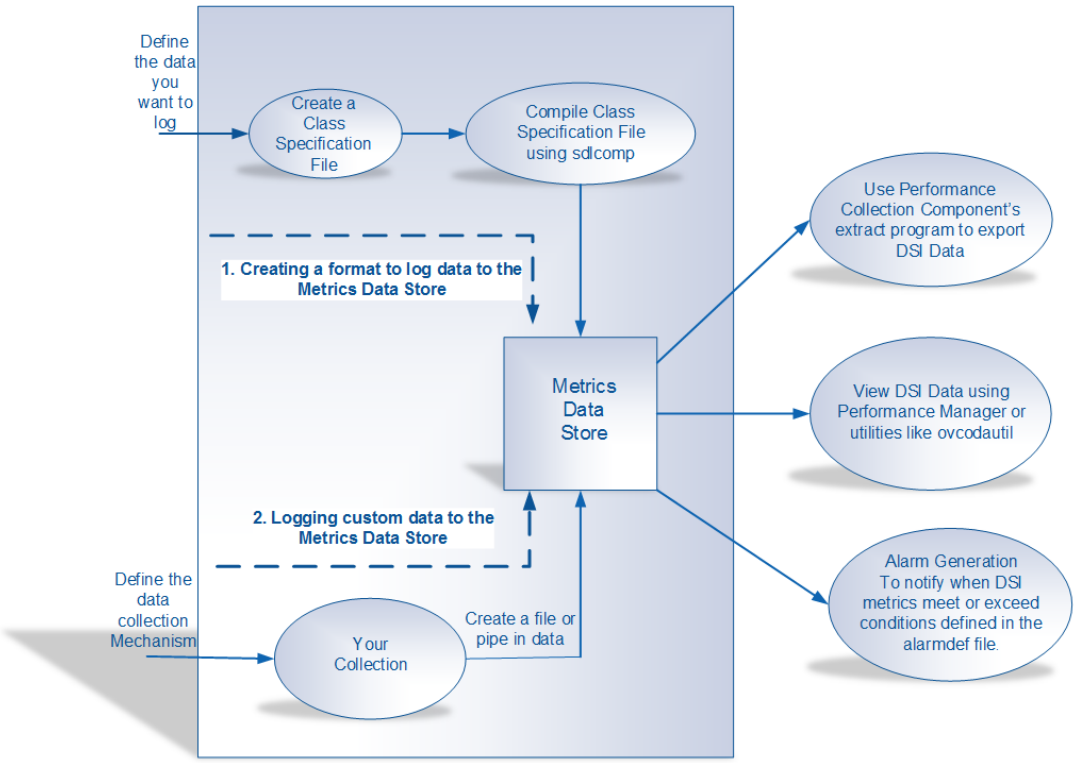
      有关详细信息，请参见使用 DSI 编译器编译类规范文件 (第 244 页)

   用于记录数据(元数据)的模型保存在度量数据存储区中。
2. 将 DSI 数据记录到度量数据存储区中。有关详细信息，请参见将 DSI 度量记录到度量数据存储区中 (第 246 页)

   通过管道将自定义数据发送到 dsilog。dsilog 会将数据记录到度量数据存储区中

下图显示如何将 DSI 收集的数据记录到度量数据存储区中。

图 17：数据源集成流程



可以使用性能收集组件的 **extract** 程序导出 DSI 数据。可以使用 HP Performance Manager 或 `ovcodautil` 等实用程序来查看 DSI 数据。可以将警报配置为在 DSI 度量超过 **`alarmdef`** 文件中定义的条件时发出通知。

第 15 章：创建模型以将 DSI 度量记录到度量数据存储区中

将 DSI 度量记录到度量数据存储区中之前，必须创建并记录用于记录 DSI 度量的模型。执行以下步骤将模型记录到度量数据存储区中：

1. 创建类规范文件
2. 将模型记录到度量数据存储区中

创建类规范文件

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模型。要创建该文件，请使用类规范语法 (第 237 页)。类规范文件包含：

- 将名称和数字 ID 分配给传入数据集的类描述。有关类描述的详细信息，请参见类描述 (第 237 页)。
- 定义要记录的度量的度量描述。度量描述定义名称并描述数据项。有关度量描述的详细信息，请参见度量描述。

要生成类规范文件，请使用任何可将文件另存为 ASCII 文本文件的文本编辑器或字处理器。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。)
- 度量名称

下面是类规范的示例：

```
CLASS VMSTAT_STATS = 10001
LABEL  "VMSTAT data"
RECORDS PER HOUR 120
;

METRICS
RUN_Q_PROCS    = 106
LABEL   "Procs in run q"
;

BLOCKED_PROCS   = 107
LABEL   "Blocked Processes"
```

```
;
```

可以在类规范文件中包括多个类。

当您运行 DSI 编译器 **sdlcomp** 时，度量数据存储区中记录了用于记录 DSI 数据的模型。

类规范语法

使用以下约定创建类规范文件：

- 方括号 [ ] 中显示的语法语句是可选的。
- 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。
- 注释以 # 或 // 开头。行内 # 或 // 后的所有内容都被忽略。
- 在每个类描述和度量描述后面添加一个分号。
- 语句不区分大小写。

备注: 用户定义的描述(如 metric_label_name 或 class_label_name)不能与 DSI 类规范语法的任何关键字元素相同。

例如：

```
CLASS class_name = class_id_number
[LABEL "class_label_name"]
[RECORDS PER HOUR  number]

;

METRICS
metric_name = metric_id_number
[LABEL "metric_label_name"]
;
```

类描述

要创建类描述，请向特定数据源中的一组度量分配名称。

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 class_name = class_id_number
[LABEL"class_label_name"]
[RECORDS PER HOUR number]
;
```

类

类名和类 ID 标识来自特定数据源的一组度量。

语法

```
CLASS class_name = class_id_number
```

如何使用

class_name 和 class_ID_number 都必须符合以下要求：

- class_name 由字母数字组成且最多包含 20 个字符。class_name 必须以字母字符开头，可以包含下划线(但不能包含特殊字符)。不区分大小写。
- class_ID_number 必须是一个数字且最多为 6 位数。
- 每个 class_name 和 class_ID_number 都必须对定义的所有类唯一，并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，请参见使用 parm 文件。

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 "class_label_name"]
```

如何使用

class_label_name 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 class_name 作为默认值。

示例

```
CLASS VMSTAT_STATS = 10001

LABEL "VMSTAT data";
```

每小时记录数

RECORDS PER HOUR 设置确定每小时写入到数据库文件中的记录数。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，然后才能成为数据库文件的一部分。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，请参见汇总方法。

语法

```
[RECORDS PER HOUR number]
```

如何使用每小时记录数

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
```

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。

示例

```
CLASS VMSTAT_STATS = 10001

LABEL  "VMSTAT data"

RECORDS PER HOUR  6;
```

在此示例中，每 10 分钟写入一条记录。

默认设置

类描述的默认设置是：

```
LABEL (class_name)

RECORDS PER HOUR 12
```

要使用默认设置，请只输入 CLASS 关键字及 class_name 和数字 class_id_number。

类规范示例

```
CLASS VMSTAT_STATS = 10001

LABEL  "VMSTAT data"

RECORDS PER HOUR 120;

METRICS

RUN_Q_PROCS  = 106

LABEL "Procs in run q"

;

BLOCKED_PROCS = 107

LABEL "Blocked Processes"

;

SWAPPED_PROCS = 108

LABEL "Swapped Processes"

;

AVG_VIRT_PAGES = 201

LABEL "Avg Virt Mem Pages"

;

FREE_LIST_SIZE = 202

LABEL "Mem Free List Size"

;
```

```
PAGE_RECLAIMS = 303
LABEL "Page Reclaims"
;
ADDR_TRANS_FAULTS = 304
LABEL "Addr Trans Faults"
;
PAGES_PAGED_IN = 305
LABEL "Pages Paged In"
;
PAGES_PAGED_OUT = 306
LABEL "Pages Paged Out"
;
PAGES_FREED = 307
LABEL "Pages Freed/Sec"
;
MEM_SHORTFALL = 308
LABEL "Exp Mem Shortfall"
;
CLOCKED_PAGES = 309
LABEL "Pages Scanned/Sec"
;
DEVICE_INTERRUPTS = 401
LABEL "Device Interrupts"
;
SYSTEM_CALLS = 402
LABEL "System Calls"
;
CONTEXT_SWITCHES = 403
```

```
LABEL "Context Switches/Sec"
;
USER_CPU = 501
LABEL "User CPU"
;
SYSTEM_CPU = 502
LABEL "System CPU"
;
IDLE_CPU = 503
LABEL "Idle CPU"
;
```

度量描述

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：

- 度量名称最多可包含 20 个字符。
- 该名称必须以字母开头。
- 度量名称只能包含字母数字字符和下划线。
- 不区分大小写。

度量 ID 号最多可包含 6 个字符。

metric-_name 和 metric_id_number 在类中定义的所有度量中都必须是唯一的。组合 class_name:metric_name 对于该系统而言必须是唯一的，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。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。

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。

备注: 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。

LABEL

度量标签标识 Performance Manager 图形和导出的数据中的度量。

语法

```
[LABEL "metric_label_name"]
```

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"。

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。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一次以下计算：

- 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。

例如：

```
METRICS
ITEM_1_3 = 11203
LABEL    "TOTAL_ORDERS"
TOTALED;
ITEM_1_5   = 11205
LABEL   "LINES_PER_ORDER"
SUMMARIZED BY ITEM_1_3;
```

使用 DSI 编译器编译类规范文件

使用类规范文件运行 DSI 编译器 **sdlcomp**。

编译器语法

```
sdlcomp <class specification file> <log file name>
```

在此实例中：

<class 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。

<log file name> 用作数据源名称。

DSI 编译器 **sdlcomp** 将不会创建基于日志文件名称的文件。日志文件名称用作数据源名称。

例如：

如果将“/tmp/test_log”或“C:\test_log”指定为日志文件集参数(在命令行中)，则 test_log 将用作数据源名称。

用于记录 DSI 数据的模型记录在度量数据存储区中

sdlcomp 编译器

sdlcomp 编译器检查类规范文件是否有错误。如果找不到错误，它会将数据源、类和度量描述添加到度量数据存储区。

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数据源：

1. 停止 `dsilog` 进程。
2. 从度量数据存储区导出所有类的数据。
3. 运行 `sdlutil` 以删除数据源。请参见使用 sdlutil 管理数据了解相关信息。
4. 更新类规范文件。
5. 运行 `sdlcomp` 以删除类规范。
6. 运行 `dsilog` 开始根据新的类规范记录。

第 16 章: 将 DSI 度量记录到度量数据存储区中

要将 DSI 度量记录到度量数据存储区中，请确保将用于记录 DSI 数据的模型记录到数据存储区中。修改警报定义文件 **alarmdef** 以通知 DSI 度量超过定义的条件的时间。有关为 DSI 度量定义警报的详细信息，请参见定义 DSI 度量的警报。从命令行启动收集进程。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**(或使用其他方法发送到 **stdin**)。

语法

```
dsilog <sdl-file-name> <data-class-name> [options]
```

在此实例中：

`<sdl-file-name>` 是根自述文件的名称。

`<data-class-name>` 是数据类的名称。

表 1：`dsilog` 参数和选项

变量和选项	定义
-c <char>	使用指定的字符作为字符串分隔符。
-i <fifo>	指示输入应来自指定的 `fifo`。fifo 是输入流的备选。
-timestamp	时间戳需要指定为输入数据文件中的第一列。这需要采用 EPOCH 格式。
-?	显示语法描述。
-vers	显示打印版本信息。

备注: 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示例，请参见数据源集成示例

dsilog 记录进程

dsilog 进程要求您创建自己的程序或使用现有进程。然后可以通过管道将此数据发送到 dsilog，后者将数据记录到数据存储区中。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 name> <class name>
```

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(指定管道)。

例如：

```
mkfifo -m 777 myfifo
dsilog logfile_set class -i myfifo &
vmstat 60 > myfifo &
```

& 指示进程在后台运行。

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。

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。

使用 sdlutil 管理数据

使用 sdlutil 程序管理 DSI 数据库文件中的数据。您可以执行以下操作：

- 在 stdout 中列出定义的类和度量信息。可将输出重定向到文件。
- 从度量数据存储区中删除数据源、类、度量和数据
- 显示版本信息。

语法

sdlutil <log file name>[option]

变量和选项	定义
logfile name	用作数据源名称。
-rm all	从度量数据存储区中删除数据源、类、度量和数据
-vers	显示版本信息。
-?	显示语法描述。

例如：

运行以下命令从度量数据存储区中删除数据源 test_log 以及类、度量和数据。

```
sdlutil /tmp/test_log -rm all
```

第 17 章: 使用记录到度量数据存储区中的 DSI 数据

将 DSI 数据记录到度量数据存储区中后，可以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 监视警报。

定义 DSI 度量的警报

在性能收集组件系统上使用 `alarmdef` 文件定义 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。`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。有关详细信息，请参见“性能警报”章节中的 USE 语句。

要激活对 `alarmdef` 文件所做的更改，请在命令行中运行 `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，请参见性能警报。

警报处理

`dsilog` 记录数据时，会将该数据与 `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。本地操作可以在性能收集组件系统上执行。还可以将警报信息发送到 HP Operations Manager。

导出 DSI 数据

要从度量数据存储区导出 DSI 数据，请使用 `extract` 程序的 `export` 函数。有关详细信息，请参见使用 extract 程序。使用 `ovcodautil -obj` 命令查看可导出的 DSI 数据源和类的列表。

使用 extract 导出 DSI 日志文件数据的示例

```
extract -xp -l logfile_set -C class [options]
```

可以使用 `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。
- 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

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。它提供要在生产率受到影响之前解决问题所需的信息。

第 18 章: 数据源集成示例

数据源集成是一项功能强大且灵活的技术。本章包含使用 DSI 收集数据并将其记录到度量数据存储区中的示例。

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** 进程接收连续输入数据流。feed_one_data_row 编写为函数，它向单个 **dsilog** 进程提供连续数据流。

```
# 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。

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。

以下示例显示了用于描述要在名为 VMSTAT_STATS 的类中记录的前两个 vmstat 数字的类规范文件。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

当您使用 sdlcomp 编译类规范文件时，系统会检查该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。

使用为类规范指定的文件名，然后指定日志文件路径。日志文件路径中的日志文件名称将用作数据源名称。

备注: 不会创建任何日志文件。

在下面的命令和编译器输出示例中，**/tmp/vmstat.spec** 是类规范文件名，/tmp/VMSTAT_DATA 是日志文件路径。

```
-> sdlcomp /tmp/vmstat.spec /tmp/VMSTAT_DATA
```

此示例创建名为 VMSTAT_DATA 的数据源。如果类规范文件中有语法错误，则显示指示问题的消息，而不创建数据源。

要验证 VMSTAT_STATS 类是否已成功添加到数据源，请运行以下命令：

```
ovcodautil -ds VMSTAT_DATA -obj
```

启动 dsilog 记录进程

现在，可以通过管道将 vmstat 的输出直接发送到 **dsilog** 记录进程。使用以下命令：

```
vmstat 60 | dsilog /tmp/VMSTAT_DATA VMSTAT_STATS &
```

此命令每隔 60 秒运行一次 vmstat，并将输出直接发送到 VMSTAT_DATA 数据源中的 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 has invalid data.Ignore to end of line, metric value exceeds maximum.
```

此消息是 **dsilog** 无法记录的 vmstat 输出中的标题行结果。尽管屏幕上显示这条消息，但 **dsilog** 会继续运行，并开始用第一个有效输入行记录数据。

访问数据

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
```

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。

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"
;
METRICS
who_wc = 151
label "who wc"
averaged
;
sdlcomp ./who_wc.spec ./who_wc_log.
```

与 sar 不同，您不能指定具有 who 的间隔或迭代值，因此创建一个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
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
```

再次需要 fifo 和一个脚本向 **dsilog** 提供数据，因此请回到分步过程。

1. 创建两个 fifo；其中一个将是用来使真实输入 fifo“保持打开”的虚拟 fifo

   ```
   .#Dummy fifo.

   mkfifo ./hold_open.fifo

   # Real input fifo for dsilog.

   mkfifo ./who.fifo
   ```

2. 使用 -i 选项启动 **dsilog** 以指定来自 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 数据源脚本 (who_data_feed)。

   ```
   ./who_data_feed &
   ```

第 19 章: 使用度量流

HPE Operations Agent 可用于将自定义度量记录到度量数据存储区和默认系统性能度量类中。

通过 HPE Operations Agent 12.01，也可使用度量流功能对自定义和系统性能度量进行流式处理。您可以使用**度量流配置**策略配置度量流。

度量流可用于将度量数据流向目标订阅者 (例如：性能引擎)并使用该数据进行绘图和分析。您可以将数据提交到 **hpsensor** 并将**度量流配置**策略从 HPE Operations Manager i (OMi) 部署到 HPE Operations Agent 以对度量数据进行流式处理。有关**度量流配置**策略的详细信息，请参见《OMi 管理指南 (OMi Administration Guide)》 。

备注: HP Compute Sensor (hpsensor) 是轻量级**性能**和**日志数据**收集进程。

流度量

要对度量数据进行流式处理，提供了基于 REST 的接口，以便将数据提交到 **hpsensor**。**hpsensor** 会在配置的间隔内将已提交的数据发布到目标订阅者。

在将数据提交到 **hpsensor** 之前，必须使用提供的 REST API 将注册信息发布到 **hpsensor**。此外，还必须在 HPE Operations Manager i (OMi) 中创建策略类型为**度量流配置**的策略，以便选择要进行流式处理的度量的列表，提供目标 URL，然后将该策略从 OMi 部署到 HPE Operations Agent。

备注: 如果目标接收器不可用，则不会对度量进行流式处理。

以下 REST API 适用于从本地系统进行注册和数据提交：

1. 注册： **`/v4/RTCM/Register`**
2. 数据提交： **`/v4/RTCM/SubmitData`**

用法：

以下 URI 可用于使用 http 客户端将数据发布到 **hpsensor**：

- 对于注册： **`http://<localhost>:<port>/hpcs/v4/RTCM/Register`**
- 对于数据提交： **`http://<localhost>:<port>/hpcs/v4/RTCM/SubmitData`**

在此实例中，

`<localhost>` 是解析为回环地址的主机名。回环地址为： `127.0.0.1`(对于 IPv4) 和 `[::1]`(对于 IPv6)。

`<port>` 是 BBC 端口(默认情况下为 383)，也可以是 **hpsensor** 端口。

hpsensor 接受采用 **v4 JSON**(JavaScript 对象表示法)格式的数据。v4 JSON 格式的意义在于度量中心数据注册和数据提交。数据应包含以下对象：

1. 标题
2. 版本
3. 包含元数据和数据字段的度量负载

下面是 v4 JSON 格式的架构：

```
[{
"Title":"Real Time Metrics",
"Version":"v4",
"Metric Payload":[
     {
     "MetaData":{
          "MetricName":{
               "description":"The unique name of the metric",
               "type":"string"
          },
          "CategoryType":{
               "description":"The category type of the metric
(attribute/counter/gauge/...)",
               "type":"string"
          },
          "DataType":{
               "description":"The data type of the metric",
               "type":"string"
          },
          "ClassName":{
               "description":"The class name of the metric",
               "type":"string"
          },
```

```
          "Unit":{
               "description":"The unit of the metric value",
               "type":"string"
          },
          "Label":{
               "description":"The label of the metric",
               "type":"string"
          },
          "DataSource":{
               "description":"The data source the metric belongs
to",
               "type":"string"
          },
          "Key":{
               "description":"Key number of the metric, key value
0... n for Key metric",
               "type":"integer"
          },
          "Interface":{
               "description":"The interface of collection, either
Legacy or Stream",
               "type":"string"
          },
          "CollectorName":{
               "description":"The name of the metric collector",
               "type":"string"
          },
          "CollectorExec":{
```

```
                "description":"The collector executable to be
executed",
                "type":"string"
           },
           "OriginalMetricName":{
                "description":"The domain the metric belongs to",
                "type":"string"
           },
           "required":[
                "MetricName",
                "ClassName",
                "DataSource",
                "Key"
                     ]
           },
           "Data":{
                "Instances":[
                     {
                          "value":{
                               "description":"The value of the
metric for an instance",
                               "type":"string/integer/real"
                          },
                          "timestamp":{
                               "description":"The epoch time when
the metric was collected",
                               "type":"integer"
                          },
                          "dimensions":{
```

```
                                "keyinstance":{
                                     "description":"The key to
uniquely identify an instance",
                                     "type":"string/integer/real"
                                },
                                "minItems":1,
                                "uniqueItems": true
                           },
                           "minItems":1,
                           "uniqueItems": true
                      }
                 ]
            }
      }
]
}]
```

数据包含以下参数：

参数	必需		描述
	对于注册	对于数据提交	
标题	是	是	必须是 **"Real Time Metrics"**。
版本	是	是	必须是 **"v4"**，因为提供的 REST API 接受采用 v4 JSON 格式的数据。
MetricName	是	是	度量的唯一名称。
CategoryType	可选	可选	度量的类别类型。接受的类别类型为：密钥、属性、量表和计数器。
DataType	可选	可选	度量的数据类型。接受的数据类型为：INT32、UINT32、DOUBLE、INT64、UINT64、STRING、BOOL 和

			TIME。
描述	可选	可选	度量的描述。
标签	可选	可选	度量的标签。
单位	可选	可选	度量的单位。例如：%、kbps、mbps 等。
Key	是	是	此字段标识此度量是否为密钥度量。值 **-1** 表示它不是密钥，值 0 或以上表示，此度量是密钥度量。密钥度量的值定义类的实例。
ClassName	是	是	度量的类名称。
DataSource	是	是	度量的数据源。
CollectorName	是	可选	度量的收集器名称。
CollectorExec	可选	可选	具有可执行命令的度量的收集器的路径。
Interface	是	可选	度量的接口。流或旧版。当您希望在不登录到度量数据存储区的情况下将度量流向 **hpsensor** 时使用流接口，当您希望使用旧版接口(如 DDF 或批量提交)将数据记录到数据存储区并将该数据流向 **hpsensor** 时使用旧版接口。如果未指定，默认接口为旧版接口。
value	必须为密钥度量	是	度量的值。指定的值应与度量的数据类型匹配。
时间戳	必须为密钥度量	是	收集度量值的 Epoch 时间。
dimensions	可选	是	度量的关键实例名称。

注册

REST API **`/v4/RTCM/Register`** 用于将注册信息发布到 **hpsensor**。注册信息必须包含以下对象：

1. 标题
2. 版本
3. 包含元数据和数据字段的度量负载(适用于**密钥**度量实例)

例如：假定希望将注册信息发布到 **hpsensor** 并对 2 个自定义度量(1 个度量，1 个密钥度量)的数据进行流式处理的用例。对于此场景，可以发布采用 v4 JSON 格式的以下注册信息：

```
[{
"Title":"Real Time Metrics",
"Version":"v4",
"Metric Payload":[
     {
     "MetaData":{
          "MetricName":"Metric_ACTIVE_PROC",
          "CategoryType":"Attribute",
          "DataType":"DOUBLE",
          "Description":"Desc",
          "Label":"Active Proc",
          "Unit":"N/A",
          "Key":-1,
          "ClassName":"Class",
          "DataSource":"DataSouce",
          "CollectorName":"Collector",
          "CollectorExec":"$OvDataDir$bin/instrumentation/RTCM_
collector.sh Collector",
          "Interface":"Stream"
     }
},
     {
     "MetaData":{
          "MetricName":"Instance_NAME",
```

```
          "CategoryType":"Attribute",
          "DataType":"STRING",
          "Description":"Desc",
          "Label":"Metric Name1",
          "Unit":"N/A",
          "Key":0,
          "ClassName":"Class",
          "DataSource":"DataSOurce",
          "CollectorName":"Collector",
          "CollectorExec":"$OvDataDir$bin/instrumentation/RTCM_
collector.sh",
          "Interface":"Stream"
     },
     "Data":{
          "Instances":[{
               "value":"inst_1",
               "timestamp":1459724247
          }, {
               "value":"inst_2",
               "timestamp":1459724247
          }]
     }
     }
     ]
}]
```

使用 REST API **/v4/RTCM/Register** 将注册信息发布到 **hpsensor**。有关如何使用此 REST API 的详细信息，请参见用法部分。

数据提交

REST API **/v4/RTCM/SubmitData** 用于将自定义度量数据提交到 **hpsensor**。已提交的数据必须包含以下对象：

1. 标题
2. 版本
3. 包含元数据和数据字段的度量负载(适用于所有度量)

备注: 对于数据提交，必须使用所有度量的维度指定实例，否则将丢弃数据提交。

例如： 假定希望将自定义度量数据提交到 **hpsensor** 并对 2 个自定义度量(1 个度量，1 个密钥度量)的数据进行流式处理的用例。对于此场景，可以提交采用 v4 JSON 格式的以下数据：

```
[{
"Title":"Real Time Metrics",
"Version":"v4",
"Metric Payload":[
     {
     "MetaData":{
          "MetricName":"Metric_ACTIVE_PROC",
          "CategoryType":"Attribute",
          "DataType":"DOUBLE",
          "Description":"Desc",
          "Label":"Active Proc",
          "Unit":"N/A",
          "Key":-1,
          "ClassName":"Class",
          "DataSource":"DataSource"
     },
     "Data":{
```

```
            "Instances":[{
                  "value":"0.0",
                  "timestamp":1459725391,
                  "dimensions":{
                        "Instance_NAME":"inst_1"
                              }
            }, {
                  "value":"0.0",
                  "timestamp":1459725391,
                  "dimensions":{
                        "Instance_NAME":"inst_2"
                              }
            }]
      }
},
      {
      "MetaData":{
            "MetricName":"Instance_NAME",
            "CategoryType":"Attribute",
            "DataType":"STRING",
            "Description":"Desc",
            "Label":"Metric Name1",
            "Unit":"N/A",
            "Key":0,
            "ClassName":"Class",
            "DataSource":"DataSource"
      },
      "Data":{
```

```
          "Instances":[{
               "value":"inst_1",
               "timestamp":1459725391,
               "dimensions":{
                    "Instance_NAME":"inst_1"
                         }
          },{
               "value":"inst_2",
               "timestamp":1459725391,
               "dimensions":{
                    "Instance_NAME":"inst_2",
                         }
          }]
     }
     ]
}]
```

使用 REST API **/v4/RTCM/SubmitData** 将度量数据提交到 **hpsensor**。有关如何使用此 REST API 的详细信息，请参见用法部分。

注册和数据提交中的特殊字符

hpsensor 接受采用 URI 格式的数据请求，并且特殊字符可能为度量名称、数据源、类名称或实例名称的一部分。下面列出了各种实体支持和不支持的特殊字符：

实体	支持的特殊字符	不支持的特殊字符
度量名称/数据源/类名称	_~`!@$%^&*()-+={[]}|_;<>.?'	/,#\:”
实例名称	_~`!@$%^&*-+={}|_.?:'	/,#\”[]();<>

备注: 对于注册和数据提交：

- 元数据字段数可以有所不同。
- 度量的实例数可以有所不同，所有度量不可能具有相同数量的实例。
- 一个实例可以有一个或多个维度。
- <*> 字符的组合可以在**度量流配置**策略中用作通配符。例如：`"Instances":["(example)","(<*>.company.com)"]`
- 可以使用实例名称中提及的不支持的字符，但不能在**度量流配置**策略中完全使用这些字符。相反，您可以使用通配符 <*> 来匹配任何字符。

配置发布间隔

hpsensor 会在配置的间隔内将自定义和系统性能度量数据发布到目标订阅者。默认情况下，发布间隔为 10 秒。您可以使用以下 XPL 变量来配置默认发布间隔：

变量	命名空间	描述	是否需要重新启动？	默认值	类型
PUBLISH_INTERVAL	hpsensor	此参数可设置将自定义和系统性能度量数据发布到目标订阅者的间隔。	否	10	整型

要配置默认发布间隔，请执行以下步骤：

1. 以管理员身份登录到 HPE Operations Agent 节点。
2. 运行以下命令：

```
<OvInstallBinDir>ovconfchg -ns hpsensor -set PUBLISH_INTERVAL
<value>
```

默认值为 10(以秒为单位)，允许的最小值为 5(以秒为单位)。

备注: 发布间隔设置将适用于所有目标。

资源利用率和可扩展性

已提交到 **hpsensor** 且用于度量流的数据将存储在内存中，并且 **hpsensor** 仅保留度量的一个实例值。如果提交多个数据点，则内存利用率将非常高。为获得

最佳的内存利用率，Hewlett Packard Enterprise 建议提交的数据点数不超过 100000 个。

第 V 部分: 事务跟踪

HPE Operations Agent 可以跟踪事务在应用程序中的进度。性能收集组件和 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。

只有在 **parm** 文件中打开事务记录后，才会监视事务。事务跟踪守护进程 `ttd` 和测量接口守护进程 `midaemon` 会在应用程序运行时收集并同步其事务数据。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。

事务跟踪提供从事务开始到结束经过的时间的客户端视图，帮助您管理服务级别协议 (SLA)，并且在服务级别目标 (SLO) 超过 `alarmdef` 文件中定义的条件时生成警报。

第 20 章: 什么是事务跟踪？

本章描述：

- 提高性能管理
- 场景：实时订单处理
- 监视事务数据

提高性能管理

您可以通过 HPE Operations Agent 和 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 24 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 25 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持续时间：

- 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 `inventory response` 和 `update inventory` 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 25 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《应用程序响应测量 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。标识这些趋势后，可以基于事务卷分配信息技术成本。性能收集组件警报可以配置为激活技术人员的寻呼机，以便可以立即调查并解决问题。有关详细信息，请参见第 26 章：高级功能。

性能收集组件是要在 Performance Manager 中查看事务数据所必需的。

... 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，您可以从数据源的“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”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，要记录 ARM 度量，**oacore** 收集器必须正在运行。`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 21 章中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 21 章中的事务配置文件 (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 21 章中的添加新应用程序)
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 26 章中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 22 章中的警报部分中的“... 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 21 章中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 22 章中的独有事务的限制)
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 21 章中的 ARM API 调用状态返回结果)
9. 使用共享库链接。(请参见按平台分类的 C 编译器选项示例 (第 306 页)部分)

第 21 章: 事务跟踪的工作原理

性能收集组件和 GlancePlus 的以下组件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` 使用 `midaemon` 从事务配置文件 `ttd.conf` 读取、注册和同步事务定义。

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、事务关联以及记录代理程序。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《应用程序响应测量 2.0 API 指南 (Application Response Measurement 2.0 API Guide)》。

备注: 《应用程序响应测量 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《应用程序响应测量 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 ARM 检测的应用程序示例：armsample1.c、armsample2.c、armsample3.c 和 armsample4.c 以及 /<InstallDir>/examples/arm/ 目录中的生成脚本 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。不同的度量值可以通过 arm_start、arm_update 和 arm_stop 调用传递。此外，当 tran 无法由 start/stop 调用分隔时，无法使用 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，则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 (_)。这是应用程序名称“WarehouseInventoryApplication”和事务名称“CallFromWestCoastElectronicSupplier”在性能收集组件或 Performance Manager 中的显示方式：

```
WarehouseInventoryApplication_CallFromWestCoastElectronicSup
```

备注: 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 使用 midaemon 从 ttd.conf 读取、注册和同步事务定义。

使用 ovpa start 命令启动性能收集组件的 **oacore** 数据收集器时，将启动 ttd。ttd 分发后在后台运行，并且错误将写入到文件 /var/opt/perf/status.ttd 中。

midaemon 也必须正在运行，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)。

警告: 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也必须停止任何 ARM 检测的应用程序。ttd 必须正在运行，才能捕获正在对系统进行的 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

使用 ovpa 脚本启动性能收集组件进程以确保进程按正确顺序启动。 ovpa stop 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 /<InstallDir>/bin/ttd -k。但是，除非重新安装性能收集组件，否则我们建议不要停止 ttd。

如果性能收集组件不在系统上，GlancePlus 将启动 midaemon。如果在 midaemon 开始处理任何测量数据之前，midaemon 未在运行，则将启动 ttd。

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

为了注册事务，ttd 进程必须始终运行。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。

如果终止并重新启动 ttd，ARM 检测的应用程序在 ARM API 调用时可能会获得返回值 -2 (TT_TTDNOTRUNNING) 和 EPIPEerrno 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。接下来，重新启动 ttd(使用 /<InstallDir>/bin/ovpa start 或 /<InstallDir>/bin/ttd)，然后重新启动应用程序。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 将大于 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。)

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 调用使用 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。它在您运行 **oacore** 进程或 perfd 进程或者启动 GlancePlus 时开始运行。

性能收集组件和 GlancePlus 都需要运行 midaemon 和 ttd，这样才能注册和跟踪事务。ovpa 脚本按照正确的顺序启动和停止性能收集组件处理，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 (第 284 页)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。

ttd.conf 的自定义为可选。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 来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。将 ttd.conf 文件中的 tran 行的 slo 和 range 值应用于新事务。

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。

上述操作导致 ttd.conf 文件使用 ttd 和 midaemon 重新读取和注册新事务及其 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。
- 使用 ovpa stop 停止 **oacore** 收集器。
- 停止 Glance 的任何使用。
- 通过执行命令 ttd -k 停止 ttd。
- 对 ttd.conf 文件做出更改后：
- 使用 ovpa start 重新启动 **oacore**。
- 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=transaction_name	必需
slo	slo=sec	可选
range	range=sec [,sec,...]	可选

下面更详细地描述了这些关键字。

tran

使用 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。必须使用 tran 关键字后才能指定可选属性 range 或 slo。tran 是配置文件中所需的唯一关键字。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 **oacore** 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

使用 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 sec 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 文件可以包含以下两种类型的条目：常规事务和特定于应用程序的事务。

在定义任何应用程序之前，常规事务应在 ttd.conf 文件中定义。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 **oacore** 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，将使用在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*。常规事务集中的其中一个条目也有名为“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事务。

应用程序 [GenLedg] 有以下事务：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 (第 276 页)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：

- 单独事务 ID 的总数不得超过 4,000 个。通常，同一事务具有多个实例比不同事务都具有单一实例的成本更低。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，开销可能会增加。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《应用程序响应测量 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 **oacore** 会生成磁盘数据库文件。

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 **oacore** 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、**oacore** 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。

第 22 章: 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，请参见事务跟踪的工作原理 (第 274 页)。有关示例，请参见事务跟踪示例 (第 293 页)。

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/`

如果系统上未安装性能收集组件且 `libarm.*` 不存在于平台的上述路径中，请参见本手册末尾的按平台分类的 C 编译器选项示例 (第 306 页)。有关如何获取它的信息，另请参阅《应用程序响应测量 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，请参见本手册末尾的 ARM 库 (libarm) (第 302 页)。

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. 要监视性能收集组件和 Performance Manager 中的事务，请确保性能收集组件 `parm` 文件已打开事务记录。然后启动或重新启动性能收集组件以读取更新的 `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。

   这些操作会重新读取 ttd.conf 文件，并使用 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. 使用 ovpa stop 停止 **oacore** 收集器。
   c. 停止 Glance 的所有使用。
   d. 使用 ttd -k 停止 ttd。

   进行更改后：
   a. 使用 ovpa start 重新启动 **oacore**。
   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，可修改性能收集组件 parm 文件将事务添加到要记录的项列表中，以供 Performance Manager 和性能收集组件使用。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 参数的默认值为 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 可以同时记录资源数据和相关器数据。

在可以收集事务数据以供性能收集组件和 Performance Manager 使用之前，必须按照如下所述激活更新后的 parm 文件：

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 会在应用程序运行时收集并同步其事务数据。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，请参见监视性能数据 (第 289 页)。

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：

- 使用错误的 id 参数(例如未初始化的变量)调用 arm_start
- 在之前未成功调用 arm_start 的情况下调用 arm_stop

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。使用性能收集组件收集此数据，然后使用 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 查看所有事务是否均按预期显示。

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 使用的共享内存段已满时，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。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。必须以 root 身份登录才能修改 ttd.conf。

有关配置文件关键字(tran、range 和 slo)的信息，请参见事务跟踪的工作原理 (第 274 页)。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
```

如果需要对 ttd.conf 文件进行添加，请执行以下操作：

- 停止所有 ARM 检测的应用程序。
- 作为 root 用户执行 ttd -hup -mi 命令。

上述操作会重新读取 ttd.conf 文件，并使用 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. 使用 ovpa stop 停止 **oacore** 收集器。

3. 停止 Glance 的所有使用。
4. 使用 `ttd -k` 停止 `ttd`。

进行更改后：

1. 使用 `ovpa start` 重新启动 **oacore**。
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。

性能收集组件和 ttd 必须正在运行才能在性能收集组件中监视事务数据。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 可导入性能收集组件数据，并使您能将该数据转换为自定义图形或数字格式。使用 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”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

要使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)”窗口的“帮助”菜单选择**在此窗口上 (On This Window)**。

第 23 章: 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。

第 24 章: 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 的 `/<InstallDir>/paperdocs/gp/C/` 下为 `gp-metrics.txt`。

  `InstallDir` 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 HPE Operations Agent Dictionary of Operating System Performance Metrics 文件，后者位于：

  在 UNIX/Linux 的 `/<InstallDir>/paperdocs/ovpa/C/` 下为 `met<platform>.txt`。

  在 Windows 的 %ovinstalldir%`paperdocs\ovpa\C` 下为 `met<platform>.txt`。

第 25 章: 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什么是事务跟踪？ (第 270 页)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
什么是事务跟踪？ (第 270 页)中描述的实时订单处理场景定义事务的 ARM API 调用。此例程将在每次操作员接听电话以处理客户订单时进行处理。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，请参见事务跟踪的工作原理 (第 274 页)

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
```

第 26 章：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
- **oacore** 检测

如何使用数据类型

下表描述了性能收集组件中使用的数据类型。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：性能收集组件中的数据类型使用

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。

性能收集组件不支持记录字符串数据。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，性能收集组件会在五分钟间隔中将 Counter32、NumericID32、Gauge32、Gauge64、

NumericID32 和 NumericID64 的 Min、Max 和 Average 记录为 32 位整数。 由于无法在性能收集组件中汇总字符串，因此会忽略 String8 和String32。 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
示例 3 NumericID32 String8 NumericID64 Gauge32 String32 Gauge64	 NumericID32 NumericID64 Gauge32 Gauge64
示例 4 String8	 (无)

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
String32	
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。

第 27 章: 事务库

此附录讨论：

- 应用程序响应测量库 (libarm)
- 按平台分类的 C 编译器选项示例
- 应用程序响应测量库 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	与 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。
/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 2 章的事务跟踪守护进程 (ttd) (第 276 页)。

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/arm64	libarmjava.so	ARM 的 32 位共享库。
/opt/perf/examples/arm/arm64	libarmjava.so	ARM 的 64 位共享库。

备注: 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。

备注: 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/。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)。同样地，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
  ```
- 对于 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
  ```

备注: 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

“无操作”库(名为 libarmNOP.*，其中 * 是 sl、so 或 a，具体取决于操作系统平台)由性能收集组件和 Glance 附带提供。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 /<InstallDir>/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 /<InstallDir>/examples/arm/ 目录中。InstallDir 是安装性能收集组件的目录。

示例

Java 包装的示例位于 /<InstallDir>/examples/arm/ 目录中。此位置还包含 README 文件，该文件解释了每个包装的功能。

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(“name”,”description”)
```

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(“transactionName”,”details”);
```

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(“transactionName”);
```

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("description");
```

Then you can create the Transaction:

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(“Transactionname”,”details”;
```

不指定详细信息

语法：

```
ARMTransaction myTransaction =

myApplication.createTransaction(“Transactionname”);
```

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_star`。

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() 方法按如下所示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 (第 309 页)部分中的示例)并调用更新:

语法:

```
myTranInstance.updateTranInstance();
```

不使用 UDM 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“心跳”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《应用程序响应测量 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_GOOD.
ARMConstants.ARM_ABORT	系统中存在基本故障时使用此值。	ARMConstants.ARM_ABORT
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。	ARMConstants.ARM_FAILED

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，可以用方法 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 /<InstallDir>/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。

第 VI 部分: 疑难解答

本部分描述使用 HPE Operations Agent 时遇到的常见问题的解决方案或变通方法。

操作监视组件

- 问题：如果 HPOM 管理的节点很多(超过 1024 个)，则您可能会遇到 HPOM 与受管节点之间的通信问题。当 HPE Operations Agent 安装在与大量受管节点(超过 1024 个)通信的 HP Performance Manager 服务器上，您也可能会看到此问题。

  解决方案：

  要避免此问题，请转到管理服务器(如果 HPOM 管理超过 1024 个节点)或 HP Performance Manager 服务器(如果代理程序安装在与超过 1024 个节点通信的 HP Performance Manager 服务器上)，然后执行以下配置步骤：

  以根身份或管理员身份登录。

  运行以下命令：

  在 32 位版本的 Windows 上：

  ```
  %ovinstalldir%bin\ovconfchg -ns xpl.net -set SocketPoll true
  ```

  在 64 位版本的 Windows 上：

  ```
  %ovinstalldir%bin\win64\ovconfchg -ns xpl.net -set SocketPoll true
  ```

  在 HP-UX/Linux/Solaris 上：

  ```
  /opt/OV/bin/ovconfchg -ns xpl.net -set SocketPoll true
  ```

  重新启动代理程序：

  在 32 位版本的 Windows 上：

  ```
  %ovinstalldir%bin\opcagt -stop
  %ovinstalldir%bin\opcagt -start
  ```

  在 64 位版本的 Windows 上：

  ```
  %ovinstalldir%bin\win64\opcagt -stop
  %ovinstalldir%bin\win64\opcagt -start
  ```

  在 HP-UX/Linux/Solaris 上：

  ```
  /opt/OV/bin/opcagt -stop
  /opt/OV/bin/opcagt -start
  ```

- 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(HPE 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：

  i. ovc -kill

  ii. 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 2：

  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 3：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：部署特定消息拦截器 (msgi) 策略且触发与此策略中条件匹配的消息后，opcmsgi 进程停止以高 CPU 利用率(在 4 个 CPU 的服务器上消耗 25%，那么单个 CPU 就是 100%)做出响应，且不再处理任何消息。

  原因：

opcmsgi 进程停止响应且不再处理任何消息是因为以下模式：

```
<*><@.variable><*>
```

解决方案：

将 <*><@.variable><*> 模式替换为 <*><@.variable><S><*>。

例如：

原始模式：

```
^\tM_Type:<@.SiSMonType><*>SiS:<
[http://<@.SiSIP>:<@>].SiSURL><*>Monitor:<@.SiSMonName><*>Group:<@.
SiSGrpName><*>Status:<@.SiSMonStatus><*>URL:<@.SiSMonURL><*>Match
Content:<@.SiSMonMatch><*>Monitor Description:DOTCOM_BACKEND:
{<@.OMTargetNode>}{<@.OMMsgGrp>}{<*.OMAppl>}{<*.OMObj>}
{<*.OMSvcID>}{<@.SiSEMSCode>}{<*.OMCIs>}{<*.OMMsgText>}
```

替换后的模式：

```
^\tM_Type:<@.SiSMonType><S><*>SiS:<
[http://<@.SiSIP>:<@>].SiSURL><S><*>Monitor:<@.SiSMonName><S><*>Gro
up:<@.SiSGrpName><S><*>Status:<@.SiSMonStatus><S><*>URL:<@.SiSMonUR
L><S><*>Match Content:<@.SiSMonMatch><S><*>Monitor
Description:DOTCOM_BACKEND:{<@.OMTargetNode>}{<@.OMMsgGrp>}
{<*.OMAppl>}{<*.OMObj>}{<*.OMSvcID>}{<@.SiSEMSCode>}{<*.OMCIs>}
{<*.OMMsgText>}
```

例如：

原始模式：

```
^\tM_Type:<@.SiSMonType><*>SiS:<
[http://<@.SiSIP>:<@>].SiSURL><*>Monitor:<@.SiSMonName><*>Group:<@.
SiSGrpName><*>Status:<@.SiSMonStatus><*>URL:<@.SiSMonURL><*>Match
Content:<@.SiSMonMatch><*>Monitor Description:DOTCOM_AUTONOMY:
{<@.OMTargetNode>}{<@.OMMsgGrp>}{<*.OMAppl>}{<*.OMObj>}
{<*.OMSvcID>}{<@.SiSEMSCode>}{<*.OMCIs>}{<*.OMMsgText>}
```

替换后的模式：

```
^\tM_Type:<@.SiSMonType><S><*>SiS:<
[http://<@.SiSIP>:<@>].SiSURL><S><*>Monitor:<@.SiSMonName><S><*>Gro
up:<@.SiSGrpName><S><*>Status:<@.SiSMonStatus><S><*>URL:<@.SiSMonUR
L><S><*>Match Content:<@.SiSMonMatch><S><*>Monitor
Description:DOTCOM_AUTONOMY:{<@.OMTargetNode>}{<@.OMMsgGrp>}
{<*.OMAppl>}{<*.OMObj>}{<*.OMSvcID>}{<@.SiSEMSCode>}{<*.OMCIs>}
{<*.OMMsgText>}
```

- 问题：在 HPE Operations Agent 上，模式匹配报告错误结果。

  原因：

在 HPE Operations Agent 上，如果模式中的 OR (|) 运算符数超过 148 个，模式匹配可能会报告错误结果。

备注: 在模式匹配中，OR (|) 运算符是一个特殊字符，它通过匹配两个表达式中的字符串来分隔这些表达式。

例如：

```
[ab|c]d
```

在上面的模式中，OR <|> 运算符匹配字符串 **abd** 和字符串 **cd**。

解决方案：

确保 OR (|) 运算符数不超过 148。

- 问题：使用嵌入式 perl 脚本在节点上运行计划任务策略后 opcmona 进程自动重新启动，且 HPOM 控制台中出现以下消息：

  ```
  (ctrl-208) pid 为 6976 的组件 'opcmona' 退出，退出值为 '-1073741819'。正在重新启动组件。
  ```

  原因：

  嵌入式 perl 脚本中引用 `exit (0)` 导致 opcmona 重新启动。

  解决方案：

  不在嵌入式 perl 脚本中使用 `exit (0)`。

- 问题：在 Red Hat Enterprise Linux 系统上，ovbbccb 无法访问时区文件并且以 UTC(GMT) 格式(而非为系统设置的实际时区)写入日期信息。此错误的时区信息记录在 `System.txt` 文件中。

  解决方案：

  要建立正确的时区，请执行以下步骤：

  a. 运行以下命令，验证 zoneinfo 文件的位置：

     ```
     ls /usr/share/zoneinfo/
     ```

  b. 运行以下命令，验证 localtime 文件的位置：

     ```
     ls -l /etc/localtime
     ```

     备注: 如果未设置“TZ”环境变量，则本地时间确定当前时区。

  c. 创建以下目录：

     ```
     mkdir -p /var/opt/OV/usr/share/lib
     ```

  d. 以递归方式将文件从一个文件夹复制到另一个文件夹：

     ```
     cp -rp /usr/share/zoneinfo /var/opt/OV/usr/share/lib
     ```

  e. 运行以下命令，在 /var/opt/OV/ 下创建名为 etc 的目录：

     ```
     mkdir /var/opt/OV/etc
     ```

f. 将 localtime 文件复制到此目录：

```
cp /etc/localtime /var/opt/OV/etc
```

g. 运行以下命令，重新启动代理程序：

i. `ovc -kill`

ii. `ovc -start`

备注：ovbbccb 稍后向 `System.txt` 文件中记录条目时，将使用当前时区。

- 问题：在将 HPE Operations Agent 从 11.14 升级到 12.01 后，**`ovcodautil -ds scope -o process -flat -rawonly`** 出现超时错误。

  原因：如果天数太多，报告工具(如 Performance Manager)将无法对选定的所有进程数据进行绘图。

  解决方案：减少度量数或/和持续时间，并将超时值增加为更高值。

  变通方法：运行以下命令提取数据：

  ```
  extract -xp -p
  ```

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 HPE Operations Agent 进程：

  ```
  /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 HPE Operations Agent 进程：

  ```
  /opt/OV/bin/opcagt -start
  ```

- 问题：安装 HPE Operations Agent 之后，如果启用跟踪机制，`System.txt` 文件中会出现以下错误消息：

  ```
  Scope data source initialization failed
  ```

  解决方案：忽略此错误。

- 问题：HPOM 控制台上显示以下错误消息：

  ```
  CODA:GetDataMatrix returned 76='Method ScopeDataView::CreateViewEntity
  failed
  ```

  原因：如果在测量阈值策略中使用带有 SCOPE 数据源的 PROCESS 对象(其中源已设置为嵌入式性能组件)，此消息将显示。

  解决方案：改为使用服务/进程监视策略。

- 问题：HP Performance Manager 或 HPE Operations Bridge Reporter 等数据分析产品无法从代理程序的数据存储区检索数据，并显示以下错误：

  ```
  Error occurred while retrieving data from the data source
  ```

  原因：代理程序的数据访问实用程序针对单个查询从客户端程序读取数据类的所有记录。通过 HP Performance Manager 等数据分析客户端将查询发送到代理程序。当数据类包含大量记录时，数据访问实用程序将无法处理查询。

  解决方案：要避免出现此问题，可配置数据访问实用程序将数据记录分多个区块传输到客户端。执行以下步骤：

  a. 以根用户或管理员身份登录到代理程序节点。

  b. 运行以下命令：

     在 Windows 上：

     ```
     %ovinstalldir%bin\ovconfchg -ns coda -set DATAMATRIX_VERSION 1
     ```

     在 HP-UX/Linux/Solaris 上：

     ```
     /opt/OV/bin/ovconfchg -ns coda -set DATAMATRIX_VERSION 1
     ```

     在 AIX 上：

     ```
     /usr/lpp/OV/bin/ovconfchg -ns coda -set DATAMATRIX_VERSION 1
     ```

     现在，对于每个查询，代理程序的数据访问实用程序先将数据分为五条记录一区块的形式，然后将这些数据发送到客户端程序。将数据分为多个块可提高数据传输过程的性能。

     可以控制代理程序可向客户端发送的每个区块中的记录数。使用 DATAMATRIX_ROWCOUNT 变量(在 coda 命名空间下)可以控制此数字(默认值是五)。

     当有海量数据时，减小 DATAMATRIX_ROWCOUNT 变量的值可稍微提高向数据存储区传输数据的速率。

     如果 DATAMATRIX_ROWCOUNT 变量设置为 0，则 HPE Operations Agent 将恢复为不分区块发送数据记录的默认行为。

     但是，建议不要更改 DATAMATRIX_ROWCOUNT 变量的默认设置。

  c. 重新启动代理程序使更改生效：

     ```
     ovc -restart coda
     ```

RTMA

- 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 HPE Operations Agent 进程：

  `/opt/OV/bin/opcagt -stop`

  c. 运行以下命令备份 `perfd`：

  `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 HPE Operations Agent 进程：

  `/opt/OV/bin/opcagt -start`

GlancePlus

问题：GlancePlus 不显示 AIX 框架上托管的所有 LPAR 实例。

原因：安装 HPE Operations Agent 的 LPAR 无法与 AIX 框架上托管的其他 LPAR 通信。

解决方案：确保安装 HPE Operations Agent 的 LPAR 可以与 AIX 框架上托管的所有其他 LPAR 通信。

在托管 HPE Operations Agent 的 LPAR 上运行以下命令，以检查它与其他 LPAR 的连接：

`xmpeek -l`<hostname>

在此实例中，<hostname> 是 LPAR 的主机名。

hpsensor

- 问题：hpsensor 报告常驻内存使用率较高

  原因：即使删除后，也不会清除分配的内部 STL 数据结构。此问题主要发生在 HPUX 平台上。

  解决方案：要解决这个问题，请执行以下步骤：

a. 以管理员身份登录到系统。
b. 转到以下目录：
   - 在 Windows 上：`%OvDataDir%hpcs\`
   - 在 Unix/Linux 上：`/var/opt/OV/hpcs/`
c. 打开 **hpcs.conf** 文件并在 **hpcs.runtime** 命名空间下将以下配置变量设置为 **true**：

   `MemMap=<true>`

其他

- 问题：`fs_type` 度量将 autofs 系统报告为 NFS(网络文件系统)。

  原因：在运行 HPE Operations Agent 12.01 的 Linux 系统上，如果 autofs 系统在内核 2.6 或更低版本上，则该系统将报告为 NFS。

  解决方案：要解决此问题，请确保 autofs 系统在内核 3.0 或更高版本上。
- 问题：在 `System.txt` 文件中记录以下错误消息后，**oacore** 进程停止：

```
Database is in an inconsistent state.
```

  原因：如果受管类信息中没有相应的数据库文件，则在 `System.txt` 文件中记录错误消息后，**oacore** 进程停止。

  备注: 全局类的 `System.txt` 消息示例：

```
0: INF:Thu Aug 20 16:14:28 2015: oacore
(21580/139960775325472):Collection intervals:Process = 60 Global = 300
DataFile Rollover% = 20.

0: ERR:Thu Aug 20 16:14:28 2015: oacore (21580/139960775325472):Database
is in an inconsistent state.No database found for Class  Scope::Global

0: INF:Thu Aug 20 16:14:28 2015: oacore (21580/139960775325472):(oacore-
84) oacore. oacore Server stopped.
```

  解决方案：要解决此问题，必须删除数据源或该数据源中的特定类。

  使用 `oadbutil.pl` 工具可删除数据源或该数据源中的特定类。

  语法：

  - `oadbutil.pl -d <datasource name>`
    删除数据源及其类。
  - `oadbutil.pl -d <datasource name> -c <class name>`
    仅删除特定类的元数据和记录(将保留有关数据源的信息)。

例如：

```
oadbutil.pl –d Scope –c Global
```

在此实例中：

<datasource name> 为 Scope

<class name> 为 Global

默认情况下，数据收集器不会创建模型以将默认性能度量类和数据源 (Scope) 记录到度量数据存储区中。删除默认性能度量类或数据源后，请执行以下步骤以重新创建该模型：

a. 运行以下命令停止 **oacore** 进程：

```
ovc -stop oacore
```

b. 运行以下命令重新创建类：

```
ovconfchg -ns oacore -set UPDATED_MODEL_AVAILABLE true
```

c. 运行以下命令启动 **oacore** 进程：

```
ovc -start oacore
```

备注：如果自定义类(通过基于 Perl 的可脚本化 API、DSI 提交)已删除，将在后续提交中自动进行重新创建。

第 VII 部分: 从早期版本升级到 HPE Operations Agent 版本 12.xx

使用 HPE Operations Agent 版本 12.01，度量数据存储区会替换基于日志文件的数据存储区。CODA 和 scope 进程(UNIX 和 Linux 节点上为 scopeux，Windows 节点上为 scopent)将合并到名为 **oacore** 进程的单个进程中。**oacore** 进程提供了针对系统性能和自定义数据的读取和写入接口。

存储在 CODA 数据库文件、SCOPE 日志文件和 DSI 日志文件中的旧数据以只读模式保留。可以通过诸如 ovcodautil、extract 等实用程序或使用诸如 HP Performance Manager 和 HP Reporter 等报告工具访问旧数据。

备注:

- HP Operations Agent 11.xx 的旧数据不会迁移到度量数据存储区中。
- 在 HPUX IA 系统上，从 HPE Operations Agent 11.xx 升级到 12.xx 时，较旧的数据库文件将进行备份，并保存在以下位置：

  `/var/opt/OV/tmp/BackUp`

  无法使用诸如 Extract、Utility 或 `ovcodautil` 等工具访问此数据。
- 在 Linux 或 Windows 系统上，升级到 HPE Operations Agent 12.xx 后，无法通过诸如 HP Performance Manager 或 HP Reporter 等工具访问旧的逻辑系统数据(保存在日志文件中)。

在从早期版本升级到 HPE Operations Agent 12.xx 的过程中，来自 DSI 日志文件的所有现有数据和 CODA 数据存储区中记录的 Smart Plug-in (SPI) 数据以及所有类和度量都会在度量数据存储区中自动创建。

DSI 数据源

在从 HPE Operations Agent 11.xx 升级到 12.xx 之前，请确保 DSI 数据源已添加到位于以下位置的 `datasources` 文件中：

在 HP-UX/Linux/Solaris 上：

```
var/opt/OV/conf/perf
```

在 Windows 上：

```
%ovdatadir%\conf\perf
```

运行以下命令将 DSI 数据源添加到 datasources 文件中。

```
DATASOURCE=<Datasource Name>LOGFILE=<DSI Logfile Path>
```

确保日志文件名称和数据源名称相同。

例如：

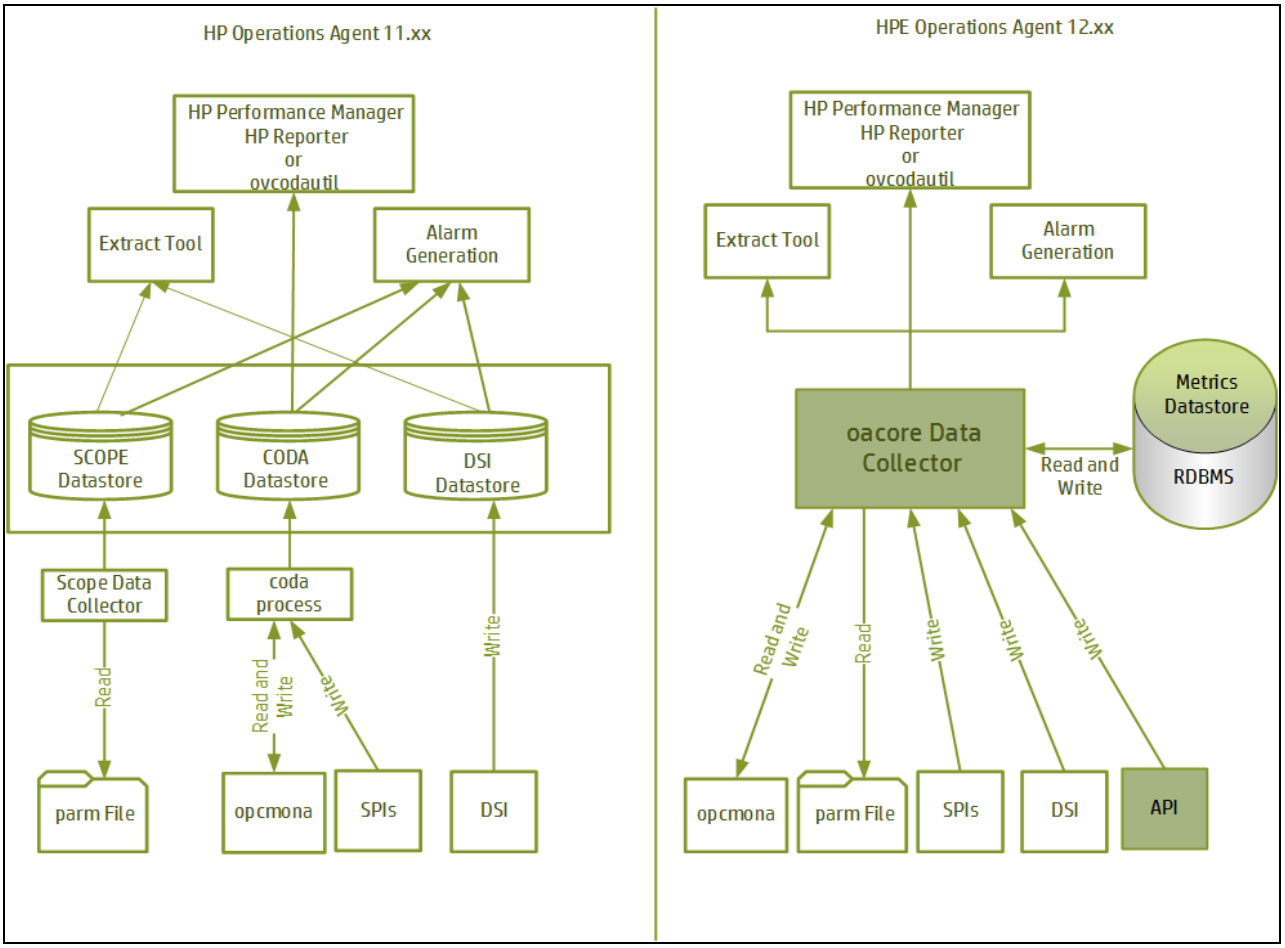
```
DATASOURCE=VMSTAT LOGFILE=/root/dsi/VMSTAT
```

仅当 DSI 数据源添加到 `datasources` 文件中时，才会在度量数据存储区中创建用于记录 DSI 数据的模型。

备注：在 HPE Operations Agent 版本 11.xx 中使用 `-timestamp` 选项记录 DSI 数据时，时间戳将采用 UTC 格式。在 HPE Operations Agent 版本 12.xx 中，使用 `-timestamp` 选项记录的 DSI 数据的时间戳将采用本地时区格式。

从 HPE Operations Agent 11.xx 升级到 12.xx 后，如果使用 extract 导出 DSI 数据，则旧数据 (11.xx) 的时间戳将采用 UTC 格式，新数据 (12.xx) 的时间戳将采用本地时区格式。

将 HPE Operations Agent 12.xx 与早期版本进行比较



将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
数据收集器	**scope** 数据收集器(在 UNIX 和 Linux 节点上是 scopeux；在 Windows 节点上是 scopent)收集并汇总对系统资源利用率的性能测量，并将数据记录在基于日志文件的数据存储区中。 根据在 **parm** 文件中指定的数据类，收集的数据将记录在以下日志文件中：logglob、logappl、logproc、logpcmd、logdev、logtran、logls 和 logindx。	**oacore** 数据收集器持续收集系统中的性能数据和运行状况数据，并将收集的数据存储在度量数据存储区中。 **备注：** CODA 和 **scope** 进程(Scopeux 和 Scopent)合并成一个进程，称为 **oacore**。**oacore** 进程提供了针对系统性能和自定义数据的读取和写入接口。
scope RUN 文件	位于 /var/opt/perf/datafiles/RUN 中的 **scope** RUN 文件指示 **scope** 是否正在运行。此文件仅在 **scope**(scopent 和 scopeux)运行时才存在。 如果 **scope** 异常终止，此文件不会删除。要重新启动 **scope**，必须首先删除 **scope** RUN 文件，然后再启动 **scope**。	此文件不再存在。 要验证 **oacore** 数据收集器的状态，请参见验证 oacore 进程的状态。
数据存储区	从受监视的系统收集的度量数据将存储在基于日志文件的数据存储区中。	从受监视的系统收集的系统性能数据和运行状况数据将存储在度量数据存储区中。度量数据存储区是单个基于关系数据库管理系统(RDBMS)的数据存储区。 根据在 **parm** 文件中指定的数据类收集度量数据。对于记录到度量数据存储区中的每个数据类，将创建一个特定的数据库文件。
代理数据源	可以托管多个来自其他系统的用作数据源的数据库文件。HP Operations Agent 11.xx 可同时读取这些数据源。	使用 HPE Operations Agent 版本 12.xx，您可以以只读模式一次只托管一个数据源。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx

		执行以下步骤托管来自其他系统的用作数据源的数据库文件：
		1. 运行以下命令停止 **oacore** 进程：
		`ovc -stop oacore`
		2. 备份现有数据库文件
		3. 运行以下命令：
		`ovconfchg -ns oacore.dml -set READ_ONLY_MODE True`
		4. 将来自远程系统中的数据库文件放置在 `datadir/databases/oa` 中。
		5. 运行以下命令启动 **oacore** 进程：
		`ovc -start oacore`
		备注: 当设置为代理模式时，将完全禁止向 度量数据存储区 记录数据。
		执行以下步骤停止托管来自其他系统的用作数据源的数据库文件：
		1. 运行以下命令停止 **oacore** 进程：
		`ovc -stop oacore`
		2. 运行以下命令：
		`ovconfchg -ns oacore.dml -clear READ_ONLY_MODE`
		3. 运行以下命令启动 **oacore** 进程：
		`ovc -start oacore`

`rtmd` 进程	`rtmd` 进程由 RTM 组件提供，用于帮助建立从节点访问实时数据的安全通信通道。	`rtmd` 进程替换为 `hpsensor` 进程。`rtmd` 的 XPL 配置不具有向后兼容性，所以在从 HPE Operations Agent 11.xx 升级到 12.xx 后将无法使用。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
		hpsensor 进程提供类似的 XPL 配置以强化安全性 (SSL)。 hpsensor 进程帮助您通过安全的通信通道以本地或远程方式访问实时性能度量。

性能收集组件

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
性能收集组件的图形用户界面	可以使用性能收集组件的图形用户界面执行以下任务：提取日志文件数据、导出日志文件数据、存档日志文件数据、调整日志文件大小、扫描日志文件、分析日志文件、配置导出模板、配置用户选项、配置收集参数、配置警报定义和检查状态。	不支持性能收集组件的图形用户界面。
ovcodautil 是用于显示记录的度量的命令行实用程序。	ovcodautil -ds SCOPE -o Global -m GBL_CPU_TOTAL_UTIL /opt/OV/bin/ovcodautil [options] -h：显示度量标题	ovcodautil -ds SCOPE -o Global -m GBL_CPU_TOTAL_UTIL /opt/OV/bin/ovcodautil [options] -header：显示度量标题 -h：显示帮助消息

parm 文件

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
scope 事务	**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。scope 事务标记确定是否将记录 **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。将始终记录第三个事务 Scope_Log_ Headers；它不受 scopetransactions=off 影响。	在 HPE Operations Agent 12.xx 中，不记录 Scope_Get_ Process_Metrics 和 Scope_Get_Global_ Metrics。
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 即在每天早上 8:30 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。 **备注:** 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。	HPE Operations Agent 版本 12.xx 不支持 `mainttime` 参数。 size 参数用于设置数据库文件的最大大小(以 MB 为单位)。当达到 **parm** 文件中指定的最大大小时，滚动更新存储默认性能度量类的数据库文件。 如果 **parm** 文件中的大小规范发生更改，**oacore** 会在启动过程中检测到这种更改。 如果 **parm** 文件中未指定大小，则在达到 1 GB 的最大大小时滚动更新数据库文件。在滚动更新期间，从数据库文件删除最旧的 20% 数据。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
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。此参数的值必须在 1 到 365 的范围内。此参数允许 **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 mainttime 自动回滚日志文件。 在回滚期间，从日志文件删除在 days 参数达到最大值之后收集的数据。 **备注:** 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。	HPE Operations Agent 版本 12.xx 不支持 days 参数。 size 参数用于设置数据库文件的最大大小(以 MB 为单位)。当达到 **parm** 文件中指定的最大大小时，滚动更新存储默认性能度量类的数据库文件。 如果 **parm** 文件中的大小规范发生更改，**oacore** 会在启动过程中检测到这种更改。 如果 **parm** 文件中未指定大小，则在达到 1 GB 的最大大小时滚动更新数据库文件。在滚动更新期间，从数据库文件删除最旧的 20% 数据。
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。在 mainttime 执行回滚。 例如，如果 **parm** 文件中设置为“maintweekday=Mon”，则一旦满足 days 参数中指定的值，将在周一的 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。 **备注:** maintweekday 参数是可选参	HPE Operations Agent 版本 12.xx 不支持 maintweekday 参数。 size 参数用于设置数据库文件的最大大小(以 MB 为单位)。当达到 **parm** 文件中指定的最大大小时，滚动更新

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
	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 parm 文件中未指定 maintweekday 但指定了 days 参数，则默认值是“maintweekday=Sun”。 例如，如果“daysglobal=30”、“application=20”、“process=30”、“device=20”、“transaction=10”、“maintweekday=Wed”，而且日志文件达到 days 参数中指定的天数，则继续收集数据，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。	存储默认性能度量类的数据库文件。 如果 **parm** 文件中的大小规范发生更改，**oacore** 仅在启动过程中才会检测到这种更改。 如果 **parm** 文件中未指定大小，则在达到 1 GB 的最大大小时滚动更新数据库文件。在滚动更新期间，从数据库文件删除最旧的 20% 数据。
javaarg	此参数仅在 UNIX/Linux 上有效。	此参数在 Windows 和 UNIX 平台上有效。
proccmd	此参数仅在 UNIX/Linux 上有效。 `proccmd` 参数允许将进程命令记录到 HP Operations Agent 数据存储区。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。	所有平台都支持此参数。 `proccmd` 参数允许将进程命令记录到数据存储区。默认情况下，此参数值设置为 0，表示禁止记录进程命令。要允许记录进程命令，请将此参数值设置为 1。 当此参数值大于等于 1 时，启用 `proccmd` 参数记录。记录的进程命令长度始终为

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
		4096，与此参数中指定的值无关。

utility 程序

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
运行 utility 程序	HP Operations Agent 11.xx 支持以批处理模式、交互模式和命令行模式运行 utility 程序。	HPE Operations Agent 12.xx 仅支持以命令行模式运行 utility 程序。
resize 命令	使用 resize 命令 -xr 管理日志文件集中的空间。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命令。	HPE Operations Agent 版本 12.xx 不支持 resize 命令。 size 参数用于设置数据库文件的最大大小(以 MB 为单位)。当达到 **parm** 文件中指定的最大大小时，滚动更新存储默认性能度量类的数据库文件。 如果 **parm** 文件中的大小规范发生更改，**oacore** 会在启动过程中检测到这种更改。 在滚动更新期间，从超过 **parm** 文件中指定大小的数据库文件中删除最旧的 20% 数据。
utility 命令： start stop exit guide logfile	这些命令受支持，您可以按如下方式进行使用： • start - 指定 analyze 或 scan 函数的开始日期和时间 • stop - 指定 analyze 或 scan 函数的结束日期和时间 • exit - 使用 exit 命令终止 **utility** 程序。	不支持这些命令。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

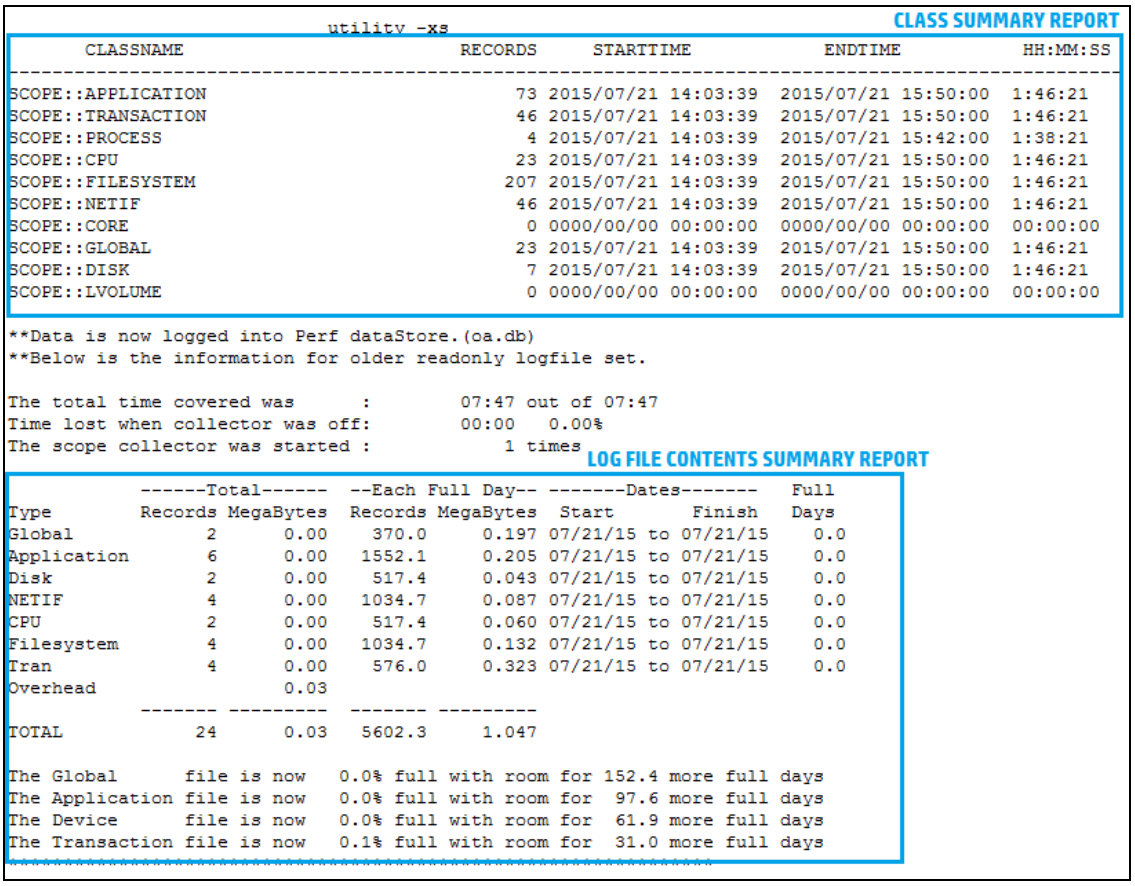
新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
menu sh show	• guide - 使用 guide 命令输入引导命令模式。引导命令界面指引您完成各种 utility 命令，并提示您执行可用的最常见任务。 • logfile - 使用 logfile 命令打开日志文件。 • menu - 使用 menu 命令打印可用 utility 命令列表。 • sh - 使用 sh 输入 shell 命令而不退出 utility。 • show - 使用 show 命令列出打开的文件的名称和可设置的 utility 参数的状态。	

实用程序扫描报告

如果从 HP Operations Agent 11.xx 升级到 HPE Operations Agent 12.xx，实用程序扫描报告将为您提供有关数据的每个类利用的磁盘空间以及存储在日志文件集中的旧数据的信息。

例如：

如果运行 utiltiy -xs 命令，则将生成类摘要报告和日志文件内容摘要报告，如下图所示：



日志文件内容摘要报告列出了存储在较旧的只读日志文件集中的信息。

例如：如果运行 utiltiy -xs -D 命令，则将生成以下报告：

初始 parm 文件全局信息

HP Operations Agent 11.xx

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。

```
04/01/2014 12:38 System ID="  "
SCOPE/UX C.05.00.100 COLLECTION INTERVALS:Process = 60, GLOBAL = 300
Seconds, Log version=D
Configuration: ia64, O/S Linux 2.6.16.21-#1 SMP Monia64 #1 SMP Monia64
CPUs=1
Logging Global(NoDisk)(NoNETIF) Application Process Transaction
        Device = Disk NETIF FileSys CPU records
```

```
Thresholds:CPU=10.00%, Disk=n/a, ProcMem=900.0 MB
Nonew=TRUE, Nokilled=TRUE, Shortlived=FALSE (<1 sec)

Javaarg = true

Parms:Buffer Cache Size = 0KB, NPROC = 0

Memory:Physical = 981 MB, Swap = 1286.4 MB,
Available to users = 853 MB

Application and device data flush frequency = 3600 seconds
01/02/2014 09:22 Data collected on 3 disk devices:
    Disk #0              = "sda"
    Disk #2              = "sdb"
    Disk #3              = "hda"
```

在此实例中，第一行中列出的日期和时间对应于全局日志文件中的最早日期和时间，它表示启动数据收集器的事件。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第一条全局记录。

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成此报告。

初始 parm 文件应用程序定义

HP Operations Agent 11.xx

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：

```
10/29/2013 15:25 Application(1) = "other"
Comment=all processes not in user-defined applications

10/29/2013 15:25 Application(2) = "network"
File=nfs*,biod,automount,amd,*inetd,snmp*,rpc*,llbd,netfmt,portmap
File=rbootd,telnet*,ftp*,*rlogin*,remsh*

File=rcp,nktl*,nvsisr,ttisr,lcsp,gcsp,strmen,strweld,vtdaemon
File=mib*,trapdest*,*web*,xntpd,yp*

File=hp_unixagt,ntl*,pty*

10/29/2013 15:25 Application(3) = "xwindows"
File=X*,xload,xclock,*term,grmd,softmsg*,vue*,ttsession,pexd
File=dt*,xfs,rpc.ttdbserver,Aserver,gdm*,kdm*
```

```
File=gnome*,kde*,soffice.bin,netscape*,mozilla*,realplay

10/29/2013 15:25 Application(4) = "memory_management"
File=swapper,vhand,syncer,pageout,fsflush,kswapd,kreclaimd,bdflush
```

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。

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成此报告。

parm 文件全局更改通知

HP Operations Agent 11.xx

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*"
```

HPE Operations Agent 12.xx

未生成此报告。

parm 文件应用程序更改

HP Operations Agent 11.xx

要获取 parm 文件应用程序更改报告，请使用带默认 detail on 的 scan 命令，并且数据存储区中要有应用程序数据。

对 parm 文件进行任何更改都需要重新启动 **scope** 才能在数据存储区中记录更改。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
```

注：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。

HPE Operations Agent 12.xx

未生成此报告。

scope 关闭时间通知

HP Operations Agent 11.xx

此报告指示 scope 重新启动的时间以及 scope 重新启动前日志文件中记录最后一个有效数据的时间

HPE Operations Agent 12.xx

未生成此报告。

特定于应用程序的摘要报告

HP Operations Agent 11.xx

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，或可以与其他应用程序合并的应用程序。可将积累的系统资源过多的应用程序拆分成多个应用程序。

只要更改应用程序定义，就会生成一个特定于应用程序的摘要报告。这使您可以在更改前后查看应用程序定义。最终报告是针对所有应用程序生成的。此报告只覆盖自生成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                    26258   3.7%   6.6% 100.0%
network                   4649   0.0%   0.4%   0.0%
xwindows                 34571  25.3%   1.9%   0.0%
memory_management         3797   0.0%   0.0%   0.0%
other_user_root          45504  71.0%  91.0%   0.0%

-------------------- ---------- ------ ------ ------

All user applications     77.1%  96.3%  93.4%   0.0%
```

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成此报告。

进程摘要报告

HP Operations Agent 11.xx

只要扫描进程数据，就会打印此报告。要获取此报告，日志文件中必须有进程数据。

此报告帮助您设置 scope 进程阈值。此报告列出了将一个进程视为“感兴趣”的进程要考虑的所有因素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:11/08/2013 10:18    to 05/05/2014 16:55
There were 3427.4 hours of process data
Process records were logged for the following reasons:

Log Reason      Records   Percent  Recs/hr
--------------- -------   -------  -------
New Processes         6      0.1%      0.0
Killed Processes      6      0.1%      0.0
CPU Threshold      6956    100.0%      2.0

NOTE:A process may 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。每个报告都覆盖自生成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。

要减少 scope 记录的进程数据量，请修改 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。要增加记录的数据量，请降低感兴趣区域的阈值。

HPE Operations Agent 12.xx

未生成此报告。

扫描开始和停止报告

HP Operations Agent 11.xx

只要扫描到有效数据，就将打印扫描开始和停止报告。该报告提供扫描开始和停止的实际日期和时间。

例如：

```
Scan started on 11/08/2013 10:18
Scan stopped on 05/05/2014 16:55
```

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成此报告

应用程序总体摘要

HP Operations Agent 11.xx

要获取此报告，HPE Operations Agent 数据存储区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
88521 out of 114779 records ( 77.1%)
832.2 out of 863.9 CPU hours ( 96.3%)
819.2 out of 877.2 M disk IOs ( 93.4%)
 0.0 out of 0.0 M trans ( 0.0%)
```

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成应用程序总体摘要报告。此报告仅包含 CODA 数据库文件、scope 日志文件和 DSI 日志文件中的旧数据。

收集器覆盖时间摘要

HP Operations Agent 11.xx

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 处于关闭状态的时间百分比很高，您应检查开始和停止 scope 的操作步骤。

例如：

```
The total time covered was :172/13:29:27 out of 178/06:37:00
Time lost when collector was off: 5/17:07:33 3.20%
The scope collector was started : 5 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。)

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才能生成此报告

备注: oacore 进程的开始和停止详细信息在 System.txt 文件中列出。

类摘要和日志文件内容摘要

HP Operations Agent 11.xx

HPE Operations Agent 11.xx 不支持类摘要报告。

只要扫描到有效数据，就将打印日志文件内容摘要报告。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。

例如：

```
            ------Total------  --Each Full Day-- -------Dates-------   Full
Type        Records MegaBytes  Records MegaBytes  Start       Finish   Days
Global         7460      3.97   1440.1     0.766 07/03/2015 to 07/08/2015    5.2
Application    9161      1.21   1768.7     0.233 07/03/2015 to 07/08/2015    5.2
Process       14920      4.45   2880.5     0.858 07/03/2015 to 07/08/2015    5.2
Disk           7451      0.63   1438.5     0.121 07/03/2015 to 07/08/2015    5.2
NETIF         14920      1.25   2880.5     0.242 07/03/2015 to 07/08/2015    5.2
CPU           29840      3.46   5761.1     0.668 07/03/2015 to 07/08/2015    5.2
Filesystem    37300      4.77   7201.3     0.922 07/03/2015 to 07/08/2015    5.2
Tran          14921      8.36   2880.2     1.613 07/03/2015 to 07/08/2015    5.2
Overhead                 0.58
            ------- ---------  ------- ---------
TOTAL        135973     28.67  26250.9     5.424

The Global      file is now  13.3% full with room for  33.9 more full days
The Application file is now   6.3% full with room for  80.3 more full days
The Process     file is now  15.0% full with room for  29.7 more full days
The Device      file is now  52.4% full with room for   4.9 more full days
The Transaction file is now  84.2% full with room for   1.0 more full days
```

HPE Operations Agent 12.xx

仅在新安装 HPE Operations Agent 12.01 时才会生成类摘要报告。该报告为您提供有关数据的每个类利用的磁盘空间的信息。

如果从 HPE Operations Agent 11.xx 升级到 12.01，则会生成类摘要报告和日志文件内容摘要报告。

日志文件空余空间摘要

HP Operations Agent 11.xx

为每个扫描的日志文件打印此摘要。

例如：

```
The Global      file is now  73.9% full with room for   0.3 more full days
The Application file is now  78.2% full with room for  21.8 more full days
The Process     file is now   7.0% full with room for  93.0 more full days
The Device      file is now  96.4% full with room for   3.1 more full days
The Transaction file is now  90.4% full with room for   3.0 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。

HPE Operations Agent 12.xx

仅在从 HPE Operations Agent 11.xx 升级到 12.01 后才会生成日志文件空余空间摘要报告。此报告仅包含 CODA 数据库文件、scope 日志文件和 DSI 日志文件中的旧数据。

extract

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
运行 extract 程序	HP Operations Agent 11.xx 支持以交互模式和命令行模式运行 extract 程序。	HPE Operations Agent 12.xx 仅支持以命令行模式运行 extract 程序。
输出文件	extract 程序执行 export 函数。该程序从日志文件中读取数据，并以 ASCII、datafile、二进制和 WK1(电子表格)格式将结果导出到	extract 程序执行 export 函数。该程序从数据存储区读取数据，并以 ASCII 格式将结果导出到输出文件。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
	输出文件。	
主机总线适配器 (HBA)	此度量类在 HP Operations Agent 11.xx 中不可用。	已添加新的度量类(主机总线适配器 (HBA))。使用以下命令导出 HBA 数据： • -h - 指定要导出的 HBA 详细信息数据。 • -H - 指定要导出的 HBA 摘要数据。
extract 命令： -s -k -ut -v -we -xt -xw -xm -xy	这些命令受支持，您可以按如下方式进行使用： • -s - 指定特定时段的开始和结束时间，排除周末。 • -k - 仅导出已终止的进程。 • -ut - 显示导出的日志文件数据中 UNIX 格式的日期和时间。	不支持这些命令。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
	• -v - 生成详细输出报告格式。 • -we - 指定要从 export 中排除的天数； 1=星期日。 • -xt - 以系统内部格式提取数据。 • -xw - 提取日历周的数据。 • -xm - 提取日历月的数据。 • -xy - 提取日历年的数据。	
文件选项： Append、Purge、New	支持	不支持
导出选项： TODAY、TOMORROW、TODAY-1 TOMORROW-1、Last、First	支持	仅当与 -b 和 -e 命令选项一起使用时，才支持 Last 和 First 选项。
导出数据的格式： ASCII、DATAFILE、WK1、电子表格或二进制	支持	仅支持 ASCII 格式
导出模板文件： Repthist、Reptall	支持	仅提供 Reptall 模板文件。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
布局： 单个、多个	支持	仅支持单个布局。
导出文件标题： !date、!time、!logfile、!class、!collector、!system_id	支持	不支持

度量

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
BLANK RECORD_TYPE DATE TIME YEAR DAY DATE_SECONDS INTERVAL STATDATE STATTIME	支持这些度量。	不支持针对任何数据类的这些度量。
事务跟踪度量	事务跟踪度量包括与监视的系统上执行的所有系统事务相关的度量。此类度量带有前缀 TTBIN_ 或	事务跟踪度量包括与监视的系统上执行的所有系统事务相关的度量。此类度量带有前缀 TT_。 不支持以下度量：

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

	TT_。	TT_USER_MEASUREMENT_NAME TT_USER_MEASUREMENT_MAX TT_USER_MEASUREMENT_MIN TT_USER_MEASUREMENT_AVG TT_USER_MEASUREMENT_NAME_2 TT_USER_MEASUREMENT_MAX_2 TT_USER_MEASUREMENT_MIN_2 TT_USER_MEASUREMENT_AVG_2 TT_USER_MEASUREMENT_NAME_3 TT_USER_MEASUREMENT_MAX_3 TT_USER_MEASUREMENT_MIN_3 TT_USER_MEASUREMENT_AVG_3 TT_USER_MEASUREMENT_NAME_4 TT_USER_MEASUREMENT_MAX_4 TT_USER_MEASUREMENT_MIN_4 TT_USER_MEASUREMENT_AVG_4 TTBIN_UPPER_RANGE_10 TTBIN_UPPER_RANGE_1 TTBIN_UPPER_RANGE_2

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

		TTBIN_UPPER_RANGE_3 TTBIN_UPPER_RANGE_4 TTBIN_UPPER_RANGE_5 TTBIN_UPPER_RANGE_6 TTBIN_UPPER_RANGE_7 TTBIN_UPPER_RANGE_8 TTBIN_UPPER_RANGE_9 TTBIN_TRANS_COUNT_10 TTBIN_TRANS_COUNT_1 TTBIN_TRANS_COUNT_2 TTBIN_TRANS_COUNT_3 TTBIN_TRANS_COUNT_4 TTBIN_TRANS_COUNT_5 TTBIN_TRANS_COUNT_6 TTBIN_TRANS_COUNT_7 TTBIN_TRANS_COUNT_8 TTBIN_TRANS_COUNT_9 TT_NUM_BINS
主机总线适配器 (HBA)	此度量类在 HP Operations Agent 11.xx 中不可用。	HBA 是随其他度量类一起记录的新度量类。它包括与监视的系统上运行的所有主机总线适配器相关的度量。此类度量带有前缀 BYHBA_。 记录以下度量： BYHBA_TIME BYHBA_INTERVAL BYHBA_ID BYHBA_NAME BYHBA_DEVNAME

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

		BYHBA_DEVNO BYHBA_CLASS BYHBA_DRIVER BYHBA_STATE BYHBA_UTIL BYHBA_THROUGHPUT_UTIL BYHBA_IO BYHBA_READ BYHBA_WRITE BYHBA_IO_RATE BYHBA_READ_RATE BYHBA_WRITE_RATE BYHBA_BYTE_RATE BYHBA_READ_BYTE_RATE BYHBA_WRITE_BYTE_RATE BYHBA_REQUEST_QUEUE BYHBA_BUSY_TIME BYHBA_AVG_WAIT_TIME BYHBA_AVG_SERVICE_TIME BYHBA_TYPE
BYLS 度量	支持此度量。	Windows 和 Linux 平台不支持 BYLS 度量。 HPE Operations Agent 12.xx 不会从 Xen、KVM、VMware vSphere、Hyper-V 和其他虚拟化域收集 BYLS 度量。

SNMP 陷阱拦截器

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
opctrapi 进程	**opctrapi** 进程配置为拦截 SNMPv1 和 SNMPv2 陷阱。	**opctrapi** 进程配置为拦截 SNMPv1、SNMPv2、SNMPv3 陷阱并发出消息通知。

SNMP_SESSION_MODE 变量	SNMP_SESSION_MODE 变量可用于将 SNMP 陷阱拦截器配置为使用不同机制绑定到端口 162 以侦听来自外部源的 SNMP 陷阱：
	• 要将 SNMP 陷阱拦截器配置为使用侦听 SNMP 陷阱的 Net-SNMP API 绑定到端口 162，请运行以下命令：
	`ovconfchg -ns eaagt set SNMP_SESSION_MODE NETSNMP`
	• 仅在 Windows 节点上。要将 SNMP 陷阱拦截器配置为使用 Microsoft 陷阱服务侦听 SNMP 陷阱，请运行以下命令：
	`ovconfchg -ns eaagt set SNMP_SESSION_MODE WIN_SNMP`
	• 仅在 UNIX/Linux 节点上。要将 SNMP 陷阱拦截器配置为使用侦听 SNMP 陷阱的 OVSNMP API 绑定到端口 162，请运行以下命令：
	`ovconfchg -ns eaagt set SNMP_SESSION_MODE NO_TRAPD`
	• 要将 SNMP 陷阱拦截器配置为使用侦听 SNMP 陷阱的 OVSNMP API 绑定到端口 162，请运行以下命令：
		不支持 SNMP_SESSION_MODE 变量。
		使用 HPE Operations Agent 12.xx，仅支持 NETSNMP 模式。

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx

	`ovconfchg -ns eaagt set SNMP_SESSION_MODE NNM_LIBS`
	• 要将 SNMP 陷阱拦截器配置为尝试订阅 NNM (7.5x) 的 PMD 守护进程以侦听 SNMP 陷阱，请运行以下命令：
	`ovconfchg -ns eaagt set SNMP_SESSION_MODE TRY_BOTH`
	• 仅在 UNIX/Linux 节点上。要将 SNMP 陷阱拦截器配置为订阅 NNM (7.5x) 的 PMD 守护进程以侦听 SNMP 陷阱，请运行以下命令：
	`ovconfchg -ns eaagt set SNMP_SESSION_MODE NNM_PMD`	

数据源集成

将 HPE Operations Agent 12.xx 与早期版本进行比较

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
数据收集	DSI 收集的数据记录到日志文件中。	DSI 收集的数据记录到度量数据存储区中。对于记录到数据存储区的每一类数据，会创建一个特定的数据库文件。 **备注:** 要使 DSI 收集自定义数据，应运行 **oacore**。 但是，为保持向后兼容性，此命令行仍然支持 logfile 参数。日志文件名从路径中提取且被视为数据源名称。 `sdlcomp <class specication file> <logfile name>`

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
		从 HPE Operations Agent 11.xx 升级到 12.xx 时，元数据或数据存储格式会从旧的日志文件复制到度量数据存储区。 **备注:** DSI 编译器 **sdlcomp** 将不基于日志文件名创建文件，日志文件名改为用作数据源名称。 要访问 DSI 日志文件中记录的旧数据，请参见“DSI 数据源”。
数据滚动更新	`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。	不支持 `INDEX BY`、`MAX INDEXES` 和 `ROLL BY`。 默认情况下，存储自定义数据的数据库文件的最大大小设置为 1 GB。无法配置此大小。 当数据库文件达到 1 GB 的最大大小时，自动滚动更新度量数据存储区中存储的数据。 在滚动更新过程中，删除 20% 的最旧数据(永久删除最旧的数据库文件)。
操作	在滚动更新之前会执行 ACTION 参数中指定的命令。	不支持此参数。
CAPACITY	CAPACITY 是类中要存储的记录数。	不支持
sdlutil 语法	`sdlutil logfile_set [option]` 在此实例中，logfile_set 是通过编译类规范创建的日志文件集的名称。 变量和选项： • -classes 类列表	`sdlutil <log file name>[option]` 变量和选项： • -rm all 从指定数据源的度量数据存储区中删除数据源、类、度量和数据。 • -vers

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
	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，则提供所有类的类描述。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。如果不列出类，则提供所有类的类描述。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。 • -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。	显示版本信息。 • -? 显示语法描述。
PRECISION	PRECISION 标识要用于度量值的小数位数。	不支持

将 HPE Operations Agent 12.xx 与早期版本进行比较(续)

新增功能	HP Operations Agent 11.xx	HPE Operations Agent 12.xx
用 sdlgendata 测试记录进程	在开始记录数据之前，可以使用 sdlgendata 程序测试已编译的日志文件集和记录进程。sdlgendata 发现一类度量(如类规范中所述)并为类中的每个度量生成数据。	不支持

常见问题

如何在升级后验证旧数据是否存在？

执行以下步骤：

1. 升级到 HPE Operations Agent 12.xx 后，以具有根特权的身份登录。
2. 运行以下命令：

   ```
   <OvInstallBinDir>ovcodautil -obj
   ```
3. 将生成包含所有数据源、类和度量的列表。

如何在升级后删除旧日志文件？

执行以下步骤删除旧日志文件：

1. 以具有根特权的身份登录到 HPE Operations Agent 12.xx。
2. 运行以下命令停止 **oacore** 进程：

   ```
   ovc -stop oacore
   ```
3. 删除以下文件：

   在 Windows 上：

   ```
   <OvDataDir>/conf/perf/datasources

   <OvDataDir>/datafiles/coda*

   <OvDataDir>/datafiles/log*
   ```

   在 HP-UX/Linux/Solaris 上：

   ```
   <OvDataDir>/conf/perf/datasources

   <OvDataDir>/datafiles/coda*

   /var/opt/perf/datafiles/log*
   ```

4. 运行以下命令启动 oacore 进程：

   ```
   ovc -start oacore
   ```

如何重新创建度量数据存储区？

执行以下步骤重新创建度量数据存储区：

1. 运行以下命令停止 **oacore** 进程：

   ```
   ovc -stop oacore
   ```

2. 可选：备份现有数据库目录 <OvDataDir>databases/oa/
3. 运行以下命令删除数据存储区：

   在 Windows 上：

   ```
   del /F "<OvDataDir>databases\oa\*"
   ```

   ```
   "<OvInstallBinDir>\sqlite3.exe" "%OvDataDir%databases\oa\oa.db"
   <"%OvDataDir%conf\oa\Model\DMLMetaMetaSchema"
   ```

   在 HP-UX/Linux/Solaris/AIX 上：

   ```
   rm -f <OvDataDir>/databases/oa/*
   ```

   ```
   <OvInstallBinDir>/sqlite3 /var/opt/OV/databases/oa/oa.db
   <OvDataDir>/conf/oa/Model/DMLMetaMetaSchema
   ```

4. 运行以下命令将配置变量 UPDATED_MODEL_AVAILABLE 设置为 True。

   ovconfchg -ns oacore -set UPDATED_MODEL_AVAILABLE TRUE

5. 运行以下命令启动 **oacore** 进程：

   ```
   ovc -start oacore
   ```

备注：在后续提交中将自动重新创建自定义类(通过 API 或 DSI 提交)。

发送文档反馈

如果对本文档有任何意见，可以通过电子邮件与文档团队联系。如果在此系统上配置了电子邮件客户端，请单击以上链接，此时将打开一个电子邮件窗口，主题行中为以下信息：

用户指南 (Operations Agent 12.01) 反馈

只需在电子邮件中添加反馈并单击“发送”即可。

如果没有可用的电子邮件客户端，请将以上信息复制到 Web 邮件客户端的新邮件中，然后将您的反馈发送至 docfeedback@hpe.com 。

我们感谢您提出宝贵的意见！