CA Application Performance Management

.NET 代理实施指南

版本 9.5

CA Technologies 产品引用

本文档涉及以下 CA Technologies 产品和功能：

- CA Application Performance Management (CA APM)
- CA Application Performance Management ChangeDetector (CA APM ChangeDetector)
- CA Application Performance Management ErrorDetector (CA APM ErrorDetector)
- CA Application Performance Management for CA Database Performance (CA APM for CA Database Performance)
- CA Application Performance Management for CA SiteMinder® (CA APM for CA SiteMinder®)
- CA Application Performance Management for CA SiteMinder® Application Server Agents (CA APM for CA SiteMinder® ASA)
- CA Application Performance Management for IBM CICS Transaction Gateway (CA APM for IBM CICS Transaction Gateway)
- CA Application Performance Management for IBM WebSphere Application Server (CA APM for IBM WebSphere Application Server)
- CA Application Performance Management for IBM WebSphere Distributed Environments (CA APM for IBM WebSphere Distributed Environments)
- CA Application Performance Management for IBM WebSphere MQ (CA APM for IBM WebSphere MQ)
- CA Application Performance Management for IBM WebSphere Portal (CA APM for IBM WebSphere Portal)
- CA Application Performance Management for IBM WebSphere Process Server (CA APM for IBM WebSphere Process Server)
- CA Application Performance Management for IBM z/OS® (CA APM for IBM z/OS®)
- CA Application Performance Management for Microsoft SharePoint (CA APM for Microsoft SharePoint)
- CA Application Performance Management for Oracle Databases (CA APM for Oracle Databases)
- CA Application Performance Management for Oracle Service Bus (CA APM for Oracle Service Bus)
- CA Application Performance Management for Oracle WebLogic Portal (CA APM for Oracle WebLogic Portal)

- CA Application Performance Management for Oracle WebLogic Server (CA APM for Oracle WebLogic Server)
- CA Application Performance Management for SOA (CA APM for SOA)
- CA Application Performance Management for TIBCO BusinessWorks (CA APM for TIBCO BusinessWorks)
- CA Application Performance Management for TIBCO Enterprise Message Service (CA APM for TIBCO Enterprise Message Service)
- CA Application Performance Management for Web Servers (CA APM for Web Servers)
- CA Application Performance Management for webMethods Broker (CA APM for webMethods Broker)
- CA Application Performance Management for webMethods Integration Server (CA APM for webMethods Integration Server)
- CA Application Performance Management Integration for CA CMDB (CA APM Integration for CA CMDB)
- CA Application Performance Management Integration for CA NSM (CA APM Integration for CA NSM)
- CA Application Performance Management LeakHunter (CA APM LeakHunter)
- CA Application Performance Management Transaction Generator (CA APM TG)
- CA Cross-Enterprise Application Performance Management
- CA Customer Experience Manager (CA CEM)
- CA Embedded Entitlements Manager (CA EEM)
- CA eHealth® Performance Manager (CA eHealth)
- CA Insight™ Database Performance Monitor for DB2 for z/OS®
- CA Introscope®
- CA SiteMinder®
- CA Spectrum® Infrastructure Manager (CA Spectrum)
- CA SYSVIEW® Performance Management (CA SYSVIEW)

联系技术支持

要获取在线技术帮助以及办公地址、主要服务时间和电话号码的完整列表，请联系技术支持：http://www.ca.com/worldwide。

目录

第 1 章： .NET 代理简介

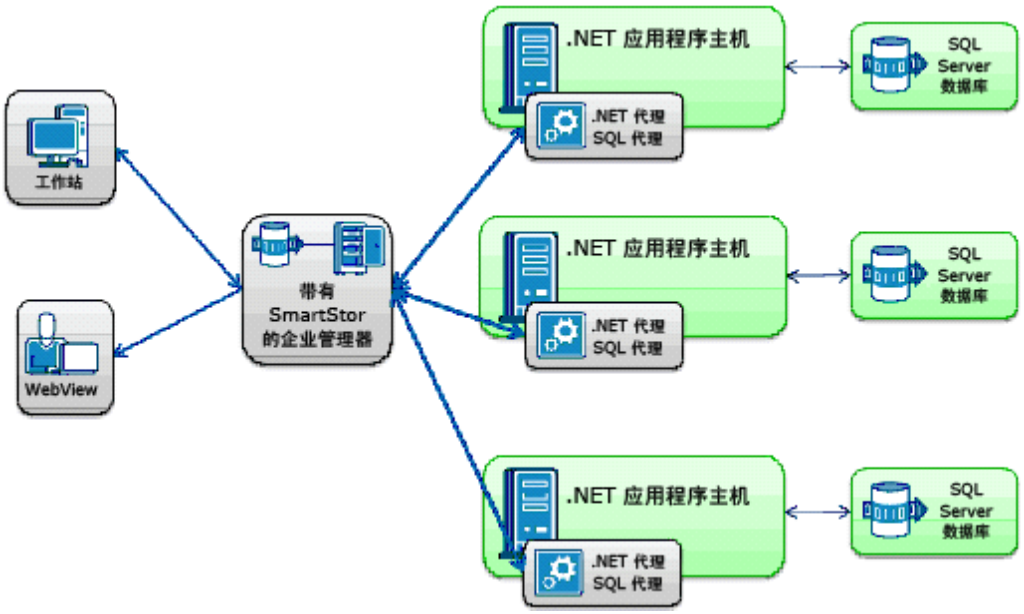
此部分包含以下主题：

关于 Introscope 部署中的 .NET 代理

.NET 代理是适用于 .NET 应用程序的应用程序管理解决方案。.NET 代理会监控 Microsoft 公共语言运行时 (CLR) 环境中运行的任务关键 .NET 应用程序，从而提供组件级别的可见性。

在 Introscope 部署中，代理收集应用程序和环境度量标准，并将其发送到企业管理器。向 Introscope 代理报告度量标准的应用程序称为“*已检测应用程序*”。在系统上安装和配置 .NET 代理后，此处运行的应用程序将在启动时自动检测。

下图显示了一个简单的 Introscope 部署，其中 .NET 应用程序连接到 SQL Server 数据库。每个应用程序服务器均托管着一个 .NET 和 SQL 代理。代理会监控应用程序活动，并将度量标准报告给企业管理器。可通过工作站和 WebView 查看度量标准。更大更复杂的部署可能包含更多代理和多个企业管理器。



默认情况下，仅检测使用 Microsoft Internet 信息服务 (IIS) 部署且在系统中处于活动状态的 ASP.NET 应用程序。您也可以检测独立的 .NET 可执行文件，或仅检测部分应用程序。

跟踪器执行的检测过程是在 ProbeBuilder 指令 (PBD) 文件中定义的。PBD 文件中的指令标识要监控的应用程序组件。跟踪器标识组件运行于 CLR 中时代理为其收集的度量标准。随后，.NET 代理将此度量标准信息发送到企业管理器。

企业管理器存储有多个代理报告的度量标准。然后您可以使用 Introscope Workstation 或 WebView 客户端应用程序来监控应用程序活动、调查性能问题的来源，并诊断问题。

详细信息：

修改默认检测 (p. 41)

.NET 代理在应用程序环境中的工作原理

在 Introscope 部署中，在运行要监控的应用程序的每台计算机上安装 .NET 代理。安装之后，Microsoft Internet 信息服务 (IIS) 会控制 .NET 代理。只有 IIS 收到用户对应用程序的请求时，.NET 代理才会活动。使用应用程序代码（例如 .aspx、.asmx）时，.NET 代理会启动，并且 NativeProfiler 会检测代码。

IIS 如何控制 .NET 代理

默认情况下，.NET 代理仅会监控由 IIS 管理并在 IIS 工作进程下运行的应用程序。以下步骤汇总了 IIS 如何控制 .NET 代理以及 .NET 应用程序启动时的检测过程。

1. IIS 收到用户对应用程序的请求。
2. IIS 启动 .NET 工作进程。
3. 请求的 .NET 应用程序启动。
4. 公共语言运行时 (CLR) 启动 NativeProfiler。
5. NativeProfiler 从全局程序集缓存 (GAC) 中加载 .NET 代理。
6. .NET 代理读取 IntroscopeAgent.profile 来确定要用于检测的 PBL 和 PBD 文件。
7. NativeProfiler 使用 PBL 和 PBD 文件中的信息在字节码中插入探测器，以从应用程序组件中收集相应的度量标准。将对应用程序进行检测。
8. 所检测应用程序开始向 .NET 代理报告度量标准。

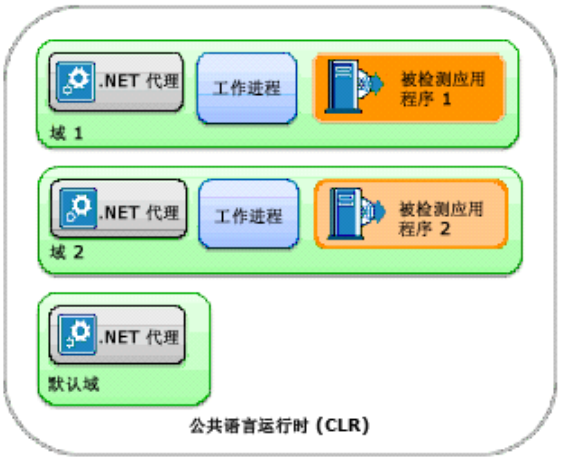
   只要 IIS 工作进程继续运行，代理就会收集度量标准，并将它们报告给企业管理器。如果所检测应用程序在一段时间内没有了用户活动，IIS 工作进程将停止应用程序进程。如果 IIS 停止了应用程序进程，就会有效地停止 .NET 代理，直到用户活动恢复。

注意：如果对 .NET 代理进行相应配置，也可以检测未使用 ASP.NET 的独立应用程序。除了会跳过步骤 1 和 2，对于独立应用程序而言，该过程是相似的，因为 Windows 操作系统可以启动独立应用程序。

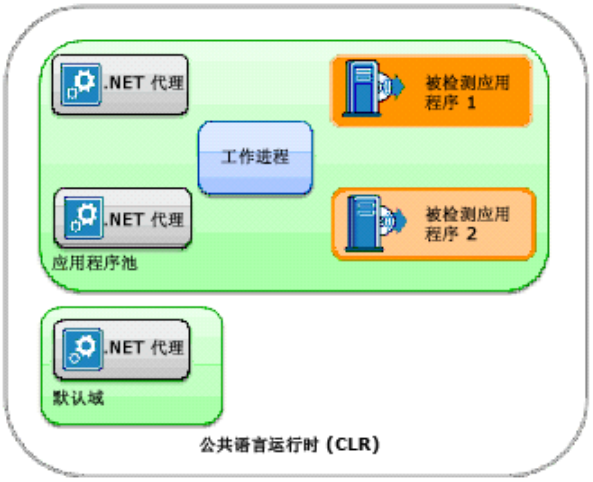
关于 .NET 代理安装和 IIS 工作进程

应在托管要监控的受管理 .NET 应用程序的每个系统上安装 .NET 代理。.NET 代理启动时，会为 CLR 的默认域创建一个代理实例。此外，还会为 CLR 中运行的每个应用程序各创建一个 .NET 代理实例。

下图显示了每个域（已启动一个 .NET 代理）中包含一个 IIS 工作进程的受管理 ASP.NET 应用程序：

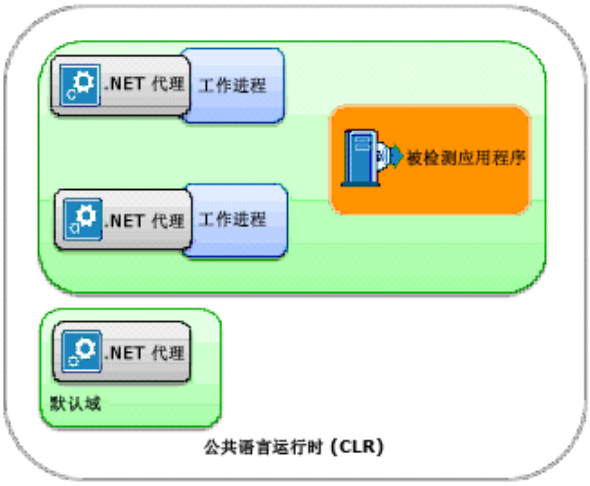


如果共享单个工作进程的 IIS 应用程序池中有多个 .NET 应用程序，则默认域有一个 .NET 代理，应用程序池中的每个应用程序也各有一个 .NET 代理，如下图所示：



由于扩展方面的原因，一些公司可能会将多个工作进程分配给单个应用程序。因此，会为默认域创建一个 .NET 代理实例，并为与应用程序关联的每个工作进程分别创建一个 .NET 代理实例。

注意：下图中显示的是最常见的配置。



如果有多个工作进程，因而有多个 .NET 代理与单个受管理应用程序关联，请将那些代理配置为虚拟代理。该配置可以聚合来自多个物理 .NET 代理的度量标准。

注意：有关详细信息，请参阅《*CA APM 配置和管理指南*》。

关于默认域中的 .NET 代理实例

如 .NET 代理实例化 (p. 15)中所述，总是会为 CLR 默认域创建一个 .NET 代理。默认情况下，默认域的 .NET 代理不会连接到企业管理器，也不会在工作站或 WebView 中显示为调查器树中的一个节点。但是，您可以根据需要将默认域的 .NET 代理配置为连接到企业管理器。有关更多信息，请参阅默认域配置 (p. 171)中的代理属性。

了解如何在企业中部署 .NET 代理

为应用程序及其运行环境开发适合的 .NET 代理配置是一个迭代过程。

安装和评估默认功能

部署代理的第一步是安装和评估默认代理配置。默认代理配置展示了对应用程序和计算环境的多个常见组件的数据收集。默认代理配置还包括几项已启用的功能，以及其他不太常用的已禁用功能。

您的目标是评估默认情况下提供的数据收集的深度和广度，并熟悉 Introscope 以及如何监控应用程序。在熟悉默认情况下代理提供的性能度量标准之后，您可以根据需要自定义代理，以便收集数据。

在评估默认性能时请切记，代理收集的度量标准越多，就会消耗越多的系统资源。代理收集的度量标准越少，您所了解的潜在问题也越少。当您优化代理配置时，请尝试打破数据收集的深度与可接受的性能水平之间的平衡。适当的检测级别通常取决于部署代理的位置。例如，在测试环境中进行监控的代理通常配置为收集大量度量标准。但是，生产服务器上的代理通常配置为提供基本信息。

确定配置要求

在部署代理之前，请确定您的数据收集要求。此信息允许您定制代理的数据收集行为，并通过代理的备用配置来评估对开销的影响。

Introscope 可在应用程序的整个生命周期中使用。例如，从开发到测试、负载验证、置备，直到生产。在生命周期的每个阶段，监控目标、环境约束和服务水平期望都会因时而异。要解决这些差异，您可以将代理配置为针对所监控的环境类型而采用不同的行为方式。

您的目标是在性能详细信息的可见性与资源开销之间确定相应的权衡。此外，请根据所监控环境的合理成本考虑最佳的可见性级别。

在预生产应用程序环境（如开发环境）中，通常应配置较高级别的数据收集，以便在应用程序性能方面提供更深的可见性。在生产或大容量事务环境中，通常应减少报告的度量标准以控制代理开销。根据您的要求，还可以将代理属性配置为控制特定的代理行为。例如，跟踪收集的最大度量标准数目以及对旧度量标准的删除。

针对您的环境，确定适当的可见性级别和可接受的性能开销，以便可以根据您的要求配置代理。

使用适当的配置属性定义基准代理配置文件

在确定要监控的应用程序环境类型后，可以创建“候选”代理配置。可使用代理配置文件 (IntroscopeAgent.profile) 中的属性控制大多数代理操作。例如，IntroscopeAgent.profile 文件可定义代理使用的 ProbeBuilder 指令文件和 ProbeBuilder 列表文件。然后，代理配置文件中列出的文件可以控制代理收集的特定度量标准。IntroscopeAgent.profile 文件还提供了用于启用或禁用特定功能或者调整操作（如轮询时间间隔）的属性。

根据您的配置和环境，可以在代理配置文件中调整属性，以便评估每项更改的影响。例如，您可以从未启用 ChangeDetector 的默认代理配置文件开始。然后，可以在添加其他任何功能之前，在配置文件中启用 ChangeDetector 属性，并在更改后评估代理性能。

评估代理性能开销

在评估代理配置时，请确定收集的度量标准是否在应用程序性能和可用性方面提供了足够的可见性。此外，还要确定度量标准的量是否会对操作环境带来无法承受的负载。代理无法报告比识别和定位性能和可用性问题所需更多的度量标准。

通过了解应用程序的性能特点，您可以有效地评估代理性能。例如，可以在实施默认监控之前和之后测试应用程序的负载，以验证影响。

要以受控方式扩展数据收集，请在实施更改之前和之后验证代理操作和开销。例如，一次只能为一个应用程序添加监控支持，这样您可以逐一评估每个附加程序的性能。

验证和部署代理配置

在验证候选代理配置是否提供所需的可见性后，在整个环境中部署该配置。

通过将以下配置项安装到目标环境，可以部署经过验证的配置：

- IntroscopeAgent.profile 文件
- 修改过或自定义的 PBD 文件

部署 .NET 代理

要部署 .NET 代理，请执行以下任务：

1. 安装 .NET 代理 (p. 21)。
2. 配置 .NET 代理 (p. 32)。
3. 配置 .NET 代理属性 (p. 47)。

详细信息：

了解 .NET 代理目录结构 (p. 30)

第 2 章： 安装 .NET 代理

本节中的说明介绍如何在 Microsoft Internet 信息服务 (IIS) 服务器上安装 .NET 代理以及将代理配置为与企业管理器进行通信。

注意：有关安装企业管理器、Workstation 和 WebView 组件的信息，请参阅《*CA APM 安装和升级指南*》。

此部分包含以下主题：

在安装 .NET 代理之前

如果您对 Introscope 或 .NET 代理不熟悉，请先查看部署过程。

请执行以下步骤：

1. 验证是否已安装了以下受支持版本的组件：
   - .NET framework

     注意：默认情况下，代理仅监控一个 Framework。通过使用 .NET Framework 4 和进程内并行执行功能，您可以配置代理属性以控制监控对象。
   - Windows 操作系统和 IIS
   - Microsoft SQL Server 或 Oracle 数据库
   - ODP.NET 驱动程序
2. 验证要监控的应用程序是否是使用 IIS 工作进程运行的。

   要验证 Web 应用程序是否在使用工作进程，请从应用程序加载一个页面，然后打开任务管理器并查找：
   - IIS 上的 aspnet_wp.exe

- IIS 上的 w3wp.exe

如果您没有看到工作进程，请使 IIS 能够处理受管理的组件。导航到您所用的 .NET 版本的 .NET Framework 目录并运行以下命令：

```
aspnet_regiis.exe -i
```

例如：

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe -i
```

3. 确认有足够的可用磁盘空间来安装代理文件。CA Technologies 建议，磁盘空间应相当于安装程序可执行文件大小的三倍。
4. 验证您打算安装新版 .NET 代理的计算机上是否已经删除了任何以前版本的 .NET 代理。

   通过打开命令窗口并在其中输入“*set*”以列出当前定义的环境变量，您可以验证 .NET 代理是否存在。如果列出了 com.wily.introscope.agentProfile=<*path*>，请先删除代理，然后才能继续操作。
5. 确认您打算安装 .NET 代理的计算机上没有安装另一监控代理或 CLR 探查器。
6. 确认您已经安装 Introscope 企业管理器和 Workstation。注意用于配置企业管理器连接的主机名和端口号。

   使用 *ping* 或 *telnet* 验证代理和企业管理器之间的连接。

注意：有关 .NET 代理支持版本和安装要求的信息，请参阅《*兼容性指南*》。

详细信息：

了解如何在企业中部署 .NET 代理 (p. 17)
配置代理属性 (p. 47)
删除 .NET 代理 (p. 44)

选择安装 .NET 代理的方法

您可以使用以下方法之一安装 .NET 代理：

- 使用图形用户界面 (GUI) 安装程序以交互方式安装 (p. 23)
- 无人值守方式，无交互提示 (p. 25)
- 使用安装存档手工安装 (p. 27)

在大多数情况下，如果要在计算机上本地安装文件，则可以使用交互式安装程序。如果要在计算机上远程安装文件或部署一组预配置安装说明，您可以在带命令行参数的脚本中运行安装程序。如果不想以交互方式或无人值守方式运行安装程序，您可以使用安装存档手工安装和配置代理。

重要信息！不管选择哪种方法，您的 SharePoint 域用户帐户都需要有“以服务身份登录”权限。

详细信息：

检查 .NET 代理是否已正确安装 (p. 127)

以交互方式安装 .NET 代理

您可以通过启动 .NET 代理安装程序并响应显示的提示来以交互方式安装 .NET 代理。

请执行以下步骤：

1. 双击 32 位或 64 位操作系统的代理安装程序可执行文件。例如，双击以下内容：

   `introscope<version>windowsAgent_dotNET_32.exe` 文件

   将显示欢迎页面。单击“下一步”开始安装。

   注意：您可以使用 .exe 或 .msi 可执行文件进行安装。提供 .msi 可执行文件是为了支持通过组策略进行软件交付或其他自动化交付，如已排定任务。.exe 可执行文件提供其他功能。例如，如果使用 .exe 可执行文件，但您未以管理员身份登录，则可以通过右键单击来以管理员身份运行此安装程序。

2. 单击“下一步”接受默认安装目录，或单击“更改”选择其他目录，然后单击“下一步”。

   .NET 代理的默认安装目录是：

   `C:\Program Files\CA APM\Introscope<version>`

   安装程序会在根安装目录内创建 *wily* 目录，即 *<Agent_Home>* 目录。

3. 输入接受代理报告度量标准的企业管理器的主机名和端口号，然后单击“下一步”。

   注意：代理的默认企业管理器是 *localhost*，此配置适用于企业管理器和代理安装在同一计算机上的实验室环境。在典型的生产环境中，请配置与远程企业管理器的连接。默认的企业管理器端口是 *5001*。

4. 选择要启用的监控选项，然后单击“下一步”。

   - ChangeDetector 将跟踪应用程序环境的变化并在 Workstation 中进行报告。如果启用了 ChangeDetector，请输入 ChangeDetector 代理的 ID。有关启用和使用 ChangeDetector 的详细信息，请参阅《*CA APM ChangeDetector 用户指南*》。
   - CA APM for SOA 可为客户端和服务器端 Web 服务提供扩展监控功能。在默认情况下，此选项处于选中状态。

     注意：默认情况下，.NET 代理可提供 Web 服务监控功能。选择 CA APM for SOA 可提供更多功能，例如，增强的事务跟踪功能、对 WCF 服务的监控以及其他依赖关系度量标准。安装后，.pbd 文件的内容取决于您是否启用了 CA APM for SOA。有关详细信息，请参阅《*CA APM for SOA 实施指南*》。
   - CA APM for Microsoft SharePoint 和 CA APM Standalone Agent for Microsoft SharePoint (SPMonitor) 支持对 Microsoft SharePoint Web 应用程序和服务进行监控。该选项仅在 Windows Server 2003 或 2008 上适用。有关详细信息，请参阅《*CA APM for Microsoft SharePoint 指南*》。

   如果不启用任何监控选项，您可以稍后手工启用监控选项。

5. 选择应当自动启动还是手工启动其他代理服务。

   如果您选择了 CA APM Standalone Agent for Microsoft SharePoint (SPMonitor)，请指定服务运行所需的 SharePoint 域用户帐户和密码。
   示例：*<domain>\<username>*

6. 单击“安装”。

   安装程序会创建 .NET 代理根安装目录，并安装代理使用的文件。

7. 安装完成后，单击“完成”。
8. 要验证代理是否已成功安装，请执行以下操作之一：

   - .exe 安装程序：

     验证 IntroscopeDotNetAgentInstall*.exe.log 文件与安装程序可执行文件是否位于同一目录中。
   - .msi 安装程序：

     验证是否在 %temp% 文件夹下创建了 MSI*.LOG 文件。日志文件仅在 Windows Installer 4.0 或更高版本中可用。

9. 重新启动 IIS 管理服务以完成安装。

注意：安装 .NET 代理后会将以下系统环境变量添加到操作系统中：*com.wily.introscope.agentProfile*、*Cor_Enable_Profiling*、*COR_PROFILER*。要向 IIS 通知系统环境变量已发生更改，需要重新启动 IIS 或重新引导本地计算机。

详细信息：

了解 .NET 代理目录结构 (p. 30)
如何配置企业管理器和代理连接 (p. 32)

以无人值守方式安装 .NET 代理

要以无人值守模式运行安装程序，请从命令行调用 .NET 代理安装程序，并指定安装说明作为命令行参数。随后安装会在后台运行，不会显示任何关于其进度的信息。由于此安装方式使您无需用户交互就可以安装代理，因此它最常用于在远程计算机上安装 .NET 代理。您也可以使用相同配置安装多个代理。您可以在命令中指定 *.exe* 或 *.msi* 可执行文件。

请执行以下步骤：

1. 打开“命令提示符”窗口。
2. 使用相应的命令行参数调用相应的 .exe 或 .msi 安装程序可执行文件。例如，要在 64 位系统上安装，您可以使用以下命令行参数：

   .msi 安装程序：

   ```
   IntroscopeDotNetAgentInstall64_9.1.0.0.msi /qn
   [INSTALLDIR="<install_dir>"] [EMHOST=myhost] [EMPORT=5001] [ENABLECD=1
   CDAGENTID=<change_dectector_agent_id>] [ENABLESOA=1] [ENABLESPP=1]
   [INSTALLSPMONITOR=1 IS_NET_API_LOGON_USERNAME=<SP Monitor Domain
   username> IS_NET_API_LOGON_PASSWORD=<SP Monitor Domain password>]
   ```

   .exe 安装程序：

   ```
   IntroscopeDotNetAgentInstall64.exe /s /v"/qn
   [INSTALLDIR="<install_dir>"] [EMHOST=myhost] [EMPORT=5001] [ENABLECD=1
   CDAGENTID=<change_dectector_agent_id>] [ENABLESOA=1] [ENABLESPP=1]
   [INSTALLSPMONITOR=1] [INSTALLSPMONITOR=1 IS_NET_API_LOGON_USERNAME=<SP
   Monitor Domain username> IS_NET_API_LOGON_PASSWORD=<SP Monitor Domain
   password>]"
   ```

   注意：您可以使用 */s /v /qn* 选项调用安装程序以无提示方式进行安装。也可以选择指定其他参数。如果您没有在命令行中指定参数，则使用该参数的默认设置。

   这些可选参数等同于您以交互方式运行安装程序时进行的选择：

   INSTALLDIR="*<root_installation_directory>*"

   指定 .NET 代理的安装目录。如果路径包含反斜线，请在路径中添加一个前导反斜线。例如：

   ```
   \"C:\IntroscopeDir"
   ```

   默认根安装目录是：

   ```
   C:\Program Files\CA APM\Introscope<version>
   ```

   要更改默认目录，请修改此属性。

EMHOST=*<host_name>*

指定接受代理报告度量标准的企业管理器的主机名。代理的默认主机名是 localhost。

EMPORT=*<port_number>*

指定代理要向其报告度量标准的企业管理器的侦听端口号。默认端口为 5001。

ENABLECD=0

指定是否启用 ChangeDetector。该参数的默认值为 0，表示不能启用 ChangeDetector。

如果您想启用 ChangeDetector，请将 *ENABLECD* 参数设置为 1。

如果保留 ChangeDetector 为禁用，相关文件将放在 *<Agent_Home>*\examples 目录中，可以稍后启用。

CDAGENTID=*<agent_name>*

指定 ChangeDetector 代理的名称。仅当 *ENABLECD* 设置为 1 时在命令行中包含此参数。ChangeDetector 代理的默认代理名称为 *SampleApplicationName*。

ENABLESOA=0

指定是否要启用 CA APM for SOA。该参数的默认值为 0，表示不能启用 CA APM for SOA。

如果您想启用 CA APM for SOA，请将 *ENABLESOA* 参数设置为 1。

如果保留 CA APM for SOA 为禁用，相关文件将放在 *<Agent_Home>*\examples 目录中，可以稍后启用。

ENABLESPP=0

指定是否要启用 CA APM for Microsoft SharePoint Portal。此参数的默认值为 0，表示不能启用 CA APM for Microsoft SharePoint。

如果您想启用 CA APM for Microsoft SharePoint Portal，请将 *ENABLESPP* 参数设置为 1。仅当操作系统是 Windows Server 2003 或 Windows Server 2008 时才启用对 Microsoft SharePoint 的监控。

如果保留 CA APM for Microsoft SharePoint Portal 为禁用，相关文件将放在 *<Agent_Home>*\examples 目录中，可以稍后启用。

INSTALLSPMONITOR=0

指定是否安装 CA APM Standalone Agent for Microsoft SharePoint Portal。此参数的默认值为 0，表示不能安装 CA APM Standalone Agent for Microsoft SharePoint。

如果您想安装 CA APM Standalone Agent for Microsoft SharePoint，请将 *INSTALLSPMONITOR* 参数设置为 1。

如果没有安装 CA APM Standalone Agent for Microsoft SharePoint，您稍后可以通过运行 CA APM Standalone Agent 安装程序来安装它。

例如，要更改默认设置（在 64 位系统上），请指定类似于以下内容的命令行：

```
IntroscopeDotNetAgentInstall64.exe /s /v"/qn
INSTALLDIR=\"C:\IntroscopeAgent\" EMHOST=dell-M65.org EMPORT=5001
ENABLECD=1 CDAGENTID=myCDAgent ENABLESOA=1 ENABLESPP=1 INSTALLSPMONITOR=1
IS_NET_API_LOGON_USERNAME=apm-domain\apmuser
IS_NET_API_LOGON_PASSWORD=apmPassword"
```

IS_NET_API_LOGON_USERNAME=*<domain><username>***

指定 Microsoft SharePoint 监视器域用户名。在 *INSTALLMONITOR* 参数设置为 0 时，提供此参数。此参数的默认值为空值。

IS_NET_API_LOGON_PASSWORD=*<password>*

指定 Microsoft SharePoint 监视器的密码。此参数的默认值为空值。

3. 要验证代理是否已成功安装，请查看如下日志文件：
   - *.exe* 安装程序：

     查看在安装程序可执行文件所在的目录中的 IntroscopeDotNetAgentInstall*.exe.log 文件。
   - *.msi* 安装程序：

     查看在 %temp% 文件夹中的 MSI*.LOG 文件。日志文件仅在 Windows Installer 4.0 或更高版本中可用。
4. （可选）创建一个使用适当的设置调用安装程序的脚本作为排定任务，或者在其他计算机上远程运行安装程序。
5. 重新启动 IIS 管理服务以完成安装。

注意：安装 .NET 代理后会将以下系统环境变量添加到操作系统中：*com.wily.introscope.agentProfile*、*Cor_Enable_Profiling* 和 *COR_PROFILER*。要向 IIS 通知系统环境变量已发生更改，需要重新启动 IIS 或重新引导本地计算机。

使用安装存档手工安装 .NET 代理

您可以将代理文件放置在系统中，而不以交互方式运行安装程序或配置响应文件。您可以通过特定于应用程序服务器的存档安装代理。

安装存档包含代理安装程序的相同文件。将存档复制到计算机之后，解压缩其中的内容并配置代理。使用这些文件在批处理作业中部署多个代理，或将这些文件保留为默认代理文件集的一个存档。

请执行以下步骤：

1. 根据您的操作系统，从 CA Support 上的 CA APM 软件下载区域下载下列安装存档之一。
   - DotNetAgentFiles-NoInstaller.x64.<APM 版本号>.zip
   - DotNetAgentFiles-NoInstaller.x86.<APM 版本号>.zip

   例如：

   ```
   DotNetAgentFiles-NoInstaller.x64.9.1.0.0.zip
   ```
2. 将安装存档内容解压缩到您所选的目录中。
3. 为 .NET 代理设置 *<Agent_Home>*\wily 目录。

   注意：运行 .NET 代理安装程序时，默认根安装目录是 C:\Program Files\CA APM\Introscope*<version>*。安装程序会从根安装目录创建 *wily* 目录，即 *<Agent_Home>* 目录。
4. 将 wily.Agent.dll 文件注册到全局程序集缓存中，如下所示：
   a. 以管理员身份打开命令提示符。
   b. 导航到 *wily**bin* 目录。例如：
      ```
      explorer <Agent_Home>\wily\bin
      ```
   c. 导航到 *assembly* 目录。例如：
      ```
      explorer C:\WINDOWS\assembly
      ```
   d. 将 wily.Agent.dll 从 *<Agent_Home>*wily\bin 目录复制到 assembly 目录。
5. 注册 wily.NativeProfiler 文件，如下所示：
   a. 从命令行中导航到 wily\bin 目录并调用以下可执行文件：
      ```
      C:\WINDOWS\SysWOW64\regsvr32.exe
      <Agent_Home>\wily\bin\x86\wily.NativeProfiler.dll
      ```
6. 在文本编辑器中打开 IntroscopeAgent.profile 文件，并配置企业管理器连接，如下所示：
   a. 找到以下行：
      introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT，并指定代理的企业管理器的主机名。例如：
      ```
      introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=sfcollect01
      ```
   b. 根据需要设置与企业管理器通信所需的其他属性。

7. 从命令行启动 PerfMon 收集器服务，如下所示：

   ```
   sc create PerfMonCollectorAgent binPath=
   "<Agent_home>\bin\PerfMonCollectorAgent.exe start= auto DisplayName= "CA
   APM PerfMon Collector Service"
   ```

8. 要验证用户是否拥有访问 *<Agent_Home>* 目录的权限，请从命令行运行 *wilypermissions* 实用工具。例如：

   ```
   wilypermissions.exe c:\WilyAgent9065\wily w3wp.exe
   ```

9. 配置下列 .NET 代理环境变量：

   - com.wily.introscope.agentProfile=*<Agent_Home>*\wily\IntroscopeAgent.profile
   - Cor_Enable_Profiling=0x1
   - COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}

10. 要完成安装，请重新启动 IIS 管理服务。

详细信息：

修改对代理目录的默认用户权限 (p. 31)
如何配置企业管理器和代理连接 (p. 32)

安装中文版或日语版的 .NET 代理

在运行 .msi 安装程序可执行文件时，您可以指定语言值。

注意：此信息仅适用于 32 位和 64 位 Windows。

请执行以下步骤：

1. 打开“命令提示符”窗口。
2. 调用具有语言值的 .msi 安装程序可执行文件，如下所示：

   - 日语：

     ```
     IntroscopeDotNetAgentInstall64_<release>.msi TRANSFORMS=1041.mst
     ProductLanguage=1041
     ```

   - 中文：

     ```
     IntroscopeDotNetAgentInstall64_<release>.msi TRANSFORMS=2052.mst
     ProductLanguage=2052
     ```

3. 要完成安装，请重新启动应用程序服务器。

   安装程序会创建 .NET 代理根安装目录，并安装代理文件。将在 %temp% 文件夹中创建 .MSI*.LOG 文件。

了解 .NET 代理目录结构

安装 .NET 代理后，会在根安装目录中创建以下 *<Agent_Home>* 目录结构：

wily

包含用于控制代理操作、度量标准数据收集和检测过程的 IntroscopeAgent.profile、ProbeBuilder 指令文件 (.pbd) 以及 ProbeBuilder 列表文件 (.pbl)。在 IntroscopeAgent.profile 中定义并在代理中部署和引用的特定属性取决于您在安装过程中所做的选择。

在 *<Agent_Home>* 目录中，子目录会提供用于启用 .NET 代理的各种功能的库和扩展文件。

bin

包含核心 .NET 代理库。

dynamic

包含用于动态检测的 PBD 文件。

examples

包含用于可选代理扩展（例如 CA APM for SOA）的文件夹和文件。如果在安装时未启用扩展，请在以后使用此目录中的文件来配置和启用扩展。

ext

包含用于已启用代理扩展或功能的文件。例如，该目录中包含用于应用程序分类地图和 LeakHunter 的文件。

hotdeploy

默认情况下该目录为空。无需编辑 IntroscopeAgent.profile 或重新启动应用程序就可以将 PBD 文件放置于该目录中来部署新的指令。但是，在将文件放置于此目录之前请谨慎考虑。如果自定义 PBD 包含无效语法或过多度量标准，检测可能会失败或应用程序性能可能会降低。

logs

存储 .NET 代理日志文件。

readme

只有在安装过程中启用了扩展才会创建该目录。如果该目录存在，其中会包含关于所启用扩展的自述信息。

tools

包含网络接口集成实用工具和 Tagscript 工具文件：TagScript.jar、TagScript.bat 和 TagScript.sh 命令行脚本

version

包含可选代理扩展的版本信息，无论您是否启用它们。

代理目录的用户权限

默认情况下，安装程序授予每个人对 <代理主目录> 目录的如下读取访问权限：

- 对 *<Agent_Home>*\wily 的读取权限
- 对 *<Agent_Home>*\bin 和 *<Agent_Home>*\ext 的读取和执行权限
- 对 *<Agent_Home>*\logs 和 *<Agent_Home>*\dynamic 的读取和写入权限

如果要限制对该目录的访问，您可以修改默认权限，以便仅允许运行 IIS 工作进程的用户帐户访问 *<代理主目录>* 目录。

详细信息：

验证对代理目录的用户权限 (p. 39)

修改对代理目录的默认用户权限

通过从命令行运行 *wilypermissions* 实用工具，可以修改对 *<代理主目录>* 目录的默认用户权限。

如果未指定任何用户，该实用工具会检测到应用程序的默认 IIS 用户并向该用户授予权限。

请执行以下步骤：

1. 导航到 *<Agent_Home>* 目录。
2. 从命令行运行 *wilypermissions.exe*，如下所示：

   ```
   wilypermissions.exe <代理主目录> [进程名称]
   ```

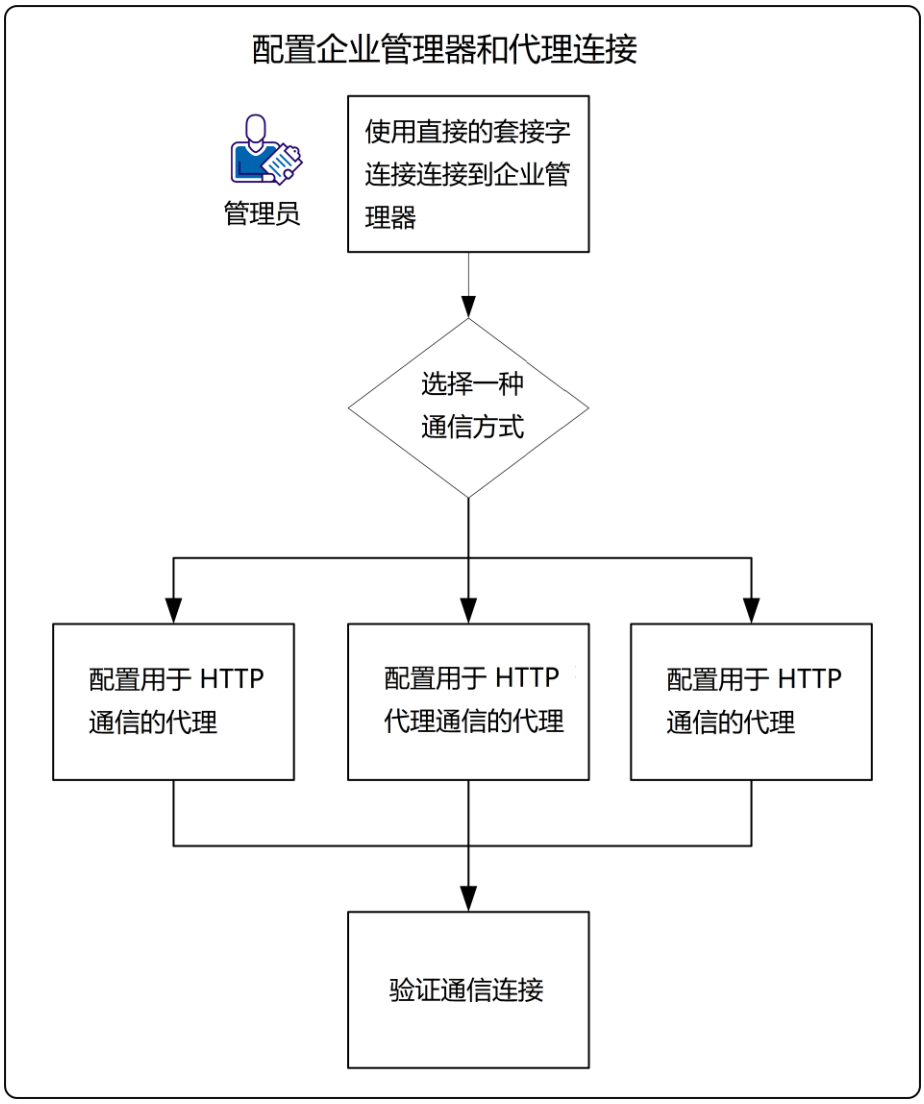
   该实用工具执行以下操作：授予用户对 *<Agent_Home>* 目录的权限，以及对性能监视器计数器的访问权限。

   注意：进程名称必须包括文件扩展名。如果未指定进程名称，则默认为 IIS 工作进程名称。

如何配置企业管理器和代理连接

在 CA APM 部署中，代理将收集 Web 应用程序和环境度量标准，并将其转发到企业管理器。代理必须连接到企业管理器才能报告度量标准。默认 CA APM 通信设置可以使代理连接到本地企业管理器。作为管理员，您可以修改默认设置，以便在企业管理器和代理位于不同的系统上时，企业管理器和代理可以进行连接。您可以配置相应设置以使用某种通信方法：HTTP、HTTP 代理或 HTTPS。

下图说明了如何以管理员身份配置企业管理器和代理连接。



要配置企业管理器和代理连接，请执行以下步骤：

1. 使用直接套接字连接来连接到企业管理器 (p. 33)。

2. 选择通信方法 (p. 34)：
   - 为代理配置 HTTP 通信 (p. 34)。
   - 为代理配置 HTTP 代理通信 (p. 35)。
   - 为代理配置 HTTPS 通信 (p. 35)。
3. 验证通信连接 (p. 38)。

使用直接套接字连接来连接到企业管理器

代理可通过直接套接字连接来连接到企业管理器。我们建议使用直接套接字连接来连接到企业管理器（如果可以）。可以配置相关通信属性，以便使用直接套接字连接来连接到企业管理器。

请执行以下步骤：

1. 在文本编辑器中打开 IntroscopeAgent.profile 文件。
2. 找到 introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT 属性，并指定在默认情况下代理所要连接到的企业管理器的主机名。例如：
   ```
   introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=sfcollect01
   ```
3. 将 introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT 属性设置为企业管理器侦听端口。例如：
   ```
   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=5001
   ```
4. 将 introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT 属性设置为用于连接到企业管理器的套接字工厂。例如：
   ```
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
   ```
   注意：在大多数情况下，您可以使用该属性的默认设置。如果要使用不同的套接字工厂，请修改该属性。
5. 保存并关闭 IntroscopeAgent.profile 文件。
6. 重新启动 IIS 管理服务。

   此时配置已完成。

选择通信方法

安装代理之后，可以修改默认代理设置，以使用下列通信方法之一：

- 超文本传输协议 (HTTP)
- HTTP 代理
- 安全超文本传输协议 (HTTPS)

要配置代理与企业管理器的连接，请选择一种通信方法，并执行与该通信方法对应的以下任务之一：

- 为代理配置 HTTP 通信 (p. 34)。
- 为代理配置 HTTP 代理通信 (p. 35)。
- 为代理配置 HTTPS 通信 (p. 35)。

为代理配置 HTTP 通信

通过 HTTP 通道，一个网络可以使用另一网络中的连接安全地发送数据。您可以将代理连接配置为允许通过防火墙（仅允许 HTTP 通信）进行通信。

注意：默认情况下，企业管理器已启用 HTTP 通道。

请执行以下步骤：

1. 导航到 <*Agent_Home*>/wily 目录。
2. 在文本编辑器中打开 IntroscopeAgent.profile 并设置以下属性：

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=hostname or
   IP_address of the Enterprise Manager
   ```

3. 将以下属性设置为企业管理器的嵌入式 Web 服务器的 HTTP 侦听端口。例如：

   ```
   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=8081
   ```

   将此值与企业管理器上 <企业管理器主目录>/config/IntroscopeEnterpriseManager.properties 文件中的 introscope.enterprisemanager.webserver.port 属性值相匹配。默认情况下，该端口为 8081。
4. 将以下属性设置为 HTTP 通道套接字工厂。例如：

   ```
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=co
   m.wily.isengard.postofficehub.link.net.HttpTunnelingSocketFactory
   ```

5. 保存并关闭文件。
6. 重新启动应用程序。

   与企业管理器 Web 服务器端口的连接可确认代理使用 HTTP 来连接到企业管理器。

为代理配置 HTTP 代理通信

您可以配置 HTTP 代理，以便通过代理服务器连接到企业管理器。如果转发代理服务器仅允许出站 HTTP 流量，并且代理在防火墙后端运行，则需要此配置。代理服务器配置适用于代理上任何已配置的 HTTP 通道连接，而不是仅适用于单个连接。在配置多个企业管理器之间的故障转移时，如果与每个企业管理器的连接均通过 HTTP 进行，请考虑此设置。

重要信息！代理服务器必须支持 HTTP Post。

重要信息！如果代理服务器不可访问，则代理绕过该代理服务器，并直接连接到企业管理器。如果代理服务器可访问，但是其身份验证失败，则代理会继续重试通过代理服务器连接到企业管理器。

请执行以下步骤：

1. 导航到 <*Agent_Home*>/wily 目录。
2. 在文本编辑器中打开 IntroscopeAgent.profile。
3. 取消注释并设置以下属性：

```
introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=hostname or IP address of the proxy server
introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=Enterprise Manager web server port
introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.HttpTunnelingSocketFactory
```

4. 取消注释以下属性，并将这些属性设置为代理主机和端口：

```
introscope.agent.enterprisemanager.transport.http.proxy.host=hostname
introscope.agent.enterprisemanager.transport.http.proxy.port=port number
```

5. 取消注释以下属性，并将该属性设置为代理用户名和密码：

```
introscope.agent.enterprisemanager.transport.http.proxy.username=username
introscope.agent.enterprisemanager.transport.http.proxy.password=password
```

6. 保存并关闭文件。
7. 重新启动应用程序。

为代理配置 HTTPS 通信

代理可以使用 HTTP over SSL（安全套接字层）(HTTPS) 来连接到企业管理器。HTTPS 加密和解密用户页面请求和 Web 服务器返回的页面。在线事务期间，通过 SSL 保护您的网站。

请执行以下步骤：

1. 导航到 <*Agent_Home*>/wily 目录。
2. 在文本编辑器中打开 IntroscopeAgent.profile。
3. 取消注释以下属性，并将该属性设置为目标企业管理器的主机名或 IP 地址：

   introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=*企业管理器的主机名或 IP 地址*
4. 取消注释以下属性，并将该属性设置为企业管理器的嵌入式 Web 服务器的 HTTP 侦听端口。例如：

   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=8444
5. 取消注释以下属性，并将该属性设置为使用 HTTPS 通道套接字工厂。例如：

   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.HttpsTunnelingSocketFactory
6. 保存并关闭 IntroscopeAgent.profile。
7. 确认代理和企业管理器具有相互通信所需的共用的密码套件。

   注意：对于代理端而言，安装在主机内的任何密码均可用于与企业管理器进行通信。主机的组策略设置中设置的优先级控制着密码的优先级。
8. 将 wily 目录中现有的用户添加到 pfx 文件中，并向它们授予适当的权限 (p. 39)。
9. 在文本编辑器中打开 IntroscopeAgent.profile。
   a. 取消注释以下属性，并按以下方式设置 pfx 文件的位置和密码：

      introscope.agent.enterprisemanager.transport.tcp.keystore.DEFAULT=<*pfx 文件的路径*>
      introscope.agent.enterprisemanager.transport.tcp.keypassword.DEFAULT=<*pfx 密码*>
   b. 保存并关闭文件。

向“受信任的根证书颁发机构”目录安装可信证书

如果您在客户端组与企业管理器之间使用安全套接字层 (SSL) 连接，则代理必须提供可信证书以进行验证。可以使用证书导入向导来安装和指定用于 SSL 连接的相应证书文件的路径。

请执行以下步骤：

1. 在应用程序服务器上，导航到“控制台根节点”\“证书”（本地计算机）。
2. 右键单击“受信任的根证书颁发机构”文件夹，然后选择“所有任务”、“导入”。

   此时将显示证书导入向导。
3. 单击“下一步”。
4. 在向导的引导下进行操作时，请填写以下字段：

   要导入的文件

   指定要导入的文件，例如：<*DVD*>\wilyHome\agentCert

   将所有证书放在以下存储区中

   将所有证书放在您指定的存储区中。

   证书存储区

   指定证书存储区，例如：受信任的根证书颁发机构。
5. 单击“下一步”，然后单击“完成”。

   证书已导入。

向“个人”目录安装可信证书

向“个人”目录安装可信证书，以使用私钥维护安全性。

请执行以下步骤：

1. 在客户端上，导航到“控制台根节点”\“证书”（本地计算机）。
2. 右键单击“个人”文件夹，然后选择“所有任务”、“导入”。

   此时将显示证书导入向导。
3. 单击“下一步”。
4. 在提示符下填写以下字段：

   文件名

   指定要导入的路径和文件，例如：<*DVD*>\wilyHome\agentCer

   将所有证书放在以下存储区中

   将所有证书放在同一存储区中。

   证书存储区

   指定将放入证书的存储区，例如：个人

5. 单击“下一步”，然后单击“完成”。

   证书已导入。

确认通信连接

配置企业管理器和代理后，请确认通信连接。例如，成功连接至端口 8081 可以确认 HTTP 通信。

请执行以下步骤：

1. 导航到 <*代理主目录*>/wily/logs 目录。
2. 打开代理日志并查找以下消息：

   ```
   [DEBUG] [IntroscopeAgent.HttpOutgoingConnection] Established client
   connection: host:hostname, port: port num
   ```

   注意：*Hostname* 为代理所连接的计算机的名称，*port num* 为代理和企业管理器所连接的端口。

   例如：如果您连接到端口 8444，日志将包括以下消息：

   ```
   [DEBUG] [IntroscopeAgent.HttpOutgoingConnection] Established client
   connection: host:hostname, port: 8444
   ```

检查用于 IIS 工作进程的用户权限

通常，.NET Web 应用程序会在 IIS 工作进程中运行。默认情况下，IIS 工作进程名称为 *w3wp.exe* 或 *aspnet_wp.exe*。运行工作进程的默认帐户为 *NETWORK SERVICE*。安装 .NET 代理时，安装程序会为创建的根安装目录设置适当的权限，允许对 .NET <*代理主目录*> 或 *wily* 目录以及 *bin*、*ext*、*log* 子目录和性能监视器计数器进行访问。

但是，可以配置其他帐户在应用程序池级别运行工作进程。如果您配置了运行 IIS 工作进程的其他用户帐户，请执行以下操作：

- 确定哪个用户帐户运行应用程序
- 验证是否运行应用程序所用的所有帐户都拥有访问 <*代理主目录*> 目录的权限
- 为运行性能监视器收集代理的帐户设置用户权限，允许其访问性能监视器计数器

确定运行应用程序的用户

在一些组织中，*NETWORK SERVICE* 之外的帐户会运行 IIS 工作进程。要确保您可以监控其他帐户运行的应用程序，您应当针对要监控的所有应用程序确定与 IIS 工作进程关联的用户名。对于识别出的每个用户名，应当验证该用户是否拥有访问 *<Agent_Home>* 目录和性能监视器计数器的权限。

请执行以下步骤：

1. 验证应用程序是否正在运行。
2. 右键单击任务栏，然后选择“任务管理器”以打开 Windows 任务管理器。
3. 单击“进程”选项卡。
4. 滚动以在“映像名称”列找到 *w3wp.exe* 或 *aspnet_wp.exe* 进程对应的条目。
5. 检查“用户名”列以确定运行进程的用户帐户。

   例如，*w3wp.exe* 工作进程的默认用户是 *NETWORK SERVICE*。如果使用其他用户帐户运行 IIS 工作进程，应注意帐户名称，以便可以验证该帐户是否拥有适当的权限。

验证对代理目录的用户权限

要使 .NET 代理正常运行，运行 IIS 工作进程的用户帐户必须拥有 *<Agent_Home>* 目录中文件的正确用户权限以及对性能监视器计数器的访问权限。默认情况下，安装程序授予每个人对 *<代理主目录>* 文件夹的读取访问权限。因此，如果要限制对该目录的访问，您应当修改默认权限，以便仅允许运行 IIS 工作进程的用户帐户访问 *<代理主目录>* 目录。

请执行以下步骤：

1. 在 Windows 资源管理器中，右键单击 .NET 代理根文件夹，然后选择“属性”。

   此时将显示该文件夹的“属性”窗口。
2. 单击“安全”选项卡。
3. 单击“添加”。

   此时将显示“选择用户或组”对话框。
4. 在“*输入要选择的对象名称*”中键入完整或部分用户名。
5. 单击“检查姓名”搜索一个或多个匹配的用户名。

6. 选择相应的用户帐户，然后单击“确定”。

   此时将显示“权限条目”对话框。

7. 在“完全控制”下，验证是否为用户选中了“*允许*”。

   启用“完全控制”可以为用户提供修改、读取和运行、列出文件夹目录、读取以及写入权限。

8. 单击“高级”。

   此时将显示“高级安全设置”对话框。

9. 选中“用在此显示的可以应用到子对象的项目替代所有子对象的权限项目”以将权限传播给子目录，然后单击“确定”。

设置“性能监控”度量标准的用户权限

要在 Introscope 调查器中查看“性能监控”度量标准，运行 IIS 工作进程的用户帐户必须具有正确的权限。如果使用了 wilypermissions 实用工具，将自动定义这些权限。如果未使用 wilypermissions 实用工具，则必须手工为运行 IIS 工作进程的用户帐户设置权限。

请执行以下步骤：

1. 从“开始”菜单导航到“设置”>“控制面板”>“管理工具”>“本地安全策略”>“本地策略”>“用户权限分配”。
2. 右键单击“配置单一进程”，然后选择“属性”。
3. 单击“添加用户或组”。
4. 将运行 IIS 工作进程的用户帐户添加到用户列表中，然后单击“确定”。
5. 右键单击“配置系统性能”，然后选择“属性”。
6. 单击“添加用户或组”。
7. 将运行 IIS 工作进程的用户帐户添加到用户列表中，然后单击“确定”。

详细信息：

修改对代理目录的默认用户权限 (p. 31)

自定义检测

默认情况下，.NET 代理会使用 NativeProfiler 检测在一个或多个 IIS 工作进程下运行的所有应用程序和应用程序池。虽然默认检测提供对 .NET 环境和 Web 应用程序的广泛和深入的监控，但是您可以根据需要进一步细化要监控的内容。

修改默认检测

默认情况下，NativeProfiler 会检测所有 IIS Web 应用程序和应用程序池，但不检测在 IIS 外部运行的任何独立应用程序。随着您的环境变得越来越复杂，您可能想要修改默认检测。例如，您可能想要：

- 更改已部署的默认 PBL 或 PBD
- 将 .NET 代理配置为检测非 IIS 应用程序。
- 阻止检测特定的进程或应用程序。
- 限制对特定应用程序池的检测。

要执行这些更改，您必须配置 .NET 代理配置文件。

指定用于部署的默认 ProbeBuilder 指令

代理配置文件提供了属性 *introscope.autoprobe.directivesFile*，指定应当部署哪些 ProbeBuilder 指令 (*.pbd*) 文件。各 ProbeBuilder 指令文件控制插入代码中的特定探测器（如计时器和计数器）。这些文件可以在 ProbeBuilder 列表 (*.pbl*) 文件中进行分组，这些列表文件定义要部署的一组 *.pbd* 文件。安装代理时，*introscope.autoprobe.directivesFile* 配置为使用 *default-full.pbl* 中列出的文件来检测应用程序。*default-full.pbl* 引用了演示对所有 .NET 组件的完整检测的 *.pbd* 文件。您可以修改 *introscope.autoprobe.directivesFile* 属性以使用不同的 *.pbl*，列出特定的 *.pbd* 文件，或修改要部署的 *.pbl* 中列出的 *.pbd 文件*列表。

更改部署的默认 ProbeBuilder 指令的最常见方式是修改 *introscope.autoprobe.directivesFile* 属性以使用 *default-typical.pbl* 文件。*default-typical.pbl* 文件引用了检测要监控的组件子集的 *.pbd* 文件。

从完整检测更改为典型检测：

1. 停止 IIS。
2. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
3. 找到 *introscope.autoprobe.directivesFile* 属性。
4. 将 *default-full.pbl* 更改为 *default-typical.pbl*。例如：
   `introscope.autoprobe.directivesFile=default-typical.pbl,hotdeploy`

5. 保存并关闭文件。
6. 重新启动 IIS。

有关使用 ProbeBuilder 指令以及使用 .NET 代理监控的默认组件的详细信息，请参阅“自定义默认数据收集”。有关 ProbeBuilder 指令文件中使用的语法以及自定义监控的详细信息，请参阅“使用 ProbeBuilder 指令 (p. 73)”。

检测在 IIS 外部运行的进程和应用程序

要监控在 IIS 外部运行的应用程序，请修改 .NET 代理配置文件以包括您想监控的应用程序。修改配置文件之前，请弄清想要包括的应用程序可执行文件的确切名称。

检测在 IIS 外部运行的进程或应用程序：

1. 停止 IIS。
2. 在文本编辑器中打开 *IntroscopeAgent.profile*。
3. 找到 *Restricted Instrumentation* 部分。
4. 将应用程序名称添加到以下属性中：
   `introscope.agent.dotnet.monitorApplications`

   默认情况下，已经为此属性列出 *w3wp.exe* 和 *aspnet_wp.exe*。可以添加其他应用程序名称，以逗号进行分隔。例如：
   ```
   introscope.agent.dotnet.monitorApplications=w3wp.exe,aspnet_wp.exe,RandomApp.exe,testapp.exe,readloop.exe
   ```

   重要信息：该属性列表区分大小写。不支持相对路径和通配符。不支持完整路径规范；请仅使用应用程序名称。
5. 将以下属性添加到 *IntroscopeAgent.profile* 中，并将其设置为 *false*：
   `introscope.agent.dotnet.runInRestrictedMode=false`
6. 保存并关闭文件。
7. 重新启动 IIS。

如果要对某个应用程序禁用监控，请将其从 *introscope.agent.dotnet.monitorApplications* 属性列表中删除。被删除的应用程序的 CLR 探查器仍然处于活动状态。NativeProfiler 被关闭，因而与应用程序关联的 .NET 代理无法连接到企业管理器，也无法报告度量标准。

检测特定的应用程序池

默认情况下，会检测所有应用程序池。但是，如果要减少开销或只想将监控重点放在最关键的资源上，可以限制要检测的 IIS 应用程序。要限制所监控的应用程序，必须在代理配置文件中标识要检测的特定应用程序池。

检测特定的应用程序池：

1. 停止 IIS。
2. 在文本编辑器中打开 *IntroscopeAgent.profile*。
3. 找到 *Restricted Instrumentation* 部分。
4. 取消注释以下属性：
   `introscope.agent.dotnet.monitorAppPools=`
5. 将要检测的应用程序池添加到逗号分隔列表内的属性中。例如：
   `introscope.agent.dotnet.monitorAppPools="NULL","DefaultAppPool","AppPool1","AppPool2"`

   注意：IIS 5 可在没有应用程序池的情况下运行。如果您要检测在 IIS 5 中运行的应用程序，请使用 "Null" 值。
6. 保存并关闭文件。
7. 重新启动 IIS。

配置应用程序空闲时间

在 Introscope 部署中，在运行要监控的应用程序的每个系统上安装 .NET 代理。安装之后，Microsoft Internet 信息服务 (IIS) 会控制 .NET 代理。如果所检测的应用程序没有了用户活动，IIS 将停止应用程序进程。如果由于非活动状态 IIS 停止了应用程序进程，Introscope 调查器 中的 .NET 代理的节点将变为不可用。

您可以使用 Internet 信息服务 (IIS) 管理器配置应用程序在变为不可用之前可以处于空闲状态的时间。通过配置空闲时间，您可以控制应用程序度量标准在 Introscope 调查器中可供使用的时间。

请执行以下步骤：

1. 导航到“开始”、“所有程序”、“管理工具”、“Internet 信息服务 (IIS) 管理器”。
2. 右键单击要配置的应用程序，然后选择“属性”。
3. 单击“虚拟目录”选项卡。
4. 在该选项卡的“应用程序设置”部分，单击“配置”。
5. 单击“选项”选项卡。
6. 验证“启用会话状态”选项。
7. 设置空闲时间（以分钟为单位），然后单击“确定”关闭“应用程序配置”对话框。
8. 单击“确定”关闭“属性”对话框。

   配置已设置。

删除 .NET 代理

要从操作系统中删除 .NET 代理，请使用下列方法之一：

- 交互式卸载 (p. 45)
- 无提示卸载 (p. 45)
- 手工卸载 (p. 46)

注意：请使用适用于您的操作系统的卸载程序。例如，如果您使用的是 .exe 安装程序，那么请使用相应的 .exe 卸载程序来删除代理。否则，该过程将无法删除代理。

以交互方式删除 .NET 代理

只要被检测应用程序正在运行，.NET 代理就会处于活动状态。删除或修改任何代理 DLL 文件之前，请确保所有被检测应用程序已停止。然后可以使用“添加或删除程序”控制面板本地删除代理文件，或使用卸载程序以无人值守方式删除代理文件。

请执行以下步骤：

1. 停止 IIS 服务。此操作可停止所有被检测应用程序。
2. 导航到“开始”>“设置”>“控制面板”>“添加或删除程序”。
3. 从当前安装的程序列表中选择以下程序之一：
   - CA APM .NET 代理*<release>*（32 位）
   - CA APM .NET 代理*<release>*（64 位）
4. 单击“删除”。

   系统会提示您是否确定要删除代理。
5. 单击“是”删除代理。

   将取消注册 .NET 代理 DLL，并删除相关环境变量和代理文件。
6. 关闭“添加或删除程序”。
7. 重新启动 IIS 管理服务或重新启动计算机。
8. 选择代理的根安装目录，单击右键，然后选择“删除”。

   将删除 .NET 代理。

以静默模式删除 .NET 代理

您能够以静默模式删除 .NET 代理和关联文件。

请执行以下步骤：

1. 停止 IIS 服务以停止所有被检测的应用程序。
2. 在命令提示符处运行以下命令之一：

   ```
   IntroscopeDotNetAgentInstall*.exe /s /x /v"qn"
   IntroscopeDotNetAgentInstall*.msi /x /qn
   ```

   将删除 .NET 代理。

手工删除 .NET 代理

您可以手工删除 .NET 代理和相关文件。

注意：只要被检测应用程序正在运行，.NET 代理就会处于活动状态。删除或修改任何代理 DLL 文件之前，必须确保所有被检测应用程序已停止。

请执行以下步骤：

1. 导航到 C:\WINDOWS\assembly 目录。
2. 右键单击 wily.Agent.dll，然后从菜单中选择“卸载”。

   代理文件即已删除。
3. 以管理员身份打开命令提示符。
   a. 删除如下 NativeProfiler 文件：
      - C:\WINDOWS\system32\regsvr32.exe /u *<Agent_Home>*\bin\wily.NativeProfiler.dll
      - C:\WINDOWS\SysWOW64\regsvr32.exe /u *<Agent_Home>*\bin\x86 \wily.NativeProfiler.dll

   b. 通过运行以下命令来删除 PerfMon 收集器服务：
      ```
      sc delete PerfMonCollectorAgent
      ```
   c. 将以下 .NET 代理环境变量设置如下：
      - com.wily.introscope.agentProfile=*<Agent_Home>*\IntroscopeAgent.profile
      - Cor_Enable_Profiling=0x1
      - COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}
4. 重新启动 IIS 以完成卸载过程。

第 3 章： 配置代理属性

可使用 *IntroscopeAgent.profile* 文件中的属性配置大多数代理操作。本节介绍了要设置的最常见的代理属性。可能还有其他属性适用于您的环境。不同版本的代理可能会提供不同的属性供您设置，也可能具有不同的默认值。

此部分包含以下主题：

如何配置备份企业管理器和故障切换属性

安装代理时，可以指定代理在默认情况下连接到的企业管理器的主机名和端口号。如有需要，还可以指定一个或多个备份企业管理器。如果代理中断了与其主企业管理器的连接，可以尝试连接到备份企业管理器之一。

要使代理能够连接到备份企业管理器，请在代理配置文件中指定企业管理器的通信属性。如果主企业管理器不可用，代理会尝试连接到允许连接列表上的下一个企业管理器。如果代理无法与其第一个备份企业管理器连接，则它会尝试列表中的下一个企业管理器。此进程会按顺序一直尝试连接列表中的每个企业管理器，直到连接成功。如果代理无法连接到任何企业管理器，则它会等待 10 秒，然后再重试。

请执行以下步骤：

1. 在文本编辑器中打开 *Introscope.Agent.profile* 文件。
2. 通过将以下属性添加到每个备份企业管理器的代理配置文件，指定一个或多个备用企业管理器通信通道：

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.NAME
   introscope.agent.enterprisemanager.transport.tcp.port.NAME
   Introscope.agent.enterprisemanager.transport.tcp.socketfactory.NAME
   ```

将 *NAME* 替换为新企业管理器通道的标识符。在创建通道时，请不要使用 DEFAULT 或现有通道的名称。例如，要创建两个备份企业管理器：

```
introscope.agent.enterprisemanager.transport.tcp.host.BackupEM1=paris
introscope.agent.enterprisemanager.transport.tcp.port.BackupEM1=5001
Introscope.agent.enterprisemanager.transport.tcp.socketfactory.BackupEM1=
com.custom.postofficehub.link.net.DefaultSocketFactory
introscope.agent.enterprisemanager.transport.tcp.host.BackupEM2=voyager
introscope.agent.enterprisemanager.transport.tcp.port.BackupEM2=5002
introscope.agent.enterprisemanager.transport.tcp.socketfactory.BackupEM2=
com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
```

3. 找到 *introscope.agent.enterprisemanager.connectionorder* 属性，并将其设置为主企业管理器和备份企业管理器标识符的逗号分隔列表。标识符的列出顺序定义了它们的连接顺序。例如：

   ```
   introscope.agent.enterprisemanager.connectionorder=DEFAULT,BackupEM1,Back
   upEM2
   ```

4. 找到 *introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds* 属性，并指定代理尝试连接其主企业管理器的频率。默认时间间隔为 120 秒（2 分钟）。例如：

   ```
   introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds=120
   ```

5. 保存更改并关闭 *IntroscopeAgent.profile* 文件。
6. 重新启动应用程序。

如何为应用程序使用特定的代理配置文件

默认情况下，会根据应用程序的虚拟目录或上下文路径为监控 Web 应用程序的 .NET 代理自动分配一个名称。会根据应用程序域名为监控非基于 Web 应用程序的代理分配一个名称。CA Technologies 建议尽可能使用自动代理命名。但是，也可以根据需要为代理明确地分配一个名称。例如，如果有多个 .NET 代理实例，您可能会希望单独管理这些实例并手工分配名称。

要手工分配代理名称，请执行以下操作：

- 为每个代理实例和应用程序创建单独的代理配置文件。
- 定义每个配置文件中的进程和代理名称属性。
- 为特定应用程序和代理实例设置代理配置文件位置。

为每个应用程序创建单独的配置文件

如果要为监控特定 .NET 应用程序的代理实例明确分配名称，必须为每个应用程序创建一个单独的 *IntroscopeAgent.profile*。通过为每个代理实例和应用程序使用单独的配置文件，您可以根据需要控制进程和代理名称以及自定义其他属性。

请执行以下步骤：

1. 复制 *<Agent_Home>* 目录中现有的 IntroscopeAgent.profile 文件。
2. 将复制的配置文件粘贴到新的目录中。
3. （可选）重命名配置文件，以便将其标识为一个代理实例配置文件。例如，*IntroscopeAgentCalc1.profile*。

定义代理名称

使用自定义进程和代理名称有助于标识监控特定应用程序的代理实例。还可以根据需要修改实例的其他属性。

请执行以下步骤：

1. 在文本编辑器中打开为代理实例和应用程序创建的 *IntroscopeAgent.profile*。例如，如果您在 *C:\NetApps\wily* 目录中创建了一个名为 *IntroscopeAgentCalc1.profile* 的自定义配置文件，则在文本编辑器中打开该文件
2. 找到 Custom Process Name 部分。
3. 使用 *introscope.agent.customProcessName* 属性指定要显示的自定义进程名称。例如：
   ```
   introscope.agent.customProcessName=ArcadeCaclcProcess
   ```
4. 使用 *introscope.agent.AutoNamingEnabled* 属性禁用自动代理命名。例如：
   ```
   introscope.agent.agentAutoNamingEnabled=false
   ```
5. 使用 *introscope.agent.agentName* 属性指定要显示的自定义代理名称。例如：
   ```
   introscope.agent.agentName=ArcadeCalcAgent
   ```
6. 保存并关闭更新的自定义配置文件。

为特定应用程序和代理实例设置代理配置文件位置

默认情况下，IntroscopeAgent.profile 文件安装在 *<Agent_Home>*\wily 目录中。例如，默认位置为 C:\Program Files\CA APM\Introscope<version>\wily。如果为监控特定应用程序的代理实例创建单独的配置文件，则必须为该应用程序和代理实例创建一个用于指定配置文件位置的配置文件。

请执行以下步骤：

1. 在文本编辑器中打开 *<Agent_Home>**Sample.exe.config* 文件。
2. 复制 *Sample.exe.config* 文件中的所有示例 *<configSections>* 和 *<com.wily.introscope.agent>* 部分。
3. 打开 .NET 应用程序的 *web.config* 文件，或打开非 IIS 应用程序可执行文件的独立配置文件。

   有关为应用程序可执行文件创建配置文件的详细信息，请参阅“特定于应用程序的配置 (p. 219)”。
4. 在 *<env.parameters>* 部分中修改代理配置文件位置。例如：

```
<env.parameters>
   <add key="com.wily.introscope.agentProfile"
value="C:\\NETApps\\wily\\IntroscopeAgentCalc1.profile" />
</env.parameters>
```

5. 保存 *web.config* 或独立配置文件。
6. 重新启动托管应用程序。

如何收集和自定义性能监视器数据

默认情况下，.NET 代理部署单独的 Windows 服务“性能监视器收集代理”，以便从所有的 Windows 性能监视器对象、计数器和实例收集度量标准。该 Windows 服务报告有关运行在 IIS 服务器上的所有代理实例和进程的信息。在安装过程中，指定该服务应自动启动还是必须手工启动。安装后，您可以使用“服务控制面板”来管理服务。例如，您可以使用“服务控制面板”来更改启动类型，或者暂停和恢复性能监视器计数器的收集。

通过在代理配置文件中修改属性，您可以自定义性能监视器收集代理所收集的数据。例如，您可以：

- 使用正则表达式筛选收集的特定度量标准。
- 控制返回的性能监视器度量标准总数。
- 控制性能监视器计数器的检查频率。
- 控制性能监视器收集代理检查新的性能监视器对象的频率。
- 防止性能监视器收集代理检查新的性能监视器对象。

注意：运行性能监视器收集代理的帐户必须拥有对性能监视器计数器的访问权限，才能在调查器中使用数据。默认情况下，性能监视器收集代理服务使用 Local System 帐户运行，该帐户有权访问性能监视器计数器。但是，您可以使用“服务控制面板”来标识运行该服务的其他用户帐户和密码。有关设置适当权限的信息，请参阅设置用于 IIS 工作进程的用户权限 (p. 38)。

使用正则表达式筛选度量标准收集

.NET 代理属性 *perfmon.metric.filterPattern* 用于指定代理读取的性能监视器计数器。默认设置为：

```
introscope.agent.perfmon.metric.filterPattern=|Processor|*|*, |.NET Data
Provider*|*|*, |.NET CLR*|{osprocessname}|*, |.NET CLR
Data|*|*, |Process|{osprocessname}|*, |ASP.NET|*
```

筛选器采用 */Object/Instance/Counter* 或 */Object/Counter* 格式（如果没有实例），其中：

- *Object* 标识性能监视器的类别，如 Memory、Processor 或 Process。
- *Instance* 标识指定对象的特定实例。某些对象（如 Memory）没有实例。
- *Counter* 标识要收集的 *Object*/*Instance* 的特定度量标准类型。例如，.NET CLR Memory 性能监视器对象包含的计数器有 # Bytes in all heaps、Gen 0 heap size、# GC handles 和 % time in GC 等。

默认筛选还包含 *{osprocessname}* 占位符。在调查器中，*{osprocessname}* 占位符会被替换，以标识监控的独立应用程序的实例或 IIS 工作进程的应用程序池名称，例如 *w3wp(BusinessServiceAppPool)*。

重要信息：使用筛选器 |*|* 相当于要求性能监视器将所有计数器枚举为缺少实例的计数器，这会导致破坏某些计数器。

可以通过修改 *introscope.agent.perfmon.metric.filterPattern* 属性值来定制 .NET 代理收集的性能监视器数据。例如，通过修改默认筛选，您可以扩大或缩小所报告数据的范围。如果已为应用程序定义自定义性能监视器计数器，也可以包括这些自定义计数器。

已保留某些性能监视器度量标准，以供 Microsoft 将来实施。在性能监视器中看到这些度量标准时，它们标记为“NotDisplayed”。在 Introscope 调查器中查看这些度量标准时，会显示占位符标记。

设置对度量标准总数的限制

默认情况下，性能监视器收集代理会收集所有可用的性能监视器对象、实例和计数器的性能监视器数据。通过对报告的性能监视器度量标准总数设置上限，您可以限制数据的总体范围。设置性能监视器度量标准最大数量有助于减少所监控服务器上的性能监视器收集代理的开销。

请执行以下步骤：

1. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
2. 找到 *perfmon.metric.limit* 属性并将其设置为每个时间间隔内允许的性能监视器度量标准的最大数目。例如：
   ```
   introscope.agent.perfmon.metric.limit=100
   ```
3. 保存并关闭 *IntroscopeAgent.profile* 文件。

控制收集性能监视器数据的频率

默认情况下，性能监视器收集代理会每隔 15 秒从所有性能监视器对象、实例和计数器检查度量标准值。此轮询时间间隔可确保及时报告以前发现的性能监视器对象的当前数据。性能监视器收集代理也会定期检查从中收集数据的新性能监视器对象。默认情况下，此浏览时间间隔是 10 分钟，这是为了确保发现可用对象并删除过时对象时不招致过多开销。您可以在代理配置文件中修改这两个时间间隔中的任何一个或同时修改两个。

请执行以下步骤：

1. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
2. 找到 *introscope.agent.perfmon.metric.pollIntervalInSeconds* 属性，并配置性能监视器收集代理的轮询时间间隔。设置的值可控制性能监视器收集代理检查度量标准值的频率。例如：
   ```
   introscope.agent.perfmon.metric.pollingIntervalInSeconds=20
   ```
3. 找到 *introscope.agent.perfmon.category.browseIntervalInSeconds* 属性，并配置性能监视器收集代理的浏览间隔。设置的值可控制性能监视器收集代理检查新的或过时的性能监视器对象的频率。例如：
   ```
   introscope.agent.perfmon.category.browseIntervalInSeconds=900
   ```
4. 保存并关闭 *IntroscopeAgent.profile* 文件。

阻止浏览性能监视器对象

性能监视器收集代理会定期查询性能监视器对象，以确保所有可用服务均考虑在内。此操作使得代理可以发现新的计数器，或消除过时的计数器（如果需要）。默认情况下此功能处于启用状态，查询之间的默认时间间隔设置为 600 秒（10 分钟）。

要减少系统开销，您可以阻止性能监视器收集代理检查新的和过时的计数器。

请执行以下步骤：

1. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
2. 找到 *introscope.agent.perfmon.category.browseEnabled* 属性并将其设置为 *false*。例如：
   ```
   introscope.agent.perfmon.category.browseEnabled=false
   ```
3. 保存并关闭 *IntroscopeAgent.profile* 文件。

启动对性能监视器数据的收集

.NET 代理部署一个外部服务（CA APM PerfMon 收集器服务）以收集和报告性能监视器度量标准。该服务使您可以在计算机级别收集度量标准，而不是收集每个 .NET 代理的度量标准。只有该 .NET 代理实例才使用 PerfMon 系统资源。

注意：在应用程序服务器上部署 .NET 代理后，请运行该服务。

请执行以下步骤：

1. 以管理员身份登录到 Windows 管理控制台。
2. 导航到 CA APM PerfMon 收集器服务。
3. 右键单击该服务，然后单击“启动”。

   将启动对数据的收集。

停止对性能监视器数据的收集

您可以使所有性能监视器计数器停止收集管理 .NET 应用程序不需要的性能监视器数据。

请执行以下步骤：

1. 打开“服务控制面板”。
2. 在服务名称列表中找到性能监视器收集器服务。
3. 选中性能监视器收集器服务，右键单击，然后选择“属性”。
4. 单击“停止”。
5. 选择“手工”或“禁用”作为“启动类型”。
6. 单击“确定”。

如何控制启动时间

许多定期使用 IIS 的组织会重新启动 IIS 服务，以便重复利用每个应用程序域的 .NET 应用程序池。只要 IIS 重新启动，会同时调用 .NET 代理以检测每个应用程序池中的应用程序。初始启动时间会因所监控的应用程序和类的数量、代理文件的配置情况以及是否存在自定义 PBD 而有所不同。

使用 NativeProfiler 检测的代理的默认设置允许代理和应用程序服务器在合理的时间内启动。要改善启动性能，您可以执行一些可选步骤。

如果要改进 .NET 代理的启动时间，请尝试执行以下任务：

- 更改代理配置文件中 *introscope.nativeprofiler.directivematching.cache.max.size* 属性的值。

  默认情况下，代理会创建先前发现的指令组的内存缓存，其中包含受监控的类。启动 IIS 时，代理会创建先前发现的类的缓存。由于应用程序代码会监视新类，缓存将随时间不断增加。默认情况下，内存缓存最多可存储 5000 个类名。如果缓存大小达到此限制，代理将在 NativeProfiler 日志文件中记录一个条目，指出缓存已饱和。

  您可以使用 *IntroscopeAgent.profile* 文件中的 *introscope.nativeprofiler.directivematching.cache.max.size* 属性增加或减少缓存的大小。如果缓存存储 5000 个以上的类名，增大该值可以改进启动时间。但是，增大该值会增加代理的内存开销。减小该属性值可减少代理的内存开销。监控的类少于 5000 个或停止监控大指令组时，应适当减少该值。

- 检查您如何识别自定义 PBD 文件中的类和指令。

  如果对同一组中的类使用 *IdentifyInheritedAs* 指令，则可以启用代理以便最为高效地使用继承层次结构。

如何启用进程内并行执行

使用 .NET Framework 4，您可以在同一进程内运行使用不同版本的 .NET Framework 的应用程序。旧的组件将继续使用旧的 .NET Framework 版本，而新组件则会使用新的 .NET Framework 版本。

默认情况下，.NET 代理会检测最早在主机进程中加载的应用程序的 .NET Framework。例如，如果启动的第一个应用程序使用 .NET Framework 2.0，则默认情况下仅检测 .NET Framework 2.0 组件。

com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs 属性使 .NET 代理在同一进程内检测多个 .NET Framework 版本的组件。

使用 *com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs* 属性，您可以指定要监控的 .NET Framework 实例。例如，如果将此属性设置为仅监控 .NET Framework 4，则仅运行在 .NET Framework 4 上的组件会报告度量标准。利用 *com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs* 属性，可通过进程内并行执行功能同时监控 .NET Framework 4 和 .NET Framework 2 应用程序和度量标准报告。

请执行以下步骤：

1. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
2. 找到 *com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs* 属性。
3. 要指示您想监控的 .NET Framework 版本，请按照以下示例所示设置此属性（用逗号隔开）：

   ```
   com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs=v2,v4
   ```
4. 保存并关闭文件。
5. 重新启动托管应用程序。

配置代理负载平衡

在工作负荷是代理报告的主要度量标准的群集中，可以通过配置 MOM 代理负载平衡来优化总体群集容量。

注意：有关 MOM 代理负载平衡的详细信息，请参阅《*CA APM 配置和管理指南*》。

如何将代理配置为收集分布统计信息度量标准

您可以将代理配置为收集通过 BlamePointTracers 分析的响应时间分布情况，以创建平均响应时间度量标准。分布统计信息度量标准可提供详细说明，可解释具体操作的变化方式。有关监控的响应时间值的其他统计分布数据可通过 getExtendedMetricData Web 服务获取。

可将 CA APM 代理配置为收集特定操作的响应时间信息。这些响应时间存储在分布统计度量标准中，它与选定操作的平均响应时间度量标准成对。

请执行以下步骤：

1. 打开 <*Agent_Home*>/wily/core/config 目录中的 *IntroscopeAgent.profile* 文件。
2. 通过取消注释并编辑 introscope.agent.distribution.statistics.components.pattern 来指定要为其创建成对的分布统计信息度量标准的平均响应时间度量标准。例如，使用以下表达式：

   ```
   ASP\\.NET\\|login_aspx:.*
   ```

   将生成 login.aspx 响应时间的分布统计信息。
3. （可选）按照以下准则创建表达式：
   a. 在度量标准节点分隔符前加上反斜杠，因为竖线和句点在正则表达式中具有特殊含义。
   b. 在反斜杠前再加上一条反斜杠，因为反斜杠在代理配置文件中具有特殊含义。

      正则表达式在代理日志中显示在特殊字符之后。例如：

      ```
      "ASP\.NET\|login_aspx:.*"
      ```

   c. 将摘要级别与各个度量标准正则表达式进行匹配。如果度量标准不匹配，则汇总或不创建摘要级别的分布统计信息。

      以下示例显示匹配的表达式：

      ASP\\.NET(\\|.*):.* 既要匹配 ASP.NET 摘要级别，又要匹配所有单个 ASP 页面

      只要没有其他度量标准路径以字母“ASP.NET”开头，就可以使用“ASP\\.NET.*”
4. 保存并关闭 IntroscopeAgent.profile 文件。
5. 重新启动代理。

分布统计信息度量标准的示例

根据您设置配置属性的方式，可以针对以下示例收集分布统计信息：

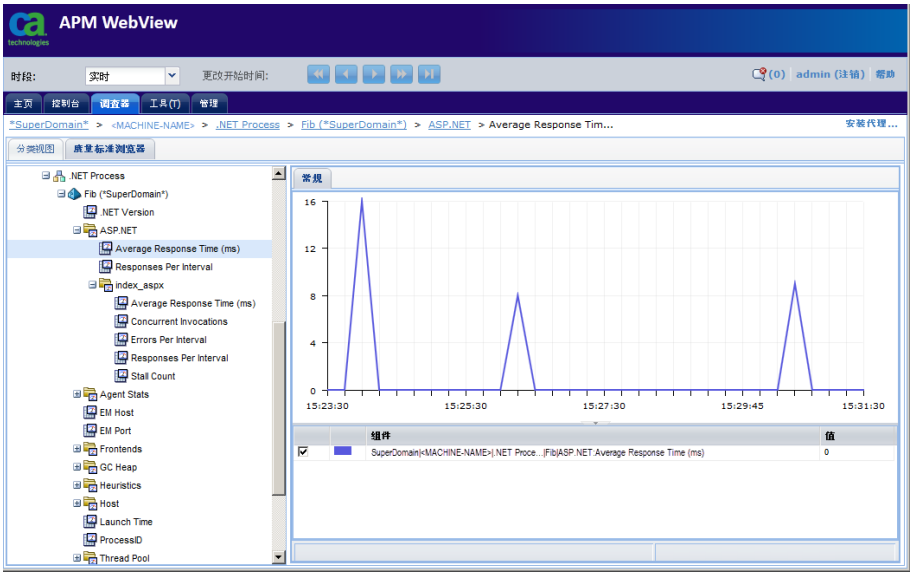
- ASP 页面和摘要级别分布度量标准 (p. 58)
- ASP 页面级别分布度量标准 (p. 59)

ASP 页面和摘要级别分布度量标准示例

通过使用以下模式收集 ASP 页面和摘要级别分布度量标准：

```
introscope.agent.distribution.statistics.components.pattern=ASP\\.NET(\\|.*):.*
```

下图显示的调查器收集 .NET 代理节点 ASP 页面和摘要级别的分布统计信息度量标准：

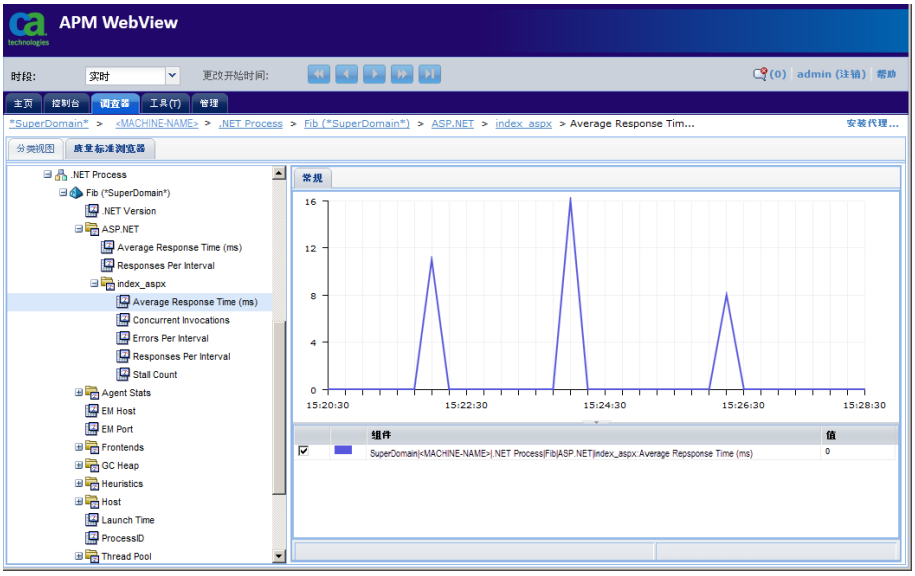


ASP 页面分布度量标准的示例

通过使用以下模式仅收集 ASP 页面分布度量标准：

```
introscope.agent.distribution.statistics.components.pattern=ASP\\,NET\\|.*
```

下图显示的调查器收集 .NET 代理节点 ASP 页面级别的分布统计信息度量标准：



配置对综合事务的检测

使用 introscope.agent.synthetic.header.names 参数已完成对综合事务监控的配置设置。

introscope.agent.synthetic.header.names 参数值列出了 HTTP 头参数，这些参数可用于确定监控的 HTTP 请求是否为综合事务的一部分。使用逗号分隔单个参数名。如果此参数是未定义的，或值为空，则检测不到综合事务。如果定义了多个 HTTP 头参数名，则会按指定的顺序对这些参数名进行检查。第一个 HTTP 参数的值可用于定义综合事务。

报告综合事务的节点取决于用于检测每个事务的特定 HTTP 头参数，如下所示：

- 如果参数值是除 *lisaframeid* 或 *x-wtg-info* 信息之外的任何值，则 HTTP 参数值本身将用作节点名称。完成适当的修改以确保使用有效的节点名称。

- 如果参数值是 *lisaframeid*，则合成的节点名称为 CA LISA。
- 如果参数值是 *x-wtg-info*，则假定 HTTP 头参数值包含一系列名称和值对。使用和号符号将对彼此分隔开。等号在每个对之内分隔属性名称和值。*group*、*name*、*ipaddress* 以及 *request_id* 的值使用 | 节点分隔符形成综合事务节点名称。

例如，给定以下参数：

```
introscope.agent.synthetic.header.names=Synthetic_Transaction,x-wtg-info,lisa
frameid
```

以下 *x-wtg-info* 头可导致在节点 SampleGroup|sample|192.168.193.1|start 下报告度量标准：

```
clear
synthetic=true&instance=ewing&name=sample&group=SampleGroup&version=4.1.0&ipa
ddress=192.168.193.1&sequencenumber=1&request_id=start&executiontime=12264550
47
```

未在 *x-wtg-info* HTTP 头参数值中定义的任何属性都具有提供的默认值，如下所示：

- *group*=unknownGroup
- *name*=unknownScript
- *ipaddress*=0.0.0.0
- *request_id*=Action

如果未定义 *introscope.agent.synthetic.header.names*，则忽略以下配置参数。

`introscope.agent.synthetic.node.name=合成用户`

报告识别的综合事务下的节点。此节点位于 *Frontends*|*Apps*|*<WebAppName>* 下，其中 *<WebAppName>* 是 Web 应用程序名称。此值默认为合成用户。

`introscope.agent.non.synthetic.node.name=实际用户`

报告未识别的综合事务下的节点。此节点位于 *Frontends*|*Apps*|*<WebAppName>* 下，其中 *<WebAppName>* 是 Web 应用程序名称。如果未定义，则没有在 *<WebAppName>* 下创建其他节点。

```
introscope.agent.synthetic.user.name=Synthetic_Trace_By_Vuser
```

其值可用作合成用户名的 HTTP 头参数的名称。合成用户名可用于分隔不同的综合事务。每个合成用户名的节点已创建在 *“合成用户”* 节点之下。如果定义了此配置参数且此名称的 HTTP 标头参数存在，则报告综合事务度量标准。报告事务的节点为 *<Synthetic Users>*/*<Synthetic User>*，其中：

- *introscope.agent.synthetic.node.name* 配置参数将确定 *<Synthetic Users>* 节点名称。
- HTTP 头参数值将确定 *<Synthetic User>* 节点名称。

注意：对这些属性所做的更改将即时生效，无需重新启动托管应用程序。

使用 TagScript 实用工具

CA TagScript 实用工具可以与 HP Vugen 一起使用以指定合成用户信息的提取。

使用 TagScript 实用工具

1. 打开 TagScript 实用工具。

   对于 Windows：

   ```
   <Agent_Home>\wily\tools\TagScript.bat
   ```

   对于 UNIX：

   ```
   <Agent_Home>/wily/tools/TagScript.sh
   ```

   系统会询问您想修改哪个环境的脚本。

2. 单击下列选项之一：
   - 性能测试－适用于 HP Loadrunner 脚本
   - 产品－适用于 HP 业务流程监视器或 Sitescope 脚本
   - 取消标记－反转标记进程
3. 导航到您的 HP Vugen 脚本所在的目录。双击每个 .c 脚本可打开它们。

   HP Vugen 脚本 .c 文件已被全部备份且被修改版替换。
4. 如果 HP Vugen 处于打开状态且执行了实用工具，系统会提示您重新加载修改的脚本。当出现提示时，单击“全是”。
5. 您可以关闭 TagScript 实用工具，或单击“文件选择”对话框的“取消”按钮。您不需要关闭 TagScript 实用工具，许多用户使用 HP Vugen 时不会关闭实用工具。如果已修改脚本，或创建了任何新的脚本，则不关闭实用工具将简化流程。

6. 确认已在以下位置标记您的脚本：
   - HP Vugen 代码的新段落已插入每个脚本的开头。
   - 标记在所有 lr_start_transaction、lr_end_transaction 之前出现，并会在脚本的结尾处出现。
7. （可选）您可以使用一组 blame 堆栈跟踪 HP Loadrunner 性能测试的每个虚拟用户。要跟踪每个用户，请在声明段落的脚本的开头取消注释以下行：

   ```
   web_add_auto_header("Synthetic_Trace_By_Vuser",vuserOverview)
   ```

   注意：如果针对标记脚本的产品取消注释此选项，则将为每个存在点或合成生成器创建一组 blame 堆栈。

第 4 章： 自定义默认数据收集

安装代理时会附带安装几个默认的 ProbeBuilder 指令 (*.pbd*) 文件，并启用对 .NET Framework 的许多常见组件和 ASP.NET 应用程序的监控。本节介绍了默认情况下提供的监控，以及如何无需编写自定义 *.pbd* 文件即可修改默认监控。

此部分包含以下主题：

关于默认 ProbeBuilder 文件

ProbeBuilder 指令 (*.pbd*) 文件指定要插入的探测器类型以及探测器应放置在代码中的位置。探测器（如计时器和计数器）控制代理向 Introscope 企业管理器报告的度量标准。为使代理能够直接收集度量标准而无需自定义，Introscope 包含了一组默认的预定义 ProbeBuilder 文件：

- *ProbeBuilder 指令 (PBD) 文件*，包含用于为应用程序插入探测器并检索度量标准的特定说明。
- *ProbeBuilder 列表 (PBL) 文件*，列出要一起部署的特定 PBD 文件，以监控所有组件或组件的一个子集。

注意：所有度量标准都使用系统时钟所设置的时间来计算。如果在事务期间重置了系统时钟，则针对该事务报告的用时可能会具有误导性。

默认的 ProbeBuilder 文件支持对 .NET 环境最常见组件的监控。您可以微调默认文件组的设置，自定义常用组件监控以适合您的环境。

默认 PBD 跟踪的组件

默认 Introscope PBD 文件会跟踪以下 .NET 组件：

- .NET Directory Services
- .NET Messaging
- .NET Remoting
- ADO.NET
- ASP.NET
- Enterprise Services
- Network Sockets
- SMTP Mail
- Web Services

默认 ProbeBuilder 指令 (PBD) 文件

下表介绍了随 .NET 代理一起安装的默认 PBD 文件：

PBD 文件名	说明
appmap.pbd	此文件为应用程序分类地图检测提供跟踪器指令。
appmap-soa.pbd	该文件为 .NET Framework 类库或 SOAP 堆栈的应用程序分类视图提供跟踪器指令，具体取决于您是否在安装过程中启用了 CA APM for SOA。
appmap-soa.spm.pbd	该文件为 .NET Framework 中受支持的 SOAP 堆栈的应用程序分类视图提供跟踪器指令。 仅当在安装过程中未启用 CA APM for SOA 时才会安装该文件。如果在安装时启用了 CA APM for SOA，该文件会重命名为 *appmap-soa.pbd*，且默认代理 *appmap-soa.pbd* 会重命名为 *appmap-soa.core.pbd*。
bizrecording.pbd	此文件提供用于设置代理业务记录的跟踪器定义和指令。
biz-trx-http.pbd	此文件为以业务为中心的 HTTP 检测提供跟踪器指令。
dotnet.pbd	此文件提供用于支持 .NET Framework 类库的指令。
errors.pbd	此文件通过指定哪些代码级别的事件构成严重错误来配置 ErrorDetector。默认情况下，仅将前端和后端错误视为严重错误。也就是说，只有明显的面向用户的错误页面或表明后端系统（ADO.NET、Messaging 等）出现问题的错误是严重错误。
httpheaderdecorator.pbd	此文件用于启用 HTTP Header Decorator，它是 CA CEM 集成解决方案的一部分。

leakhunter.pbd	此文件为 Introscope LeakHunter 提供检测设置。通常，不需要修改此文件的内容。
sharepoint-full.pbd	此文件提供打开/关闭开关（以 *TurnOn* 指令的形式），用于跟踪 Microsoft SharePoint 数据。大多数跟踪器组都已打开。
sharepoint-typical.pbd	此文件提供打开/关闭开关（以 *TurnOn* 指令的形式），用于跟踪 Microsoft SharePoint 数据。只有一小部分跟踪器组已打开。
skips.pbd	此文件指示 NativeProfiler 跳过可重入问题的特定程序集、类和方法。
spm-correlation.pbd	此文件提供控制组件间事务跟踪关联的指令。使用 CA APM for SOA 时，需要此文件以支持跨进程事务跟踪。
sqlagent.pbd	此文件是 SQL 代理配置文件。可使用此文件检测 ADO.NET 供应商库 (.dll)。通常不需要编辑此文件。
toggles-full.pbd	此文件提供打开/关闭开关（以 *TurnOn* 指令的形式），用于执行其他指令文件中提供的跟踪。大多数跟踪器组都已打开。
toggles-typical.pbd	此文件提供打开/关闭开关（以 *TurnOn* 指令的形式），用于执行其他指令文件中提供的跟踪。只有一小部分跟踪器组已打开。
webservices.pbd	此文件提供用于支持对 .NET Web 服务进行监控的指令。 此文件的具体内容取决于是否在安装过程中启用了 CA APM for SOA。如果启用，将监控 .NET Web 服务和 SOAP 堆栈。否则，仅监控 .NET Web 服务。

详细信息：

默认跟踪器组和 toggles 文件 (p. 66)
打开或关闭跟踪器组 (p. 67)

先前版本中的默认 PBD 文件

默认情况下，代理使用当前版本中的 PBD 和 PBL 文件。但是，产品在 <*代理主目录*>/wily/examples/legacy 目录中提供了先前版本中的 PBD 和 PBL 文件。该目录中的每个文件名都带有 -legacy 后缀，例如，default-full-legacy.pbl。

默认 ProbeBuilder 列表 (PBL) 文件

每个代理有两组可用的 .pbl 文件：

- **default-full.pbl**：此文件是默认的 .pbl 文件。该文件引用其中大多数跟踪器组已打开的 PBD 文件。默认情况下，Introscope 使用此组文件来演示 Introscope 的完整功能。
- **default-typical.pbl**：在引用的 .pbd 文件中，有一部分跟踪器组已打开。典型组包含通用设置，您可以针对特定环境自定义该组文件。

可在 PBD 文件中找到跟踪器组，并在 PBL 文件中加以引用。它们会报告有关一组类的信息。在 .pbd 文件中，通过术语 "flag" 来表示跟踪器组信息。例如，*TraceOneMethodIfFlagged* 或 *SetFlag* 定义了跟踪器组信息。

默认跟踪器组和 toggles 文件

跟踪器组包括一组跟踪器，可应用于一组类。例如，有些跟踪器组会报告所有系统消息传送类的响应时间和响应率。

您可以通过打开或关闭某些跟踪器组来优化系统上的度量标准收集。这会影响开销使用情况，是增加还是减少则取决于您如何配置跟踪器组。

跟踪器组在 *toggles-full.pbd* 和 *toggles-typical.pbd* 文件中进行修改，并由 *default-full.pbl* 和 *default-typical.pbl* 文件引用。在 *toggles-full.pbd* 文件中，对所有默认跟踪器组启用跟踪，从而支持对所有已发现的 .NET 组件的跟踪。

在 *toggles-typical.pbd* 文件中，对跟踪器组的一个子集禁用跟踪。默认情况下，在 *toggles-typical.pbd* 文件中对以下默认跟踪器组禁用跟踪：

- 网络配置：SocketTracing
- 事务实用工具跟踪：ContextUtilTracing
- 运行时远程处理跟踪：RemotingWebServiceTracing
- 系统 Web Mail 跟踪：WebMailTracing

在大多数情况下，默认的 *toggles-full.pbd* 和 *toggles-typical.pbd* 文件可以不进行编辑直接使用。但是，您可以通过打开或关闭某些跟踪器组来优化度量标准收集。例如，如果您使用的是 *toggles-typical.pbd* 文件，通过找到 *Network Configuration* 部分并启用 *SocketTracing* 组（取消注释 *TurnOn* 语句），可以添加对套接字度量标准的跟踪：

```
TurnOn: SocketTracing
```

同样，通过注释掉 *TurnOn* 语句，可以停止对特定跟踪器组的跟踪，从而节约系统开销。例如，如果您使用的是 *toggles-full.pbd* 文件，但是不想跟踪 SQL 代理操作，通过找到 *SQL Agent Tracing* 部分并注释掉 SQL 代理跟踪器组的 *TurnOn* 语句，可以停止对那些操作进行跟踪：

```
#TurnOn: SQLAgentCommands
#TurnOn: SQLAgentDataReaders
#TurnOn: SQLAgentTransactions
#TurnOn: SQLAgentConnections
```

您也可以通过将类添加到现有的跟踪器组来自定义跟踪。

打开或关闭跟踪器组

您可以通过打开或关闭某些跟踪器组来优化系统上的度量标准收集。

请执行以下步骤：

1. 找到并打开 *toggles-full.pbd* 或 *toggles-typical.pbd*，具体取决于正在使用的文件类型，对应于：*default-full.pbl* 或 *default-typical.pbl*。这些文件可在 *<Agent_Home>* 目录中找到。
2. 找到要打开或关闭的跟踪器组。
3. 通过在行首添加或删除英镑符号 (#) 对该行进行注释或取消注释。下例中的指令表示跟踪器组已打开或已关闭：

   ```
   TurnOn: SocketTracing
   ```

   此跟踪器组已打开。已取消注释该行。

   ```
   #TurnOn: SocketTracing
   ```

   此跟踪器组已关闭。已注释该行。
4. 保存 *toggles-full.pbd* 或 *toggles-typical.pbd*。

   配置已设置。

配置代理连接度量标准

默认情况下，Introscope 会生成有关连接到企业管理器的代理的连接状态的度量标准，您可以对度量标准进行监控。您可以监控代理连接度量标准，以确定代理与企业管理器之间的当前连接状态。

代理连接度量标准显示在企业管理器进程（自定义度量标准主机）下的 Workstation 调查器中：

自定义度量标准主机（虚拟）\ 自定义度量标准进程（虚拟）\ 自定义度量标准代理（虚拟）（*超级域*）\ 代理 \ <主机名> \ <代理进程名称> \ <代理名称> \ 连接状态

连接度量标准具有以下值：

- 0－未提供有关代理的任何数据
- 1－代理已连接
- 2－代理报告迟缓
- 3－代理已断开连接

断开连接的代理也会生成“需要关注”事件。如同其他事件一样，用户可以使用历史查询界面来查询代理断开连接事件。代理断开连接事件是在 Workstation 控制台的“概览”选项卡中用于评估应用程序运行状况的数据的一部分。

代理与企业管理器断开连接后，Introscope 会继续生成断开连接状态度量标准，直到代理超时。代理超时后，将不再生成连接度量标准，也不再向企业管理器报告连接度量标准。

请执行以下步骤：

1. 在安装有企业管理器的计算机上，打开位于 *<Introscope 主目录>/config* 目录中的 *IntroscopeEnterpriseManager.properties* 文件。
2. 修改以下属性：

   introscope.enterprisemanager.agentconnection.metrics.agentTimeoutInMinutes

   时间增量以分钟为单位。
3. 保存 *IntroscopeEnterpriseManager.properties*

注意：有关企业管理器属性的信息，请参阅《*CA APM 配置和管理指南*》。

关闭套接字度量标准

默认情况下，.NET 代理配置为收集各套接字的输入和输出带宽度量标准。但是，跟踪每个套接字带宽的度量标准可能会产生较高的开销。如果收集套接字级别网络度量标准占用了大量的处理器或 I/O 时间，可以完全关闭对这些套接字度量标准的收集。

关闭套接字度量标准报告：

1. 在文本编辑器中打开 *IntroscopeAgent.profile* 文件。
2. 找到“Agent Socket Rate Metrics”部分。
3. 将 *introscope.agent.sockets.reportRateMetrics* 属性设置为 *false*。例如：
   ```
   introscope.agent.sockets.reportRateMetrics=false
   ```
4. 保存并关闭 *IntroscopeAgent.profile*。

配置 .NET 代理日志记录选项

以下部分介绍如何在详细模式下运行 .NET 代理以及如何为代理设置日志文件选项。为实现这些功能，Introscope 的 .NET 代理使用了 *Log4net* 功能。如果要使用其他 *Log4net* 功能，请参阅 Log4net 文档，网址为 http://logging.apache.org/log4net/release/features.html。

以详细模式运行 .NET 代理

以详细模式运行 .NET 代理，可将大量详细信息记录到日志文件中，这在调试时会很有帮助。

以详细模式运行 .NET 代理：

1. 停止 .NET 代理。
2. 打开 *logging.config.xml* 文件。
3. 将 *level value* 属性更改为 **VERBOSE**。默认值为 *INFO*。
   ```
   <root>
   <level value="VERBOSE" />
   <appender-ref ref="logfile" />
   <appender-ref ref="console" />
   </root>
   ```
4. 保存 *logging.config.xml* 文件并重新启动 .NET 代理。

更改 .NET 代理日志文件的位置

默认情况下，日志文件会写入 *<Agent_Home>\logs* 目录，通常是 *C:\Program Files\CA Wily\Introscope<version>\wily\logs*，其中 *<version>* 是所安装的 Introscope 的版本。为了方便使用，您可能希望更改 .NET 代理日志文件的位置。

更改 .NET 代理日志文件的位置：

1. 停止 .NET 代理。
2. 打开 *logging.config.xml* 文件。
3. 将 **file value** 属性更改为日志文件的目标位置，例如：
   `<file value="c:\introscope_logs\IntroscopeAgent.log" />`
4. 保存 *logging.config.xml* 文件。
5. 重新启动 .NET 代理。

如果已经为代理配置了名称，则已命名的代理日志会写入 *logs* 目录（默认位置或者您刚刚指定的新位置）。

.NET 代理日志文件和自动代理命名

默认情况下，.NET 代理会自动获取名称。自动找到 .NET 代理名称后，还将使用相同的信息自动命名与该代理关联的日志文件。代理生成的日志文件记录有关所使用的 PBD 以及在检测过程中插入的探测器的信息。默认情况下，如果使用自动命名，创建的日志文件最初使用文件名中的时间戳进行命名。例如：

```
AutoProbe20060928-175024.log
```

一旦代理名称可用，日志文件将重命名以包含代理名称。例如，如果代理名称是 *MyDomain//MyAgent*，其中 *MyDomain* 是域，*MyAgent* 是实例：

```
AutoProbeMyDomain_MyStuff.log
```

如果您的日志文件的名称是时间戳而不是日志的实际名称，则在获取代理名称之前进程可能已超时。此外，如果使用了 Log4Net 高级功能，自动命名功能可能无法正常工作。

注意：在加载某个类路径上的资源中的 .NET 代理配置文件时，NativeProfiler 无法写入日志文件，因为 *IntroscopeAgent.profile* 文件位于资源内部。

如果要禁用日志文件自动命名，请在代理配置文件中将 introscope.agent.disableLogFileAutoNaming (p. 146) 属性设置为 true。

默认域日志

默认域不会连接到企业管理器以报告度量标准，也不会自行运行任何应用程序。但是，由于 .NET 代理会处理默认域承载的所有应用程序域的所有字节代码检测，因此驻留在默认域上的 .NET 代理仍会生成日志文件。其中一个文件 *AutoProbe.DefaultDomain.log* 包含有关默认域中的字节代码检测的信息。由于所有字节码检测都是在默认域中进行的，因此这些日志文件包含与检测有关的重要信息。

默认域还会为 .NET 代理生成 *IntroscopeAgent.DefaultDomain.log* 文件。

第 5 章： 使用 ProbeBuilder 指令

默认情况下，代理可提供针对 .NET 应用程序的许多组件进行监控。但是，要充分利用 Introscope，通常至少要对特定于您的应用程序的一些类和方法进行检测。本节介绍了如何使用 ProbeBuilder 指令关键字以及如何创建自定义 PBD 文件。

重要信息！ PBD 和 PBL 仅支持 ASCII 字符。在 PBD 或 PBL 中放置其他字符（如 Unicode 字符）可能导致出现问题。

此部分包含以下主题：

向现有跟踪器组中添加类

您可以通过将特定类添加到现有跟踪器组来打开对该类的跟踪。要将某个类标识为跟踪器组的一部分，请使用其中一个标识关键字。

例如，将类 *System.EnterpriseServices.ServicedComponent* 添加到跟踪器组 *ServicedComponentTracing* 中：

```
IdentifyClassAs:
System.EnterpriseServices.ServicedComponent ServicedComponentTracing
```

创建自定义跟踪器

您可以通过创建自定义 .pbd 文件来进一步优化度量标准收集。创建自定义指令（通过创建跟踪器来跟踪特定于应用程序的量度）需要使用特定的语法和关键字。

要编写自定义跟踪器，您必须定义：

- 指令类型（通常指示要跟踪的类或方法的数量）。
- 要跟踪的特定类或方法。

- 类或方法中要跟踪的信息类型（例如，时间、比率或计数）。
- 要在其下显示此信息的完全限定度量标准名称（包括资源路径）。

创建了自定义 .pbd 后，Introscope 会立即将其视为即开即用的 .pbd。您可以在创建的度量标准中设置报警，将报警保存到 SmartStor，或者在 Introscope 工作站中创建自定义控制板时使用报警。

注意：请务必谨慎选择要跟踪的方法，因为跟踪的方法越多，意味着系统的开销越大。

各节主题

常见自定义跟踪器示例

BlamePointTracer 是最常用的跟踪器。此跟踪器为关联的方法或类生成五个单独的度量标准：

- 平均响应时间 (毫秒)
- 并发调用
- 每个时间间隔的错误
- 每个时间间隔的响应数
- 停顿计数

下面是 *BlamePointTracer* 的一个示例。已为类 "petshop.catalog.Catalog" 中名为 "search" 的方法设置了 *BlamePointTracer*。"MSPetShop|Catalog|search" 是 Introscope 调查器中节点的名称，BlamePoint 度量标准将显示在该节点下：

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamePointTracer
"MSPetShop|Catalog|search"
```

跟踪器语法

除用于将跟踪器关联到组或启用/禁用组的简单关键字外，PBD 文件还包含跟踪器定义。要使 Introscope 能够识别并处理跟踪器，必须在构建自定义跟踪器时使用特定的语法。跟踪器由指令和有关要跟踪的方法或类的信息组成，格式如下：

```
<directive>: [arguments]
```

其中，*[arguments]* 特定于列表和指令。跟踪指令中使用的参数包括 *<Tracer-Group>*、*<class>*、*<method>*、*<Tracer-name>* 和 *<metric-name>*。

注意：根据所使用的指令，仅需要这些参数中的一部分。

<directive>

有六个主要指令可用于自定义跟踪：

- **TraceOneMethodOfClass**－跟踪指定类中的指定方法。
- **TraceAllMethodsOfClass**－跟踪指定类中的所有方法。
- **TraceOneMethodIfInherits**－跟踪指定类的所有直接子类或指定接口的所有直接接口实施中的一个方法。
- **TraceAllMethodsIfInherits**－跟踪指定类的所有直接子类或指定接口的所有直接接口实施中的所有方法。

注意：只能跟踪已实施的具体方法，只有此类方法会在运行时报告度量标准数据。在自定义跟踪器中指定抽象的方法将导致无法报告度量标准数据。

- **TraceOneMethodIfFlagged**－如果指定的类包含在已使用 *TurnOn* 关键字启用的跟踪器组中，则跟踪一个方法。
- **TraceAllMethodsIfFlagged**－如果指定的类包含在已使用 *TurnOn* 关键字启用的跟踪器组中，则跟踪所有方法。

<Tracer-Group>

与跟踪器相关联的组。

<class>

要跟踪的完全限定类或接口名称。完全限定类包括类的完整程序集名称以及类名称，例如：

```
[MyAssembly]com.mycompany.myassembly.MyClass
```

注意：程序集名称必须括在方括号 [] 内。

<method>

- 方法名称（例如 *MyMethod*）

或者

- 包含返回类型和参数的完整方法签名（例如 *myMethod;[mscorlib]System.Void([mscorlib]System.Int32*）。有关方法签名的更多信息，请参阅区分签名 (p. 79)。

<Tracer-name>

指定要使用的跟踪器类型。例如，*BlamePointTracer*。请参阅跟踪器名称 (p. 76)了解跟踪器名称以及跟踪内容。

<metric-name>

控制所收集的数据如何在 Introscope 工作站中显示。

以下示例介绍了三种在度量标准树中的不同级别上指定度量标准名称和位置的方法。

- **metric-name**—度量标准直接显示在代理节点内。
- **resource:metric-name**—度量标准显示在代理节点下的一个资源（文件夹）内。
- **resource|sub-resource|sub-sub-resource:metric-name**—度量标准显示在代理节点下的多个资源（文件夹）级别中。使用管道字符（|）分隔资源。

跟踪器名称

下面介绍了跟踪器名称以及跟踪内容。

BlamePointTracer

提供标准度量标准组，包括平均响应时间、每个时间间隔的计数、并发、停顿和问题组件的错误。

ConcurrentInvocationCounter

报告方法已启动但未完成的次数。将在*调查器*树中的跟踪器 <metric-name> 中指定的度量标准名称下报告结果。此跟踪器的一个使用示例是对同时进行的数据库查询数目进行计数。

DumpStackTraceTracer

此跟踪器针对堆栈跟踪所应用到的方法，将堆栈跟踪转储到被检测应用程序的标准错误中。Dump Stack Tracer 所抛出的异常堆栈跟踪不是真正意义上的异常－它是一种输出方法堆栈跟踪的机制。

此功能对于确定方法的调用路径非常有用。

重要信息！此功能会大大增加系统开销。强烈建议仅在诊断上下文（此时容许系统开销急剧增加）中使用此跟踪器。

MethodTimer

平均方法执行时间（毫秒），会在度量标准树中的跟踪器 *<metric-name>* 中指定的度量标准名称下报告该时间。

PerIntervalCounter

每个时间间隔内的调用数。此时间间隔将基于数据使用方（例如，调查器中的“视图”窗格）的查看时间段进行更改。将在调查器树中的跟踪器 *<metric-name>* 中指定的度量标准名称下报告该时间间隔。

自定义方法跟踪器示例

自定义跟踪器可包含带有空格的度量标准名称。在自定义度量标准名称中使用空格时，CA Technologies 建议在所有度量标准名称两侧加上引号 ("")。

重要信息！请不要在类名称两侧加引号。否则会引起自定义跟踪器出现故障。例如：

正确

```
IdentifyClassAs: My.Name.Space.MyClass MyTracers
```

不正确

```
IdentifyClassAs: "My.Name.Space.MyClass" MyTracers
```

如果创建包含类名称的度量标准名称，则必须在整个度量标准名称两侧加上引号。允许在度量标准名称中使用空格，度量标准名称中的所有空格都必须包含在引号内。例如，度量标准名称 "{classname}|Test One Node" 将如下所示：

正确

TraceOneMethodIfFlagged: MyTracers AMethod BlamePointTracer "{classname}|Test One Node"

不正确

TraceOneMethodIfFlagged: MyTracers AMethod BlamePointTracer {classname}|Test One Node

以下部分是方法跟踪器示例。在以下示例中，在度量标准名称两侧使用了引号 ("")。CA Technologies 强烈建议在创建自定义度量标准名称时在所有度量标准名称两侧加上引号。

平均值跟踪器示例

此跟踪器将跟踪指定方法的平均执行时间（毫秒）。

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamedMethodTimer
"MSPetShop|Catalog|search:Average Method Invocation Time (ms)"
```

比率跟踪器示例

此跟踪器将对每秒内的方法调用次数进行计数，并在指定的度量标准名称下报告此比率。

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamedMethodRateTracer
"MSPetShop|Catalog|search:Method Invocations Per Second"
```

每个时间间隔内的计数器跟踪器示例

此方法跟踪器将对每个时间间隔内的方法调用次数进行计数，并在指定的度量标准名称下报告每个时间间隔内的计数。

```
TraceOneMethodOfClass: petshop.catalog.Catalog search PerIntervalCounter
"MSPetShop|Catalog|search:Method Invocations Per Interval"
```

时间间隔由企业管理器中的监控逻辑（例如，图表频率）决定。

Introscope 调查器中的预览窗格默认采用 15 秒的时间间隔。

计数器跟踪器示例

此跟踪器将对调用方法的总次数进行计数。

```
TraceOneMethodOfClass: petshop.cart.ShoppingCart placeOrder
BlamedMethodTraceIncrementor "MSPetShop|ShoppingCart|placeOrder:Total Order
Count"
```

组合计数器跟踪器示例

这些跟踪器将增量跟踪器和减量跟踪器组合起来，以保持计数的连续性。

```
TraceOneMethodOfClass: petshop.account.LoginComponent login
MethodTraceIncrementor "MSPetShop|Account:Logged In Users"
TraceOneMethodOfClass: petshop.account.LogoutComponent logout
MethodTraceDecrementor "MSPetShop |Account:Logged In Users"
```

创建高级自定义跟踪器

以下部分详细介绍了如何创建高级自定义跟踪器（例如，单度量标准跟踪器、跳过以及组合自定义跟踪器）。

各节主题

高级单度量标准跟踪器

指令和跟踪器将跟踪方法、类和类组。单度量标准跟踪器将报告特定方法的特定度量标准，这是 Introscope 可跟踪的最小单位。可通过以下多种方式创建单度量标准跟踪器：通过方法签名、通过替换关键字或者通过使用度量标准名称参数。

区别签名

可基于方法签名将跟踪器应用到方法。

要使用特定签名跟踪某个方法的单个实例，请将签名附加到使用内部方法描述符格式指定的方法名称（包含返回类型）。

例如，*myMethod;[mscorlib]System.Void([mscorlib]System.Int32)* 将跟踪具有 *int* 参数和 *void* 返回类型的方法实例。

基于关键字的度量标准名称替换

基于关键字的替换允许在运行时将值替换为度量标准名称。

会在运行时将跟踪器中的度量标准名称中代表实际值的参数替换为度量标准名称。此功能可与任何指令一起使用。下表列出了参数及其运行时替换项：

参数	运行时替换项
{method}	所跟踪方法的名称
{classname}	所跟踪类的运行时类名称
{namespacename}	所跟踪类的运行时命名空间名称
{namespaceandclassname}	所跟踪类的运行时命名空间和类名称
{assemblyname}	所跟踪程序集的名称。
{fullclassname}	报告完整的类名称（包括程序集名称）。

注意：如果 Introscope 处理某个不具有 *namespace* 的类，它会将 *{namespacename}* 替换为字符串 *“<Unnamed Namespace>”*。

基于关键字的替换：示例 1

如果 .pbd 文件中跟踪器的度量标准名称为：

```
"{namespacename}|{classname}|{method}:Response Time (ms)"
```

且将一个跟踪器应用到了 *myNamespace* 中具有运行时类 *myClass* 的方法 *myMethod*，将得到以下度量标准名称：

```
"myNamespace|myClass|myMethod:Response Time (ms)"
```

基于关键字的替换：示例 2

如果将 .pbd 文件中具有度量标准名称

```
"{namespaceandclassname}|{method}:Response Time (ms)"
```

的跟踪器应用到了示例 1 的同一方法，将得到以下度量标准名称：

```
"myNamespace.myClass|myMethod:Response Time(ms)"
```

注意：在本例中，会在 *命名空间*和 *类*之间使用 .（句点），而不使用第一个示例中的 |（管道符号）。

基于度量标准名称的参数

您可以创建一个用于创建度量标准名称的单方法跟踪器。此度量标准名称基于使用 TraceOneMethodWithParametersofClass 关键字并通过以下格式传递给某一方法的参数：

```
TraceOneMethodWithParametersOfClass: <类> <方法> <跟踪器名称> <度量标准名称>
```

可在度量标准名称中使用参数。在度量标准名称中使用参数值替换占位符字符串。可使用字符串 "{#}" 作为占位符，其中 # 是要替换的参数的索引。索引从零开始计数。可按任何顺序使用任何数目的参数替换项。在替换为度量标准名称之前，所有参数都将转换为字符串。应谨慎使用除字符串之外的对象参数，因为它们是使用 ToString() 方法进行转换的。

重要信息！ 如果您不清楚参数将转换成什么字符串，请不要在度量标准名称中使用该参数。

基于度量标准名称的示例

Web 站点使用名为 *order* 的类，其中方法名称为 *process*。该方法使用以下参数来表示不同类型的顺序：*book* 或 *music*。

您可以创建如下跟踪器：

```
TraceOneMethodWithParametersOfClass: order process;
[mscorlib]System.Void([mscorlib]System.Int32) MethodTimer
"Order|{0}Order:Average Response Time (ms)"
```

此跟踪器将生成如下度量标准：

```
Order
    BookOrder
        平均响应时间（毫秒）
    MusicOrder
        平均响应时间（毫秒）
```

您还可以使用 *TraceOneMethodWithParametersIfInherits* 关键字。

跳过指令

通过使用跳过指令，您可以避免检测特定包、类或方法。跳过指令可让 ProbeBuilder 跳过命名空间、类或程序集。

您也可以通过减少匹配来限制所检测的包、类或方法的数量。这可通过将 *IdentifyAllClassAs* 和/或 *TraceAllMethodsIfFlagged* 指令替换为使用范围较小的指令来实现。

组合自定义跟踪器

您可以使用影响同一度量标准的多个跟踪器，实际上就是将这些跟踪器组合在一起。这是增量器和减量器最常使用的方法。

本示例创建了一个名为 *Logged-in Users* 的度量标准。使用类 *user* 以及方法 *login* 和 *logout* 创建以下跟踪器：

```
TraceOneMethodOfClass user login MethodTraceIncrementor "Logged-in Users"
TraceOneMethodOfClass user logout MethodTraceDecrementor "Logged-in Users"
```

当有人登录时，会增加 *Logged-in Users* 度量标准，当有人注销时，会减少 *Logged-in Users* 度量标准。

有关特定跟踪器的注意事项

以下标识符和跟踪器执行特定于 .NET 环境的操作：

- *IdentifyAnnotatedClassAs*: <attribute-class-name> <Tracer-group>

  将使用指定属性类注释的所有类与指定的跟踪器组相关联。

  某些类可使用属性类进行注释，以向该类提供额外功能。在下例中，将名为 *System.EnterpriseServices.Transaction* 的属性类附加到类 *ServicedComponent*：

  ```
  [Transaction]
  Public class ServicedComponent {
  }
  ```

  在 .pbd 中写入以下内容：

  ```
  IdentifyAnnotatedClassAs: System.EnterpriseServices.Transaction
  MyTracerGroup
  ```

  这将标识所有带有 *[Transaction]* 注释的类，包括 *ServicedComponent*。

注意：使用此标识符时，Introscope .NET 代理不会跟踪继承的属性，但跟踪应用到基类的属性。

- *TraceAnnotatedMethodsIfFlagged*: <Tracer-group> <attribute-class-name> <Tracer-name> <metric-name>

  针对与指定的跟踪器组相关联的类，跟踪所有由指定类进行注释的方法。

显式接口实施

.NET 代理在 .pbd 文件中使用显式接口实施。如果您要跟踪的类的方法与其他类的其他方法具有相同的名称，则必须明确命名该方法以及该方法所属的接口。例如，如果 *InterfaceA* 和 *InterfaceB* 都有一个名为 *MethodX* 的方法，在调用 *InterfaceA* 的 *MethodX* 时，必须同时命名接口和方法：*InterfaceA.MethodX*。

以下是跟踪类的方法（具有显式接口实施）的示例：

```
SetFlag: customInterfaceTracing
TurnOn: customInterfaceTracing
IdentifyInheritedAs: EdgeCaseInterface customInterfaceTracing
TraceOneMethodIfFlagged: customInterfaceTracing EdgeCaseInterface.method2
BlamePointTracer "Interface|{namespaceandclassname}|{method}"
```

检测和继承

Introscope 不会自动检测类层次结构中级别较深的类。例如，假定在类层次结构中，*ClassB* 扩展了 *ClassA*，*ClassC* 扩展了 *ClassB*，如下所示：

```
Interface\ClassA
    ClassB
        ClassC
```

当您检测 *ClassA* 时，也会检测 *ClassB*，因为它明确扩展了 *ClassA*。但是，Introscope *不会*检测 *ClassC*，因为 *ClassC* 未明确扩展 *ClassA*。要检测 *ClassC*，必须明确识别 *ClassC*。

应用 ProbeBuilder 指令

准备好实施 ProbeBuilder 指令文件后，可通过以下三种方法实施新文件：

- 使用 hotdeploy 目录 (p. 84)
- 使用 <Agent_Home>/wily 目录 (p. 84)
- 自定义位置和权限 (p. 85)

使用 hotdeploy 目录

通过 *hotdeploy* 目录，Introscope 管理员可以在不编辑 *IntroscopeAgent.profile* 甚至不重新启动应用程序的情况下，更加快速简便地部署新指令。在使用此功能时需特别谨慎。如果您的自定义 PBD 包含无效语法或者配置为收集过多度量标准，将会更快地受到影响。无效的 PBD 会导致 NativeProfiler 关闭，收集过多度量标准的 PBD 会影响应用程序的性能。为解决此问题，CA Technologies 建议：

- 在将 QA 和性能环境中的所有指令推行到生产环境之前，先进行测试和验证。
- 确保您的服务器环境的变更控制过程已更新，使之反映用于部署 PBD 的新选项。

当新 PBD 放在 hotdeploy 目录下时，.NET 代理会自动部署该新 PBD。但是，只有重新启动应用程序后，新的或更改过的 PBD 才会影响已在运行的类和应用程序。当新 PBD 放在此目录下时，您无需编辑 IntroscopeAgent.profile 以获取新的或更改过的 PBD。

使用 hotdeploy 目录应用 .pbd：

- 将自定义文件或修改过的文件（.pbd 和 .pbl）复制到 *<Agent_Home>\wily\hotdeploy* 目录。

使用 <Agent_Home>/wily 目录

要部署新的或更改的 PBD 和 PBL，您必须将其包括在 *introscope.autoprobe.directivesFile* 属性中，并且将文件放在 *IntroscopeAgent.profile* 文件所在的同一目录中或 *IntroscopeAgent.profile* 文件位置的相对目录中。

如果将文件放在任何其他目录中，设置 *introscope.autoprobe.directivesFile* 属性时必须指定文件的完整路径。

部署新的和更改的 .pbd 和 .pbl 文件：

1. 将自定义文件或修改过的文件（PBD 和 .PBL）复制到 *<Agent_Home>* 目录。
2. 更新 *IntroscopeAgent.profile* 文件中的 *introscope.autoprobe.directivesFile* 属性，使之包含所有新文件的名称，并以逗号分隔。

   例如，将自定义 *petstore.pbd* 文件添加到该属性中：

   ```
   introscope.autoprobe.directivesFile=default-full.pbl,petstore.pbd,hotdeploy
   ```
3. 保存并关闭 *IntroscopeAgent.profile*。
4. 重新启动应用程序或 IIS 服务。

注意：请不要删除为该属性定义的现有 *.pbl* 或 *.pbd* 文件，除非您要禁用受现有 .pbl 或 .pbd 文件控制的监控。

自定义位置和权限

除如上所述使用 *hotdeploy* 目录或 *wily* 目录外，您还可以将 PBD 放在所选择的自定义位置（以上所述的两个目录除外）。

如果将 .pbd 文件放在自定义位置，则必须在 *IntrosocopeAgent.profile* 中指定 .pbd 文件的位置。例如，如果将 *leakhunter.pbd* 放在 *C:* 驱动器上的自定义位置，应按以下方式更新 *introscope.autoprobe.directivesFile* 属性：

```
introscope.autoprobe.directivesFile=default-full.pbl,C:\\sw\\leakhunter.pbd
```

将 .pbd 放在自定义位置时，启动 IIS 进程的用户应该对该自定义位置（在上例中为 *C:\\sw*）具有相应权限。如果启动 IIS 进程的用户对此位置没有权限，将在默认域日志中报告一条错误消息，且自定义位置中的 .pbd 不会生效。

重要信息！ CA Technologies 强烈建议您将 PBD 放在 *hotdeploy* 目录下。

事务跟踪和动态检测

当从 Introscope Workstation 运行事务跟踪时，您可以选择执行动态检测。使用动态检测，您可以选择一种或多种运行时检测方式，而无需创建 PBD 文件或修改代理配置文件。使用动态检测，您可以完成以下操作：

- 查看和检测属于事务组件的一个、多个或所有调用方法。
- 查看对临时检测方法的跟踪。
- 查看收集的关于临时检测方法的度量标准。
- 使临时检测永久。

动态检测是一个占用大量 CPU 的操作。CA Technologies 强烈建议您使用可以最大程度地减少要检测的类的配置。

注意：对 .NET 操作环境的支持受到限制。有关详细信息，请参阅《*CA APM Workstation 用户指南*》中的“事务跟踪”。

详细信息：

动态检测 (p. 172)

第 6 章： 配置 LeakHunter

Introscope LeakHunter 是一个附加组件，旨在通过观测随着时间推移而逐渐增大的集合实例来帮助查找潜在的内存泄漏源－也就是说，观测存储在集合中的对象数是否随着时间的推移而增长。

短期（数分钟或数小时）运行的程序中发生内存泄漏可能不会引起大问题。但是，对于每天运行 24 小时（例如网站）的应用程序来说，很小的内存泄漏也可能很快升级为大问题。

Introscope LeakHunter 旨在通过打开、发现集合类，然后在收集信息之后关闭来跟踪值得注意的内存泄漏的相关信息。这种使用 LeakHunter 的方法只会产生很小的临时开销。

此部分包含以下主题：

LeakHunter 的工作原理

启用 LeakHunter 的同时，还应定义 LeakHunter 寻找新的潜在泄漏的超时时间。然后重新启动托管应用程序。

如果 LeakHunter 发现了随着时间推移而逐渐增长的集合，会执行以下操作：

- 通过唯一的 ID 识别集合
- 将有关集合的信息作为度量标准数据报告给企业管理器
- 将有关集合的信息报告到代理计算机上的日志文件
- 继续跟踪和报告该集合的数据

如果 LeakHunter 发现集合不再泄漏，它会将此事报告给企业管理器和日志文件，但会继续跟踪和报告该集合的数据。

LeakHunter 会继续寻找潜在泄漏并监控已识别的潜在泄漏，直到超时期满。超时期满之后，LeakHunter 会停止在新分配的集合中寻找潜在泄漏，并仅继续检查已识别为潜在泄漏的集合。这大大地减少了 LeakHunter 的开销，可以腾出更多开销来监控其他潜在泄漏。LeakHunter 会继续监控已识别的潜在泄漏，直到关闭托管应用程序。

要查找内存泄漏源，可以浏览 Introscope 调查器中的度量标准数据或查看日志文件。

LeakHunter 在 .NET 中跟踪哪些内容

将跟踪以下 .NET 专用集合：

各节信息

专用集合

- System.Collections.ArrayList
- System.Collections.BitArray
- System.Collections.CollectionBase
- System.Collections.DictionaryBase
- System.Collections.Hashtable
- System.Collections.Queue
- System.Collections.SortedList
- System.Collections.Stack
- System.Collections.Specialized.HybridDictionary
- System.Collections.Specialized.ListDictionary
- System.Collections.Specialized.NameObjectCollectionBase
- System.Collections.Specialized.NameValueCollection
- System.Collections.Specialized.StringCollection
- System.Collections.Specialized.StringDictionary
- System.Collections.Specialized.OrderedDictionary

专用接口

- System.Collections.ICollection
- System.Collections.IDictionary
- System.Collections.IList
- System.Collections.Specialized.IOrderedDictionary

LeakHunter 会跟踪 System.Collections 中 .NET IList、ICollection 和 IDictionary 的实施。

有关 ICollection、IDictionary 和 IList 的信息，请参阅：

- IList 实施：
  http://msdn2.microsoft.com/en-us/library/system.collections.ilist(VS.71).aspx
- ICollection 实施：
  http://msdn2.microsoft.com/en-us/library/system.collections.icollection(VS.71).aspx
- IDictionary 实施：
  http://msdn.microsoft.com/en-us/library/system.collections.idictionary%28VS.71%29.aspx

LeakHunter 还可以跟踪 System.Collections.Specialized 中的 IOrderedDictionary 实例。但是，.NET 框架中没有此类实施，因此 LeakHunter 仅跟踪 IOrderedDictionary 的应用程序实施。

常规集合

- System.Collections.Generic.List
- System.Collections.Generic.SortedList
- System.Collections.Generic.Dictionary
- System.Collections.Generic.SortedDictionary
- System.Collections.Generic.LinkedList
- System.Collections.Generic.Queue
- System.Collections.Generic.Stack
- System.Collections.Generic.SynchronizedKeyedCollection
- System.Collections.Generic.SynchronizedCollection
- System.Collections.Generic.KeyedByTypeCollection

- System.Collections.Generic.HashSet
- System.Collections.ObjectModel.KeyedCollection
- System.Collections.ObjectModel.Collection
- System.Collections.ObjectModel.ObservableCollection

常规接口

- System.Collections.Generic.IList
- System.Collections.Generic.ICollection
- System.Collections.Generic.IDictionary

LeakHunter 不跟踪哪些内容

Introscope LeakHunter 不跟踪：

- 不是由集合引起的泄漏。
- 自定义集合实施或引用数量逐渐增长的其他数据结构。
- 未检测的泄漏集合。

启用和禁用 LeakHunter

LeakHunter 作为代理扩展运行，因此不需要更新任何类路径。默认情况下，LeakHunter 在安装之后不会启用。您必须启用 LeakHunter 以使用其功能。

启用 LeakHunter：

1. 打开代理配置文件 *IntroscopeAgent.profile*。
2. 在“*LeakHunter 配置*”标题下，找到 *introscope.agent.leakhunter.enable* 属性，并输入 *true* 值。
3. 保存代理配置文件。
4. 打开 *IntroscopeAgent.profile* 属性 *introscope.autoprobe.directivesFile* 中列出的 **.typical.pbl* 或 **.full.pbl* 文件。
5. 取消注释 *leakhunter.pbd*。

6. 保存并关闭 **.typical.pbl* 或 **.full.pbl* 文件。
7. 重新启动应用程序。

注意：默认情况下，可从 *<代理主目录>\ext* 目录查找并引用 LeakHunter 等代理扩展。但是，您可以在代理配置文件中更改代理扩展目录的位置。如果您更改了扩展目录的位置，请务必同时移动扩展目录的内容。

禁用 LeakHunter：

1. 打开代理配置文件 *IntroscopeAgent.profile*。
2. 在“*LeakHunter 配置*”标题下，找到 *introscope.agent.leakhunter.enable* 属性，并输入 *false* 值。
3. 保存代理配置文件。
4. 重新启动应用程序。

配置 LeakHunter 属性

LeakHunter 配置属性位于 *<Agent_Home>/wily* 目录下的代理配置文件 *IntroscopeAgent.profile* 中。

配置 LeakHunter：

1. 打开代理配置文件 *IntroscopeAgent.profile*。
2. 根据需要配置以下 LeakHunter 属性：
   - introscope.agent.leakhunter.logfile.location
   - introscope.agent.leakhunter.logfile.append
   - introscope.agent.leakhunter.leakSensitivity
   - introscope.agent.leakhunter.timeoutInMinutes
   - introscope.agent.leakhunter.collectAllocationStackTraces
   - introscope.agent.leakhunter.ignore.0

   有关 LeakHunter 属性的更多信息，请参阅 LeakHunter 属性 (p. 177)。

   注意：*IntroscopeAgent.profile* 包含用于控制 LeakHunter 忽略哪些软件包的默认属性。在大多数情况下，这些属性配置为忽略会导致性能下降的收集。这些属性在默认情况下处于启用状态。如果注释掉这些属性，代理日志中将报告异常。

忽略会导致性能下降的集合

如果集合的 size() 方法的执行时间与集合中的对象数目成正比，则此类集合会降低性能。换言之，如果集合的 size() 方法需要的执行时间越来越长（例如，对于要获取列表大小的、实施不当的 LinkedList，我们需要遍历列表的每个元素并进行计数），这将对应用程序的性能造成负面影响。

应使用 IntroscopeAgent.profile 中的忽略属性来忽略此类收集。

运行 LeakHunter

LeakHunter 作为代理扩展运行。

运行 LeakHunter：

1. 在文本编辑器中打开 *IntroscopeAgent.profile*。
2. 将 *introscope.agent.leakhunter.enable* 属性设置为 true，并根据需要设置其他 LeakHunter 属性。
3. 重新启动应用程序。

使用集合 ID 标识潜在泄漏

LeakHunter 使用唯一的集合 ID 来标识每个潜在泄漏，该 ID 可用于将调查器树中的度量标准数据与日志文件中的数据关联。集合 ID 也在各应用程序之间提供稳定的名称。

集合 ID 格式如下：

```
<method>-<4 digit hash code>#<unique number>
```

- <method>：分配集合的方法名称
- <4 digit hash code>：包含方法的类全名的哈希代码
- #<unique number>：附加到具有相同方法和哈希代码的潜在泄漏的一个数字，用于在代理运行过程中确保使用唯一的集合 ID。

下面是一些集合 ID 示例：

```
theLookupTable-6314#1
getLoginID-1234#1
getLoginID-1234#2
getLoginID-1234#3
verifyCart-5678#1
verifyCart-0012#1
```

LeakHunter 日志文件

LeakHunter 日志文件包含 Introscope LeakHunter 在托管应用程序中识别的潜在泄漏的相关信息；每个条目包含与一个潜在泄漏有关的信息。LeakHunter 日志文件包括有关四个方案的条目：

- 当首次识别到潜在泄漏时－请参阅首次识别潜在泄漏日志条目 (p. 93)
- 当已识别的泄漏已停止泄漏时－请参阅已识别的潜在泄漏停止泄漏日志条目 (p. 94)
- 当之前识别的泄漏再次开始泄漏时－请参阅已识别的潜在泄漏再次泄漏日志条目 (p. 94)
- 当发生 LeakHunter 超时时－请参阅 LeakHunter 超时日志条目 (p. 95)

首次识别潜在泄漏日志条目

此类型的 LeakHunter 日志条目包含有关某个潜在泄漏首次被识别时的信息：

- 当前时间戳（写入日志的时间）
- 集合 ID
- 集合的类
- 集合的分配方法
- 收集的分配时间
- 集合的分配堆栈跟踪
- 收集所分配到的字段名称
- 集合的当前大小

注意：LeakHunter 日志文件中记录的已泄漏收集的当前大小不会动态更新。日志文件标识了首次识别泄漏时的泄漏大小。要查看有关已泄露连接大小的最新信息，请单击 Introscope 工作站中的“泄漏”选项卡。

.NET 日志文件示例：首次识别潜在泄漏时

本示例假定您使用 AutoProbe 作为检测方式。

```
10/18/2007 11:54:47 AM
Potential leak identified
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObject, munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
```

```
Allocation Method:
[munger]com.wily.tools.munger.MungerFactory.createNewInstance[mscorlib,
Version=2.0.0.0,
PublicKeyToken=b77a5c561934e089]System.Object([mscorlib]System.String)
Allocation Timestamp: 10/18/2007 11:54:47 AM
Allocation Stack Trace:
Field Name(s):
    未知
Current Size: 10500
```

已识别的潜在泄漏停止泄漏日志条目

此类型的 LeakHunter 日志条目包含有关已停止泄漏的潜在泄漏的信息：

- 当前时间戳（写入日志的时间）
- 集合 ID
- 集合的类
- 集合的当前大小

.NET 日志文件示例：潜在泄漏已停止

```
10/18/2007 11:55:47 AM
Potential leak no longer appears to be leaking
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObjec
t, munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
Current Size: 28000
```

已识别的潜在泄漏又开始泄漏日志条目

此类型的条目包含有关再次开始泄漏的潜在泄漏的信息：

- 当前时间戳（写入日志的时间）
- 集合 ID
- 集合的类
- 集合的当前大小

.NET 日志文件示例：潜在泄漏再次开始泄漏

```
10/18/2007 11:57:47 AM
Potential leak appears to be leaking again
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObjec
t, munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
Current Size: 35000
```

LeakHunter 超时日志条目

这种类型的 LeakHunter 日志条目包含将继续跟踪的潜在泄漏数目。

日志文件示例：发生超时时

```
LeakHunter timeout occurred at 4/27/04 1:32:12 PM PDT
LeakHunter will only continue to track the 3 potential leaks
```

使用 LeakHunter

有关如何使用 LeakHunter 的详细信息，请参阅《*CA APM Workstation 用户指南*》。

第 7 章： 配置 ErrorDetector

此部分包含以下主题：

ErrorDetector 概览

使用 Introscope ErrorDetector，应用程序支持人员可以检测和诊断严重错误的原因，这些错误会防止用户完成 Web 事务。此类应用程序可用性问题通常会导致向用户发出错误消息，如 "404 Not Found" 等，但是错误消息缺少 IT 人员确定问题根本原因所需的具体信息。使用 Introscope ErrorDetector，当应用程序运行中发生这些严重错误时，您可以监控错误、确定错误的频率和本质，并将有关根本原因的具体信息发送给开发者。

ErrorDetector 是唯一可通过启用严重应用程序错误的根本原因分析，来帮助确保卓越用户体验并提高事务完整性的应用程序管理解决方案。

通过 Introscope ErrorDetector，IT 团队可以：

- 确定异常事务的频率
- 确定所记录的异常是否影响用户
- 查看错误在事务路径中发生的精确位置
- 获取再现、诊断和消除严重错误所需的信息

Introscope ErrorDetector 集成到了 Introscope 中，使您可以监控 Introscope 工作站中的错误。当确实发生应用程序错误时，您也可以使用实时错误查看器来检查每个错误的详细信息。

各节信息

错误类型

CA Technologies 基于 NET 规范中包含的信息定义了一组条件，用来描述“严重”错误。ErrorDetector 将错误和异常均视为错误。最常见的错误类型是抛出的异常。

常见错误包括：

- HTTP 错误（404、500 等）

  有时，HTTP 404 错误源自 Web 服务器，而不是应用程序服务器。如果发生此错误，ErrorDetector 将不会通过代理检测到 Web 服务器错误。

- SQL 语句错误
- 网络连接错误（超时错误）
- 后端错误（例如，无法通过 JMS 发送消息、无法将消息写入消息队列）

CA Technologies 所认为的重要错误可能与您认为的重要错误不同。如果您认为 ErrorDetector 跟踪的某些错误不重要，可以选择忽略。如果想要跟踪其他错误，可以使用错误跟踪器来创建新指令来跟踪这些错误。

ErrorDetector 的工作原理

Introscope 在安装代理时附带了一个名为 *errors.pbd* 的 ProbeBuilder 指令 (PBD) 文件。此 PBD 中的跟踪器可捕获严重错误。

安装代理时会自动安装 ErrorDetector。安装 ErrorDetector 之后，将 Introscope 配置为使用 *errors.pbd*，并启用 ErrorDetector 功能。

Introscope 代理根据 *errors.pbd* 文件中的定义收集错误信息。

在工作站中，您可以查看：

- 调查器中的错误度量标准数据
- 实时错误查看器中的实时错误
- 新错误快照中的错误详细信息，其中显示了有关错误如何发生的组件级别信息

ErrorDetector 集成了事务跟踪器，使您能够确切地了解事务路径上下文中为什么会发生严重错误以及这些错误是如何发生的。另外，所有错误和事务都一直存储在 CA Wily 的事务事件数据库中，您可以通过分析历史数据发现其中的趋势。

Introscope 将事务定义为服务的调用和处理。在 Web 应用程序上下文中，事务是发送自 Web 浏览器的 URL 的调用和处理。在 Web 服务上下文中，事务是 SOAP 消息的调用和处理。

在 .NET 代理中启用 ErrorDetector

默认情况下，会启用 .NET 代理来捕获错误数据。您可以通过在 *IntroscopeAgent.profile* 中将属性 *introscope.agent.errorsnapshots.enable* 设置为 *false* 来禁用错误数据捕获。默认情况下，*errors.pbd* 文件会在 .NET 代理的 *default-full.pbl* 文件和 *default-typical.pbl* 中列出，不需要进一步配置。

配置 ErrorDetector 选项

您可以将代理配置为忽略不想跟踪的错误。您指定用来标记错误的信息可以是精确的错误消息，也可以是消息的任何部分加上星号通配符。

忽略特定错误（可选）：

1. 打开代理配置文件 *IntroscopeAgent.profile*。
2. 对于 *introscope.agent.errorsnapshots.ignore* 属性，将值定义为任何您认为会识别该错误类型的信息。

   例如，以下忽略属性会忽略在它内部找到的任何带有短语 "IOException" 的错误：

   ```
   introscope.agent.errorsnapshots.ignore.0=*IOException*
   ```
3. 要忽略其他错误，请按顺序添加其他忽略属性。例如，要忽略两种错误类型，属性应如下所示：

   ```
   introscope.agent.errorsnapshots.ignore.0=*IOException*
   introscope.agent.errorsnapshots.ignore.1=*HTTP Error Code *500*
   ```
4. 保存对代理配置文件所做的更改。

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

高级错误数据捕获

虽然 ErrorDetector 在默认情况下可捕获许多常见的错误类型，但是 CA Technologies 还提供了一些选项，可自定义错误检测机制以符合您的需求。

您可以使用以下与错误相关的跟踪器创建 ProbeBuilder 指令 (PBD) 来捕获错误。

- **ExceptionErrorReporter** 可报告标准异常。
- **ThisErrorReporter** 将当前对象报告为错误。
- **HTTPErrorCodeReporter** 可捕获 HTTP 错误代码和关联的错误消息。
- **MethodCalledErrorReporter** 报告是否调用了特定的方法。

您应将使用这些跟踪器创建的任何新指令放置在 *<代理主目录>*/wily 目录下的 *errors.pbd* 文件中。

定义新错误类型

可使用 PBD 为 ErrorDetector 定义错误。还有几个特殊的跟踪器，用于确认错误是否存在并捕获（或构建）错误消息。通过将这些跟踪器放在正确的位置，可以让 ErrorDetector 了解应用程序或基础架构中的新错误。

ExceptionErrorReporter

ExceptionErrorReporter 跟踪器可用于检查从已检测的方法抛出的异常。如果抛出了异常，此跟踪器会将异常当作错误，并从异常中获取错误消息。这是对错误的最常见定义。

为了捕获错误消息，ExceptionErrorReporter 跟踪器必须与 "...WithParameters" 指令配合使用。例如：

TraceOneMethodWithParametersOfClass: com.bank.CustomerAccount getBalance ExceptionErrorReporter "CustomerAccount:Errors Per Interval"

此指令指定从 *CustomerAccount* 上的 *getBalance()* 方法抛出的任何异常均视为错误。

请注意，您必须使用“...WithParameters”指令增加“每个时间间隔的错误”度量标准，但对于任一方法，您只需指定“...WithParameters”指令一次，该方法上的参数即可对所有追踪器可用。例如，您可以指定：

TraceOneMethodWithParametersOfClass: com.myClass myMethod BlamePointTracer

此指令使 *com.myClass myMethod* 方法的参数可用于其他追踪器，包括 *ExceptionErrorReporter* 追踪器。

MethodCalledErrorReporter

MethodCalledErrorReporter 跟踪器作用于方法，调用方法即表示已发生错误。例如：

TraceOneMethodOfClass: com.bank.CheckingAccount cancelCheck MethodCalledErrorReporter "CustomerAccount:Canceled Checks Per Interval"

此指令指定只要调用了 cancelCheck() 方法（不管出于任何原因），就意味着发生了错误。错误消息仅指出了所调用的类和方法。

如果您不知道哪些方法会抛出异常或错误，请尝试使用 ThisErrorReporter 跟踪器。

ThisErrorReporter

ThisErrorReporter 跟踪器与 *MethodCalledErrorReporter* 类似，但是它通过对已检测的对象调用 *toString()* 来构建错误消息。这对于指出异常类的构建器非常有用。例如：

```
TraceOneMethodWithParametersOfClass:
ezfids.util.exception.EasyFidsException .ctor ThisErrorReporter
"Exceptions|{packageandclassname}:Errors Per Interval"
```

注意：为了捕获错误消息，*ThisErrorReporter* 跟踪器必须与“...WithParameters”指令配合使用。

指令指定只要调用了 *InvalidPINException* 的构建器（“init”或“.ctor”），就构成错误。错误消息是通过对 *InvalidPINException* 调用 *toString()* 来确定的，InvalidPINException 通常会返回应用程序开发者指定的错误消息。

如果您有一个基于自己的异常类型的自定义错误管理系统，此跟踪器将非常有用。

HTTPErrorCodeReporter

HTTPErrorCodeTracer 跟踪器可报告 ASP.NET 页面中的错误代码。这是一个按时间间隔的计数器，用于计算以下事件的数目：

- HTTP 响应代码 400 或更高代码。
- ProcessRequest 的 *System.Web.IHttpHandler* 子类调用，在 .NET 环境中适用于代码 400 或更高代码

有关使用情况示例，请参阅 *errors.pbd*。

警告

请合理使用前面几节中介绍的跟踪器。最佳做法是使与错误跟踪相关联的开销仅报告最严重的问题，如后端系统中出现的无法恢复的问题。

默认的 *errors.pbd* 旨在报告严重错误，同时最大程度地减少开销。过度使用错误跟踪（例如，将 ExceptionErrorReporter 应用于每个受监控的方法）会导致大量“误报”。例如，如果用户在数字字段中输入“加利福尼亚”，可能导致 *NumberFormatException*，而您并不希望将其报告为严重问题。

使用 ErrorDetector

有关如何使用 ErrorDetector 的详细信息，请参阅《*APM Workstation 指南*》。

第 8 章： 配置 Boundary Blame

此部分包含以下主题：

了解 Boundary Blame

使用 Introscope 的 Blame 技术，您可以查看受管理的 .NET 应用程序中应用程序层上的度量标准：应用程序的前端和后端。此功能称为 Boundary Blame，使用户可以将问题归类到应用程序的前端或后端。

本节介绍了 Introscope 的 Boundary Blame 功能的配置选项。

Introscope 利用 SQL 代理的 SQL 语句监控功能来自动检测后端。如果 SQL 代理不可用，Introscope 会自动将套接字调用检测为后端，如客户端/服务器数据库，或通过套接字访问的 LDAP 服务器。

有关 Boundary Blame 在 Introscope 调查器中如何显示的信息，请参阅《Introscope 工作站用户指南》。

使用 URL 组

URL 组是根据 URL 路径前缀定义的命名事务组。Introscope 会汇集每个 URL 组的度量标准，并在 Introscope 调查器的 *Frontends*|*Apps*|*<ApplicationName>*|*URLs* 节点下显示这些度量标准。

路径前缀是 URL 中与主机名相同的部分。例如，在以下 URL 中：

```
http://burger1.com/TestWar/burgerServlet?viewItem&category=
11776&item=555066263O&rd=1
```

路径前缀为：

```
/TestWar
```

您可以为能够从 URL 路径前缀派生出的、任何有用的请求类别定义 URL 组。例如，根据应用程序 URL 的形式，您可以为应用程序支持的每个客户、每个主要应用程序或者子应用程序定义 URL 组。这使得您可以在所提交的服务级别上下文中监控性能，或者监控应用程序的关键业务部分的性能。

默认情况下，所有 URL 均分配到名为 *Default* 的 URL 组。这可以防止由于无效 URL（例如会导致 404 错误的 URL）不创建唯一的一次性度量标准而产生的开销。通过配置有实际意义的 URL 组，用户可以在子应用程序级别上监控性能。

相关信息

URL 组属性 (p. 104)
示例 URL 组 (p. 104)
定义 URL 组 (p. 104)
URL 组高级命名技巧（可选） (p. 106)

URL 组属性

使用以下属性在 *IntroscopeAgent.profile* 文件中定义 URL 组：

- *introscope.agent.urlgroup.***keys**
- *introscope.agent.urlgroup.***group.default.pathprefix**
- *introscope.agent.urlgroup.***group.default.format**

示例 URL 组

以下列表摘自代理配置文件，该配置文件说明了如何定义 URL 组。列表后面的部分介绍了如何配置必需属性。

```
introscope.agent.urlgroup.keys=alpha,beta,gamma
introscope.agent.urlgroup.group.alpha.pathprefix=/testWarintroscope.agent.urlgroup.group.alpha.format=foo {host} bar {protocol} baz {port} quux {query_param:foo} red {path_substring:2:5} yellow {path_delimited:/:0:1} green{path_delimited:/:1:4} blue {path_substring:0:0}
introscope.agent.urlgroup.group.beta.pathprefix=/nofilterWarintroscope.agent.urlgroup.group.beta.format=nofilter foo {host} bar {protocol} baz {port} quux{query_param:foo} red {path_substring:2:5} yellow{path_delimited:/:0:1} green {path_delimited:/:1:4} blue{path_substring:0:0}
introscope.agent.urlgroup.group.gamma.pathprefix=/examplesWebAppintroscope.agent.urlgroup.group.gamma.format=Examples Web App
```

定义 URL 组

以下部分提供了有关用于配置 URL 组的属性的信息。

各节信息

定义 URL 组的键

使用 *introscope.agent.urlgroup.keys* 属性可定义所有 URL 组的 ID（即键）列表。URL 组的键可在声明 URL 组属性的其他属性定义中引用。本示例定义了三个 URL 组的键：

```
introscope.agent.urlgroup.keys=alpha,beta,gamma
```

为 URL 组指定键时的顺序十分重要。有关更多信息，请参阅 URL 分组 (p. 215)。

定义每个 URL 组的成员资格

通过根据所匹配的 URL 路径前缀来指定模式，使用 *introscope.agent.urlgroup.group.*GroupKey.*pathprefix* 来定义哪些请求属于 URL 组。

示例 1

此属性定义将 URL 路径部分以 */TestWar* 开始的所有请求分配给键为 *alpha* 的 URL 组：

```
introscope.agent.urlgroup.group.alpha.pathprefix=/TestWar
```

与指定的 *pathprefix* 相匹配的请求包括：

```
http://burger1.com/TestWar/burgerServlet?ViewItem&category=11776&item=5550662630&rd=1
http://burger1.com/TestWar/burgerServlet?Command=Order&item=5550662630
```

示例 2

如果公司提供了呼叫中心服务，可通过为每个应用程序功能设置一个 URL 组来监控功能区域的响应时间。如果客户通过以下 URL 访问服务：

```
http://genesystems/us/application_function/
```

其中，*application_function* 对应于应用程序（如 *OrderEntry*、*AcctService* 和 *Support*），每个 URL 组的 *pathprefix* 属性指定相应的 *application_function*。

注意：您可以在 *pathprefix* 属性中使用星号 (*) 作为通配符。

定义 URL 组的名称

使用 *introscope.agent.urlgroup.group.GroupKey.format* 可确定 URL 组的名称，在工作站中，URL 组的响应时间度量标准（其键为 *GroupKey*）会显示这些名称下。

通常情况下，*format* 属性用于分配文本字符串作为 URL 的名称。在本示例中，键为 *alpha* 的 URL 组度量标准以 *Alpha Group* 名称显示在工作站中：

```
introscope.agent.urlgroup.group.alpha.format=Alpha Group
```

URL 组高级命名技巧（可选）

如有需要，您可以从请求元素（如服务器端口、协议）或从请求 URL 的子字符串中派生 URL 组名称。如果通过检查请求能够轻易地区分应用程序模块，这将非常有用。以下部分介绍了 *format* 属性的高级形式。

各节信息

使用主机作为 URL 组名称

要将 URL 组度量标准组织到反映与请求相关联的 HTTP 服务器主机名的名称下，请将 *format* 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format={host}
```

当 *format={host}* 时，以下请求的统计信息将分别显示在 *us.mybank.com* 和 *uk.mybank.com* 度量标准名称下：

```
https://us.mybank.com/mifi/loanApp...
https://uk.mybank.com/mifi/loanApp...
```

使用协议作为 URL 组名称

要将 URL 组统计信息组织到反映与请求相关联的协议的名称下，请将 *format* 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format={protocol}
```

当 *format={protocol}* 时，统计信息在调查器中将分组到请求 URL 的协议部分所对应的度量标准名称下。例如，这些请求的统计信息将显示在 *https* 度量标准名称下：

```
https://us.mybank.com/cgi-bin/mifi/scripts.....
https://uk.mybank.com/cgi-bin/mifi/scripts.....
```

使用端口作为 URL 组名称

要将 URL 组统计信息组织到反映与请求相关联的端口的名称下，请将 *format* 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format={port}
```

当 *format={port}* 时，统计信息将分组到请求 URL 的端口部分所对应的名称下。例如，这些请求的统计信息将显示在 *9001* 名称下。

```
https://us.mybank.com:9001/cgi-bin/mifi/scripts.....
https://uk.mybank.com:9001/cgi-bin/mifi/scripts.....
```

使用参数作为 URL 组名称

要在调查器中将 URL 组统计信息组织到反映与请求相关联的参数值的度量标准名称下，请将 format 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format={query_param:param}
```

当 *format={query_param:param}* 时，统计信息在调查器中将分组到指定参数值所对应的度量标准名称下。没有参数的请求将列在 *<empty>* 下。例如，使用以下参数定义：

```
introscope.agent.urlgroup.group.alpha.format={query_param:category}
```

这些请求的统计信息将显示在度量标准名称 "734" 下。

```
http://ubuy.com/ws/shopping?viewItem&category=734&item=3772&tc=photo
http://ubuy.com/ws/shopping?viewItem&category=734&item=8574&tc=photo
```

使用请求路径子字符串作为 URL 组名称

要将 URL 组统计信息组织到反映请求 URL 路径部分字符串的名称下，请将 *format* 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format={path_substring:m:n}
```

其中，*m* 是第一个字符的索引，*n* 是大于最后一个字符的索引的索引。例如，使用此设置：

```
introscope.agent.urlgroup.group.alpha.format={path_substring:0:3}
```

此请求的统计信息将显示在度量标准节点 *"/ht"* 下

```
http://research.com/htmldocu/webL-12.html
```

使用请求路径中由分隔符分隔的部分作为 URL 组名称

要将 URL 组统计信息组织到反映请求 URL 路径部分字符串的名称下，请将 *format* 参数定义如下：

```
introscope.agent.urlgroup.group.alpha.format=
{path_delimited:delim_char:m:n}
```

其中，*delim_char* 是用于分隔路径中各区段的字符，*m* 是要选择的第一个区段的索引，*n* 是大于要选择的最后区段的索引的索引。例如，使用此设置：

```
introscope.agent.urlgroup.group.alpha.format={path_delimited:/:2:4}
```

以下形式的请求的统计信息：

```
http://www.buyitall.com/userid,sessionid/pageid
```

将显示在度量标准名称 */pageid* 下

请注意：

- 一个分隔字符计为一个区段
- 区段计数从 0 开始

此表显示上例的区段以斜杠字符进行分隔：

区段索引	0	1	2	3
区段字符串	/	*userid*、*sessionid*	/	*pageid*

您可以根据需要指定多个分隔符。例如，使用此设置：

```
introscope.agent.urlgroup.group.alpha.format={path_delimited:/,:3:4}
```

上面所示形式的请求的统计信息将显示在度量标准名称 *sessionid* 下。

此表显示上例的区段以斜杠和逗号字符进行分隔：

区段索引	0	1	2	3	4	5
区段字符串	/	*userid*	,	*sessionid*	/	*pageid*

为 URL 组使用多种命名方法

您可以在单个 *format* 字符串中结合使用多种命名方法，如下所示：

```
introscope.agent.urlgroup.group.alpha.format=red {host} orange {protocol}
yellow {port} green {query_param:foo} blue {path_substring:2:5} indigo
{path_delimited:/:0:1} violet {path_delimited:/:1:4} ultraviolet
{path_substring:0:0} friend computer
```

使用 Blame 跟踪器标记 Blame 点

Introscope Blame 技术可跟踪 .NET 应用程序的性能，从而使您能够在应用程序的前端和后端查看度量标准。此功能称为 Boundary Blame，使用户可以将问题归类到应用程序的前端或后端。

以下部分介绍了如何使用跟踪器在应用程序中明确标记前端和后端。

前端和后端度量标准信息还用于在应用程序分类图中显示有关应用程序的信息。

各节主题

Blame 跟踪器

Introscope 提供了用于捕获前端和后端度量标准的跟踪器：FrontendTracer 和 BackendTracer。这两个跟踪器分别用于明确地标记前端和后端。

您可以使用 FrontendTracer 和 BackendTracer 检测自己的代码（如访问后端的代码），以捕获和显示自定义组件的度量标准。

如果没有配置 FrontendTracer，Blame 堆栈的第一个组件将作为默认前端。如果没有配置 BackendTracer，Introscope 将推断一个后端，也就是说，如果没有明确标记后端，打开客户端套接字的任何组件将作为默认后端。

要防止将特定类标记为前端，可以指定 PBD 参数 **is.frontend.unless**。有关更多信息，请参阅自定义 FrontendMarker 指令 (p. 110)。

如果使用 BackendTracer 为 Introscope 检测为后端的项分配一个所需的名称，或者使用 BackendTracer 标记 Introscope 所不检测的自定义套接字，将会非常有用。

FrontendTracer 和 BackendTracer 是 BlamePointTracer 的实例，BlamePointTracer 提供了平均响应时间、每个时间间隔的计数、并发以及停顿等度量标准。BlamePointTracer 可应用到粒度更高的 Blame 堆栈的“中间”组件。然而，BlamePointTracer 不会在 Introscope 调查器中填充“每个时间间隔的错误”度量标准。

标准 PBD 中的 Blame 跟踪器

随 Introscope 和 .NET 代理提供了两个标准 PBD（dotnet.pbd 和 sqlagent.pbd），可实施 Boundary Blame 跟踪：

- *dotnet.pbd* 中的 *PageInfoTracer* 是 FrontendTracer 的实例。
- *sqlagent.pbd* 中的 *SQLBackendTracer* 是 BackendTracer 的实例。

自定义 FrontendMarker 指令

Introscope 9.0 介绍了 PBD 参数 **is.frontend.unless**。使用此参数，FrontendMarker（或其子类，如 PageInfoTracer）所检测的某些类可不被标记为前端组件。该参数应设置为一个以逗号分隔的绝对类名称列表。如果第一个组件是用于将请求转发给更加具体的组件（专门处理所收到的请求类型）的常规“分发程序”，这将非常有用。因此，第二个组件将是更好的前端标记。默认值为空列表。PBD 参数不是动态的，因此如果此参数的值发生更改，则需要重新启动被检测应用程序服务器。

重要信息！应当仅使用逗号而**不是**空格来分隔类名称。使用空格将导致 SetTracerParameter 指令无效。

在参数列表中指定的、由跟踪器（此参数所应用到的跟踪器）检测的任何类都不会指定为前端，也不会在 Introscope 调查器的“前端”节点中生成度量标准。

例如，要防止使用名为 *PageInfoTracer* 的 FrontendMarker 检测的类 *ASP.index_aspx* 和 *Fib.CalculatorController* 在包 *com.ABCCorp* 中被视为前端，应使用以下 PDB 指令：

```
SetTracerParameter: PageInfoTracer is.frontend.unless
ASP.index_aspx,Fib.CalculatorController
```

第 9 章： 事务跟踪器选项

此部分包含以下主题：

事务跟踪新模式

CA APM Introscope 9.1 引入了新的代理跟踪体系结构和新的跟踪器。此事务跟踪新模式将替换之前的跟踪器在早期版本中使用的事务 blame 堆栈。此新模式实施可优化计算和处理，从而提高代理性能。

CA APM Introscope 9.1 仍允许您使用事务 blame 堆栈配置和运行代理。代理的此“传统”模式受到完全支持，但未来版本中的代理增强功能将仅针对新模式实施。当在“传统”模式下运行代理时，以下功能将不可用：

- 版本 9.1 中的代理 CPU 使用量和响应时间优化
- .NET 代理的动态检测
- SQL 代理属性

  `introscope.agent.sqlagent.sql.artonly`

  `introscope.agent.sqlagent.sql.turnoffmetrics`

重要信息！在新模式下使用 CA APM 代理运行传统跟踪器称为“混合模式”配置。请**不要**在混合模式下运行，因为可能会出现内存消耗和服务中断。

如果运行的是传统模式跟踪器，请在传统模式下运行代理。当客户的传统跟踪器不兼容时，以下消息将写入代理日志中以通知客户。

> “检测到使用传统 API 的代理跟踪器。不建议在代理使用新模式的情况下运行传统的跟踪器。请联系支持人员，他会向您介绍有关如何升级传统跟踪器的文档。临时情况下，请配置代理使用传统模式或使用预配置版本的传统代理包。例如，UNIX 上 Oracle WebLogic 代理的传统软件包为 IntroscopeAgentFiles-Legacy-NoInstallerx.x.x.xweblogic.unix.tar”

恢复到在传统模式下运行代理，直到您运行新模式跟踪器。

如果您使用的是以下开箱即用型扩展，请将您的代理配置为在传统模式下运行：

- CA APM for IBM Websphere Portal
- CA APM for IBM CICS Transaction Gateway
- CA APM for IBM z/OS
- CA APM for CA SiteMinder Web Access Manager
- CA APM Integration for CA LISA

将代理配置为使用传统模式事务跟踪

转换为传统模式事务跟踪仅适用于具有新模式的 CA APM 版本。版本 9.1 之前的 CA APM 版本不具有新模式。

提供了两个选项来将代理配置为使用传统模式：

- 部署预配置的传统代理软件包。可在代理的仅限于文件、无安装程序的软件包中获取这些软件包。例如，UNIX 上 Oracle WebLogic 代理的传统软件包为：

  `IntroscopeAgentFiles-Legacy-NoInstaller<版本号>weblogic.unix.tar`

- 针对传统模式手工配置。

请执行以下步骤：

1. 停止受监控的应用程序。
2. 存档并删除 <Agent_Home>/logs 目录中的现有日志文件，从而为新日志做好准备。
3. 备份 <Agent_Home>/core/config 目录中的现有 .pbl 和 .pbd 文件。
4. 备份现有文件 <Agent_Home>/core/config/IntroscopeAgent.profile。
5. 将旧版的 .pbl 和 .pbd 文件从 <Agent_Home>/examples/legacy 目录复制到 <Agent Home>/core/config 目录。
6. 打开 <Agent_Home>/core/config/IntroscopeAgent.profile 并进行以下更改：
   - 添加属性 introscope.agent.configuration.old=true。
   - 更新代理属性 introscope.autoprobe.directivesFile，以便指向已复制的相应传统 .pbl 和 .pbd 文件。例如，使用 spm-legacy.pbl 替换 spm.pbl。
7. 重新启动受监控的应用程序。

要配置特定 CA APM 扩展以便在典型安装中使用传统模式，请参阅下表中列出的 .pbl 和 .pbd 文件。

扩展名称	旧版的典型 pbl 或 pbd 文件
CA APM for Oracle Weblogic Server	ppweblogic-legacy.pbd spm-legacy.pbl
CA APM for Oracle Weblogic Portal	powerpackforweblogicportal-legacy.pbl spm-legacy.pbl
CA APM for Oracle Service Bus	OSB-typical-legacy.pbl spm-legacy.pbl
CA APM for IBM Websphere Portal	powerpackforwebsphereportal.pbl spm-legacy.pbl
CA APM for IBM Websphere MQ	webspheremq-legacy.pbl
CA APM for IBM Websphere Process Server /Business Process Management	wps-legacy.pbd wesb-legacy.pbd spm-legacy.pbl
CA APM for TIBCO BusinessWorks	tibcobw-typical-legacy.pbl spm-legacy.pbl
CA APM for Software AG Webmethods Integration Server	webmethods-legacy.pbl spm-legacy.pbl
CA APM for CA SiteMinder Web Access Manager	smwebagentext.pbd
CA APM for IBM CICS Transaction Gateway	PPCTGTranTrace.pbd PPCTGServer-typical.pbd PPCTGClient-typical.pbd
CA APM for IBM z/OS	zos-full.pbl
CA APM for CA LISA	lisa-typical.pbl
CA APM for SYSVIEW	CTG_ECI_Tracer_For_SYSVIEW-legacy.pbd HTTP_Tracer_For_SYSVIEW-legacy.pbd WS_Tracer_For_SYSVIEW-legacy.pbd

控制事务跟踪采样

默认情况下，Introscope 代理每小时跟踪一次应用程序中每个标准化的唯一 URL，以提供对事务行为的采样。通过此采样，无需显式运行事务跟踪，即可对可能有问题的事务类型进行历史分析。

默认情况下，事务跟踪采样处于启用状态。您可以通过在代理配置文件 IntroscopeAgent.profile 中取消注释以下属性来禁用该行为：

```
introscope.agent.transactiontracer.sampling.enabled
```

取消注释并设置为 false 可禁用事务跟踪采样。

通过在代理配置文件中取消注释以下属性，您可以配置在每个间隔中对多少个事务进行采样以及该间隔时间为多久。

重要信息！ 这些配置通常是在企业管理器中进行的。在代理配置文件中配置以下属性将覆盖企业管理器中所做的任何配置。

```
introscope.agent.transactiontracer.sampling.perinterval.count
```

取消注释并设置在每个间隔中对多少事务进行采样。默认值为 1。

```
introscope.agent.transactiontracer.sampling.interval.seconds
```

取消注释并设置采样间隔时间（秒）。默认值为 120 秒。

事务跟踪组件限定

Introscope 设置了一个限定（默认设置为 5000 个组件）来限制跟踪的大小。到达此限制时，日志中将显示警告，跟踪将停止。

这样您就可以限定包含大量组件的事务（组件计数超出预期）。例如，当组件执行数百个对象交互和后端 SQL 调用时， 如果没有限定，事务跟踪器会将其视为一个事务而不停地继续。在某些极端的情况下，如果没有限定，CLR 会在跟踪完成之前耗尽内存。

用于限定事务的属性位于 *IntroscopeAgent.profile* 文件中：

```
introscope.agent.transactiontrace.componentCountClamp=5000
```

对于生成限定组件的跟踪（即超过 CountClamp 的跟踪），将使用星号进行标记，并显示与自身相关联的工具提示。有关查看这些跟踪的详细信息，请参阅《*CA APM Workstation 用户指南*》。

如果限定设置得太低，当启动应用程序时，可能会遇到性能监控 (PerfMon) 或 LeakHunter 异常。如果遇到这种情况，必须重新启动受管理的 .NET 应用程序。

事务跟踪器选项

可以将 Introscope 事务跟踪器配置为仅跟踪满足您所指定的条件的事务。您可以按用户 ID 数据或 HTTP 请求和会话属性进行筛选。

重要信息！您可以按用户 ID *或* HTTP 属性进行筛选，但不能同时按两者进行筛选。请不要同时配置这两种类型的筛选－这种配置可能会产生错误的度量标准。

控制跟踪哪些事务：

1. 启用 .NET 代理可报告筛选参数，如启用筛选参数收集 (p. 115)中所述。
2. 配置按用户 ID 或 HTTP 请求数据进行筛选：
   - 按用户 ID 筛选事务跟踪 (p. 116)
   - 按 HTTP 请求数据筛选事务跟踪 (p. 117)。

启用筛选参数收集

默认情况下，.NET 代理仅报告它所跟踪的事务的 URL。将限制对各个 HTTP 属性的报告，以将事务跟踪对系统开销的影响降到最小。要启用筛选，必须首先通过将以下行添加到代理配置文件来启用 HTTP 属性收集：

```
introscope.agent.asp.disableHttpProperties=false
```

这将启用对以下内容的收集：

- 应用程序名称
- 会话 ID
- 上下文路径
- 服务器名称
- URL
- 上下文路径
- 规范化 URL
- HTTP 标头
- HTTP 参数
- HTTP 属性

您可以通过将该行添加到代理配置文件来启用对其他属性的收集：

```
introscope.agent.asp.optionalProperties=true
```

这将启用对以下内容的收集：

- 方案
- URL 参考地址
- 方法

按用户 ID 筛选事务跟踪

要将 .NET 代理配置为在前端组件中按用户 ID 筛选事务跟踪，请确定应用程序如何指定用户 ID。开发应用程序的应用程序架构师可以提供此信息。

Introscope 事务跟踪器可以识别通过以下方法之一访问的用户 ID：

- HTTP 上下文身份
- HTTP 请求标头
- URL 用户信息
- HTTP 会话的一个属性

重要信息！ 请仅执行适用于在应用程序中指定用户 ID 的方法的配置过程。

各节信息

按上下文身份筛选 (p. 116)
按 URL 用户筛选 (p. 116)
按请求标头筛选 (p. 117)
按会话属性筛选 (p. 117)

按上下文身份筛选

如果用户 ID 是通过 HTTP 上下文身份访问的，请在代理配置文件中取消注释此属性：

```
introscope.agent.transactiontracer.userid.method=HttpContext.User.Identity.Name
```

按 URL 用户筛选

如果用户 ID 是通过 URL 用户信息访问的，请在代理配置文件中取消注释此属性：

```
introscope.agent.transactiontracer.userid.method=HttpContext.Request.Uri.UserInfo
```

按请求标头筛选

如果用户 ID 是通过 HTTP 请求标头确定的，请在代理配置文件中取消注释以下属性对，并为第二个属性定义键字符串：

```
introscope.agent.transactiontracer.userid.method=HttpRequest.Headers.Get
introscope.agent.transactiontracer.userid.key=<application defined key string>
```

按会话属性筛选

如果用户 ID 是 HTTP 会话中的一个属性，请在代理配置文件中取消注释以下属性对，并为第二个属性定义键字符串：

```
introscope.agent.transactiontracer.userid.method=HttpContext.Session.Contents
introscope.agent.transactiontracer.userid.key=<application defined key string>
```

按 HTTP 请求数据筛选事务跟踪

可以根据 HTTP 请求属性筛选事务跟踪，包括：

- 请求标头
- 请求参数
- 会话属性

您可以使用多个属性进行筛选，例如，按请求参数和会话属性。

重要信息！ 如果已经配置按用户 ID 进行筛选，请不要配置按 HTTP 属性进行筛选。

请执行以下步骤：

1. 打开 IntroscopeAgent.profile 文件。
2. 在“事务跟踪器配置”标题下找到事务跟踪器属性。
3. 要收集特定的 HTTP 请求标头，请取消注释以下属性，并在以逗号分隔的列表中指定要跟踪的 HTTP 请求标头：
   ```
   introscope.agent.transactiontracer.parameter.httprequest.headers=User-Agent
   ```
4. 要收集 HTTP 请求参数数据，请取消注释此属性，并在以逗号分隔的列表中指定要跟踪的 HTTP 请求参数：
   ```
   introscope.agent.transactiontracer.parameter.httprequest.parameters=parameter1,parameter2
   ```
5. 要收集 HTTP 会话属性数据，请取消注释此属性，并在以逗号分隔的列表中指定要跟踪的 HTTP 会话属性，例如：
   ```
   introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute1,attribute2
   ```

6. 保存对 IntroscopeAgent.profile 文件所做的更改。
7. 重新启动应用程序。

配置组件停顿报告

Application Performance Management 停顿是指在某个定义的时间长度内没有来自已探测组件的响应。默认情况下，APM Introscope 代理可检测出此状况并报告停顿度量标准。

每当代理检查停顿时，会检查方法堆栈上所有最常检测的组件。当组件停顿时，会创建停顿度量标准和停顿快照。还会针对订阅下游监控的每个组件创建停顿度量标准。

首先针对方法堆栈上的最常检测的组件创建停顿度量标准。然后再针对所需时间超过 *introscope.agent.stalls.thresholdseconds* 值的组件创建这些度量标准。

下游订户组件停顿

在调用停顿的下游组件时，订阅下游监控的组件将停顿。

默认情况下，以下组件会订阅下游停顿监控：

- FrontendTracer 探测组件
- BackendTracer 探测组件
- WebServiceBlamepointTracer@Servicelevel 探测组件
- WebServiceBlamepointTracer@Opertationlevel 探测组件

示例

前端组件对调用停顿后端组件的 Web 服务进行调用。

前端--> Web 服务-->后端

将针对后端、Web 服务和前端组件创建停顿度量标准，因为它们在默认情况下都订阅下游监控。

上游继承组件停顿

停顿检查器首先检查组件堆栈的顶级成员，查看其是否已停顿。如果顶级组件未停顿，停顿检查器会查找已停顿的父组件。当父组件停顿时，将针对该组件和堆栈中的每个上游父组件创建停顿度量标准。

如果组件 M1() 调用多个组件 M2()（每个仅需花费几秒种），则可以为 M1() 报告停顿度量标准。

示例：

停顿阈值设置为 15 秒。方法 M1() 循环调用方法 M2() 100 次。M2() 从不停顿，因为每次响应只需 2 秒。但是，M1() 最终将停顿。

禁止将停顿作为事件进行捕获

默认情况下，Introscope 在事务事件数据库中将事务停顿作为事件进行捕获，并从已检测的事件生成停顿度量标准。将为事务中的第一个和最后一个方法生成停顿度量标准。用户可以在工作站历史事件查看器中查看停顿事件以及关联的度量标准。

注意：生成的停顿度量标准始终可用，但是停顿事件只有在安装 Introscope 错误检测器后才可见。停顿作为普通错误进行存储，可以在“错误”选项卡视图和实时错误查看器中查看。也可以通过查询“type:errorsnapshot”在历史查询查看器中查看停顿。停顿的事务在实时错误查看器中和“错误”选项卡下被报告为错误。当使用“Errors matching *”条件进行跟踪时，停顿的事务在“事务跟踪”窗口中并不作为错误进行报告，因为此行为是设计的行为。

要禁止将停顿作为事件进行捕获、更改停顿阈值，或更改 .NET 代理检查停顿的频率，请使用以下属性：

- introscope.agent.stalls.enable－控制代理是否检查停顿，并为已检测到的停顿创建事件。
- introscope.agent.stalls.thresholdseconds－指定响应时间最小阈值，达到该时间后，事务将视为已停顿。
- introscope.agent.stalls.resolutionseconds－指定代理检查停顿的频率。

重要信息！如果您禁止将停顿作为事件进行捕获，将不再支持 PBD 中的停顿跟踪器。

非标识事务跟踪

默认情况下，即使在 CEM UI 中启用非标识事务跟踪功能，也不对非标识事务进行跟踪。

要启用对非标识事务的跟踪，请在 introscopeAgent.profile 中将以下属性值设置为 FALSE。

```
introscope.agent.bizdef.turnOff.nonIdentifying.txn=FALSE
```

第 10 章： 配置 Introscope SQL 代理

此部分包含以下主题：

SQL 代理概览

Introscope SQL 代理可将详细的数据库性能数据报告给企业管理器。通过跟踪托管应用程序与数据库之间的交互，SQL 代理可以了解应用程序中各个 SQL 语句的性能。

SQL 代理监控 SQL 语句的方法与 .NET 代理监控 .NET Web 应用程序的方法相同。SQL 代理是非侵入式的，可使用非常低的开销来监控应用程序或数据库。

为了在单个 SQL 语句级别上提供有意义的性能度量标准，SQL 代理通过剥离特定于事务的数据并将原始 SQL 语句转换为特定于 Introscope 的规范化语句来汇总性能数据。由于规范化语句不包括敏感信息（如信用卡号），因此该过程还可以保护您数据的安全。

例如，SQL 代理将此 SQL 查询：

```
SELECT * FROM BOOKS WHERE AUTHOR = 'Atwood'
```

转换为此规范化语句：

```
SELECT * FROM BOOKS WHERE AUTHOR = ?
```

同样，SQL 代理将此 SQL 更新语句：

```
INSERT INTO BOOKS (AUTHOR, TITLE) VALUES ('Atwood', 'The Robber Bride')
```

转换为此规范化语句：

```
INSERT INTO BOOKS (AUTHOR, TITLE) VALUES (?, ?)
```

只有引号中的文本 ('xyz') 会进行规范化。

规范化语句的度量标准汇集在一起，可以在工作站调查器的 JDBC 节点中进行查看。

SQL 代理文件

SQL 代理包含在所有代理安装中。提供 SQL 代理功能的文件包括：

- *wily/ext/wily.SQLAgent.ext.dll*
- *wily/sqlagent.pbd*

默认情况下，代理扩展（如 *wily.SQLAgent.ext.dll* 文件）安装在 *wily/ext* 目录下。可以使用代理配置文件中的 *introscope.agent.extensions.directory* 属性更改代理扩展目录的位置。如果您更改了 */ext* 目录的位置，请务必同时移动 */ext* 目录的内容。

SQL 语句规范化

有些应用程序会生成大量唯一的 SQL 语句。如果正在使用 Hibernate 等技术，生成较长的唯一 SQL 语句的可能性将会增加。长 SQL 语句会在代理中造成度量标准爆发，从而导致性能下降以及其他系统问题。

有关 Hibernate 的更多信息，请参阅 http://www.hibernate.org。

各节信息

编写不当的 SQL 语句如何导致度量标准爆发

一般情况下，SQL 代理度量标准的数目应接近唯一 SQL 语句的数目。如果即使应用程序使用一小组 SQL 语句，SQL 代理仍显示大量且越来越多的唯一 SQL 度量标准，问题可能出在 SQL 语句的编写方式上。

SQL 语句可导致度量标准爆发的情况很常见。例如，每次执行用于包括嵌入的注释、创建临时表或插入值列表的 SQL 语句时，都会无意中创建唯一的度量标准名称。

示例：SQL 语句中的注释

度量标准爆发的一个常见原因是在 SQL 语句中使用注释的方法有误。例如，如果您的 SQL 语句如下：

```
"/* John Doe, user ID=?, txn=? */ select * from table..."
```

SQL 代理将创建度量标准名称中包含注释的度量标准：

```
"/* John Doe, user ID=?, txn=? */ select * from table..."
```

SQL 语句中嵌入的注释对于数据库管理员查看由谁执行何种查询、哪个执行查询的数据库忽略查询非常有用。然而，SQL 代理在捕获 SQL 语句时无法解析注释字符串。因此，对于每个唯一的用户 ID，SQL 代理会创建唯一的度量标准，这可能导致度量标准爆发。

可以通过将 SQL 注释放在单引号中来避免此问题。例如：

```
"/*' John Doe, user ID=?, txn=? '*/ select * from table..."
```

然后，SQL 代理将创建以下度量标准，其中注释不再产生唯一的度量标准名称：

```
"/* ? */ select * from table..."
```

示例：临时表或自动生成的表名称

度量标准爆发的另一个可能原因是：应用程序在 SQL 语句中引用临时表或自动生成名称的表。例如，如果打开调查器在“后端|{backendName}|SQL|{sqlType}|sql”下查看度量标准， 可能会看到与下面类似的度量标准：

```
SELECT * FROM TMP_123981398210381920912 WHERE ROW_ID = ?
```

此 SQL 语句正在访问将唯一标识符附加到表名的临时表。附加到 *TMP_* 表名的其他数字会在每次执行语句时创建一个唯一的度量标准名称，从而导致度量标准爆发。

示例：可生成值列表或插入值的语句

度量标准爆发的另一个常见原因是：SQL 语句生成值列表或对值进行大量修改。例如，假设系统已警告您可能会发生度量标准爆发，并且您通过调查对以下 SQL 语句进行了检查：

```
#1 INSERT INTO COMMENTS (COMMENT_ID, CARD_ID, CMMT_TYPE_ID,
CMMT_STATUS_ID,CMMT_CATEGORY_ID, LOCATION_ID, CMMT_LIST_ID, COMMENTS_DSC,
USER_ID,LAST_UPDATE_TS) VALUES (?, ?, ?, ?, ?, ?, ?, "CHANGE CITY FROM CARROLTON,TO
CAROLTON, _ ", ?, CURRENT)
```

在研究代码时，请注意 *"CHANGE CITY FROM CARROLTON, TO CAROLTON, _ "* 会生成一个城市数组。

同样，如果您正在调查可能的度量标准爆发，可能会复查与下面类似的 SQL 语句：

```
CHANGE COUNTRY FROM US TO CA _ CHANGE EMAIL ADDRESS FROM TO BRIGGIN @ COM _ "
```

在研究代码时，会注意到 CHANGE COUNTRY 产生很长的国家/地区列表。此外，国家/地区的引号放置可导致在电子邮件地址中插入 SQL 语句，从而创建可能会引起度量标准爆发的唯一度量标准。

SQL 语句规范化选项

为了解决长 SQL 语句问题，SQL 代理提供了以下规范化程序以供使用：

- 默认 SQL 语句规范化程序 (p. 124)
- 自定义 SQL 语句规范化程序 (p. 125)
- 正则表达式 SQL 语句规范化程序
- 命令行表达式 SQL 语句规范化程序

默认 SQL 语句规范化程序

默认情况下，标准的 SQL 语句规范化程序在 SQL 代理中处于启用状态。它对单引号内的文本 ('xyz') 进行规范化。例如，SQL 代理将此 SQL 查询：

```
SELECT * FROM BOOKS WHERE AUTHOR = 'Atwood'
```

转换为此规范化语句：

```
SELECT * FROM BOOKS WHERE AUTHOR = ?
```

规范化语句的度量标准汇集在一起，可以在工作站调查器中进行查看。

自定义 SQL 语句规范化程序

SQL 代理允许用户添加扩展来执行自定义规范化。为此，请创建一个 DLL 文件，其中包含由 SQL 代理实施的规范化方案。

应用 SQL 语句规范化程序扩展：

1. 创建扩展 DLL 文件。

   SQL 规范化程序扩展文件的入口点类必须实施 *com.wily.introscope.agent.trace.ISqlNormalizer* 接口。

   创建 DLL 扩展文件包括创建清单文件，该清单文件包含下面步骤 2 中详细介绍的 SQL 规范化程序扩展的特定键。然而，要使扩展能够正常运行，还需要其他常规键。这些键是您将要用来构建任意扩展文件的类型。创建的扩展文件与数据库 SQL 语句文本规范化相关，例如 *Backends|{backendName}|SQL|{sqlType}|{actualSQLStatement}* 节点下的度量标准。*{actualSQLStatement}* 由 SQL 规范化程序进行规范化。

2. 将以下键放置在所创建扩展的清单中：

   ```
   com-wily-Extension-Plugins-List:testNormalizer1
   ```

   注意：此键的值可为任意值。在本例中，*testNormalizer1* 用作示例。无论您为此键指定何值，请在以下键中使用该值。

   ```
   com-wily-Extension-Plugin-testNormalizer1-Type: sqlnormalizer
   com-wily-Extension-Plugin-testNormalizer1-Version: 1
   com-wily-Extension-Plugin-testNormalizer1-Name: normalizer1
   ```

   注意：上述键应当包含规范化程序的唯一名称，例如 normalizer1。

   ```
   com-wily-Extension-Plugin-testNormalizer1-Entry-Point-Class:
   <Thefully-qualified classname of your implementation of ISQLNormalizer>
   ```

3. 将创建的扩展文件放在 *<Agent_Home>\wily\ext* 目录中。

4. 在 *IntroscopeAgent.profile* 中，找到以下属性并进行设置：

   ```
   introscope.agent.sqlagent.normalizer.extension
   ```

   从所创建扩展的清单文件中，将属性设置为 *com-wily-Extension-Plugin-{plugin}-Name*。此属性的值不区分大小写。例如：

   ```
   introscope.agent.sqlagent.normalizer.extension=normalizer1
   ```

   注意：这是一个热属性。更改扩展名称将导致重新注册扩展。

5. 在 *IntroscopeAgent.profile* 中，您可以选择性地添加以下属性来设置错误限制计数：

   ```
   introscope.agent.sqlagent.normalizer.extension.errorCount
   ```

   有关错误和异常的更多信息，请参阅异常 (p. 126)。

   注意：如果自定义规范化程序扩展抛出的错误超过了错误限制计数，将禁用扩展。

6. 保存 *IntroscopeAgent.profile*。
7. 重新启动应用程序。

异常

如果创建的扩展针对某个查询抛出了异常，默认 SQL 语句规范化程序将为该查询使用默认的规范化方案。发生这种情况时，将记录一条 ERROR 消息，指出扩展抛出了异常，还会记录一条包含堆栈跟踪信息的 DEBUG 消息。但是，抛出五个此类异常之后，默认 SQL 语句规范化程序将禁用所创建的扩展，并停止尝试将创建的扩展用于将来的查询，直到更改了规范化程序。

空字符串

如果创建的扩展针对某个查询返回了空字符串，StatementNormalizer 将为该查询使用默认的规范化方案，并记录一条 INFO 消息，指出扩展返回了空值。但是，返回五个此类空字符串之后，StatementNormalizer 将停止记录消息，但是将尝试继续使用扩展。

第 11 章： 故障排除和提示

本节包含使用 .NET 代理的常见配置问题和建议的做法。

此部分包含以下主题：

检查 .NET 代理是否已正确安装

在某些情况下，您可能不知道是否已经在特定计算机上安装或禁用 .NET 代理，或者不知道存在安装或配置错误。如果您不确定是否已安装 .NET 代理，或者要验证 .NET 代理是否配置正确，您可以通过执行以下操作来检查环境：

- 确认 .NET 代理环境变量已设置并且值正确。

  打开命令窗口，键入 *set* 并将输出定向到一个文本文件中，然后查看该文件以验证是否已设置下列环境变量：

  ```
  com.wily.introscope.agentProfile=C:\Program Files\CA
  Wily\Introscope9.1.0.0\wily\IntroscopeAgent.profile
  Cor_Enable_Profiling=0x1
  COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}
  ```

- 导航到 Windows 系统文件夹中的 *assembly* 目录（如 *C:\WINDOWS\assembly*），并确认 *wily.Agent* 已作为注册的程序集名称列出且版本信息正确。

  如果未看到 *wily.Agent* 列出，请使用 *wilyregtool.exe* 或 *gacutil.exe* 在全局程序集缓存中手工注册该程序集。

- 检查运行 IIS 工作进程的用户帐户，并确认运行工作进程的所有帐户对 *<代理主目录>* 目录有“完全控制”权限。

更正代理扩展的版本信息

扩展的版本号可能与 .NET 代理的版本号不同，具体取决于何时安装 .NET 代理以及任何可选扩展。在大多数情况下，如果 .NET 代理和扩展之间的版本信息不同，代理会记录错误消息，而扩展将无法正常运行。要更正此类问题，可以手动更新版本信息。根据您的环境，执行以下操作之一：

- 将应用程序逐个配置为使用正确的代理版本号。
- 将所有应用程序配置为全局使用正确的代理版本号。

重要信息：仅选择一个选项；不要同时执行两个选项。

逐个为应用程序配置版本信息：

1. 如果要安装的扩展是常规的 *.exe*，请创建一个名为 *<Extension_Name>.exe.config* 的文件，并将此文件放在原始 *.exe* 所在的同一目录下。
2. 如果扩展是 IIS 应用程序，请创建一个名为 *w3wp.exe.config* 的文件，并将其放在 *w3wp.exe* 所在的同一目录下。这适用于默认域。
3. 如果扩展是 IIS 应用程序，还要将以下内容添加到每个应用程序的 *web.config*：

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="..." .../>
      <bindingRedirect oldVersion="0.0.0.0 - 65535.65535.65535.65535"
newVersion="<AGENT.VERSION.NUMBER>"/>
   </dependentAssembly>
</assemblyBinding>
```

   为 *assemblyIdentity name* 输入相应的内容，并将代理版本号替换为您的 .NET 代理版本号。版本信息应包括四位数。例如，如果安装了 **9.0.7** 版本的 .NET 代理，则应添加以下内容：

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="wily.Agent"
publicKeyToken="2B41FDFB6CD662A5"/>
      <bindingRedirect oldVersion="9.0.5.0 - 65535.65535.65535.65535"
newVersion="9.0.7.0" />
   </dependentAssembly>
</assemblyBinding>
```

注意：如果以上文件已存在，请在 *<runtime>* 下添加 *<assemblyBinding>* 节点。

全局配置所有应用程序：

- 将以下内容添加到 *machine.config* 文件中，该文件通常位于应用程序的 *%runtime install path%\Config* 目录下：

```
assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="..." .../>
      <bindingRedirect oldVersion="0.0.0.0 - 65535.65535.65535.65535"
newVersion="<AGENT.VERSION.NUMBER>"/>
   </dependentAssembly>
</assemblyBinding>
```

  为 *assemblyIdentity name* 输入相应的内容，并将代理版本号替换为您的 .NET 代理版本号。版本信息应包括四位数。例如，如果安装了 **9.0.7** 版本的 .NET 代理，应添加以下内容：

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="wily.Agent"
publicKeyToken="2B41FDFB6CD662A5"/>
      <bindingRedirect oldVersion="9.0.5.0 - 65535.65535.65535.65535"
newVersion="9.0.7.0" />
   </dependentAssembly>
</assemblyBinding>
```

注意：将该代码段添加到 *machine.config* 中会全局影响所有应用程序。

选择使用还是禁用 hotdeploy 目录

通过 *hotdeploy* 目录，您可以在不编辑 *IntroscopeAgent.profile* 甚至不重新启动托管应用程序的情况下部署新指令。在使用此功能时需特别谨慎。如果您的自定义 PBD 包含无效语法或者配置为收集过多度量标准，将会更快地受到影响。无效的 PBD 会导致 NativeProfiler 关闭，收集过多度量标准的 PBD 会影响应用程序的性能。

为解决此问题，CA Technologies 建议：

- 在将 QA 和性能环境中的所有指令推行到生产环境之前，先进行测试和验证。
- 确保您的服务器环境的变更控制过程已更新，使之反映用于部署 PBD 的新选项。

当新 PBD 放在 *hotdeploy* 目录下时，.NET 代理会自动部署该新 PBD。但是，只有重新启动应用程序后，新增或更改过的 PBD 才会影响已在运行的类和应用程序。

注意：您只能部署 *hotdeploy* 目录中的 PBD 文件。代理将忽略放置在该目录中的任何 ProbeBuilder 列表 (PBL)。

如果要防止自动部署无效的 PBD 文件，可以禁用 *hotdeploy* 目录。

禁用 *hotdeploy* 目录：

1. 将存储在 *hotdeploy* 目录中的所有自定义 PBD 移到 *<Agent_Home>* 目录。
2. 在文本编辑器中打开 *IntroscopeAgent.profile*。
3. 从 *introscope.autoprobe.directivesFile* 属性中删除 *hotdeploy*。
4. 将要使用的自定义 PBD 添加到 *introscope.autoprobe.directivesFile*。例如：

   ```
   introscope.autoprobe.directivesFile=default-typical.pbl,custom1.pbd,custom2.pbd,custom3.pbd
   ```

5. 保存 *IntroscopeAgent.profile* 并重新启动代理。

附录 A： .NET 代理属性

本部分包括以下主题：

.NET 代理到企业管理器的连接

您可以控制代理如何连接到企业管理器。

introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT

指定运行代理默认与之连接的企业管理器的计算机主机名。

默认

localhost

示例

```
introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT
=localhost
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT

指定托管企业管理器的计算机上用于侦听来自代理的连接的端口号。

默认

默认端口取决于要配置的通信通道的类型。对于代理和企业管理器之间的直接通信，默认端口为 5001。

示例

```
introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT
=5001
```

注释

- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。
- 如果想要通过 HTTPS（基于 SSL 的 HTTP）连接到企业管理器，默认端口为 8444。如果想要通过 SSL 连接到企业管理器，默认端口为 5443。但是，默认情况下，这些默认设置已注释掉。

introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT

指定默认客户端套接字工厂，以用于从代理到企业管理器的连接。

默认

默认客户端套接字工厂取决于要配置的通信通道的类型。对于代理与企业管理器之间的直接通信，默认的套接字工厂如下所示：

com.wily.isengard.postofficehub.link.net.DefaultSocketFactory

示例

```
introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

代理故障转移

如果代理与其主企业管理器的连接断开，这些属性会指定代理将尝试连接的备份企业管理器，以及代理尝试重新连接到其主企业管理器的频率。

introscope.agent.enterprisemanager.connectionorder

指定代理断开与默认企业管理器的连接时使用的备份企业管理器连接顺序。

属性设置

代理可以连接到的其他企业管理器的名称。

默认

企业管理器由默认主机名、端口号和套接字工厂属性定义。

示例

```
introscope.agent.enterprisemanager.connectionorder=DEFAULT
```

注释

- 使用逗号分隔列表。
- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds

指定被拒绝的代理尝试重新连接到以下企业管理器的间隔秒数：

- 基于在代理配置文件 introscope.agent.enterprisemanager.connectionorder 属性值中配置的顺序的企业管理器。
- 根据 loadbalancing.xml 配置允许连接的任何企业管理器。

如果代理无法连接到企业管理器，可以按下列方式处理连接：

- 尝试连接到下一个允许的企业管理器。
- 不连接到任何不允许连接的企业管理器。

注意：有关配置 loadbalancing.xml 和代理与企业管理器的连接的信息，请参阅《*CA APM 配置和管理指南*》。

默认

默认时间间隔是 120 秒。

示例

```
introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds=120
```

注释

必须重新启动托管应用程序，对此属性所做的更改才能生效。

默认情况下，此属性注释掉。

该属性在允许代理跨以下 CA APM 组件进行连接的环境中非常有用：

- 群集。
- 收集器和独立企业管理器。
- 群集、收集器和独立企业管理器的任意组合。

如果代理可以连接到不同群集中的企业管理器且未设置该属性，则以下示例显示了会发生的状况：

1. 连接到群集中企业管理器的代理断开连接。
2. 代理以不允许的模式连接到群集 2 中的企业管理器。
3. 代理不知道群集 1 中允许的企业管理器何时变得可用。

该属性强制代理持续尝试连接到允许的企业管理器，直到某个企业管理器可供连接。

代理度量标准老化

代理度量标准老化功能可定期从代理内存缓存中删除死度量标准。死度量标准是指在所配置的一段时间内没有报告新数据的度量标准。删除旧的度量标准有助于提高代理性能，避免潜在的度量标准爆发。

注意：如果无意中将代理设置为报告的度量标准多于系统可以处理的度量标准，将发生度量标准爆发。如果报告了太多度量标准，代理会影响应用程序服务器的性能，在极个别情况下，还会使服务器完全无法运行。

仅当组中的所有度量标准均被视为删除候选项时，才会删除组中的度量标准。目前，只有 *BlamePointTracer* 和 *MetricRecordingAdministrator* 度量标准可按组删除。其他度量标准均单独删除。

MetricRecordingAdministrator 具有以下用于创建、检索或删除度量标准组的接口：

- *getAgent().IAgent_getMetricRecordingAdministrator.addMetricGroup*

  字符串组件，收集度量标准。组件名称是度量标准组的度量标准资源名称。度量标准必须位于同一个度量标准节点下，才能称之为一个组。度量标准是 *com.wily.introscope.spec.metric.AgentMetric* 数据结构的集合。您只能将 *AgentMetric* 数据结构添加到此集合中。

- *getAgent().IAgent_getMetricRecordingAdministrator.getMetricGroup*

  字符串组件。根据组件名称（即度量标准资源名称），您可以获得度量标准集合。

- *getAgent().IAgent_getMetricRecordingAdministrator.removeMetricGroup*

  字符串组件。度量标准组基于组建名称（即度量标准资源名称）进行删除。

- *getAgent().IAgent_getDataAccumulatorFactory.isRemoved*

  检查度量标准是否已删除。如果在扩展中保留一个累积器实例，则使用此接口。如果累积器由于度量标准老化而被删除，可使用此接口防止保留死引用。

重要信息！ 如果要创建使用 *MetricRecordingAdministrator* 界面的扩展（例如，用于与其他 CA Technologies 产品配合使用），请确保删除自己的累积器实例。如果度量标准因未被调用而过期，并且稍后数据可用于该度量标准，则旧的累积器实例将不会创建新的度量标准数据点。要避免这种情况，请不要删除您自己的累积器实例，并改用 *getDataAccumulatorFactory* 接口。

配置代理度量标准老化

默认情况下，代理度量标准老化处于打开状态。您可以选择使用属性 introscope.agent.metricAging.turnOn (see page 139) 关闭此功能。如果从 *IntroscopeAgent.profile* 中删除此属性，默认情况下将关闭代理度量标准老化。

代理度量标准老化将针对代理中的心跳运行。心跳使用属性 introscope.agent.metricAging.heartbeatInterval (see page 140) 进行配置。请务必将心跳的频率维持在较低状态。较高的心跳频率将影响代理和 CA Introscope® 的性能。

在每个心跳期间，会检查一组特定的度量标准。这可通过属性 introscope.agent.metricAging.dataChunk (see page 140) 进行配置。将此值维持在较低状态也很重要，因为较高的值将影响性能。默认值是每个心跳检查 500 个度量标准。将逐个检查这 500 个度量标准以查看是否是删除候选项。例如，如果将此属性设置为每个心跳检查 500 个度量标准区块，并且代理内存中共有 10,000 个度量标准，则它将在对性能造成较小影响的前提下花费更长时间来检查所有 10,000 个度量标准。但是，如果将此属性设置为较高的值，您可以更快地检查这 10,000 个度量标准，但是开销可能会很大。

如果度量标准在特定时间段之后没有收到新数据，该度量标准将成为删除候选项。您可以使用属性 introscope.agent.metricAging.numberTimeslices (see page 141) 来配置此时间段。默认情况下，此属性设置为 3000。如果某个度量标准满足删除以上条件，将执行检查以查看其所在组中的所有度量标准是否是删除候选项。如果也满足此项要求，则删除该度量标准。

introscope.agent.metricAging.turnOn

打开或关闭代理度量标准老化。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.metricAging.turnOn=true
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.metricAging.heartbeatInterval

指定检查待删除度量标准的时间间隔（以秒为单位）。

默认

1800

示例

```
introscope.agent.metricAging.heartbeatInterval=1800
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.metricAging.dataChunk

指定在每个时间间隔内检查的度量标准数。

默认

500

示例

```
introscope.agent.metricAging.dataChunk=500
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.metricAging.numberTimeslices

在成为删除候选项之前，此属性会指定要检查的时间间隔数（没有任何新数据）。

默认

3000

示例

```
introscope.agent.metricAging.numberTimeslices=3000
```

注释

对此属性所做的更改会立即生效，不需要重新启动托管应用程序。

introscope.agent.metricAging.metricExclude.ignore.0

使指定的度量标准不被删除。要防止一个或多个度量标准老化，请将度量标准名称或度量标准筛选添加到列表中。

属性设置

以逗号分隔的度量标准列表。可以在度量标准名称中使用星号 (*) 作为通配符。

默认

默认情况下，度量标准名称以 Threads 开头 (*Threads**)。

示例

```
introscope.agent.metricAging.metricExclude.ignore.0=Threads*
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

代理度量标准限定

您可以配置代理以便大致限定发送给企业管理器的度量标准数目。如果生成的度量标准数目超过该属性值，则代理停止收集和发送新度量标准。

introscope.agent.metricClamp

将代理配置为大致限定发送给企业管理器的度量标准数目。

默认

5000

示例

```
introscope.agent.metricClamp=5000
```

注释

- 如果不设置该属性，则不会发生度量标准限定。旧的度量标准仍将报告值。
- 对此属性所做的更改可立即生效，不需要重新启动托管应用程序。
- 该限定属性与位于 apm-events-thresholds-config.xml 文件中的 introscope.enterprisemanager.agent.metrics.limit 属性一起使用。

  注意：有关 introscope.enterprisemanager.agent.metrics.limit 属性的信息，请参阅《*CA APM 配置和管理指南*》。

如果 introscope.enterprisemanager.agent.metrics.limit 限定值在 introscope.agent.metricClamp 值之前触发，企业管理器会读取代理度量标准，但是不会在调查器度量标准浏览器树中报告它们。

如果 introscope.agent.metricClamp 限定值在 introscope.enterprisemanager.agent.metrics.limit 限定值之前触发，代理将停止向企业管理器发送度量标准。

introscope.agent.simpleInstanceCounter.referenceTrackingLimit

SimpleInstanceCounter 度量标准跟踪为每个类创建的实例数。过多的实例可能会导致大量的 .NET 代理开销。使用该属性可限制每个类的跟踪实例数。

选项

整数

默认

25000

示例

```
introscope.agent.simpleInstanceCounter.referenceTrackingLimit=25000
```

注释

对此属性所做的更改会立即生效，不需要重新启动托管应用程序。

代理内存开销

只有在某些极端情况下代理内存开销才会明显。降低内存消耗的典型代价是可能会延长响应时间。但是，每个应用程序都是唯一的，内存使用情况和响应时间之间的权衡可能会因应用程序本身而异。

introscope.agent.reduceAgentMemoryOverhead

此属性可指定要使用的代理配置。如果想要尝试减少代理内存开销，请取消注释该属性。默认情况下，此属性设置为 true，且已注释掉。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.reduceAgentMemoryOverhead=true
```

注释

必须重新启动托管应用程序，以便对此属性所做的更改生效。

代理命名

您可以控制自动代理和日志文件命名以及与命名相关的操作。

introscope.agent.agentAutoNamingEnabled

指定是否使用代理自动命名为支持的应用程序服务器获取 .NET 代理名称。

选项

True 或 False

默认

True

示例

```
introscope.agent.agentAutoNamingEnabled=true
```

注释

- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。
- 要求为 WebLogic 指定 Startup Class；要求为 WebSphere 指定 Custom Service。
- 设置为 true，在支持的应用程序服务器附带的代理配置文件中未注释掉。

introscope.agent.agentAutoNamingMaximumConnectionDelayInSeconds

指定代理在连接到企业管理器之前获取命名信息所等待的最大秒数。

属性设置

代表所延迟秒数的正整数。

默认

默认值为 120 秒。

示例

```
introscope.agent.agentAutoNamingMaximumConnectionDelayInSeconds=120
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.agentAutoRenamingIntervalInMinutes

指定代理检查以确认是否已重命名的频率。代理会在每个您指定的时间间隔内检查确认自己的名称是否更改。

属性设置

代表所间隔分钟数的正整数。

默认

默认为每 10 分钟一次。

示例

```
introscope.agent.agentAutoRenamingIntervalInMinutes=10
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.disableLogFileAutoNaming

指定是否禁用代理日志文件的自动命名。默认情况下，如果将代理配置为使用自动命名，其日志文件也将使用代理名称或时间戳自动命名。将该属性设置为 true 可禁用默认行为。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.disableLogFileAutoNaming=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.agentName

指定在所有其他代理命名方法失败时要用于该代理的名称。如果此属性的值无效或者此属性已从配置文件中删除，则默认代理名称为 *Unknown Agent*。默认情况下，此属性注释掉。如果其他代理命名方法失败，可以取消注释该属性以提供默认代理名称。

属性设置

用于在使用任何其他方法都无法确定代理名称时作为代理名称的文本字符串。

默认

AgentName

示例

```
introscope.agent.agentName=AgentName
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.clonedAgent

使您能够在同一台计算机上运行应用程序的相同副本。如果在同一台计算机上运行应用程序的相同副本，请将该属性设置为 true。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.clonedAgent=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.display.hostName.as.fqdn

该属性指定是否将代理名称显示为完全限定域名 (fqdn)。要启用完全限定域名，请将该属性值设置为 true。默认情况下，代理显示主机名。

注意：对于 Catalyst 集成，请将该属性设置为 true。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.display.hostName.as.fqdn=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

代理记录（业务记录）

您可以控制代理处理业务事务记录的方式。

注意：有关代理业务记录的详细信息，请参阅*《CA APM 事务定义指南》*。

introscope.agent.bizRecording.enabled

对代理启用或禁用业务事务记录。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.bizRecording.enabled=true
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

要进一步为代理配置业务事务记录，请参阅应用程序分类视图的其他属性。

详细信息：

应用程序分类视图 (p. 150)
应用程序分类图业务事务 POST 参数 (p. 155)

代理线程优先级

您可以控制代理线程的优先级。

introscope.agent.thread.all.priority

指定代理线程的优先级。

属性设置

值的范围为 0（低）到 4（高）。

默认

默认优先级为 2。

示例

```
introscope.agent.thread.all.priority=2
```

应用程序分类视图

您可以配置应用程序分类视图数据。

注意：有关如何使用应用程序分类视图的信息，请参阅《*CA APM Workstation 用户指南*》。

introscope.agent.appmap.enabled

为应用程序分类视图启用或禁用对受监控代码的跟踪。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.appmap.enabled=true
```

注释

默认情况下处于启用状态。

introscope.agent.appmap.metrics.enabled

为应用程序分类视图节点启用或禁用度量标准跟踪。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.appmap.metrics.enabled=false
```

注释

默认情况下，此属性注释掉。

introscope.agent.appmap.queue.size

设置应用程序分类视图的缓冲区大小。

属性设置

正整数。

默认

1000

示例

```
introscope.agent.appmap.queue.size=1000
```

注释

- 该值必须是正整数。
- 如果该值设置为 0，则缓冲区没有边界。
- 默认情况下，此属性注释掉。

introscope.agent.appmap.queue.period

设置将应用程序分类视图数据发送到企业管理器的频率（以毫秒为单位）。

属性设置

正整数。

默认

1000

示例

```
introscope.agent.appmap.queue.period=1000
```

注释

- 必须是正整数。
- 如果值设置为 0，将使用默认值。
- 默认情况下，此属性注释掉。

introscope.agent.appmap.intermediateNodes.enabled

启用或禁用包括应用程序前端和后端节点之间的中间节点的功能。

属性设置

True 或 False

默认

False

示例

```
#introscope.agent.appmap.intermediateNodes.enabled=true
```

注释

- 对此属性所做的更改可立即生效，不需要重新启动托管应用程序。
- 如果将此属性设置为 true，代理性能将会降低。
- 默认情况下，此属性注释掉。

应用程序分类视图和 Catalyst 集成

您可以为 Catalyst 集成配置应用程序分类视图数据。

注意：有关如何使用应用程序分类视图的信息，请参阅《*CA APM Workstation 用户指南*》。

配置发送信息的功能

通过此属性，可以启用或禁用代理发送与 Catalyst 集成所需的其他信息的功能。

请执行以下步骤：

1. 在文本编辑器中打开默认的 IntroscopeAgent.profile 文件。

找到 introscope.agent.appmap.catalystIntegration.enabled=<false|true> 行，并如下所示设置值：

true

启用代理发送与 Catalyst 集成所需的其他信息的功能。

false

禁用该配置。

以下示例显示了格式：

```
introscope.agent.appmap.catalystIntegration.enabled=false
```

注意：默认情况下，此属性会注释掉。

2. 保存并关闭文件。

代理将设置为使用该配置。

配置可用网络的列表

introscope.agent.primary.net.interface.name 属性指定代理针对 Catalyst 集成使用的主计算机的主网络接口名称。您可以更改该属性的配置，并且更改将自动应用。

注意：代理日志记录级别设置成调试时，可用于配置的网络接口名称相关信息将显示在日志文件中。另外，您还可以使用 Network Interface 实用程序来确定该属性的主网络接口名称。

请执行以下步骤：

1. 在文本编辑器中打开默认的 IntroscopeAgent.profile 文件。
2. 找到行：
   introscope.agent.primary.net.interface.name=<false|true>，并指定名称值。

   下面的示例显示了该名称格式：
   `introscope.agent.primary.net.interface.name=eth4`

   注意：未定义默认值。不设置该属性时，代理会将第一个可用网络接口指定为主接口。您可以使用 Network Interface 实用程序来确定用于该属性的名称值。
3. （可选）通过指定子接口编号（从 0 开始）允许多个网络地址。

   以下示例显示了子接口编号格式：
   `introscope.agent.primary.net.interface.name=eth4.1`
4. 保存并关闭文件。

   此时配置文件设置为使用该配置。

详细信息：

使用网络接口实用工具 (p. 221)

应用程序分类图业务事务 POST 参数

您可以通过匹配 POST 参数，配置 Local Product Shorts 以执行更复杂的监控。

introscope.agent.bizdef.matchPost

此属性确定何时匹配 POST 参数。

属性设置

此属性的有效设置包括 *never*、*before* 或 *after*。

- 将属性设置为 **never** 可获取完整的代理功能和更好的性能。此设置使得应用程序可以使用 URL、cookie 或标头参数来识别所有业务事务，但是无法匹配任何通过 POST 参数单独识别的业务事务。
- 将属性设置为 **before** 可获取完整的代理性能。此设置允许应用程序使用 POST 参数来识别部分或全部业务事务，但是从不直接访问 servlet 数据流来获取 HTTP 表单请求。该属性设置为 before 时，部署的新应用程序也必须符合标准 API。

  重要信息！将此属性设置为 *before* 时，可能会对应用程序造成具有潜在危险的影响。在实施之前，请与 CA Technologies 代表一起检查该属性设置。

- 将属性设置为 **after**，可使业务事务与 POST 参数安全匹配，但代理功能将受到限制。将此属性设置为 *after* 时，代理将无法在进程之间映射由 POST 参数识别的业务事务，或为这些业务事务生成全套度量标准。与其他选项相比，此设置还会消耗稍多的 CPU 时间，但是如果需要 POST 参数功能，则该设置被视为是最安全的。它允许应用程序使用 POST 参数来识别部分或全部业务事务，但不保证从不直接访问 servlet 数据流。

示例

```
introscope.agent.bizdef.matchPost=after
```

注释

- *never*—从不试图匹配 POST 参数。这是最快的选项，但可能导致不准确的业务事务组件匹配。
- *before*—在 servlet 执行之前匹配 POST 参数。
- *after*—在 servlet 执行之后匹配 POST 参数模式。跨进程映射和一些度量标准将不可用。这是该参数的默认设置。

已知限制

使用代理记录定义的度量标准显示在调查器的应用程序分类视图中。在配置代理记录时，使用正则表达式时会有一些已知的限制。大部分限制与 POST 参数有关。

已知的限制包括：

- POST 参数值不支持行终止符 (.)。
- 如果 POST 参数定义依赖于业务事务定义，则仅向业务事务组件提供三个度量标准。这些度量标准为：
  - 平均响应时间
  - 每个时间间隔的响应数
  - 每个时间间隔的错误

- 如果 POST 参数定义依赖于业务事务定义，则事务跟踪组件中的业务组件名称将有一个通用名称，而不是业务服务、业务事务和业务事务组件的特定名称。这也适用于业务事务定义，这些定义依赖于不匹配的 POST 参数定义。
- JBoss 和 Tomcat 的某些版本可能会将头键保存为小写值，从而导致 *caseSensitiveName* 属性无法正确处理 *HEADER_TYPE*。

注意：有关代理记录的详细信息，请参阅 *《CA APM 事务定义指南》*。

AutoProbe

您可以控制代理与 AutoProbe 的交互方式。

重要信息！以下属性是必需的参数。如果未设置 AutoProbe 属性或者值无效，Introscope 不会运行。

introscope.autoprobe.directivesFile

指定要部署的 ProbeBuilder 指令 (*.pbd*) 和 ProbeBuilder 列表 (*.pbl*) 文件。为该属性列出的文件确定检测的组件。此属性是必需属性。

属性设置

单个条目或以逗号分隔的条目列表。该列表可能包括以下内容的任意组合：

- ProbeBuilder 指令 (*.pbd*) 文件名
- ProbeBuilder 列表 (*.pbl*) 文件名
- *hotdeploy* 目录名称（放在 *hotdeploy* 目录下的 *.pbd* 文件会自动加载，无需编辑代理配置文件）

您可以指定列出的文件名的绝对路径，也可以使用 *IntroscopeAgent.profile* 文件位置的相对路径指定文件名。

默认

默认条目取决于安装代理时您所做的选择。

示例

```
introscope.autoprobe.directivesFile=default-full.pbl,hotdeploy
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.autoprobe.enable

此属性设置为 false 时，出现下列情况：

- 禁用 AutoProbe
- .NET 代理不连接到企业管理器
- .NET 代理不显示在调查器中
- .NET 代理不报告度量标准

属性设置

选项

true 或 false

默认

true

示例

```
introscope.autoprobe.enable=true
```

注意：JVM 的重新启动对于任何更改的生效是必需的。

introscope.autoprobe.logfile

检测日志文件的名称和位置。将此属性设置为将日志文件的位置移动到默认位置以外的位置。

属性设置

检测日志文件的绝对或相对路径。

默认

默认位置是相对于 *<Agent_Home>* 目录的 *logs\AutoProbe*.log。

示例

```
introscope.autoprobe.logfile=logs/AutoProbe.log
```

要禁用日志记录，请按以下方式注释掉日志文件：

```
introscope.autoprobe.logfile=logs/AutoProbe.log
```

注释

- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。

CA CEM 代理配置文件属性

您可以配置与 CA CEM 相关的 IntroscopeAgent.profile 属性。CA Introscope® 代理配置文件位于 *<代理主目录>*\wily 目录中。

为了使 CA CEM 与 CA Introscope® 能够正常集成，所有与 CA CEM 相关的属性均预配置为必需选项。

introscope.autoprobe.directivesFile

您需要通过 directivesFile 属性配置启用 ServletHeaderDecorator/HTTPHeaderDecorator 和 CEMTracer。

指令文件属性指定了应从何处查找 AutoProbe 的指令文件 (PBD) 或指令列表 (PBL)。

AutoProbe 使用指令来启用您的应用程序，并确定代理将哪些度量标准报告给企业管理器。

设置

根据所安装的代理应用程序服务器，格式为 <*appserver*>-full.pbl 或 <*appserver*>-typical.pbl。

默认

```
default-typical.pbl
```

示例

```
introscope.autoprobe.directivesFile=weblogic-typical.pbl
```

注释

虽然可以直接向此属性列表的末尾添加“ServletHeaderDecorator.pbd”或“httpheaderdecorator.pbd”，但最佳做法是：

1. 找到属性中指定的 PBL 文件（在上面的示例中为 weblogic-typical.pbl）。
2. 在文本编辑器中打开该 PBL 文件。
3. 对于 Java 代理，取消注释以启用 ServletHeaderDecorator.pbd 行。
4. 对于 .NET 代理，取消注释以启用 httpheaderdecorator.pbd 行。
5. 保存对 PBL.profile 文件所做的更改。

introscope.agent.remoteagentconfiguration.allowedFiles

此属性可确定允许从任何计算机远程复制到代理目录的文件。

企业管理器可使用此属性中的文件名来识别要发送给代理的有效 CA CEM 域配置文件。域配置文件包含 CA CEM 业务服务和事务定义。

设置

使用有效的文件名。

默认

domainconfig.xml

示例

```
introscope.agent.remoteagentconfiguration.allowedFiles=domainconfig.xml
```

注释

此属性也适用于 Introscope 命令行 Workstation (CLW)“发送配置文件”命令。

有关详细信息，请参阅使用命令行 Workstation。

此属性适用于 CA CEM 版本。

introscope.agent.remoteagentconfiguration.enabled

如果此布尔值设置为 true，则允许将远程文件从其他计算机复制到代理。

企业管理器要求将此属性设置为 true，以便将 CA CEM 域配置文件发送到代理。域配置文件包含 CA CEM 业务服务和事务定义。

设置

true 或 false

默认

- 对于 Java 代理，设置为 true
- 对于 .NET 代理，设置为 false

示例

```
introscope.agent.remoteagentconfiguration.enabled=true
```

注释

远程用户也可以使用 CA Introscope® 命令行 Workstation (CLW)“发送配置文件”命令将 introscope.agent.remoteagentconfiguration.allowedFiles 属性中指定的文件复制到代理目录中。

此属性适用于 CA CEM 4.0 和 4.1 版本。仅当您已在“Introscope 设置”页面上选择“CEMTracer 4.0/4.1 支持”选项时，此属性才适用于 CA CEM 4.2/4.5。

通过“CEMTracer 4.0/4.1 支持”选项，可以随时将代理从 4.0 或 4.1 交错迁移到 4.2/4.5。仅在需要时使用此选项。

对于不兼容的代理（即，不支持的 .NET 代理、EPA 代理或其他非 Java 代理），请将 introscope.agent.remoteagentconfiguration.enabled 属性设置为 false。

introscope.agent.decorator.enabled

如果此布尔值设置为 true，它会将代理配置为将其他性能监控信息添加到 HTTP 响应标头中。ServletHeaderDecorator/HTTPHeaderDecorator 可将 GUID 附加到每个事务，并将 GUID 插入 HTTP 标头 x-apm-info。

此属性可启用 CA CEM 与 CA Introscope® 之间的事务关联。

设置

true 或 false

默认

- 对于 Java 代理，设置为 false
- 对于 .NET 代理，设置为 true

示例

```
introscope.agent.decorator.enabled=false
```

introscope.agent.decorator.security

此属性确定发送到 CA CEM 的已修饰 HTTP 响应标头的格式。

设置

- clear－明文编码
- encrypted－标头数据已加密

默认

clear

示例

```
introscope.agent.decorator.security=clear
```

注释

默认设置 clear 适用于初始测试。但是，此设置可能会泄露不希望防火墙之外的用户知道的事务标头中的信息。请将该属性设置为 encrypted，以实现更安全的生产环境。

要将此属性设置为 encrypted，请使用支持的 JVM。

注意：有关 JVM 支持信息，请参阅《*兼容性指南*》。

introscope.agent.cemtracer.domainconfigfile

用于指定 CA CEM 业务服务和事务层次结构的 CA CEM 域配置文件的名称。CEMTracer 会在其安装目录中查找具有该名称的文件。

每当 CA CEM 管理员单击 CA CEM 中的“同步所有监视器”时，都会将域配置文件推送到企业管理器，然后企业管理器再转而将文件推送到每个连接的代理。

有关特定于 CA CEM 版本的信息，请参阅**注释**（见下文）。

设置

可以是任何有效的文件名。

默认

```
domainconfig.xml
```

示例

```
introscope.agent.cemtracer.domainconfigfile=domainconfig.xml
```

注释

此属性适用于 CA CEM 4.0 和 4.1 版本。仅当您已在“Introscope 设置”页面上选择“CEMTracer 4.0/4.1 支持”选项时，此选项才适用于 CA CEM 4.2/4.5。

通过“CEMTracer 4.0/4.1 支持”选项，可以随时将您的代理从 4.0 或 4.1 交错迁移到 4.2/4.5；请仅在需要时使用。

- 如果代理未连接到企业管理器，则不会发送域配置文件。

- 如果代理目录为只读，则无法写入域配置文件。
- 如果未在代理上启用 CEMTracer 4.0/4.1，则发送域配置文件后不会对其执行任何操作。

introscope.agent.cemtracer.domainconfigfile.reloadfrequencyinminutes

代理重新加载域配置文件的频率（以分钟为单位）。（4.0/4.1 代理不会在每次企业管理器发送域配置文件后自动重新加载。如果域配置文件没有更改，则代理不会进行重新加载。）

有关特定于 CA CEM 版本的信息，请参阅**注释**（见下文）。

设置

数值

默认

1

示例

```
introscope.agent.cemtracer.domainconfigfile.reloadfrequencyinminutes=1
```

注释

此属性适用于 CA CEM 4.0 和 4.1 版本。仅当您已在“Introscope 设置”页面上选择“CEMTracer 4.0/4.1 支持”选项时，此选项才适用于 CA CEM 4.2/4.5。

通过“CEMTracer 4.0/4.1 支持”选项，可以随时将您的代理从 4.0 或 4.1 交错迁移到 4.2；请仅在需要时使用。

introscope.agent.distribution.statistics.components.pattern

要从 BlamePointTracers 收集响应时间分布信息，请取消注释并编辑此属性。您可以使用此响应时间信息来创建平均响应时间度量标准。

有关更多信息：

如何将代理配置为收集分布统计信息度量标准 (p. 56)

ChangeDetector 配置

您可以控制本地代理与 ChangeDetector 配合使用的方式。

注意：有关使用 ChangeDetector 的详细信息，请参阅《*CA APM ChangeDetector 用户指南*》。

introscope.changeDetector.enable

指定是启用还是禁用 ChangeDetector。将该属性设置为 true 可启用 ChangeDetector。默认情况下该属性会注释掉，并设置为 false。如果启用 ChangeDetector，还应设置与 ChangeDetector 相关的其他属性。

属性设置

True 或 False

默认

False

示例

```
introscope.changeDetector.enable=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.changeDetector.agentID

指定 ChangeDetector 用来识别本地代理的文本字符串。默认情况下，此属性注释掉。如果启用 ChangeDetector，应取消注释此属性并将其设置为相应的值。

默认

默认值为 *SampleApplicationName*。

示例

```
introscope.changeDetector.agentID=SampleApplicationName
```

introscope.changeDetector.rootDir

为 ChangeDetector 文件指定根目录。根目录是 ChangeDetector 在其中创建本地缓存文件的文件夹。

属性设置

文本字符串形式的 ChangeDetector 文件根目录完整路径。

默认

默认路径为 *c://sw//AppServer//wily//change_detector*。

示例

```
introscope.changeDetector.rootDir=c://sw//AppServer//wily//change_detector
```

注释

请使用反斜线转义反斜线字符，如示例中所示。

introscope.changeDetector.isengardStartupWaitTimeInSec

指定启动代理后等待 ChangeDetector 尝试连接到企业管理器的时间。默认情况下，此属性注释掉。

默认

默认值为 15 秒。

示例

```
introscope.changeDetector.isengardStartupWaitTimeInSec=15
```

introscope.changeDetector.waitTimeBetweenReconnectInSec

指定 ChangeDetector 在重新尝试连接到企业管理器之前等待的秒数。默认情况下，此属性注释掉。

默认

默认值为 10 秒。

示例

```
introscope.changeDetector.waitTimeBetweenReconnectInSec=10
```

introscope.changeDetector.profile

指定 ChangeDetector 数据源配置文件的绝对路径或相对路径。默认情况下，此属性注释掉。

默认

默认值为 ChangeDetector-config.xml。

示例

```
introscope.changeDetector.profile=CDConfig\\changedetector-config.xml
```

注释

请使用反斜线转义反斜线字符，如示例中所示。

introscope.changeDetector.profileDir

指定包含数据源配置文件的目录的绝对路径或相对路径。如果设置了此属性，除了使用 *introscope.changeDetector.profile* 属性指定的任何文件以外，还会使用此目录中的所有数据源配置文件。默认情况下，此属性注释掉。

默认

默认值为 changeDetector_profiles。

示例

```
introscope.changeDetector.profileDir=c:\\CDconfig\\changeDetector_profiles
```

注释

使用反斜杠来转义反斜杠字符。

跨进程事务跟踪

您可以启用对由于存在尾部筛选器而产生的下游跟踪的自动收集。启用此属性并对尾部筛选器长时间运行事务跟踪会话，会导致将大量不需要的跟踪发送给企业管理器。

introscope.agent.transactiontracer.tailfilterPropagate.enable

控制尾部筛选的存在是否会触发从下游代理自动收集跟踪。此属性不影响对由于通过了头部筛选器而产生的自动下游跟踪的收集。

属性设置

True 或 False

默认

false；注释掉。

示例

```
introscope.agent.transactiontracer.tailfilterPropagate.enable
=false
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

默认域配置

您可以控制默认域如何连接到企业管理器。

introscope.agent.dotnet.enableDefaultDomain

确定已连接到默认域的代理是否连接到了企业管理器。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.dotnet.enableDefaultDomain=false
```

注释

您必须将此属性添加到 *IntroscopeAgent.profile* 以将其启用。

当此属性设置为 true 时，监控默认域的代理也将在调查器中进行报告。

动态检测

您可以动态检测类和方法，而无需写入自定义 PBD、重新启动应用程序服务器或重新启动代理。

注意：有关从 Introscope Workstation 使用事务跟踪和动态检测的详细信息，请参阅《*CA APM Workstation 用户指南*》。

introscope.agent.remoteagentdynamicinstrumentation.enabled

此属性启用或禁用动态检测的远程管理。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.remoteagentdynamicinstrumentation.enabled=true
```

注释

- 重新启动托管应用程序，以使更改生效。
- 动态检测是一个占用大量 CPU 的操作。请使用可以最大程度地减少要检测的类的配置。
- 如果已启用动态检测，则无法在同一进程内监控多个 CLR（比如无法进行进程内并行执行）。有关详细信息，请参阅 “com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs (p. 188)”属性。

ErrorDetector

您可以控制代理与 ErrorDetector 的交互方式。

introscope.agent.errorsnapshots.enable

使代理能够捕获有关严重错误的事务详细信息。默认情况下，代理上安装有 Introscope ErrorDetector。该属性必须设置为 true，才能查看错误快照。

属性设置

True 或 False

默认

True

注释

此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.errorsnapshots.throttle

指定代理在 15 秒时间内可以发送的最大错误快照数目。

默认

10

示例

```
introscope.agent.errorsnapshots.throttle=10
```

注释

此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.errorsnapshots.ignore.<index>

指定一个或多个错误消息筛选。通过在属性名称后附加索引标识符（例如 .0、.1、.2...），可以根据需要指定任意数量的筛选。您可以使用通配符 (*)。将忽略满足您指定的条件的错误消息。对于符合您所定义的筛选的错误，不生成任何错误快照；也不会将有关这些错误的任何错误事件发送到企业管理器。

重要信息！ 此属性无法用于筛选 SOAP 错误消息。

默认

提供了示例定义，并注释掉，如下所示。

示例

```
introscope.agent.errorsnapshots.ignore.0=*com.company.HarmlessException*
introscope.agent.errorsnapshots.ignore.1=*HTTP Error Code: 404*
```

注释

此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

扩展

您可以配置代理扩展的位置。

introscope.agent.extensions.directory

指定代理加载的所有扩展的位置。您可以指定该目录的绝对路径或相对路径。如果您不指定绝对路径，则您指定的值将相对于 *IntroscopeAgent.properties* 文件位置进行解析。

默认

默认位置为 *<Agent_Home>*/ext 目录中的 *ext* 目录。

示例

```
introscope.agent.extensions.directory=../ext
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

已筛选的参数

您可以启用对已筛选参数的收集。

introscope.agent.asp.disableHttpProperties

默认情况下，.NET 代理仅报告它所跟踪的事务的 URL。将限制对各个 HTTP 属性的报告，以将事务跟踪对系统开销的影响降到最小。

要启用筛选，必须首先通过将属性设置为 false 来启用 HTTP 属性收集。

启用对以下项的收集：应用程序名称、会话 ID、上下文路径、服务器名称、URL、规范化的 URL、HTTP 标头、HTTP 参数、HTTP 属性

属性设置

选项

True 或 False

默认

False

示例

```
#introscope.agent.asp.disableHttpProperties=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

HTTP Header Decorator

您可以启用 Servlet Header Decorator，它是 CA CEM 集成解决方案的一部分。

introscope.agent.decorator.enabled

如果此布尔值设置为 true，它会将代理配置为将其他性能监控信息添加到 HTTP 响应标头中。

HTTPHeaderDecorator 可将 GUID 附加到每个事务，并将 GUID 插入 HTTP 标头 x-wily-info。这将启用 CA CEM 与 Introscope 之间的事务关联。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.decorator.enabled=false
```

LeakHunter 配置

您可以控制代理与 LeakHunter 的交互方式。

introscope.agent.leakhunter.enable

控制是否启用 LeakHunter 功能。将值设置为 true 可启用 LeakHunter。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.leakhunter.enable=false
```

注释

- 启用该选项可能会导致 CPU 和内存使用率较高。仅当其他度量标准表明存在内存泄漏时才应启用该功能。
- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.leakhunter.logfile.location

控制 *LeakHunter.log* 文件的位置。可以指定文件名的绝对路径或相对路径。相对路径相对于 *<Agent_Home>* 目录进行解析。如果不希望 LeakHunter 将数据记录到日志文件中，请将值保留为空白或注释掉。

默认

默认路径为 *../../logs/LeakHunter.log*。将在 *<Agent_Home>logs* 目录中找到日志文件。

示例

```
introscope.agent.leakhunter.logfile.location=../../logs/LeakHunter.log
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.leakhunter.logfile.append

指定在应用程序重新启动时替换日志文件还是向现有日志文件中添加信息。

属性设置

True 或 False

- 设置为 False 将替换日志文件。
- 设置为 True 将向现有日志文件中添加信息。

默认

False

示例

```
introscope.agent.leakhunter.logfile.append=false
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.leakhunter.leakSensitivity

控制 LeakHunter 泄漏检测算法的敏感度级别。较高的敏感度设置将导致报告较多的潜在泄漏，较低的敏感度设置则导致报告较少的潜在泄漏。

属性设置

泄漏敏感度范围必须是 1（低）到 10（高）之间的正整数值。

默认

5

示例

```
introscope.agent.leakhunter.leakSensitivity=5
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.leakhunter.timeoutInMinutes

控制 LeakHunter 寻找新的潜在泄漏所用的时间长度（以分钟为单位）。在指定的时间过后，LeakHunter 将停止寻找新的潜在泄漏。它将继续跟踪以前识别的潜在泄漏。

属性设置

必须是正整数（不能是负数）。

默认

默认值为 120 分钟。

示例

```
introscope.agent.leakhunter.timeoutInMinutes=120
```

注释

- 如果希望 LeakHunter 始终寻找新的潜在泄漏，请将值设置为零。
- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.leakhunter.collectAllocationStackTraces

控制 LeakHunter 是否为潜在泄漏生成分配堆栈跟踪。将该属性设置为 *true* 将获得更精确的有关潜在泄漏分配的数据，但是需要额外的内存和 CPU 开销。因此，默认设置为 *false*。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.leakhunter.collectAllocationStackTraces=
false
```

注释

- 将该属性设置为 *true* 可能会在 CPU 使用和内存方面造成较高的系统开销。
- 此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.leakhunter.ignore.0

在检查潜在泄漏时，可以使用一个或多个忽略属性指定希望 LeakHunter 忽略的特定集合。对于常规集合，请使用包含常规类型限定的语法，例如：*system.Collections.Generic.List`1* 。满足您所指定条件的集合将被忽略。

属性设置

以逗号分隔的类匹配模式列表。

默认

无

示例

```
#introscope.agent.leakhunter.ignore.0=
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

日志记录

您可以使用属性来配置日志记录。

introscope.agent.log.config.path

此属性指向 Log4Net 配置文件。

属性设置

默认

`logging.config.xml`

示例

```
introscope.agent.log.config.path=logging.config.xml
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

NativeProfiler

您可以在 *IntroscopeAgent.profile* 中设置以下属性来控制 NativeProfiler 操作。这些属性可以控制对使用本机映像生成器创建的应用程序的监控、类名内存中缓存的大小以及 NativeProfiler 日志文件的位置和内容。

introscope.nativeprofiler.clrv4.transparency.checks.disabled

.NET 4 CLR 会对透明程序集进行检查，这些程序集可通过探查器使检测代码失效。要阻止这些检查，请将此属性的值设为 "true"。

属性设置

true 或 false

默认

true

示例

```
#introscope.nativeprofiler.clrv4.transparency.checks.disable=
true
```

注意：必须重置 IIS 才能让该值生效。

introscope.nativeprofiler.logfile

设置记录 NativeProfiler 读取的指令以及 NativeProfiler 检测的方法等相关信息的日志文件的路径。

属性设置

文件位置的绝对或相对路径。

默认

logs/nativeprofiler.log

示例

```
introscope.nativeprofiler.logfile=logs/nativeprofiler.log
```

注释

NativeProfiler 会记录配置其读取的 PBD 和 PBL 中处于活动状态的所有指令的相关信息，以及记录所检测的特定方法以供您参考。

introscope.nativeprofiler.logBytecode

确定 NativeProfiler 是否列出了已检测的字节码。如果设置为 true，NativeProfiler 日志文件会列出已检测的字节码。默认情况下，此属性设置为 false 且已注释掉。

属性设置

True 或 False

默认

False

示例

```
#introscope.nativeprofiler.logBytecode=false
```

注释

默认情况下，此属性注释掉。

introscope.nativeprofiler.logAllMethodsNoticed

如果启用此属性，则会记录 NativeProfiler 发现的所有方法，包括未检测的方法。

属性设置

True 或 False

默认

False

示例

```
introscope.nativeprofiler.logAllMethodsNoticed=false
```

注释

默认情况下处于禁用状态。

introscope.nativeprofiler.directivematching.cache.max.size

指定代理可以存储在内存中缓存的类名的最大数。默认情况下，代理会创建先前发现的指令组的内存缓存，其中包含要监控的类。每次启动 IIS 时，代理都会创建先前发现的类的缓存。缓存会随时间而增加，因为要监控应用程序代码所使用的新类。默认情况下，内存缓存最多可存储 5000 个类名。

如果缓存必须存储 5000 个以上的类名，则增大此属性的值可以改进启动时间。但是，增大该值会增加代理所需的内存开销。减小该属性值可减少代理的内存开销。如果监视少于 5000 个类，则可以适当地减小该值。

注意：缓存不能存储类对象。

默认

默认情况下，缓存最多可以存储 5000 个类名。

示例

```
introscope.nativeprofiler.directivematching.cache.max.size=5000
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.nativeprofiler.generic.agent.trigger.enabled

借助此属性，可使用常规触发器启动 .NET 代理。如果启用此属性，则会在第一个用户代码执行之后，使用常规触发器启动该代理。此操作将启动可以限制的默认域探测。如果禁用，则在 IIS 运行时，会使用 main 方法或 start-up 方法启动该代理。

属性设置

True 或 False

默认

False

示例

```
introscope.nativeprofiler.generic.agent.trigger.enabled=true
```

注释

- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。
- 在 ProcSxS 中，代理名称显示如下：
  `<ProcessName>_<ApplicationDomainName>`

com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs

控制代理如何处理对公共语言运行时的多个版本的监控。默认情况下，代理仅基于首先加载的 CLR 版本监控 .NET Framework 的一个版本。使用该属性可修改默认行为，并指定是监控 .NET Framework 2.0 组件还是监控 .NET Framework 4 组件。

属性设置

有效值包括：

- **无**—指定要使用默认行为，并选择基于首先加载的 CLR 进行监控的 .NET Framework 版本。
- **V2**—指定仅监控 .NET Framework 2.0 组件。或者，可以指定 CLR 版本号，例如 v2.0.50727。
- **V4**—指定仅监控 .NET Framework 4 组件。或者，可以指定 CLR 版本号，例如 v4.0.30319。
- **默认值**—默认值为“无”。该值表示仅监控 .NET Framework 的一个版本。首先加载的应用程序确定将监控 .NET Framework 的哪个版本。

注意：不能同时监控 .NET Framework 2.0 和 .NET Framework 4 组件。启用动态检测时，此功能无法使用。

示例

```
com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs=v2
```

注释

必须重新启动应用程序，对该属性所做的更改才能生效。

性能监视器数据收集

您可以配置属性来控制从性能监视器计数器收集数据。

introscope.agent.perfmon.metric.filterPattern

指定标识要监控的性能监视器计数器的正则表达式。

属性设置

用于匹配性能监视器计数器名称的文本字符串正则表达式。要添加新的计数器（如特定于进程的线程度量标准），请将新的表达式添加到列表的末尾并以逗号分隔。例如：*. . . , /Thread/{osprocessname}*/**

默认

默认筛选器将监控处理器、.NET Data Provider、.NET CLR 以及 ASP.NET 性能监视器计数器、对象和实例。

示例

```
introscope.agent.perfmon.metric.filterPattern=|Processor|*|*, |
.NET Data Provider*|*|*, |.NET CLR*|{osprocessname}|*, |.NET CLR
Data|*|*, |Process|{osprocessname}|*, |ASP.NET|*
```

注释

使用筛选器 |*|* 相当于要求性能监视器将所有计数器枚举为缺少实例的计数器。该设置对某些计数器无效。

introscope.agent.perfmon.metric.limit

指定每个时间间隔内可以报告的性能监视器度量标准的最大数目。

默认

默认值为 1000 个度量标准。

示例

```
introscope.agent.perfmon.metric.limit=1000
```

introscope.agent.perfmon.metric.pollIntervalInSeconds

指定性能监视器收集代理检查性能监视器对象、计数器和实例中新的度量标准值的频率。默认轮询时间间隔为每 15 秒检查一次所有度量标准值。

默认

默认时间间隔是 15 秒。

示例

```
introscope.agent.perfmon.metric.pollIntervalInSeconds=15
```

introscope.agent.perfmon.category.browseEnabled

启用或禁用新性能监视器计数器的发现功能。

属性设置

True 或 False

默认

True（启用）

示例

```
introscope.agent.perfmon.category.browseEnabled=true
```

introscope.agent.perfmon.category.browseIntervalInSeconds

确定性能监视器收集代理检查要发现的新性能监视器对象的频率。

属性设置

代表时间间隔秒数的正整数。

默认

默认时间间隔为 600 秒（10 分钟）。

示例

```
introscope.agent.perfmon.category.browseIntervalInSeconds=600
```

进程名称

您可以配置属性以定义进程名称。

introscope.agent.customProcessName

当进程显示在企业管理器和工作站中时，指定进程名称。取消注释此属性并定义进程名称，从而在企业管理器和工作站中显示自定义名称。

默认

CustomProcessName

示例

```
#introscope.agent.customProcessName=CustomProcessName
```

注释

- 默认情况下，此属性注释掉。
- 您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.defaultProcessName

如果没有定义任何自定义进程名称，并且代理无法确定主应用程序类的名称，将使用默认进程名称。

默认

.NET Process

示例

```
introscope.agent.defaultProcessName=.NET Process
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

限制检测配置

您可以配置对进程或可执行文件目标组的检测。

注意：有关限制检测的详细信息，请参阅“配置检测”。

introscope.agent.dotnet.monitorApplications

指定要检测的进程和应用程序。

选项

不支持完全限定路径。请仅使用应用程序可执行文件的进程映像名称。名称区分大小写，且必须与报告给 Windows 进程管理的名称精确匹配。

默认

```
w3wp.exe,aspnet_wp.exe
```

示例

```
introscope.agent.dotnet.monitorApplications=w3wp.exe,aspnet_wp.exe,dllhost.exe
```

introscope.agent.dotnet.monitorAppPools

指定要检测的应用程序池。

选项

取消注释此属性，并指定要检测的 IIS 应用程序池。

默认

"NULL","DefaultAppPool","AppPool1","AppPool2"

示例

```
#introscope.agent.dotnet.monitorAppPools=
"NULL","DefaultAppPool","AppPool1","AppPool2"
```

注释

- 应用程序池名称必须用引号引起来。使用 "NULL" 可指定未在任何应用程序池中运行的应用程序。
- 保留已注释状态可检测所有应用程序池，或者，保留取消注释状态并仅列出想要检测的应用程序池。

套接字度量标准

以下属性控制套接字度量标准的生成。

introscope.agent.sockets.reportRateMetrics

启用对各个套接字输入/输出带宽速率度量标准的报告。

属性设置

True 或 False

默认

True

示例

```
introscope.agent.sockets.reportRateMetrics=true
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

SQL 代理

以下属性适用于 SQL 代理。

introscope.agent.sqlagent.sql.maxlength

限制显示在 SQL 代理度量标准的调查器树中的 SQL 语句的大小（以字节为单位）。

默认

默认限制为 990 字节。

示例

```
introscope.agent.sqlagent.sql.maxlength=990
```

注释

默认情况下，该属性不在 *IntroscopeAgent.profile* 中显示。要更改该值，请将该属性添加到代理配置文件中。

introscope.agent.sqlagent.normalizer.extension

此属性可指定用于覆盖预配置规范化方案的 SQL 规范化程序扩展的名称。

要使自定义规范化扩展生效，其清单属性 *com-wily-Extension-Plugin-{pluginName}-Name* 的值必须与此属性中给定的值匹配。

如果您指定逗号分隔的名称列表，代理将使用默认规范化程序扩展。

例如，根据以下内容设置，RegexSqlNormalizer 可用于规范化：

```
introscope.agent.sqlagent.normalizer.extension=ext1, ext2
```

此属性可限制将 SQL 语句中的多少内容（字节）显示在 SQL 代理度量标准的调查器树中。

属性设置

用于覆盖预配置规范化方案的 SQL 规范化程序扩展的名称。

默认

RegexSqlNormalizer

示例

```
introscope.agent.sqlagent.normalizer.extension=RegexSqlNormalizer
```

注释

如果使用默认设置，您还必须配置正则表达式 SQL 语句规范化程序属性：

- introscope.agent.sqlagent.normalizer.regex.matchFallThrough (see page 201)
- introscope.agent.sqlagent.normalizer.regex.keys (see page 197)
- introscope.agent.sqlagent.normalizer.regex.key1.pattern (see page 198)
- introscope.agent.sqlagent.normalizer.regex.key1.replaceAll (see page 199)
- introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat (see page 200)
- introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive (see page 201)

对此属性所做的更改将即时生效，无需重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.keys

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。此属性用于指定正则表达式组键。按顺序对其进行评估。

默认

key1

示例

```
introscope.agent.sqlagent.normalizer.regex.keys=key1
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.key1.pattern

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。此属性用于指定将用于与 SQL 匹配的正则表达式模式。

属性设置

此处可使用 *java.util.Regex* 包允许的所有有效正则表达式条目。

默认

.*call(.*\)\.FOO(.*\)

示例

```
introscope.agent.sqlagent.normalizer.regex.key1.pattern=.*call(.*\)\.FOO(.*\)
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.key1.replaceAll

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。如果该属性设置为 false，它会将 SQL 查询中首次出现的匹配模式替换为替换字符串。如果设置为 true，它将使用替换字符串替换 SQL 查询中匹配模式的所有实例。

属性设置

True 或 False

默认

false

示例

```
introscope.agent.sqlagent.normalizer.regex.key1.replaceAll=false
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。此属性指定替换字符串的格式。

属性设置

java.util.Regex 包和 *java.util.regex.Matcher* 类允许的所有有效正则表达式条目均可以在此处使用。

默认

$1

示例

```
introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat
=$1
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。此属性指定模式匹配是否区分大小写。

属性设置

true 或 false

默认

false

示例

```
introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive
=false
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.normalizer.regex.matchFallThrough

将此属性与 introscope.agent.sqlagent.normalizer.extension (see page 195) 结合使用，可设置正则表达式 SQL 语句规范化程序。此属性设置为 true 时，它将根据所有正则表达式键组评估 SQL 字符串。

实施是连锁的。例如，如果 SQL 匹配多个键组，则 group1 的规范化 SQL 输出将作为 group2 的输入，依此类推。

如果该属性设置为 false，只要键组匹配，就会返回来自该组的规范化 SQL 输出。

属性设置

True 或 False

默认

false

示例

```
introscope.agent.sqlagent.normalizer.regex.matchFallThrough=false
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.sqlagent.sql.artonly

introscope.agent.sqlagent.sql.artonly 属性可用于将代理配置为只创建和发送平均响应时间度量标准。该属性可影响后端下的所有 SQL 代理度量标准。该属性的值为 true 时，可以改善代理在 SQL 度量标准和事务跟踪方面的性能。

注意：设置 introscope.agent.sqlagent.sql.turnoffmetrics (see page 203)=true 将覆盖该属性。

重要信息！必须设置以下跟踪器参数，此属性设置才能正常工作：

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

默认情况下，该属性处于关闭状态：

```
introscope.agent.sqlagent.sql.artonly=false
```

对该属性所做的更改将立即生效，可以使用“管理”界面进行更改。

注意：该属性无法控制自定义度量标准，如连接计数。

introscope.agent.sqlagent.sql.rawsql

introscope.agent.sqlagent.sql.rawsql 属性可将代理配置为将*未规范化的* SQL 作为 SQL 组件的参数添加到事务跟踪。该属性的值为 true 时，可以改善代理在 SQL 度量标准和事务跟踪方面的性能。

默认情况下，该属性处于关闭状态：

introscope.agent.sqlagent.sql.rawsql=false

对该属性所做的更改将在重新启动托管应用程序之后生效。

重要信息！启用该属性会导致事务跟踪中显示密码和敏感信息。

introscope.agent.sqlagent.sql.turnoffmetrics

可以使用 introscope.agent.sqlagent.sql.turnoffmetrics 属性关闭 SQL 语句度量标准，从而将较少的度量标准从代理发送到企业管理器。该属性的值为 true 时，可以改善代理在 SQL 度量标准和事务跟踪方面的性能。

重要信息！必须设置以下跟踪器参数，此属性设置才能正常工作：

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

默认情况下，该属性处于关闭状态：

```
introscope.agent.sqlagent.sql.turnoffmetrics=false
```

此属性将覆盖 introscope.agent.sqlagent.sql.artonly 属性。

对该属性所做的更改将立即生效，可以使用“管理”用户界面进行更改。

introscope.agent.sqlagent.sql.turnofftrace

introscope.agent.sqlagent.sql.turnofftrace 属性可控制代理是否针对后端下的 SQL 语句创建事务跟踪组件，并将其发送给企业管理器。该属性的值为 true 时，可以改善代理在 SQL 度量标准和事务跟踪方面的性能。

重要信息！必须设置以下跟踪器参数，此属性设置才能正常工作：

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

默认情况下，该属性处于关闭状态：

```
introscope.agent.sqlagent.sql.turnofftrace=false
```

对该属性所做的更改将立即生效，可以使用“管理”用户界面进行更改。

停顿度量标准

以下部分定义了与停顿度量标准有关的属性。

introscope.agent.stalls.thresholdseconds

该属性指定执行过程被视为停顿过程之前的秒数。要确保准确的“停顿计数”度量标准，请将停顿阈值设置为 15 秒或更多。此设置允许企业管理器完成其搜集周期。

默认

默认值为 30 秒。

示例

```
introscope.agent.stalls.thresholdseconds=30
```

注释

此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.stalls.resolutionseconds

该属性指定代理检查停顿的频率。要确保准确的“停顿计数”度量标准，请勿将停顿解析度设置为少于 10 秒。此设置允许企业管理器完成其搜集周期。

默认

默认为每 10 秒。

示例

```
introscope.agent.stalls.resolutionseconds=10
```

注释

此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

事务跟踪

以下属性适用于事务跟踪。

introscope.agent.crossprocess.compression

使用此属性可减小跨进程事务跟踪数据的大小。

属性设置

lzma、gzip、none

默认

lzma

示例

```
introscope.agent.crossprocess.compression=lzma
```

注释

- 此选项将增加代理 CPU 开销，但会减小跨进程标头的大小。
- *lzma* 压缩比 *gzip* 压缩更加高效，但可能会占用更多的 CPU。
- .NET 代理不支持 *gzip* 选项，因此，如需具备互操作性，请不要使用 *gzip*。
- 此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.crossprocess.correlationid.maxlimit

允许的跨进程参数数据的最大大小。

如果跨进程参数数据的总大小超过了此限制，即使应用了压缩，一些数据仍将被丢弃，并且某些跨进程关联功能将无法正常工作。

但是，此设置可以避免用户事务由于标头太大而导致网络传输失败。

默认

4096

示例

```
introscope.agent.crossprocess.correlationid.maxlimit=4096
```

注释

- 与上述 *introscope.agent.crossprocess.compression* 和 *introscope.agent.crossprocess.compression.minlimit* 属性一起使用
- 此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.crossprocess.compression.minlimit

使用此属性可设置要应用压缩的跨进程参数数据的最小长度。

属性设置

可以设置的值范围是 0 到 introscope.agent.crossprocess.correlationid.maxlimit (see page 207) 中设置的最大总限制的两倍。

如果设置为小于默认值 1500，则压缩将更加频繁地运行并消耗更多 CPU。在正常情况下，默认设置 1500 通常不会对 CPU 开销造成影响。

默认

1500

示例

```
introscope.agent.crossprocess.compression.minlimit=1500
```

注释

- 与上面的 introscope.agent.crossprocess.compression 属性一起使用。
- 此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。

introscope.agent.transactiontrace.componentCountClamp

限制事务跟踪中允许的组件数量。

默认

5000

重要信息！如果限定大小增加，则需要更多内存。在极个别情况下，可能需要调整 JVM 的最大堆大小，否则托管应用程序可能会耗尽内存。

示例

```
introscope.agent.transactiontrace.componentCountClamp=5000
```

注释

- 将在代理中丢弃任何超出限定值的事务跟踪，并在代理日志文件中记录一条警告消息。
- 此属性是动态属性。您可以在运行时更改此属性的配置，更改会自动生效。
- 到达设置的限制时，日志中将显示警告，跟踪将停止。
- 零**不是**有效值。请**不要**设置 introscope.agent.transactiontrace.componentCountClamp=0。

introscope.agent.transactiontrace.headFilterClamp

指定标头筛选中允许的最大组件深度。标头筛选会检查事务的开头，以便潜在地收集整个事务。标头筛选将检查每个组件，直到第一个出现问题的组件退出。如果不进行限定，在具有非常深的调用堆栈的事务中会出现问题。通过强制代理仅查询到某个固定深度，限定值可以限制该行为对内存和 CPU 使用率的影响。

默认

30

警告！如果限定大小增加，则需要更多内存。将影响垃圾回收行为，从而在应用程序范围内对性能造成影响。

示例

```
introscope.agent.transactiontrace.headFilterClamp=30
```

注释

- 对此属性所做的更改可立即生效，不需要重新启动托管应用程序。
- 将不再检查深度超过限定的事务跟踪是否存在收集；*除非*可使用一些其他机制（如采样或用户启动事务跟踪）选择要收集的事务。

introscope.agent.transactiontracer.parameter.httprequest.headers

指定要捕获的 HTTP 请求标头数据（在一个逗号分隔列表中）。使用逗号分隔列表。

默认

注释掉；*User-Agent*

示例

```
introscope.agent.transactiontracer.parameter.httprequest.headers=User-Agent
```

注释

IntroscopeAgent.profile 包含将此属性的值设为空值的已注释掉的语句。用户可以选择性地取消注释该语句并提供所需的标头名称。

introscope.agent.transactiontracer.parameter.httprequest.parameters

指定要捕获的 HTTP 请求参数数据（在一个逗号分隔列表中）。

默认

注释掉；常规参数。

示例

```
introscope.agent.transactiontracer.parameter.httprequest.parameters=parameter1,parameter2
```

注释

IntroscopeAgent.profile 包含将此属性的值设为空值的已注释掉的语句。用户可以选择性地取消注释该语句并提供所需的参数名称。

introscope.agent.transactiontracer.parameter.httpsession.attributes

指定要捕获的 HTTP 会话属性数据（在一个逗号分隔列表中）。

默认

注释掉；常规参数。

示例

```
introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute1,attribute2
```

注释

IntroscopeAgent.profile 包含将此属性的值设为空值的已注释掉的语句。用户可以选择性地取消注释该语句并提供所需的参数名称。

introscope.agent.transactiontracer.sampling.enabled

取消注释以下属性可禁用事务跟踪器采样。

属性设置

True 或 False

默认

False

示例

```
introscope.agent.transactiontracer.sampling.enabled=false
```

注释

对此属性所做的更改可立即生效，不需要重新启动托管应用程序。

introscope.agent.transactiontracer.sampling.perinterval.count

此属性通常在企业管理器中进行配置。在代理中配置此属性将禁用在企业管理器中的配置。请参阅*《CA APM 配置和管理指南》*以获取更多信息。

默认

1

示例

```
introscope.agent.transactiontracer.sampling.perinterval.count
=1
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

introscope.agent.transactiontracer.sampling.interval.seconds

此属性通常在企业管理器中进行配置。在代理中配置此属性将禁用在企业管理器中的配置。

注意：有关详细信息，请参阅*《CA APM 配置和管理指南》*。

默认

120

示例

```
introscope.agent.transactiontracer.sampling.interval.seconds=
120
```

注释

您必须重新启动托管应用程序，对此属性所做的更改才能生效。

配置会话 ID 收集

introscope.agent.transactiontracer.parameter.capture.sessionid 属性可启用或禁用 TransactionTracer 数据中会话 ID 的收集。默认情况下，在 TransactionTracer 数据中启用并记录该属性。禁用该属性时，无法在筛选中使用数据。

请执行以下步骤：

1. 在文本编辑器中打开 IntroscopeAgent.profile 文件。

   查找下列行：

   ```
   # Uncomment the following property to disable sessionid
   capture in TransactionTracer data.
   # By default, it is enabled and recorded in the TT Data.

   #
   introscope.agent.transactiontracer.parameter.capture.sess
   ionid=true
   ```

2. 按照说明，通过注释或取消注释以下行来启用或禁用该属性：

```
#
introscope.agent.transactiontracer.parameter.capture.sessionid=true
```

3. 保存并关闭文件，然后重新启动代理。

   代理配置设置为使用为收集会话 ID 而指定的值。

introscope.agent.transactiontracer.userid.key

用户定义的键字符串。

默认

注释掉；常规参数。

示例

```
#introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute1,attribute2
```

注释

IntroscopeAgent.profile 包含将此属性的值设为空值的已注释掉的语句。如果在您的环境中使用 *HttpServletRequest.getHeader* 或 *HttpServletRequest.getValue* 访问用户 ID，则用户可以选择取消注释该语句并提供正确的值。

更多信息请参阅 introscope.agent.transactiontracer.userid.method (see page 215)。

introscope.agent.transactiontracer.userid.method

指定返回用户 ID 的方法。代理配置文件针对三个允许值中的每一个都包含一个已注释掉的属性定义。

基于 getRemoteUser、getHeader 或 getValue 是否访问了用户 ID 来取消注释相应语句。

属性设置

可用的值有：

- *HttpServletRequest.getRemoteUser*
- *HttpServletRequest.getHeader*
- *HttpServletRequest.getValue*

默认

注释掉；请参见上面的选项。

示例

IntroscopeAgent.profile 包括三个允许值中每一个的已注释掉的属性定义。可以取消注释要使用的属性。

```
introscope.agent.transactiontracer.userid.method=HttpServletR
equest.getRemoteUser
#introscope.agent.transactiontracer.userid.method=HttpServlet
Request.getHeader
#introscope.agent.transactiontracer.userid.method=HttpSession
.getValue
```

URL 分组

这些属性可用于为前端度量标准配置 URL 组。

introscope.agent.urlgroup.keys

前端命名的配置设置。

默认

default

示例

```
introscope.agent.urlgroup.keys=default
```

注释

如果某个 URL 地址属于两个 URL 组，则 URL 组的键在此属性中的排列顺序很重要。狭义模式所定义的 URL 组应先于广义模式所指定的 URL 组。

例如，如果具有 alpha 键的 URL 组包含单个地址，而具有 beta 键的 URL 组包含网段上的所有地址（该网段包含第一个 URL 组中的地址），则 alpha 在 keys 参数中应先于 beta。

introscope.agent.urlgroup.group.default.pathprefix

前端命名的配置设置。

默认

*

示例

```
introscope.agent.urlgroup.group.default.pathprefix=*
```

introscope.agent.urlgroup.group.default.format

前端命名的配置设置。

默认

default

示例

```
introscope.agent.urlgroup.group.default.format=default
```

附录 B： 特定于应用程序的配置

.NET Framework 允许使用带 *.config* 扩展名的可选 XML 格式文件来配置特定于应用程序的参数。

添加配置属性

对于 ASP.NET 应用程序，*web.config* 是特定于应用程序的设置和配置的主文件。此文件存储在应用程序根目录中。

对于其他 .NET 可执行文件，配置文件的命名方式与应用程序相同，并附加一个 *.config* 扩展名。该文件与应用程序可执行文件存储在同一个目录中。例如，对于 *testapp.exe*，可选的配置文件为 *testapp.exe.config*。

可以将特定于 Introscope 的配置添加到 *.config* 文件。例如，通过设置参数，使各个应用程序可以引用各自的 *IntroscopeAgent.profile* 文件实例（允许不同的应用程序具有不同的代理配置），还可以为 Web 服务启用跨进程事务关联。

请执行以下步骤：

1. 打开应用程序配置文件。
2. 将 *sectionGroup* 和 *section* 添加到您的应用程序配置文件中。如下所示对其进行命名：

```
<configuration>
   <configSections>
      <sectionGroup>
         <sectionGroup name="com.wily.introscope.agent">
         <section
name="env.parameters"type="System.Configuration.NameValueSectionHandler"
/>
      </sectionGroup>
```

3. 将新属性添加到 *env.parameters* 部分。例如：

```
<com.wily.introscope.agent>
   <env.parameters>
      <add key="com.wily.introscope.agentProfile"
value="e:\\junkyard\\dotnettest\\Agent.profile" />
   </env.parameters>
</com.wily.introscope.agent>
```

有关示例，请参阅 *sample.exe.config*。

注意：安装 .NET 代理不需要配置文件。

附录 C： 使用网络接口实用工具

可以使用网络接口实用工具来确定代理用于 Catalyst 集成的主计算机的网络接口名称值。

此部分包含以下主题：

确定网络接口名称 (p. 221)

确定网络接口名称

网络接口实用工具为 introscope.agent.primary.net.interface.name 属性提供名称和子接口值。

请执行以下步骤：

1. 导航到以下目录：

   <代理主目录>\wily\tools

2. 执行 NetInterface.exe。

   浏览器会在“网络接口”选项卡上显示 .NET 支持的网络接口名称的列表。

详细信息：

配置可用网络的列表 (p. 154)