CA Application Performance Management

ChangeDetector 用户指南

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： CA Application Performance Management ChangeDetector 概览

通过 CA Application Performance Management ChangeDetector，CA Introscope 可以监控和报告应用程序文件和配置中的更改。 CA APM ChangeDetector 可以检测生产 Web 应用程序的更改，从而帮助您识别哪项更改是导致 Web 应用程序性能问题的根本原因。在识别导致问题的更改后，可以使用 CA APM ChangeDetector 来诊断该问题－通过 CA APM ChangeDetector，您可以看到代码、应用程序服务器配置和连接系统配置中可能导致性能问题的更改。

由于更改发生在应用程序的许多不同部分，因此，环境性能代理 (EPAgent) 也支持 CA APM ChangeDetector。 通过 EPAgent，可以收集来自几乎任何源的应用程序性能信息，从而监控特定于您环境的更改数据。 有关 EPAgent 的详细信息，请参阅 *《CA APM 环境性能代理实施指南》*。

本章介绍了 CA APM ChangeDetector 及其如何适用于 CA Introscope 环境，并提供了典型使用方案。

此部分包含以下主题：

关于本指南

本指南包含 CA APM ChangeDetector 图形用户界面组件的安装、配置、部署和使用信息。

除非特别说明，否则这些信息应该适用于所有平台。 例如，如果某部分只适用于 .NET 平台，它将包含一条如下所示的注释：

注意：本部分仅适用于 .NET 平台。

本指南中使用的目录命名约定

该指南对安装目录使用以下命名约定。

约定：

<EM_Home>

指代：

安装企业管理器的目录。 通常在 Program Files 目录下。

约定：

<Workstation_Home>

指代：

安装 Workstation 的目录。 通常在 Program Files 目录下。

约定：

<Agent_Home>

指代：

安装 CA Introscope 代理的目录。 通常是 CA APM 代理目录。

约定：

<ProductName_Home>

指代：

第三方产品或应用程序的安装目录。 例如，如果您使用的是 WebLogic，则是应用程序服务器的安装目录。

约定：

文件名<版本号><操作系统或其他标识符>.文件类型

指代：

包含特定标识信息的文件名。

例如，如果要从 UNIX 操作系统上 CA APM ChangeDetector 8.1 的 tar 包中提取文件，应下载以下文件：
ChangeDetector8.1.0.0.unix.tar

但是，本指南中将其显示为：
ChangeDetector<VersionNumber>.unix.tar

关于 CA APM ChangeDetector

CA APM ChangeDetector 是 CA Introscope 的一组扩展，可用来监控应用程序环境中的更改。 CA APM ChangeDetector 直接与 CA Introscope 集成，提供低开销的实时更改检测。 出现生产问题时，CA APM ChangeDetector 可以帮助 CA Introscope 用户将应用程序更改与应用程序性能的更改相关联，从而找出导致该问题的更改。

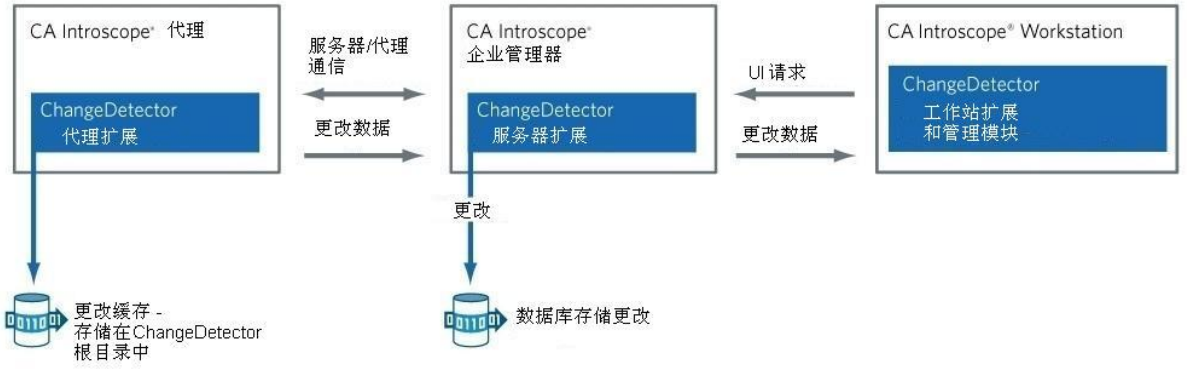
CA APM ChangeDetector 可以报告代理中的应用程序代码、配置和环境在多个时间段之间的差异，因此您可以看出应用程序实例在时间段之间（例如，今天和昨天或者今天和上周）的不同。

安装 CA APM ChangeDetector 后，您可以查看：

- 更改事件的图形视图，包括文件（文本和二进制文件）、存档、系统属性、应用程序代码、数据库表和特定数据库查询结果集。
- 更改事件的详细信息，包括检测到更改的时间、更改的特性以及文本文件的各版本之间的差异。
- CA APM ChangeDetector 显示板，显示更改数据的高级别视图。
- 更改的分层视图和历史视图（今天、昨天、上周）。
- 更改报告，汇总应用程序中的更改。

CA APM ChangeDetector 和您的 CA Introscope 环境

下图显示 CA APM ChangeDetector 组件与 CA Introscope 的交互方式：



CA APM ChangeDetector 使用方案

下列方案说明了如何使用 CA APM ChangeDetector 来检测更改，将更改关联到问题，确定更改的内容以及决定如何更正问题。

将更改归为导致问题的原因：星期一早上蓝色

星期一早上 9:00 点，银行的网上银行应用程序开始遇到性能问题。 CA Introscope 提醒 IT 团队，登录及其他关键事务响应时间降级并违反了 SLA 级别。 愤怒的用户开始打电话给客户服务中心，抱怨他们无法访问自己的银行帐户。 呼叫中心经理将问题上报给应用程序支持小组，该小组立即开始调查。

“更改了什么？ 应用程序上周工作正常。”

应用程序支持团队查看了他们的 CA Introscope 显示板以检查性能下降情况。 他们查看了性能图表－在安装 CA APM ChangeDetector 后，他们可查看直接集成到 CA APM ChangeDetector 图表中的更改数据。 团队检查图表上提供的更改标注和详细信息，注意到在当前性能问题发生之前，即上周末进行了一系列更改。 通过进一步调查，他们意识到更改是在周六晚上的应用程序维护窗口期间发生的。 团队浏览了 CA Introscope 调查器的树视图，检查各个应用程序实例的性能图；从相似的图表中清楚地看到每个实例大约在相同的时间经历了类似的一系列更改。

由于怀疑这些更改是导致问题的原因，应用程序支持团队更深入地调查。他们在 CA Introscope 调查器内单击应用程序的每个组件时，CA APM ChangeDetector 视图显示了每个组件发生的更改。 将范围锁定在应用程序文件时，他们注意到 37 个文件发生了更改，其中包括 3 个配置文件。CA APM ChangeDetector 帮助团队检验是否每个更改都是修改操作，是否每个更改都是在维护窗口期间检测到的。 通过选择其中一个应用程序服务器配置文件，他们将服务器上的文件的当前版本与更改前的上一版本进行了比较。

现在，原因很清楚。 有一个原先设置为 100 的数据库连接池参数现在更改为 10。 在获得此信息，应用程序支持经理告知开发经理发生了键入错误，并确认速度减慢是由于数据库连接中的意外更改而导致的。

前瞻性地检测有可能成为问题的更改：当场捕获

CA Introscope 报警响起。 通知应用程序支持团队 Java 应用程序发生更改。 因为不是预期的更改（即不在预定的维护窗口之内），所以应用程序支持经理开始调查其性能图。 应用程序目前工作正常，但是图表显示几分钟之前有少量更改。 将光标放在更改上将显示更改详细信息。 它们似乎是对系统变量的修改。

应用程序支持经理首先浏览了 CA Introscope 调查器，查找各个应用程序组件的更改。CA APM ChangeDetector 提供了应用程序更改的树视图。在检查更改时，应用程序支持经理发现应用程序文件和配置文件未发生任何更改，APM 数据库配置也未发生任何更改。然而，通过确认较早的 CA Introscope 报警，支持经理注意到 JVM 和应用程序服务器的四个系统属性发生了更改。

通过单击每个检测到的变量更改，应用程序支持经理查看了特定的变量名、检测到更改的时间以及更改的值。 Java 堆大小变量已降低，如果应用程序负载增加，可能会出现问题。 通过与 UNIX 系统管理员确认，得知已进行常规内部处理，并没有意识到更改的潜在影响。 通过将更改恢复为原始值，避免了一个潜在的性能问题。

第 2 章： 安装和配置 CA APM ChangeDetector

本章介绍了如何使用配置向导或者通过手工编辑 XML 配置文件来安装和配置 CA APM ChangeDetector 数据源。

此部分包含以下主题：

安装和启用 CA APM ChangeDetector

CA APM ChangeDetector 在代理安装期间安装并启用。可使用独立代理安装程序或使用无人值守模式安装选项安装 CA APM ChangeDetector。安装 CA APM ChangeDetector 后，必须对其进行配置。

遵循这些步骤:

1. 安装代理。

   有关安装代理的详细信息，请参阅 *《CA APM .Net 代理实施指南》* 或 *《CA APM Java 实施指南》*。默认情况下，CA APM ChangeDetector 处于启用状态。

a. 可以通过打开 *<EM_Home>/config* 目录中的 *IntroscopeEnterpriseManager.properties* 文件，将以下属性设置为 *false* 并重新启动企业管理器来启用 CA APM ChangeDetector：

```
introscope.changeDetector.disable=false
```

重要信息！ 如果您针对 WebSphere 6.1 或 7.0 应用程序运行 CA APM ChangeDetector，AIX 环境中的 Oracle 数据库可能收到类似如下的异常：

java.security.AccessControlException: Access denied (java.net.SocketPermission hostname:port connect,resolve) at java.security.AccessController.checkPermission(AccessController.java:104)

修改 *<WebSphere_Home>/properties/server.policy* 以授予 Java 安全权限，或者在 WebSphere 管理控制台中清除“启用应用程序安全性”复选框以防止出现该异常。

2. 配置 CA APM ChangeDetector。

在配置 CA APM ChangeDetector 之前，请先熟悉各种配置选项，然后参阅在配置 CA APM ChangeDetector 之前 (p. 14)。

注意： CA APM ChangeDetector 安装了一个本机库，用于报告文件所有者数据以及针对 Windows (cdnativefile.dll) 和 Linux (libcdnativefile.so) 操作系统已进行更改的数据。有关支持的平台版本的列表，请参阅 *《CA APM 兼容性指南》*。

在配置 CA APM ChangeDetector 之前

在配置 CA APM ChangeDetector 之前，请先熟悉各种配置选项，并选择最适合您环境的方法：

- 使用 CA APM ChangeDetector 配置向导 (p. 16)—使用该向导，可以通过在一系列引导您配置设置的页面上输入信息来配置 CA APM ChangeDetector。
- 修改 CA APM ChangeDetector 配置文件 (p. 25)—此方法适用于所有平台。 该方法允许您通过手工编辑 XML 文件来配置 CA APM ChangeDetector。如果您选择此方法，可使用 CA APM ChangeDetector 附带的一个示例 XML 文件。

另外，熟悉以下概念：

- 使用多个 CA APM ChangeDetector 配置文件 (p. 15)
- 了解数据源 (p. 15)

详细信息：

示例配置文件 (p. 73)

使用多个 CA APM ChangeDetector 配置文件

您可以指定包含多个 CA APM ChangeDetector 配置文件的目录，而不是指定一个文件。指定目录后，CA APM ChangeDetector 在启动时会读取该目录中的所有有效配置文件。

指定 CA APM ChangeDetector 配置目录：

- 编辑 *IntroscopeAgent.profile*，设置以下属性：
  `introscope.changeDetector.profileDir=<ChangeDetector 目录的路径>`

  该属性的默认值为：
  `<IntroscopeAgent.profile 的路径>/changeDetector_profiles`。

  如果您也为以下属性指定一个文件：
  `introscope.changeDetector.profile`

  CA APM ChangeDetector 将读取该文件以及指定的配置目录中的所有文件。

了解数据源

数据源是一组资源，其类型由数据源类型定义－例如，文件、数据库表列值或运行时类实例。资源按名称来标识，例如 *C:\Introscope\Introscope Enterprise Manager.exe*、数据库中的 *buffer_pool* 列名或 *java.lang.System*。一个数据源可以有很多资源；资源可以有很多值，但一次只有一个值。

注意：数据源中的资源名称必须是唯一的。

使用两种方法之一来定义数据源：

- 使用 CA APM ChangeDetector 配置向导 (p. 16)
- 修改 CA APM ChangeDetector 配置文件 (p. 25)

您在配置文件中指定的数据源显示在启用了 CA APM ChangeDetector 的代理的调查器中的“更改”选项卡中。 有关查看更改数据的信息，请参阅查看 CA APM ChangeDetector 数据 (p. 51)。

每个数据源定义都必须包含在 *datasource-type* 标记内，且必须包含 *name* 和 *class* 属性。*name* 之后会在配置文件中被引用到，它可以是任何内容。*class* 属性指定使用哪个类来解析这种类型的数据源的任何 datasource-instance 定义。 该类必须实现以下接口：

```
com.wily.cd.agent.config.IDataSourceConfig
```

datasource-type 元素指示代理有哪些类型的数据源可用。 要充分利用 *datasource-type* 元素，还需定义 *datasource-instance* 元素。 每个元素对应代理监控的一个物理数据源。

注意：数据源实例不能包含多个同名的资源－例如，同一数据源实例中具有相同完整路径和名称的重复文件是无效的，同名的 *javaenv* 属性也是如此。 CA APM ChangeDetector 中的所有数据都是按数据源实例来组织的。

使用 ChangeDetector 配置向导

使用配置向导配置 CA APM ChangeDetector。 然而，您也可以更新 *ChangeDetector-config.xml* 来执行此操作－请参阅示例配置文件 (p. 73)。

可以使用配置向导完成以下操作：

- 运行配置向导 (p. 16)
- 使用配置向导添加数据源 (p. 17)
- 使用配置向导修改数据源 (p. 18)
- 使用配置向导删除数据源 (p. 18)

注意：可以使用多个 CA APM ChangeDetector 配置文件。 有关详细信息，请参阅使用多个 CA APM ChangeDetector 配置文件 (p. 15)。

运行配置向导

使用配置向导，可以通过图形用户界面来设置数据源属性，而无需手动编辑 XML。

运行配置向导：

1. 安装 CA APM ChangeDetector 之后，转至 *<Workstation_Home>/tools*，在命令提示符下运行 *configwizard.bat* 以启动配置向导。

   注意：如果未设置 *JAVA_HOME*，则将 JRE 的路径作为参数在命令提示符下运行 *configwizard.bat*。 例如：*configwizard.bat s:\sw\sun\jre*。

2. 在该向导中，可以执行下列操作：
   - 使用配置向导添加数据源 (p. 17)
   - 使用配置向导修改数据源 (p. 18)
   - 使用配置向导删除数据源 (p. 18)

使用配置向导添加数据源

可以使用配置向导添加以下数据源：

- 使用向导配置程序集监视器 (p. 19)

  注意：这仅适用于 .NET 平台。
- 使用向导配置 Java 类监视器 (p. 20)

  注意：这仅适用于 Java 平台。
- 使用向导配置数据库监视器 (p. 21)
- 使用向导配置文件系统监视器 (p. 22)
- 使用向导配置 .NET 环境变量监视器 (p. 24)

  注意：这仅适用于 .NET 平台。
- 使用向导配置 Java 系统属性监视器 (p. 25)

  注意：这仅适用于 Java 平台。

添加数据源：

1. 运行配置向导 (p. 16)。
2. 选择平台，然后单击“下一步”。
3. 选择“创建新 CA APM ChangeDetector 配置文件”，然后单击“下一步”。

   从向导的这一页可以创建新的配置文件或修改现有配置文件。
4. 选择一个选项，然后单击“下一步”。

   创建数据源后，它们将显示在向导窗口的“当前数据源”部分。 此后，可以随时编辑和删除这些数据源。

   添加数据源后，它们将显示在当前数据源中。 如果需要，可以进一步编辑不同字段和元素或删除数据源。

5. 选择要添加的数据源类型，然后单击“下一步”。
6. 输入要添加的数据源的名称，然后单击“下一步”。这些选项因上一步中指定的平台而异。
   - 继续配置所有适用的数据源（请参阅使用向导配置数据源 (p. 19)），并根据需要定义可选的 CA APM ChangeDetector Workstation 和企业管理器属性（请参阅可选的配置属性 (p. 46)）。
7. 添加完所有数据源后，单击“另存为”保存 XML 文件。

使用配置向导修改数据源

添加数据源后，可以对它们进行修改。

注意：如果要升级至更高版本，可以在升级时保存原始文件的备份，也可以覆盖现有文件。

修改数据源：

1. 运行配置向导 (p. 16)。
2. 选择一个平台，然后单击“下一步”。
3. 选择“修改现有的 ChangeDetector”，然后单击“下一步”。
4. 选择要修改的配置文件。
5. 在向导的左窗格中选择要修改的数据源，并进行修改。

   每个数据源都有可修改的不同字段和元素。 您可以更改字段，也可以选择元素，然后对其进行修改。
6. 进行更改后，单击“保存”，然后单击“退出”。

使用配置向导删除数据源

添加数据源后，可以将它们删除。

删除数据源：

1. 运行配置向导 (p. 16)。
2. 选择平台，然后单击“下一步”。
3. 选择“删除所选数据源”，然后单击“下一步”。
4. 选择要从中删除数据源的配置文件。
5. 要删除数据源，应在向导左窗格中将其选中。
6. 进行所有更改后，单击“保存”，然后单击“退出”。

使用向导配置数据源

使用配置向导可以配置以下数据源。 将数据源添加到新配置文件之后，单击“另存为”以命名并保存 XML 文件：

- 使用向导配置程序集监视器 (p. 19)

  注意：程序集监视器仅适用于 .NET 平台。

- 使用向导配置 Java 类监视器 (p. 20)
- **注意：**Java 类监视器仅适用于 Java 平台。
- 使用向导配置数据库监视器 (p. 21)
- 使用向导配置 .NET 环境变量监视器 (p. 24)

  注意：.NET 环境变量监视器仅适用于 .NET 平台。

- 使用向导配置 Java 系统属性监视器 (p. 25)

  注意：Java 系统属性仅适用于 Java 平台。

使用向导配置程序集监视器

对于 .NET 平台，可以添加或修改程序集监视器数据源。

使用向导配置程序集监视器数据源：

1. 输入字母数字字符组成的数据源名称，然后单击“下一步”。

   此时会显示向导的下一页。

2. 输入以下属性配置信息，然后单击“下一步”：

   - **每次迭代的类数**－定义的每次迭代加载的类数。 值越低，使用的 CPU 越少，但扫描所需的时间越长。
   - **在迭代之间延迟**－两次迭代之间经过的时间（单位为秒、分钟或小时）。在该时间内不监控类。 值越高，使用的 CPU 越少，但扫描所需的时间越长。
   - **初始等待时间**－监控类之前的初始时间（单位为秒、分钟或小时）。

   此时会显示向导的下一页。

3. 要为程序集和类添加排除元素，请执行以下步骤：

   a. 选择“添加新排除元素”。

   b. 单击“下一步”。

   c. 在提供的字段中键入排除模式。

   d. 要添加包括模式，请单击“添加”并在提供的字段中键入模式。 根据需要重复此步骤。

   e. 单击“下一步”。

   f. 重复以上步骤添加更多排除模式，或者单击“完成”。

   注意：第一个排除元素排除监控与 regex 匹配的程序集，第二个排除元素排除监控与 regex 匹配的类。
   使用排除模式可缩小扫描范围，并且可针对排除模式的任何例外情况添加包括模式。 例如，排除模式可以是 .*，包括模式可以是 java\.*
   对排除或包括模式使用正则表达式。 例如：
   foo
   .*bar.
   .*
   com\.wily\.(.*)

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

使用向导配置 Java 类监视器

对于 Java 平台，可以添加或修改 Java 类监视器数据源。

使用向导配置 java 类监视器数据源：

1. 使用字母数字字符输入数据源名称，然后单击“下一步”。

   此时会显示向导的下一页。

2. 输入以下属性配置信息，然后单击“下一步”：

   - **每次迭代的类数**－值越低，使用的 CPU 越少，但扫描所需的时间越长。
   - **在迭代之间延迟**－值越高，使用的 CPU 越少，但扫描所需的时间越长。

   此时会显示向导的下一页。

3. 选择“添加新排除元素”，然后单击“下一步”。
4. 在提供的字段中输入排除模式。
5. 要添加包括模式，请单击“添加”，然后在提供的字段中输入模式。根据需要重复此步骤，然后单击“下一步”。
6. 重复以上步骤添加更多排除模式，或者单击“完成”。

   注意：使用排除模式可缩小扫描范围，并且可针对排除模式的任何例外情况添加包括模式。 例如，排除模式可以是 .*，包括模式可以是 java\.*
   对排除或包括模式使用正则表达式。 例如：
   foo
   .*bar.
   .*
   com\.wily\.(.*)

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

使用向导配置数据库监视器

可以使用向导添加或修改数据库监视器数据源。

使用向导配置数据库监视器数据源：

1. 使用字母数字字符输入数据源名称，然后单击“下一步”。

   此时会显示向导的下一页。
2. 针对 Java 平台输入以下数据库信息，然后单击“下一步”：
   - **JDBC 驱动程序**—该数据库的驱动程序。 例如：
     oracle.jdbc.driver.OracleDriver
   - **JDBC 驱动程序类路径**—您的数据库驱动程序的路径（也可以浏览该路径）。
   - **JDBC URL**—输入数据库的 JDBC URL。
   - **用户名**—输入使用数据库的人员的用户名。
   - **密码**—输入您数据库的密码。密码在配置文件中将自动经过模糊处理。

- **确认密码**－出于验证的目的重新输入密码。
- **排定类型**－“重复性”或“后继启动”。 对很少更改或应用程序仅仅在启动时检查的数据库选择后继启动。
- **重复时间间隔**－如果您选择的排定类型是“重复性”，请选择时间间隔。 值越高，使用的 CPU 越少，但扫描所需的时间越长。
- **在迭代之间延迟**－值越高，使用的 CPU 越少，但扫描所需的时间越长。

此时会显示向导的下一页。

3. 针对 .NET 平台输入以下数据库信息，然后单击“下一步”：
   - **URL**－输入数据库的 URL。
   - **用户名**－输入使用数据库的人员的用户名。
   - **密码**－输入您数据库的密码。密码在配置文件中将自动经过模糊处理。
   - **确认密码**－出于验证的目的重新输入密码。

   注意：用户名和密码可以在 .NET 平台的 URL 属性内指定，也可以没有值。 向导不检查 URL 的这些值。
   - **排定类型**－“重复性”或“后继启动”。 对很少更改或应用程序仅仅在启动时检查的数据库选择后继启动。
   - **重复时间间隔**－如果您选择的排定类型是“重复性”，请选择时间间隔。 值越高，使用的 CPU 越少，但扫描所需的时间越长。

   此时会显示向导的下一页。
4. 要添加 SQL 语句，请选择“添加新 SQL 语句”，然后单击“下一步”。

   可以重复上述步骤添加更多 SQL 语句，或者单击“完成”。 例如：

   ```
   SELECT name, value FROM v$parameter
   ```

   注意：向导不验证 SQL 语句。 请确保对数据库使用有效的 SQL 语句。

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

使用向导配置文件系统监视器

可以使用向导添加或修改文件系统监视器数据源。

使用向导配置文件系统监视器数据源：

1. 使用字母数字字符输入数据源名称，然后单击“下一步”。

   此时会显示向导的下一页。

2. 选择“添加新的 scan-directory 元素”，然后单击“下一步”。

3. 输入目录名称和必填字段：“目录名称”、“递归”、“文件集”和“已启用”：

   - **目录名称**－输入您希望 CA APM ChangeDetector 扫描的目录路径。 例如：*C:\\WebLogic\\myApplicationHome*
   - **递归**－如果值为 True，则 CA APM ChangeDetector 将查看指定目录下的任何子文件夹。
   - **文件集**－选择与该扫描目录关联的文件集。
   - **已启用**－您可能要出于测试或其他目的禁用扫描目录。 然而，如果您在将其启用后再禁用，CA APM ChangeDetector 会将该扫描目录元素中所包含的文件报告为已删除。

4. 要添加排除模式，请单击“添加新的”并输入模式。根据需要重复此步骤。

5. 单击“创建或修改文件集”添加文件集或编辑现有文件集。

   - 选择“添加新文件集”，然后单击“下一步”。
   - 输入文件集名称，然后单击“下一步”。

6. 单击“下一步”，通过重复上述步骤添加更多扫描目录元素，然后单击“完成”。

7. 输入包括/排除元素可缩小扫描范围，并且可针对排除模式的任何例外情况添加包括模式：

   - 选择“添加新排除元素”，然后单击“下一步”。
   - 在提供的字段中键入排除模式。
   - 要输入包括模式，请单击“添加”并在提供的字段中键入包括模式。根据需要重复此步骤。

8. 输入以下信息，然后单击“下一步”：

   - **每次迭代的文件数**－值越低，使用的 CPU 越少，但扫描所需的时间越长。
   - **在文件迭代之间延迟**－值越高，使用的 CPU 越少，但扫描所需的时间越长。
   - **要上载文件的最大大小**－可上传到服务器的最大 ASCII 文件大小。 可以在 CA APM ChangeDetector 中的差异视图中查看上传的 ASCII 文件。

   注意：以下存档属性仅 Java 平台。

- **使用摘录**－选择“总是”、“从不”或“根据需要”。 摘录是一种形式的消息摘要，如 MD5。 它使 CA APM ChangeDetector 可以通过执行哈希比较来忽略文件的修饰性更改。 如果选择“总是”，可能会影响性能。 如果选择“根据需要”，将只有在时间戳和文件大小发生更改时才使用摘要。
- **展开存档**－如果您要扫描存档（如 *.zip* 或 *.jar*）中的各个文件，请选择 True。
- **每次迭代的存档数**－如果您选择展开存档，请指定该值。值越低，扫描所需的时间越长。 由于存档文件是作为一个整体扫描的，因此，如果您的存档中有大量的文件，应当限制该数目。
- **在存档迭代之间延迟**－值越高，使用的 CPU 越少，但扫描所需的时间越长。

9. 添加文件集，在本部分选择“完成”，然后单击“下一步”。

   注意：要使 CA APM ChangeDetector 能够检测到文件，受监控的文件系统必须具有读取权限，并且如果这些系统在网络上，还要求网络正常工作。

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

使用向导配置 .NET 环境变量监视器

对于 .NET 平台，可以添加或修改环境变量数据源。

使用向导配置文件系统监视器数据源：

1. 使用字母数字字符输入数据源名称，然后单击“下一步”。
2. 通过选择“添加新排除元素”来添加排除元素，然后单击“下一步”。

   可以在提供的字段内输入排除模式。 使用排除模式可缩小扫描范围，并且可针对排除模式的任何例外情况添加包括模式。 例如，排除模式可以是 .*，包括模式可以是 java\.*

   对排除或包括模式使用正则表达式。 例如：

   foo

   .*bar.

   (.*)

   java\.*

3. 通过选择“添加新的”来添加包括模式，然后单击“下一步”。

   您可以在提供的字段内输入包括模式。

4. 根据需要重复必要的步骤继续添加包括/排除模式，然后单击“完成”。

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

使用向导配置 Java 系统属性监视器

对于 Java 平台，可以添加或修改 Java 系统属性监视器数据源。

使用向导配置 Java 系统监视器数据源：

1. 使用字母数字字符输入数据源名称，然后单击“下一步”。
2. 通过选择“添加新排除元素”来添加排除元素，然后单击“下一步”。

   可以在提供的字段内输入排除模式。 使用排除模式可缩小扫描范围，并且可针对排除模式的任何例外情况添加包括模式。 例如，排除模式可以是 .*，包括模式可以是 java\.*

   对排除或包括模式使用正则表达式。 例如：

   foo

   .*bar.

   (.*)

   java\.*

3. 通过选择“添加新的”来添加包括模式，然后单击“下一步”。

   您可以在提供的字段内输入包括模式。

详细信息：

使用配置向导添加数据源 (p. 17)
使用配置向导修改数据源 (p. 18)

修改 CA APM ChangeDetector 配置文件

通过编辑 XML 配置文件，您可以手动修改配置设置。 该方法可用于所有平台。 如果您选择此方法，可使用 CA APM ChangeDetector 附带的一个示例 XML 文件。

详细信息：

示例配置文件 (p. 73)

关于 ChangeDetector-config.xml 文件

您可以基于 ChangeDetector-config.xml 创建自定义配置文件，从而*指定*您希望 CA APM ChangeDetector 监控的更改类型。 CA APM ChangeDetector 按数据源对更改进行分组。 使用该文件，您可以更改以下配置：

- 手工配置数据库监视器属性 (p. 27)—*database* 数据源定义如何从兼容的数据库中收集更改数据。
- 手工配置文件系统监视器属性 (p. 29)—*file* 数据源定义要监控哪些文件的更改。
- 手工配置 Java 类监视器属性 (p. 35)—*classmonitor* 数据源定义要监控哪些 Java 类。

  注意：这仅适用于 Java 平台。
- 手工配置 Java 系统属性监视器 (p. 37)—*javaenv* 数据源定义要监控哪些 Java 过程系统属性。

  注意：这仅适用于 Java 平台。
- 手工配置程序集监视器属性 (p. 37)—表示 .NET 环境的程序集监视器的 *classmonitor* 数据源指示 CA APM ChangeDetector 监控哪些程序集。

  注意：这仅适用于 .NET 平台。
- 手工配置 .NET 环境变量监视器属性 (p. 40)—表示 .NET 环境的系统监视器属性的 *javaenv* 数据源指示 CA APM ChangeDetector 监控哪些系统属性。

  注意：这仅适用于 .NET 平台。

重要信息！使用完整文件路径搜索文件可能会导致 CPU 使用率猛增。要避免该问题，请仅使用完整路径文件名进行搜索。 要仅使用完整路径文件名进行搜索，请通过添加 *fullpath="true"* 属性来编辑 *ChangeDetector-config.xml* 文件，如下所示：

```
<scan-directory recursive="true" name="/opt/Oracle" fileset="default"
enabled="true" fullpath="true" > </scan-directory>
```

在配置文件中使用系统属性或代理属性

可以将系统属性或代理配置文件属性用作配置文件中的任何 XML 属性的值。 这样，便可以在 CA APM ChangeDetector 配置文件中使用运行时解析的属性。 如果为某个值同时提供了系统属性和代理配置文件属性，则系统属性优先。

下面是一个示例：

```
<scan-directory name="${APPLICATION_HOME}/bin/" recursive="true"
fileset="default"/>
```

在上面的示例中，*${APPLICATION_HOME}* 将在运行时替换为其当前值。

注意：这些属性的值必须在运行时映射到有效的系统或代理配置文件属性。

详细信息：

示例配置文件 (p. 73)

手工配置数据库监视器属性

database 数据源实例指示 CA APM ChangeDetector 如何从兼容的数据库中收集更改数据。 请确保有数据库支持的 JDBC 或 OLEDB 驱动程序。

注意：根据使用的平台的不同，配置属性有些微的差异。 如果属性仅适用于某些平台，则会在本部分给出相应的注释。

注意：该元素在示例 Java ChangeDetector-config.xml 文件 (p. 74)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="Orcl_on_aserver" type="database" version="8.0"
driver="oracle.jdbc.driver.OracleDriver"
driverClasspath="C:\\somePathTo\\classes12.zip"
url="jdbc:oracle:thin:@aserver:1521:orcl" username="a3f973777b9d"
password="f478831d9bcd65" isClearText="false" >
SQL Server
SELECT name, value FROM v$parameter
</sql>
<schedule type="repetitive" interval="1" unit="min" />
</datasource-instance>
```

使用类路径分隔符 ; 或 : 可提供数据库驱动程序的多个类路径。

可以提供数据库的用户名和密码（如果有）。 但是不能只为其中的一项提供值。

注意：为安全起见，您输入的用户名和密码值将自动替换为经过模糊处理的值。 要更改这些值，请先设置属性 *isClearText="true"*，然后再进行更改。 代理下次运行时，*isClearText* 属性将自动重置为 *false*。

使用数据库监视器时，除了所有 *datasource-instance* 元素都必须定义的 *name* 和 *type* 属性外，还必须在 *datasource-instance* 元素中设置以下属性：

- *driver*—要使用的 JDBC 兼容驱动程序的类名。 此示例连接到 Oracle 数据库，因此该自定义配置文件中提供的驱动程序为：
  `oracle.jdbc.driver.OracleDriver`

注意：此属性仅适用于 Java 平台。

- *driverClasspath*—包含 driver 属性中所引用驱动程序的存档的路径。可以通过适用平台的类路径分隔符来指向多个驱动程序。

注意：此属性仅适用于 Java 平台。

- *url*—要用来连接到数据库的 JDBC URL。
- *username*—要用来连接到数据库的用户名。 即使在重新启动代理时 *cleartext* 属性设置为 *true*，也将对该值进行模糊处理。
- *password*—要用来连接到数据库的密码。 即使在重新启动代理时 *cleartext* 属性设置为 *true*，也将对该值进行模糊处理。
- *isClearText*—指定是否对用户名和密码进行模糊处理。如果值为 *true*，将对用户名和密码进行模糊处理，并使用模糊处理后的值重写配置文件。

出于安全考虑，此属性默认情况下设置为 *false*。 如果将该属性设置为 *true*，然后重新启动代理，则该属性将重新设置为 *false*。

可以在类型为 *database* 的 datasource-instance 元素中定义两个顶级元素：*sql* 和 *schedule*。

sql 元素是用来从您计划监控的表中收集信息的 SQL 语句。 此类型的 *datasource-instance* 元素中可以有任意数目的 SQL 元素。

注意：无论定义了多少个 SQL 元素，SQL 语句都必须仅生成两列，且第一列中的值必须是唯一的（主键）。 *resultset* 的第一列中不允许出现空值。第二列中出现空值被视为相应的行不存在(即 *resultset* 中没有该行)。

schedule 元素定义数据库监视器扫描更改的频率。 必须为此元素定义 *type* 属性。 此属性的有效值为：

- *repetitive*—如果 *type* 属性的值为 *repetitive*，必须定义 *interval* 和 *unit* 属性。 *interval* 属性必须是整数值。 *unit* 属性必须为下列值之一：*分钟*、*小时*、*秒*。这两个属性指示 CA APM ChangeDetector 扫描数据库更改的频率。 在示例中，CA APM ChangeDetector 每分钟扫描一次。 如果希望每 10 秒钟扫描一次，应当为 *interval* 提供值 *10*，为 *unit* 提供值*秒*。
- *post-startup*—如果 *type* 属性的值为 *post-startup*，则不需要定义其他属性。对 *type* 属性使用该值将指示 CA APM ChangeDetector 在代理启动后扫描提供的 SQL 语句一次。

手工配置文件系统监视器属性

通过 CA APM ChangeDetector 文件监控系统，可以控制对需求不同且容许的 I/O 开销不同的组织中文件更改的监控。 这样，便可以平衡从发生更改到检测到更改期间的处理和 I/O 成本及时间。

注意：文件监控属性区分大小写。 例如，如果要监控的目录名为 *i18n*，但 *scan-directory name* 属性设置为 *I18N*，CA APM ChangeDetector 将无法找到该目录。

注意：该元素在示例 Java ChangeDetector-config.xml 文件 (p. 74)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="C_Drive" type="file" version="8.0">
      <!-- datasource-instance specific XML here -->
      <property name="explodeArchiveFiles" value="true" />
      <!-- Accepted units are hour, min, sec -->
      <property name="delayBetweenIterations" value="1" unit="sec" />
      <property name="filesPerIteration" value="1" />
      <property name="delayBetweenArchiveIterations" value="10" unit="sec" />
      <property name="archiveFilesPerIteration" value="1" />
      <!-- Accepted units are bytes, KBytes, MBytes -->
      <property name="maxFileSizeToUpload" value="4" unit="KB" />
      <property name="useDigest" value="needed" />
      <fileset name="default">
         <exclude pattern="(.*)\.err" />
         <exclude pattern="(.*)\.log" />
         <exclude pattern="(.*)\.lok" />
         <exclude pattern="(.*)\.tlog" />
         <exclude pattern="(.*)\.log0(.*)" />
      </fileset>
      <fileset name="NoCode">
         <include-fileset name="default" />
         <exclude pattern="(.*)\.jar" >
             <include pattern="(.*)wily(.*)\.jar" />
         </exclude>
         <exclude pattern="(.*)\.zip" />
      </fileset>
      <!--  typically, the wily agent is installed in the "wily" directory. -->
      <scan-directory name="wily" recursive="true"
         fileset="default">
         <exclude name="data" />
      </scan-directory>
      <!--  scan the java home directory, as specified in the java.home system
property -->
      <scan-directory recursive="true" fileset="default"
         name="${java.home}" enabled="false">
         <exclude name="lib/zi" />
      </scan-directory>
      <!--  directories to be scanned in a typical jboss installation -->
      <scan-directory name="." recursive="true" fileset="default"
         enabled="false">
         <exclude name="log" />
         <exclude name="tmp" />
      </scan-directory>
      <!--  test scan directory -->
      <scan-directory recursive="true" fileset="NoCode" name="."
         enabled="true" />
      <!--  directories to be scanned in a typical WebSphere 5.0ee installation
-->
      <!--  basically exclude the following dirs:
         _uninst, _uninstPME, BRBeans, classes, installableApps, logs, temp,
tranlog, wstemp -->
      <scan-directory recursive="true" name="." fileset="default"
         enabled="false">
         <exclude name="_uninst" />
         <exclude name="_uninstPME" />
```

```
        <exclude name="logs" />
        <exclude name="temp" />
        <exclude name="tranlog" />
        <exclude name="wstemp" />
      </scan-directory>
    </datasource-instance>
```

定义类型为 File 的数据源实例中的元素

可以在类型为 file 的数据源实例中定义三个顶级元素：

- 属性 (p. 31)
- fileset (p. 34)
- scan-directory (p. 35)

在配置文件的开头，将数据源实例类型 *file* 定义为文件系统监视器。

文件更改监控系统按顺序扫描由 *file* 数据源实例配置指定的所有文件。将分配 CPU 以收集一组工作，完成后将重新释放 CPU 以执行其他任务。

property 元素

每个 *property* 元素都必须具有一个 *name* 属性和一个 *value* 属性。此处定义的某些属性可能还包含 *单位*属性。

注意：.NET 平台不支持存档，因此只能针对 Java 定义所有与存档相关的属性。

此类型的数据源实例支持以下属性：

- *explodeArchiveFiles*

  仅当您使用的是 Java 代理时，此属性才适用。 它与运行 .NET 代理的环境不兼容。

  其值属性可以为 *true* 或 *false*。

  如果值设置为 *true*，CA APM ChangeDetector 将报告存档文件的内容。 如果 CA APM ChangeDetector 正在监控的存档文件中发生更改，将至少发送两个更改事件——一个是修改存档文件，另一个是修改存档文件中的内容文件。 如果修改的内容文件也是存档文件，CA APM ChangeDetector 还会打开该存档，并发送嵌套存档的内容文件的更改事件，依此类推。

  仅支持 ZIP 和 GZIP 文件格式的存档（例如 *zip*、*gzip*、*jar*、*ear*、*war*、*rar*、*sar*）。 CA APM ChangeDetector 不支持 *tar* 文件。

  如果 *explodeArchiveFiles* 值设置为 false，则 CA APM ChangeDetector 不查看存档内部。 对存档进行的任何更改均显示为对存档本身的修改/添加/删除。

  默认值为 *false*。

- *delayBetweenIterations*

  值为整数，单位为 *秒*、*分钟*或 *小时*。

  此属性定义文件队列的每次迭代之间的休眠时间，与 *filesPerIteration* 属性相关联。

  默认值为 3 秒。

- *filesPerIteration*

  值为整数，定义与 *delayBetweenIterations* 属性相关的一组工作的文件数。

  此属性指定在放弃 CPU 之前扫描的文件数。如果每次迭代有 10 个文件，CA APM ChangeDetector 将扫描 10 个文件中的更改，然后休眠，持续时间为 *delayBetweenIterations* 属性中定义的值。休眠时间过后，它将扫描另 10 个文件并再次休眠，依此类推。

  默认值为 5。

- *delayBetweenArchiveIterations*

  值为整数，单位为*秒*、*分钟*或*小时*。

  此属性控制存档队列的休眠时间。当 CA APM ChangeDetector 扫描某文件并将其识别为存档时，CA APM ChangeDetector 会将该文件置于存档队列中以便可以检查其内容。

  仅当 *explodeArchiveFiles* 设置为 *true* 时此属性才有意义；否则，存档文件将被视为常规文件。

  默认值为 10 秒。

- *archiveFilesPerIteration*

  允许采用与 *filesPerIteration* 属性 (property) 相同的属性 (attribute) 值。与 *filesPerIteration* 属性一样，*archiveFilesPerIteration* 是存档队列的一方面，控制每次迭代时扫描的存档数量。仅当 *explodeArchiveFiles* 设置为 *true* 时此属性才有意义；否则，存档文件将被视为常规文件。

  默认值为 1。

  注意：存档的完整内容是一次性上传的－*filesPerIteration* 的值在这里不适用。 但是，如果在当前扫描的存档内发现另一个存档，则会将嵌套存档添加到存档队列中。

- *maxFileSizeToUpload*

  值属性值为整数，单位属性值为 *B*、*KB* 或 *MB*。此属性定义其内容将发送到服务器的文件的最大大小。 当前只发送大小低于此属性定义的文件大小的 ASCII 文件。

  默认值为 50KB。

- *useDigest*

  此属性定义如何使用消息摘要（如 MD5）检测更改。 使用摘要，CA APM ChangeDetector 可以通过执行哈希比较来忽略文件的修饰性更改。 此属性的值为：

  - 从不－不使用摘要
  - 总是－总是进行摘要比较
  - 根据需要－只有在时间戳和文件大小已更改时才使用摘要（默认值）

fileset 元素

下面是 *fileset* 元素的示例：

```
<fileset name="default">
        <exclude pattern="(.*)\.err" />
        <exclude pattern="(.*)\.log" />
        <exclude pattern="(.*)\.lok" />
        <exclude pattern="(.*)\.tlog" />
        <exclude pattern="(.*)\.log0(.*)" />
</fileset>
<fileset name="NoCode">
        <include-fileset name="default" />
        <exclude pattern="(.*)\.jar" >
            <include pattern="(.*)wily(.*)" />
        </exclude>
        <exclude pattern="(.*)\.zip" />
</fileset>
<fileset name="all">
        <exclude pattern="(.*)>
            <include pattern="(.*) />
        </exclude>
</fileset>
```

文件集定义扫描中包括或排除的文件的模式。 fileset 元素允许的子元素有：

- *exclude*
- *include-fileset*

exclude 元素必须定义 *pattern* 属性。 *exclude* 元素可以包含 0 个或 0 个以上的 *include* 元素作为子节点。 *include* 元素必须定义 *pattern* 属性。 *include* 元素用于修正 *exclude* 元素。 在上面的示例中，对于文件集 *NoCode*，要排除所有以 *.jar* 结尾的文件名，但名称中包含 *wily* 的文件除外。

include-fileset 元素必须定义 *name* 属性。 使用 *include-fileset* 元素时，要包括的文件集必须已在之前定义好。在上面的示例中，文件集 *NoCode* 包括文件集 *default*。如果交换两个文件集的顺序，使 default 成为定义的第二个文件集，则配置将无效。 当文件集使用 *include-fileset* 元素时，所引用文件集的所有包括模式和排除模式均成为主元素的一部分。 在上面的示例中，文件集“*NoCode*”包含排除模式 **.jar*、**.zip*、**.err*、**.log*、**.lok* 等。

fileset 元素中 *exclude* 和 *include-fileset* 元素的顺序无关紧要。如果某文件不匹配定义的排除模式，将包括在扫描中。 如果某文件匹配定义的排除模式，则检查其是否匹配在该排除模式内定义的任何包括模式。 如果匹配一个包括模式，则将文件包括在扫描中。 如果不匹配任何包括模式，则从扫描中排除该文件。

scan-directory 元素

scan-directory 元素定义要扫描的目录。 *scan-directory* 元素的属性为 *name*、*recursive*、*fileset* 和 *enabled*：

- *name* 属性定义要扫描的目录。 必需。 例如，*c:\\test* 或 *C:/test*。
- *recursive* 属性的值为 *true/false*，它定义文件系统监视器是否递归扫描它找到的任何目录。 默认值为 *true*。
- fileset 属性定义此数据源实例要使用的文件集。必须在配置文件中提前定义所提供的文件集。 这是必需属性。
- *enabled* 属性的值为 *true/false*，它启用或禁用 *scan-directory* 元素。

scan-directory 元素可包含 *exclude* 子元素。 *exclude* 元素需要 *name* 属性。 *name* 属性必须映射到目录。

注意：为 name 属性提供的值不能是正则表达式。 它必须是与实际目录一致的文本字符串。要基于模式排除内容，请修改所使用的 *fileset* 元素。

以下是在示例 Java ChangeDetector-config.xml 文件 (p. 74)中定义的此元素的示例：

```
<scan-directory name="." recursive="true" fileset="default"
   enabled="false">
   <exclude name="log" />
   <exclude name="tmp" />
</scan-directory>
```

手工配置 Java 类监视器属性

注意：这仅适用于 Java 平台。

classmonitor 数据源实例指示 CA APM ChangeDetector 要监控哪些 Java 类。 类名及其类加载器用作资源名称，与类定义对应的字节数组用作资源值。

每个 CA APM ChangeDetector 实例只允许有一个此类型的 *datasource* 实例。

注意：由于 Java 采用按需加载的方式工作，CA APM ChangeDetector 无法确定是类不再属于二进制执行的一部分，还是尚未加载；因此 *classmonitor* 数据源不生成删除更改。

注意：该元素在示例 Java ChangeDetector-config.xml 文件 (p. 74)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="Java class monitor" type="classmonitor"
version="8.0">
      <property name="delayBetweenIterations" value="2" unit="sec"/>
      <property name="classesPerIteration" value="100" />
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

exclude 元素的语法与 Java 系统属性监视器中的语法相同。

详细信息：

手工配置 Java 系统属性监视器 (p. 37)

定义类型为 Classmonitor 的数据源实例中的元素

注意：这仅适用于 Java 平台。

可以为类型为 *classmonitor* 的数据源实例定义两个 property 元素。 其中的每个 *property* 元素都必须有一个 *name* 属性和一个 *value* 属性。

- *delayBetweenIterations*

  值为整数，单位为*秒*、*分钟*或小时。

  此属性定义类队列的每次迭代之间的休眠时间，与 *classesPerIteration* 属性相关联。

  默认值为 2 秒。

- *classesPerIteration*

  值为整数，定义与 *delayBetweenIterations* 属性相关的一组工作的类数。

  此属性指定在放弃 CPU 之前扫描的类数量。 如果每次迭代有 10 个类，CA APM ChangeDetector 将扫描 10 个类中的更改，然后休眠，持续时间为 *delayBetweenIterations* 属性中定义的值。 休眠时间过后，它将扫描另 10 个类并再次休眠，依此类推。

  默认值为 100。

手工配置 Java 系统属性监视器

注意：这仅适用于 Java 平台。

javaenv 数据源实例指示 CA APM ChangeDetector 要监控哪些 Java 系统属性。 CA APM ChangeDetector 只监控两次重新启动进程之间 Java 系统属性的更改，而不监控由正在运行的 Java 程序调用的运行时更改－此类型的收集仅在启动时运行一次。 属性名称用作资源名称，而属性值构成这些资源的值。

注意：该元素在示例 Java ChangeDetector-config.xml 文件 (p. 74)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="Java system properties" type="javaenv"
version="8.0">
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

在 *javaenv* 数据源中，只能将 *exclude* 元素定义为顶级元素。*exclude* 元素可包含子元素。

使用 *exclude* 元素可排除不希望 CA Introscope 显示的属性。 *exclude* 元素必须定义 *pattern* 属性。 例如，如果不希望 CA Introscope 显示 *foo* 属性，请使用示例中所示的 *exclude* 元素。

exclude 元素可定义任意数量的 *include* 子节点。 与 *exclude* 元素相同，*include* 元素必须定义 *pattern* 属性。*include* 子节点用于重写 *exclude* 元素的行为。如示例所示，可以排除匹配 *.*bar.** 的所有属性，其中匹配 *hello* 或 *.*world.** 的属性除外。

手工配置程序集监视器属性

注意：这仅适用于 .NET 平台。

classmonitor 数据源表示 .NET 环境的程序集监视器。 它指示 CA APM ChangeDetector 要监控哪些程序集。

程序集监视器每次加载一个方法。 方法中包含的元数据如下：

- 程序集名称
- 版本
- 类
- 方法
- 方法签名

可以监控同名程序集的不同版本之间元数据的变化。例如，*cd_sample.dll 1.0.0* 和 *cd_sample.dll version 1.0.1* 中的类将被视为相同元数据的不同版本。 将在 Workstation 调查器中监控并显示数据的变化。 但是，如果程序集的名称变化，程序集将被视为新资源，其中的类也将是新资源，将被视为其他事件。

注意：该元素在示例 .NET ChangeDetectorDotnet-config.xml 文件 (p. 79)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="Assembly Monitor" type="classmonitor" version="8.0">

    <property name="initialWaitTime" value="30" unit="sec" />
    <property name="delayBetweenIterations" value="2" unit="min" />
    <property name="classesPerIteration" value="5" />
    <excludeassembly pattern=".\mscorlib\.dll"/>
    <excludeassembly pattern=".\System\.dll"/>
    <excludeassembly pattern=".\System\.Xml\.dll"/>
    <excludeassembly pattern=".\System\.Web\.dll"/>
    <excludeassembly pattern=".\System\.Configuration\.dll"/>
    <excludeassembly pattern=".\wily\.."/>
    <excludeassembly pattern=".\Microsoft\.JScript\.dll"/>
    <excludeassembly pattern=".\VJSharpCodeProvider\.dll"/>
    <excludeassembly pattern=".\System\.Data\.dll"/>
    <excludeassembly pattern=".\Oracle\.DataAccess\.dll"/>
    <excludeassembly pattern=".\System\.Web\.Mobile\.dll"/>
    <excludeassembly pattern=".\System\.ServiceModel\.dll"/>
    <excludeassembly pattern=".\SMDiagnostics\.dll"/>
    <excludeassembly pattern=".\System\.Drawing\.dll"/>
    <excludeassembly pattern=".\System\.Web\.RegularExpressions\.dll"/>
    <excludeassembly pattern=".\Microsoft\.VisualBasic\.dll"/>
    <excludeassembly pattern=".\CppCodeProvider\.dll"/>
    <excludeassembly pattern=".\System\.EnterpriseServices\.dll"/>
    <excludeassembly pattern=".\System\.Transactions\.dll"/>

    <exclude pattern="com\.wily\.(.*)"/>

</datasource-instance>
```

这些元素的语法如下：

- *excludeassembly*

  <!-- exclude assemblies -->

  <excludeassembly pattern=".\mscorlib\.dll"/>

  <excludeassembly pattern=".\System\.dll"/>

  <excludeassembly pattern=".\System\.Xml\.dll"/>

  <excludeassembly pattern=".\System\.Web\.dll"/>

  <excludeassembly pattern=".\System\.Configuration\.dll"/>

  <excludeassembly pattern=".\wily\.."/>

  <excludeassembly pattern=".\Microsoft\.JScript\.dll"/>

  <excludeassembly pattern=".\VJSharpCodeProvider\.dll"/>

  <excludeassembly pattern=".\System\.Data\.dll"/>

  <excludeassembly pattern=".\Oracle\.DataAccess\.dll"/>

  <excludeassembly pattern=".\System\.Web\.Mobile\.dll"/>

  <excludeassembly pattern=".\System\.ServiceModel\.dll"/>

  <excludeassembly pattern=".\SMDiagnostics\.dll"/>

  <excludeassembly pattern=".\System\.Drawing\.dll"/>

  <excludeassembly pattern=".\System\.Web\.RegularExpressions\.dll"/>

  <excludeassembly pattern=".\Microsoft\.VisualBasic\.dll"/>

  <excludeassembly pattern=".\CppCodeProvider\.dll"/>

  <excludeassembly pattern=".\System\.EnterpriseServices\.dll"/>

  <excludeassembly pattern=".\System\.Transactions\.dll"/>

- exclude

  <exclude pattern="com\.wily\.(.)"/>*

- include

  <excludeassembly pattern=".\System\.dll"/>

  <exclude pattern="abc\.xyz\.(.)">*

  <include pattern="abc\.xyz\.asdf\.(.)"/> </exclude>*

可以为类型为 *classmonitor* 的数据源实例定义 property 元素。 其中的每个 *property* 元素都必须有一个 *name* 属性和一个 *value* 属性。

- *initialWaitTime*

  值为整数，单位为秒、分钟或小时。

  此属性定义最后一个程序集加载后、代理在开始扫描任何类之前等待的时间。

- *delayBetweenIterations*

  值为整数，单位为秒、分钟或小时。

  此属性定义类队列的每次迭代之间的休眠时间，与 classesPerIteration 属性相关联。

  默认值为 2 秒。

- *classesPerIteration*

  值为整数，定义与 delayBetweenIterations 属性相关的一组工作的类数。

  此属性指定在放弃 CPU 之前扫描的类数量。 如果每次迭代有 10 个类，CA APM ChangeDetector 将扫描 10 个类中的更改，然后休眠，持续时间为 delayBetweenIterations 属性中定义的值。 休眠时间过后，它将扫描另 10 个类并再次休眠，依此类推。

  默认值为 100。

手工配置 .NET 环境变量监视器属性

注意：这仅适用于 .NET 平台。

.NET 环境变量监视器数据源 (javaenv) 实例指示 CA APM ChangeDetector 要监控哪些环境变量。CA APM ChangeDetector 仅监控两次重新启动进程之间环境变量的更改，而不监控由正在运行的应用程序所调用的运行时更改－此类型的收集仅在启动时运行一次。 变量名称用作资源名称，而变量值构成这些资源的值。

注意：该元素在示例 .NET ChangeDetectorDotnet-config.xml 文件 (p. 79)中所示的自定义配置文件示例中已定义。 本示例提供的示例数据可能会影响（也可能不会影响）数据内容。 如果您使用的是其他版本的 CA APM ChangeDetector，请在相应的位置替换为适当的内容：

```
<datasource-instance name="System Properties" type="javaenv"  version="8.0">
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

在 javaenv 数据源中，只能将 exclude 元素定义为顶级元素。 exclude 元素可包含子元素。

使用 exclude 元素可排除不希望 CA Introscope 显示的属性。 exclude 元素必须定义 pattern 属性。 例如，如果不希望 CA Introscope 显示 foo 属性，请使用示例中所示的 exclude 元素。

exclude 元素可定义任意数量的 include 子节点。 与 exclude 元素相同，include 元素必须定义 pattern 属性。include 子节点用于重写 exclude 元素的行为。如示例所示，可以排除匹配 .*bar.* 的所有属性，其中匹配 hello 或 .*world.* 的属性除外。

升级现有的 ChangeDetector-config.xml 文件

升级 CA APM ChangeDetector 时，配置文件自动升级为新格式。

注意：您必须拥有配置文件及其当前目录的写权限，才能将现有配置文件升级为原始名称 *.bak*。

CA APM ChangeDetector 重命名现有配置文件并将其替换为升级后的文件。 如果无法重命名现有文件，或无法写入其所在目录，新的升级配置文件将保存在临时目录中，并且系统会指示您用新文件覆盖现有文件。

修改代理配置文件

修改 *IntroscopeAgent.profile* 可指定 CA APM ChangeDetector 代理扩展 ID、CA APM ChangeDetector 配置文件的路径和故障切换机制。有时并不需要进行这些修改，而是可以选择使用自动 ID 分配。

设置 ChangeDetector 代理 ID 和配置文件的路径（可选）：

1. 使用易于记忆的新名称（例如，*IntroscopeAgentBackup.profile*），将每个 ChangeDetector 扩展代理实例中的 *IntroscopeAgent.profile* 备份到单独的目录中。
2. 编辑 *IntroscopeAgent.profile* 设置以下属性：

   introscope.changeDetector.agentID=<ChangeDetector 代理 ID>

   agentID 属性只能包含字母数字字符以及下划线 (_) 和连字符 (-) 这两个特殊字符。

   此 ID 与 CA APM ChangeDetector 代理的全局 ID 一致，并且必须在连接到企业管理器或企业管理器群集的所有代理中是唯一的。

3. 编辑每个 ChangeDetector 扩展代理实例上的 *introscopeAgent.profile*，以包括含文件名的配置文件路径：

   introscope.changeDetector.profile=<ChangeDetector-config.xml 的路径>

   如果使用的是示例配置文件且未更改名称，则文件名为 *ChangeDetector-config.xml*。 如果更改了名称，请指定该名称。

   输入路径名时，请使用绝对路径名和两个反斜杠。 例如，如果引用的是 WebLogic 应用程序服务器，则属性值中输入的路径应类似于：*<ProductName_Home>**<Agent_Home>*。

   注意：请确保为每个代理指定配置文件路径。 如果多个代理通过一个 *IntroscopeAgent.profile* 运行，请在命令行定义属性（例如，在应用程序服务器的启动脚本中）。 这样，CA Introscope 便可以区分不同的 CA APM ChangeDetector 代理扩展配置。 有关如何创建新的 Introscope 代理配置文件的信息，请参阅 *《CA APM Java 代理实施指南》*。

详细信息：

ChangeDetector 代理 ID 命名选项 (p. 44)

在负载平衡环境中配置 CA APM ChangeDetector

可以将 CA APM ChangeDetector 代理扩展配置为在负载平衡的代理环境中工作。 通过将具有 CA APM ChangeDetector 代理扩展的代理配置为连接到管理器中的管理器 (MOM) 来执行此操作。 之后，MOM 会将代理负载分布到相应的收集器，具体取决于 weight 属性 (*introscope.enterprisemanager.clustering.login.em1.weight*) 的配置。

为 CD 配置负载平衡：

- 通过设置以下属性将 ChangeDetector 代理配置为连接到 MOM：

  ```
  introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=<MOM 主机>
  introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=5001
  introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
  ```

- 通过设置以下属性为 MOM 配置负载平衡：

  （收集器 1）

  ```
  introscope.enterprisemanager.clustering.login.em1.host=sqw32vserv12
  introscope.enterprisemanager.clustering.login.em1.port=5001
  introscope.enterprisemanager.clustering.login.em1.publickey=internal/server/EM.public
  introscope.enterprisemanager.clustering.login.em1.weight=50
  ```

（收集器 2）

```
introscope.enterprisemanager.clustering.login.em2.host=sqw32vserv10
introscope.enterprisemanager.clustering.login.em2.port=5002
introscope.enterprisemanager.clustering.login.em2.publickey=internal/server/EM.public
introscope.enterprisemanager.clustering.login.em2.weight=50
```

有关代理属性的详细信息，请参阅*《CA APM Java 代理实施指南》*或*《CA APM .NET 代理实施指南》*。有关企业管理器属性的详细信息，请参阅*《CA APM 配置和管理指南》*。

为 CA APM ChangeDetector 配置代理故障切换机制

如果安装了 CA APM ChangeDetector 代理扩展来使用代理故障切换机制，请按照以下过程配置主企业管理器收集器和后备企业管理器收集器。

在故障切换的情况下，CA APM ChangeDetector 代理扩展开始将更改数据发送到指定为后备企业管理器的企业管理器。 但是，由于后备企业管理器上的更改数据库 (changes.db) 与主企业管理器不同步，CA APM ChangeDetector 将关闭。但是，CA Introscope 代理将继续运行，且 CA APM ChangeDetector 会重试主（默认）企业管理器，直到它恢复运行。

注意：如果 CA Introscope 代理直接连接到 MOM，将不会报告更改。

为 CD 配置代理故障切换机制：

在 *IntroscopeAgent.profile* 中配置以下属性：

- 将代理的主企业管理器收集器（不是 MOM）指定为启用 CA APM ChangeDetector。 设置以下属性：

  ```
  introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=<主 EM 收集器主机>
  introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=<主 EM 收集器端口>
  introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
  ```

- 将 MOM 企业管理器指定为后备企业管理器。 设置以下属性：

  ```
  introscope.agent.enterprisemanager.transport.tcp.host.FALLBACK =<MOM EM 收集器主机>
  introscope.agent.enterprisemanager.transport.tcp.port.FALLBACK =<MOM EM 收集器端口>
  introscope.agent.enterprisemanager.transport.tcp.socketfactory.FALLBACK=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
  ```

- 通过设置以下属性提供连接顺序和重试时间间隔：

```
introscope.agent.enterprisemanager.connectionorder=DEFAULT, FALLBACK
introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds=
<故障切换重试时间（秒）>
```

有关代理属性的详细信息，请参阅*《CA APM Java 代理实施指南》*或*《CA APM .NET 代理实施指南》*。

在 .NET 中运行具有各自配置文件的多个应用程序

如果用户要在 .NET 中的 IIS 下运行多个应用程序，且希望每个应用程序分别报告其更改，则必须在以逗号分隔的列表中标识每个应用程序的配置文件夹。 每个应用程序都必须有自己的 ChangeDetector 配置文件夹。此文件夹包含每个应用程序的 ChangeDetector 配置 XML。 路径中最终文件夹的名称应部分匹配应用程序名称（AppDomain 友好名称）。

例如，如果要监控有各自配置文件夹的 BalloonShop 和 Petshop 的更改，应如以下示例所示配置 *introscope.changeDetector.profileDir*：

```
introscope.changeDetector.profileDir=S:\sw\CA_Wily\wily_dotnet\Introscope<版本
号>\wily\CD-config\balloonshop,S:\sw\CA_Wily\wily_dotnet\Introscope<版本
号>\wily\CD-config\petshop
```

禁用 CA APM ChangeDetector

禁用 CA APM ChangeDetector：

将以下属性设置为 false：

```
introscope.changeDetector.enable=false
```

ChangeDetector 代理 ID 命名选项

安装的每个 CA APM ChangeDetector 代理扩展的 ChangeDetector 代理 ID 都必须是唯一的。 在只有一个 CA Introscope 代理的简单环境中，可以让 CA APM ChangeDetector 自动为您分配 ID。

在较复杂的环境中，可能需要使用其他方法来获取唯一的 ChangeDetector 代理 ID。 您的选择取决于：

- 您是否有多个启用了 CA APM ChangeDetector 的代理
- 是每个代理都有唯一的配置文件 (IntroscopeAgent.profile) 还是所有代理都使用同一个配置文件

- 是否要对 CA APM ChangeDetector 代理扩展使用自动 ID 生成
- 是使用 java 系统属性、.NET 环境变量监视器还是代理属性来标识 CA APM ChangeDetector 代理扩展

注意：如果有多个 ID 相同的 CA APM ChangeDetector 代理扩展运行于同一个 CA Introscope 代理上，启动的第二个实例将自我关闭并显示一条错误消息。

下表汇总了一些建议的选项。

环境	建议的代理 ID 设置
每个启用了 CA APM ChangeDetector 的代理均使用位于唯一目录中的唯一代理配置文件。	■ 使用 CA APM ChangeDetector 的自动 ID 分配。 对此无需任何操作。
多个启用了 CA APM ChangeDetector 的代理使用同一个代理配置文件	■ 在 Java 启动命令中，使用 -D 参数指定 *introscope.changeDetector.agentID* 属性。 ■ 在 Java 命令行或 *IntroscoptAgent.profile* 中，使用将在运行时解析的、基于任何有用的 Java 系统属性、.NET 环境变量监视器或代理配置文件属性的表达式。

详细信息：

使用 -D Java 系统参数来分配 ID (p. 46)
基于 Java 系统或代理配置文件属性分配 ID (p. 46)

使用自动 ID 分配

仅当每个启用了 CA APM ChangeDetector 的代理均有一个 IntroscopeAgent.profile 时，才可以使用自动 ID 分配。

在这种情况下，如果未在 Java 系统命令或 *IntroscopeAgent.profile* 文件中定义属性 *introscope.changeDetector.agentID*，则 CA APM ChangeDetector 会为该属性生成一个唯一值。 该值存储于 CA APM ChangeDetector 根目录的“.id”文件中。 不应删除或修改该文件。

注意：默认情况下，在包含 *IntroscopeAgent.profile* 文件的目录中创建 CA APM ChangeDetector 根目录 (change_detector)。 但是，可以定义属性 *introscope.changeDetector.rootDir* 来指定不同的根目录。

使用 -D Java 系统参数来分配 ID

可以在 Java 命令行中定义 CA APM ChangeDetector 代理扩展的 ID。为此，请将 *introscope.changeDetector.agentID* 属性与 -D 参数一起使用，其配置如下例所示：

```
java -Dintroscope.changeDetector.agentID=<value>
```

基于 Java 系统或代理配置文件属性分配 ID

通过使 ID 基于任意 Java 系统属性或代理配置文件属性，可以在运行时动态分配 ChangeDetector 代理 ID。如果您具有复杂的环境并且已使用其他某个属性来区分不同的进程，可能需要使用此方法。

基于其他属性分配 ChangeDetector 代理 ID（选项 1）：

- 在 *IntroscopeAgent.profile* 文件中，为代理 ID 属性输入一个表达式，如下面的示例所示，其中前缀和后缀为简单常量字符串：

  ```
  introscope.changeDetector.agentID=<前缀>${任意 Java 系统属性}<后缀>
  introscope.changeDetector.agentID=<前缀>${任意代理配置文件属性}<后缀>
  ```

  可以使用前缀或后缀，或同时使用两者。

基于其他属性分配 ChangeDetector 代理 ID（选项 2）：

- 在 Java 启动命令中，使用与上面类似的一个表达式，如以下示例所示，其中前缀和后缀为简单常量字符串：

  ```
  java -Dintroscope.changeDetector.agentID=<前缀>${任意 Java 系统属性}<后缀>
  java -Dintroscope.changeDetector.agentID=<前缀>${任意代理配置文件属性}<后缀>
  ```

  可以使用前缀或后缀，或同时使用两者。

可选的配置属性

CA APM ChangeDetector 包括可选属性，即您可以选择在 CA Introscope 代理、Workstation 和企业管理器配置文件中设置的属性。

可选的代理属性

在 *IntroscopeAgent.profile* 中设置以下属性：

- *introscope.changeDetector.rootDir*

  指定 CA APM ChangeDetector 根目录的位置或将创建 CA APM ChangeDetector 根目录（如果尚不存在）的位置。如果未指定，则根目录默认为 *<Agent_Home>*

  注意： CA APM ChangeDetector 使用根目录来创建正常处理所需的文件，因此不得删除此目录。 定义目录位置的属性是可选的。

- *introscope.changeDetector.compressEntries.enable*

  如果该值设置为 false，将禁用数据压缩。 数据压缩在 CA APM ChangeDetector 数据缓冲区中启用压缩功能。 如果启动时消耗内存，则此属性很有用。 默认值为 *true*。 如果设置为 *false*，则必须重新启动应用程序才能应用该设置。

- *introscope.changeDetector.compressEntries.batchSize*

  此属性是 *introscope.changeDetector.compressEntries.enable* 属性附带的。 它定义压缩作业的批处理大小。 默认值为 1000。

- *introscope.changeDetector.isengardStartupWaitTimeInSec*

  这是默认属性。 它让您可以指定代理启动后、尝试连接到企业管理器之前所等待的时间（秒）。 默认值为 15。

- *introscope.changeDetector.waitTimeBetweenReconnectInSec*

  这是默认属性。 它使您可以指定代理尝试重新连接到企业管理器之前的时间（秒）。 默认值为 10。

可选的 Workstation 属性

可以在 *IntroscopeWorkstation.properties* 中设置以下属性：

- *introscope.changeDetector.defaultDataWindowValue=<value>*

  指定在调查器中查看实时数据的时间长度。 此属性与 *defaultDataWindowUnit* 属性配合使用，后者指定查看实时数据的时间长度的单位值。 默认值为 1。

- *introscope.changeDetector.defaultDataWindowUnit*=<value>

  指定在调查器中查看实时数据的时间长度单位。 此属性与 *defaultDataWindowValue* 属性配合使用。此属性的有效值为 *秒*、*分钟*、*小时*、*天*、*周*和 *月*。 默认值为 *周*。 指定天数或月份数时使用的实际起始日期取决于区域设置，例如，在某些区域设置中一周从星期日开始，而在其他区域设置中则从星期一开始。

- *introscope.changeDetector.useCh*angeTime=false

  默认情况下，CA APM ChangeDetector 显示检测到文件更改的时间，而不显示更改时间。 要同时显示检测时间和更改时间，请将此属性设置为 *true*。 此属性仅适用于文件更改，而不适用于系统属性、环境变量、数据库或 java 类更改。

配置 EPAgent 插件以发送 CA APM ChangeDetector 数据

注意：这仅适用于 Java 平台。

如果将 CA APM ChangeDetector 用于 EPAgent 插件或扩展，请配置 EPAgent。

CA APM ChangeDetector 使用显示到 STDOUT 的 XML 将其数据发送到企业管理器，因此必须设置 EPAgent 数据的格式，以便 CA APM ChangeDetector 可以理解：

```
<changeData dataSource="dataSource name">
<resource name="resource1" value="resource1 value"/>
<resource name="resource2" value="resource2 value"/>
</changeData>
```

注意：将 ChangeDetector XML 显示到 STDOUT 时，XML 必须位于一行（上面的示例 XML 采用多行格式是为了便于阅读）。 当您以单行显示到 STDOUT 时，XML 将如下所示：

<changeData dataSource="dataSource name"><resource name="resource1" value="resource1 value"/><resource name="resource2" value="resource2 value"/></changeData>

每个 changeData 元素都必须定义 dataSource 属性。 这是您在查看 CA APM ChangeDetector 数据时将在 CA Introscope Workstation 中看到的名称。 每个 changeData 元素都链接到特定的数据源。 如果要报告多个数据源的 CA APM ChangeDetector 数据，必须分别在单独的行中显示每个 changeData 元素。

changeData 元素可包含任意数量的 resource 元素。 每个 resource 元素都必须定义两个属性－与所报告的资源名称相对应的 name 属性；与所报告的资源的当前值相对应的 value 属性。

每次显示 changeData 元素时，都必须包括所有已知资源的当前状态：

- 如果某个资源在您上次显示该资源的 changeData 元素时未包括，而这次您打算包括该资源，应考虑将该资源添加到系统中。
- 如果资源的 value 属性在两次迭代间发生更改，则该资源被视为已修改。
- 如果某个资源在您上次显示该资源的 changeData 元素时已包括，而这次您不打算再包括该资源，应考虑将该资源从系统中删除。

可选的企业管理器属性

可以在 *IntroscopeEnterpriseManager.properties* 中设置以下属性：

- *introscope.changeDetector.storage.purge.maxDataAgeInDays=*<值>

  指定更改可保存的最长时间（天），之后将从数据存储中删除。 默认值为 90。

- *introscope.changeDetector.storage.purge.offsetHour=*<值>

  指定发生清除（如果启用）的时间。 如果保留为空，则默认为 4（凌晨 4 点）。

- *introscope.changeDetector.storage.perst.dbfile=*<存储文件的路径>

  指定 ChangeDetector 数据所使用的存储文件的位置。 该位置可以是相对路径，也可以是完全指定的路径。 默认值为 *{$EM_INSTALL}/data/changes.db*。

- *introscope.changeDetector.jdbc.maxNumRows=*<值>

  指定允许 SQL 查询处理的最大行数。 执行的查询将以行格式返回已经报告给企业管理器的所有更改。更改是从 *changes.db* 文件报告的。默认值为 10000。

第 3 章： 查看 CA APM ChangeDetector 数据

CA APM ChangeDetector 直接与 CA Introscope 集成，它对 CA Introscope 进行了扩展，使您可以轻松监控应用程序环境中的更改。

安装 CA APM ChangeDetector 后，可在 CA Introscope 中查看这些更改：

- 在 CA Introscope 控制台中，CA APM ChangeDetector 显示板显示更改数据的高级视图。
- 在 Workstation 中，“更改”选项卡以分层树视图或表视图的形式显示更改数据。
- 在 Workstation 中，通过更改查看器，您可以设置选项来指定要查看的更改。
- 在 CA Introscope 度量标准图和显示板中，通过注释来标识更改事件。
- 在 CA Introscope 报表中，可以查看更改历史。

注意：如果在所有 CA Introscope 代理均连接到企业管理器之前连接到 MOM，则 Workstation 中显示的更改数和 CA Introscope 代理数将不正确。单击其他节点或打开其他调查器，几分钟后，该问题即更正。

此部分包含以下主题：

在 CA Introscope 中查看更改数据

安装和配置 CA APM ChangeDetector 时，可标识要监控的 CA Introscope 代理，并指定 CA APM ChangeDetector 要收集的更改数据类型。 还可以启用 Workstation 来查看更改数据和创建报告。

在 Workstation 中，可以在以下位置查看更改数据：

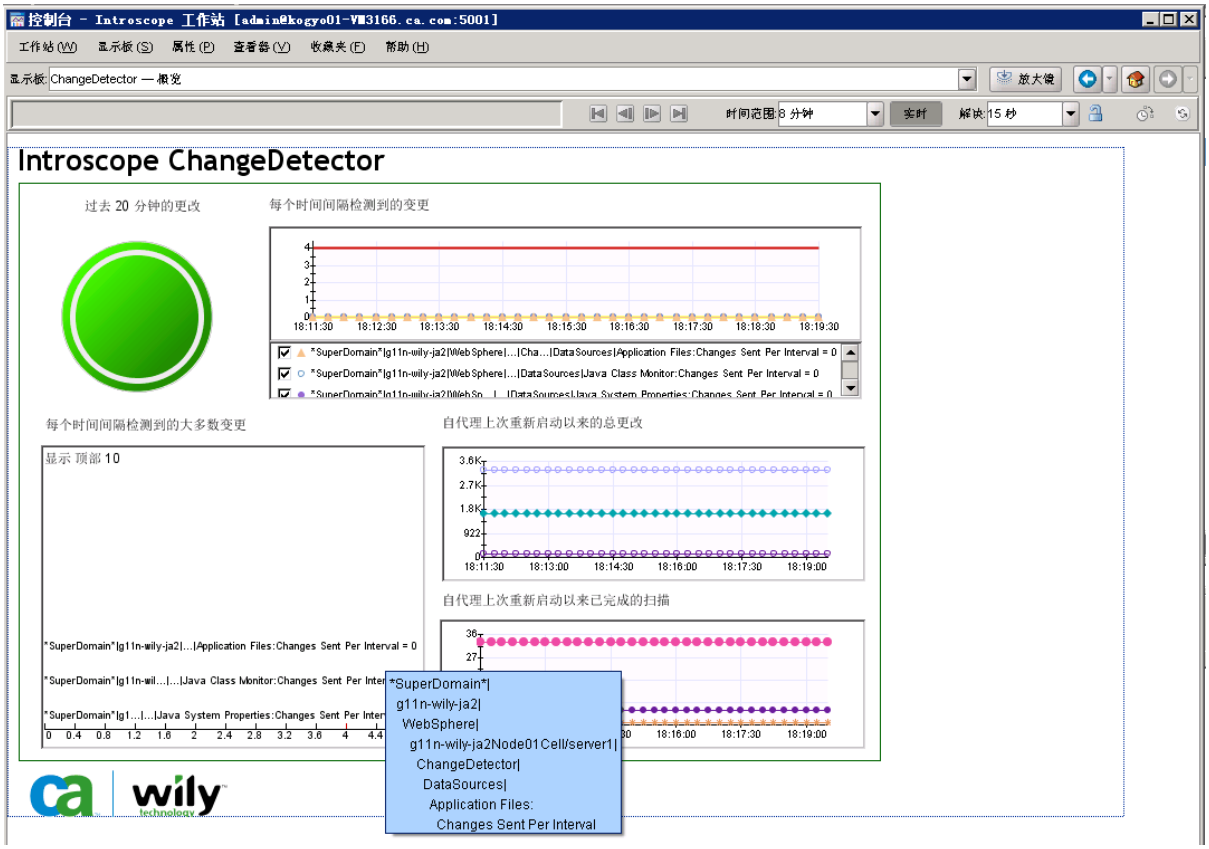
- CA APM ChangeDetector 显示板 (p. 52)
- 调查器中的树视图 (p. 53)。
- 调查器中的表视图 (p. 59)。 表视图显示的所选资源的信息与树视图中显示的信息相同，但采用的格式不同。

调查器中的“更改”表包括“启动更改查看器”按钮。 通过更改查看器中的选项，您可以更好地控制要查看哪些更改数据。

注意：如果在所有 CA Introscope 代理均连接到企业管理器之前连接到 MOM，则 Workstation 中显示的更改数和 CA Introscope 代理数将不正确。单击其他节点或打开其他调查器，几分钟后，该问题即更正。

在 CA APM ChangeDetector 显示板中查看更改数据

CA Introscope 控制台包括一个 CA APM ChangeDetector 显示板，其中显示更改数据的高级视图：



显示板包括更改数据的以下视图：

- 报警指示器显示过去 20 分钟的更改。

  注意：CA APM ChangeDetector 检查过去 20 分钟是否有更改。 但是，这不是指过去 20 分钟的*累积*值。 例如，如果过去 20 分钟有 10 个更改，但其中的每个更改分别是按不同的时间间隔发送的，指示灯将显示为黄色。 只有当一个时间间隔内有 5 个或 5 个以上的更改时，指示灯才会显示为红色。

- 图表显示自 CA Introscope 代理上次重新启动以来每个时间间隔检测到的更改。
- 图表显示自 CA Introscope 代理上次重新启动以来完成的扫描数。

打开 CA APM ChangeDetector

安装 CA APM ChangeDetector 时，CA Introscope Workstation 包括“更改”选项卡。

打开 CA APM ChangeDetector：

- 打开新的 CA Introscope 调查器并单击“更改”选项卡。 CA APM ChangeDetector 在树视图模式下打开。

在树视图中查看更改数据

调查器中的 CA APM ChangeDetector 树视图以分层形式显示更改数据，并首先按根节点*更改*然后按日期和时间对这些数据进行分组。 树视图是默认调查器视图。

还可以通过单击 CA Introscope 调查器树中特定代理的 ChangeDetector 节点或单击 CA Introscope 代理的任意 ChangeDetector 数据源来查看该代理的更改数据。

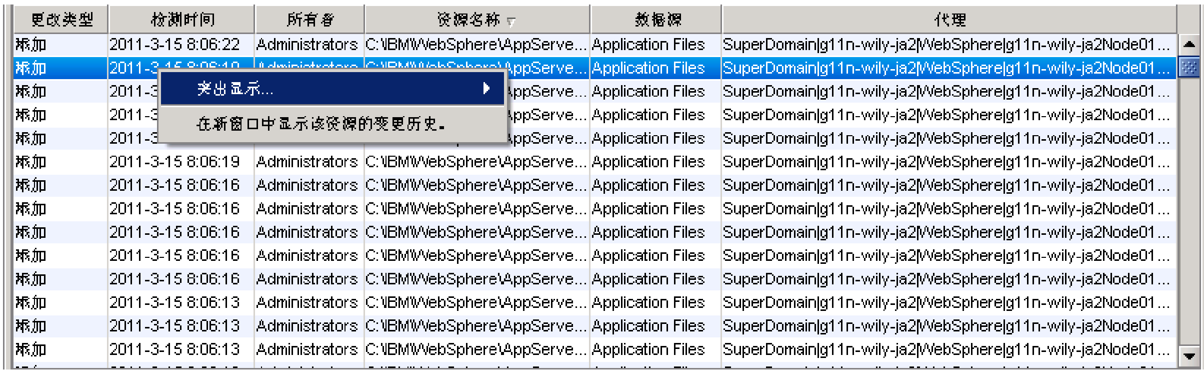
如果 ChangeDetector 数据源关闭，将不再能在“常规”选项卡中查看度量标准数据，但可以继续在“更改”选项卡中查看更改数据。

Introscope“更改”选项卡显示最近一周的更改数据，以及会话过程中显示的实时数据。选择其他时间范围时，您将看到该时间范围内的更改数据。

可以通过使用 ChangeDetector 配置参数来修改显示更改数据的默认时间段—请参阅第 13 页的*安装和配置 ChangeDetector*。

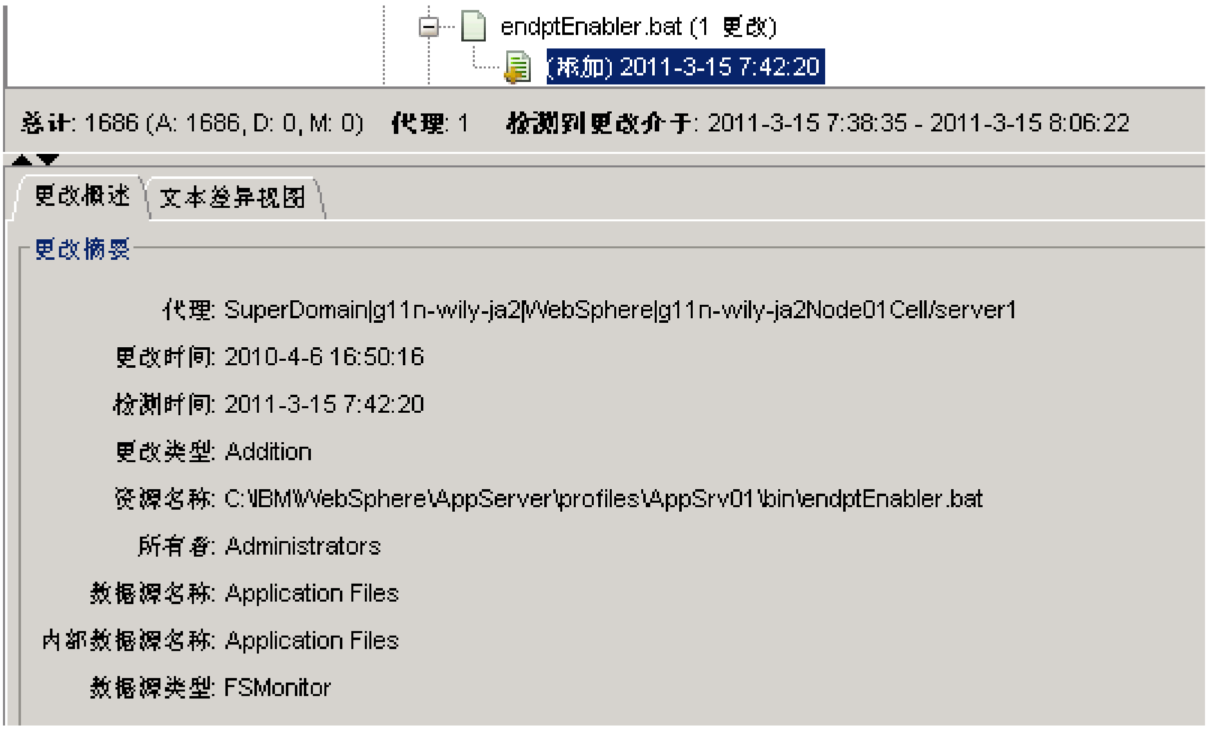
可以按时间或资源对树视图中的更改数据进行分组：

如果单击包含多个更改的节点，底部窗格将以表格形式列出该节点下的所有更改事件。

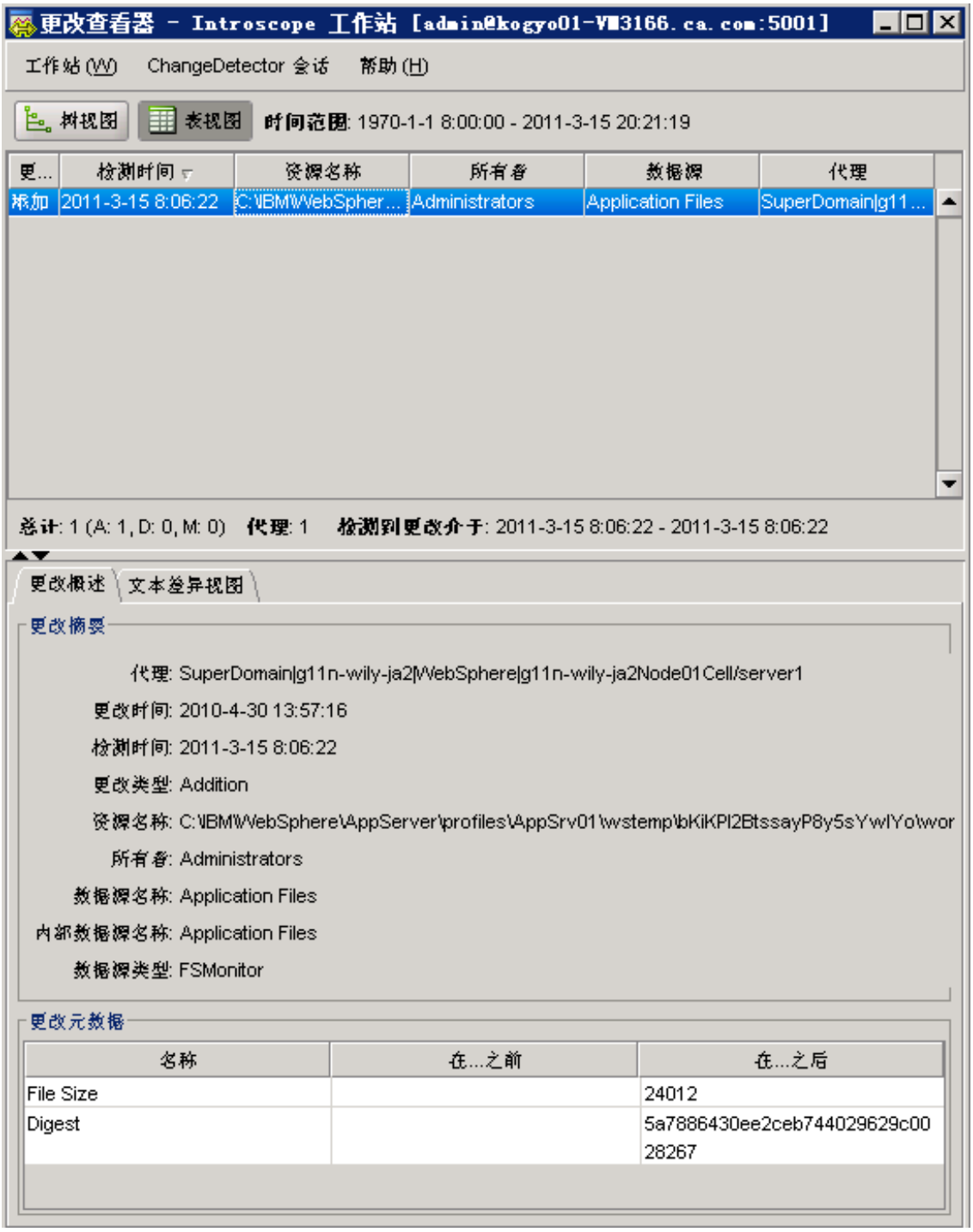


右键单击底部窗格中的更改事件可进一步选择查看选项：

- 选择“突出显示”可使用表格突出显示来更轻松地查看特定类型的更改（请参阅使用表格突出显示 (p. 60)）。
- 单击“在父视图中选择此更改”将使树视图突出显示此更改，并在“更改概述”选项卡中显示特定更改数据：



- 单击“在新窗口中显示该资源的更改历史”将在一个单独的窗口中打开表视图，并显示所选资源的更改历史。



在“更改概述”选项卡中查看信息

只要在树中选择了一个更改，就会打开“更改概述”选项卡。 “更改概述”选项卡始终包括摘要面板。 还可能包含元数据和详细信息面板：

- **更改摘要**

  每当您选择了一个更改时，就会显示在“更改概述”选项卡中。摘要显示有关更改的基本信息－代理 ID、更改时间（仅适用于文件－请参阅下面的注释）、检测时间、更改类型、资源和数据源名称以及数据源类型。

- **更改元数据**

  只要更改数据包括元数据，就会显示在“更改概述”选项卡中。 例如，对于文件更改，元数据包括上次修改时间和文件大小。

- **更改详细信息**

  仅当更改详细信息可显示在前面/后面的表中时，才会显示在“更改概述”选项卡中－例如，环境属性、数据库属性和 java 类等数据类型。

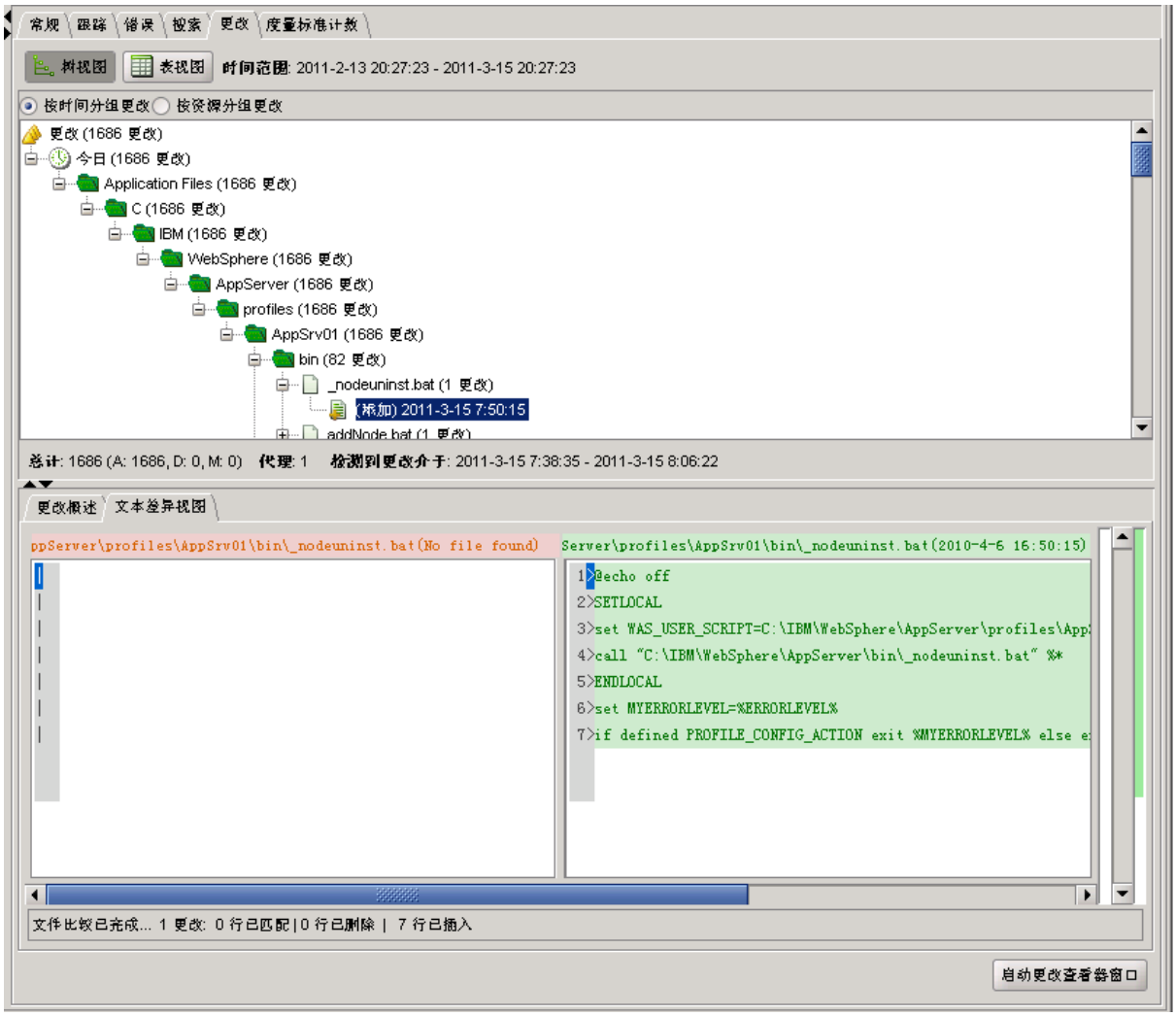
注意：对于文件，如果设置了可选的 Workstation 属性，CA APM ChangeDetector 可区分检测到更改的时间和上次修改文件的时间（请参阅可选的配置属性 (p. 46)）。首次启动启用了 CA APM ChangeDetector 的代理时，如果文件的上次修改时间在实时时间窗口（默认为 7 天）之前，则实时模式下可能不会显示文件的添加事件。但是，可以通过在“时间范围”下拉列表中选择其他时间范围，来访问当前时间窗口中不可见的数据。

在某些情况下，如果“检测时间”和“更改时间”之间的时差长于完全扫描所需的时间，则 CA APM ChangeDetector 可能需要较完全扫描更长的时间来检测文件更改。 发生这种情况的原因有很多。 例如，文件可能重命名，覆盖了现有文件。CA APM ChangeDetector 将此情况视为已覆盖文件的修改更改事件，但文件的上次修改时间戳仍是重命名之前的时间戳（因操作系统而异），从而造成了此影响。 此外，在某些操作系统中，用户可使用系统实用工具修改文件的上次修改时间戳，这可能会导致同样的影响。

要配置 CA APM ChangeDetector 以显示更改时间，请参阅可选的配置属性 (p. 46)。

在“文本差异视图”选项卡中查看信息

可使用 ChangeDetector 的“文本差异视图”选项卡来识别文本文件内容的差异。



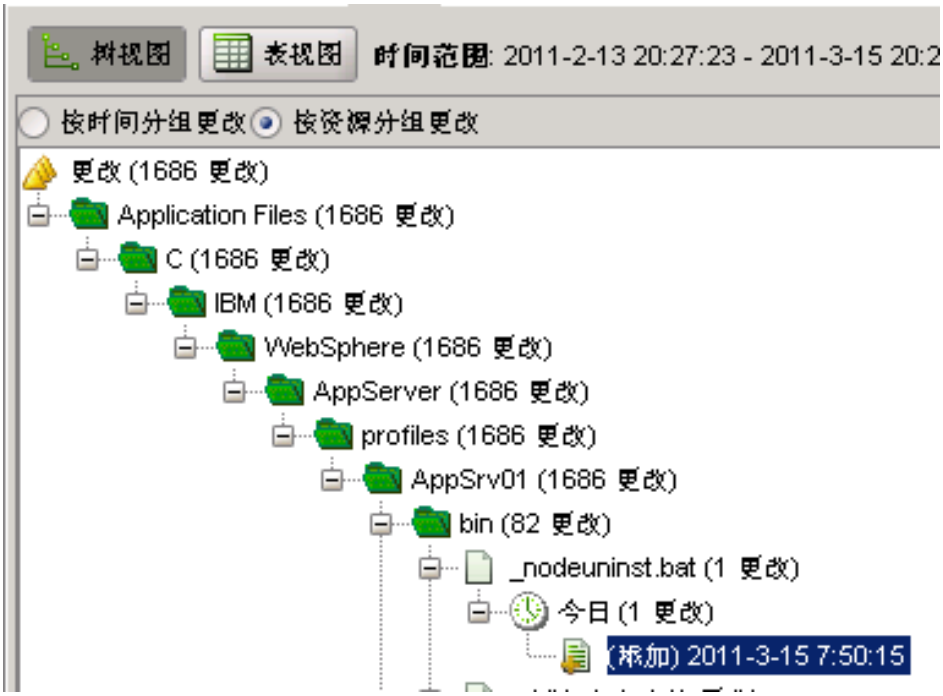
当您选择以下更改时，将在 ChangeDetector 的树视图或表视图中显示“文本差异视图”选项卡：

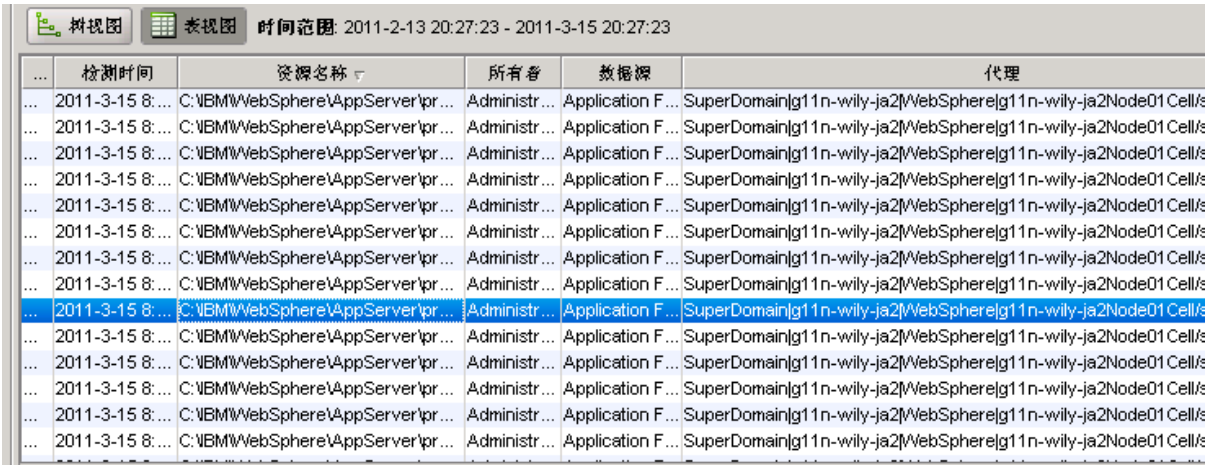
- 文本文件（例如，配置文件）的更改，且
- 文件大小小于配置文件 (*ChangeDetector-config.xml*) 中定义的 *maxFileSizeToUpload* 属性值。

大小超过 maxFileSizeToUpload 值的文本文件在 Workstation 的文本差异视图中不可见。

在表视图中查看更改数据

在调查器中选择某个资源时，可以在树视图或表视图中查看该资源的更改数据。 在树视图中选择某个更改事件（如删除或修改）时，表视图将对显示所选事件详细信息的行突出显示。

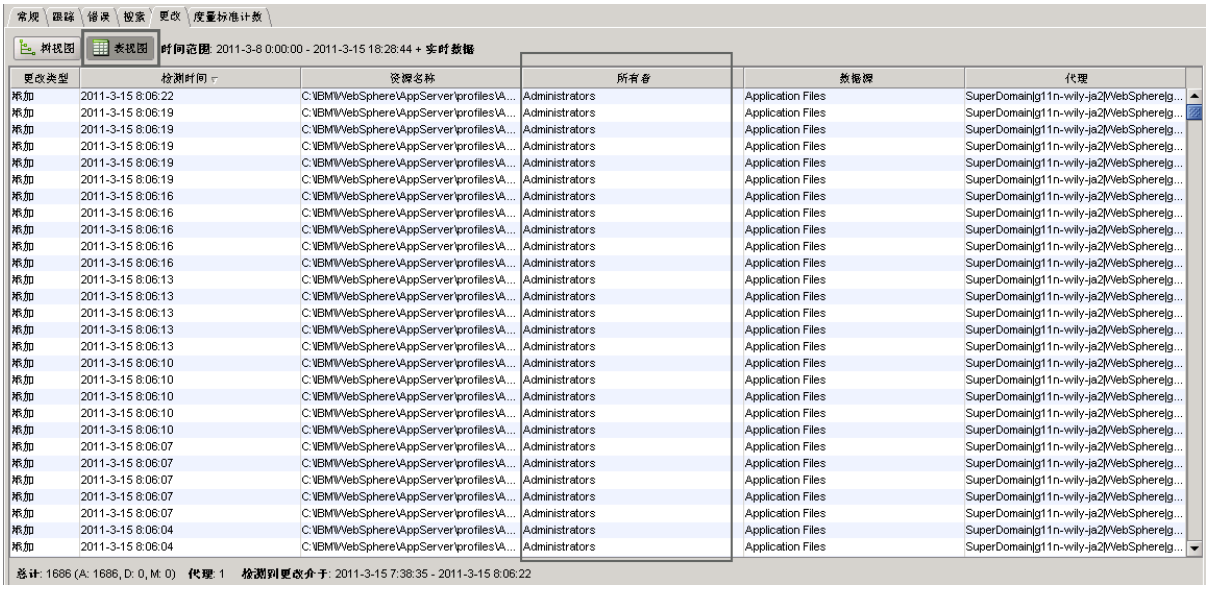




使用表视图对显示在不同轴（如更改类型、检测时间、资源名称、数据源和代理 ID）上的数据进行排序。

在表视图中查看所有者数据

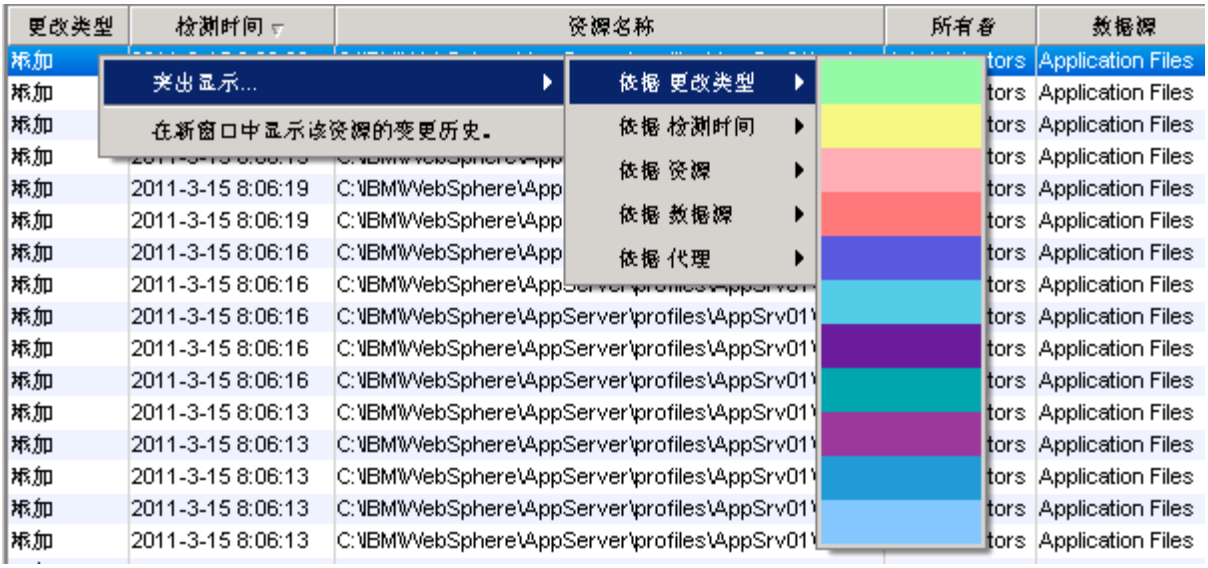
CA APM ChangeDetector 识别提交文件系统监视器数据源更改的用户。在表视图中查看文件系统数据源时，将会在“所有者”列中显示用户标识信息：



如果查看其他任何数据源，“所有者”列虽然显示在表视图中，但不含任何数据。

使用表格突出显示

为了便于查看特定类型的更改数据，可使用颜色在表视图中按更改类型、检测时间、资源、数据源或代理突出显示事件。



应用突出显示：

1. 选择具有要突出显示的特征的事件－例如，特定资源中的事件或修改事件。
2. 右键单击该事件，选择“突出显示”及要突出显示的特征：按更改类型、按检测时间、按资源、按数据源或按代理。

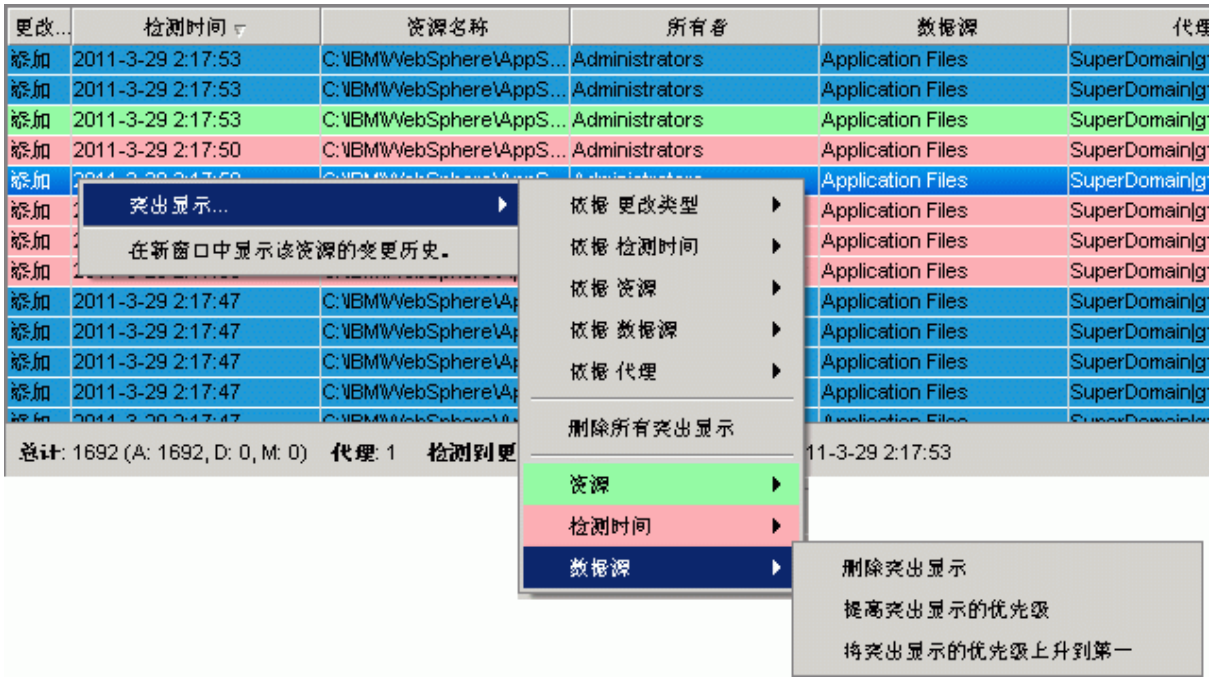
   如果选择“按更改时间”，还必须指定日期范围。
3. 选择一种突出显示颜色。

   现在具有所选特征的全部事件均突出显示。

使用突出显示

- 突出显示按它们的定义顺序的优先级应用，且不重叠。 例如，如果按更改类型定义红色突出显示，则该更改类型的所有事件均以红色突出显示。 如果之后按资源定义黄色突出显示，则以红色突出显示的事件将不会改变颜色。 黄色突出显示将应用于匹配所选资源的余下事件。
- 可以更改任何突出显示的优先级顺序。
- 可以分别删除每个突出显示，也可以一次删除所有突出显示。

  请注意此示例中的“突出显示”菜单。 定义了三个突出显示。 “按资源”、“按检测时间”和“按数据源”。

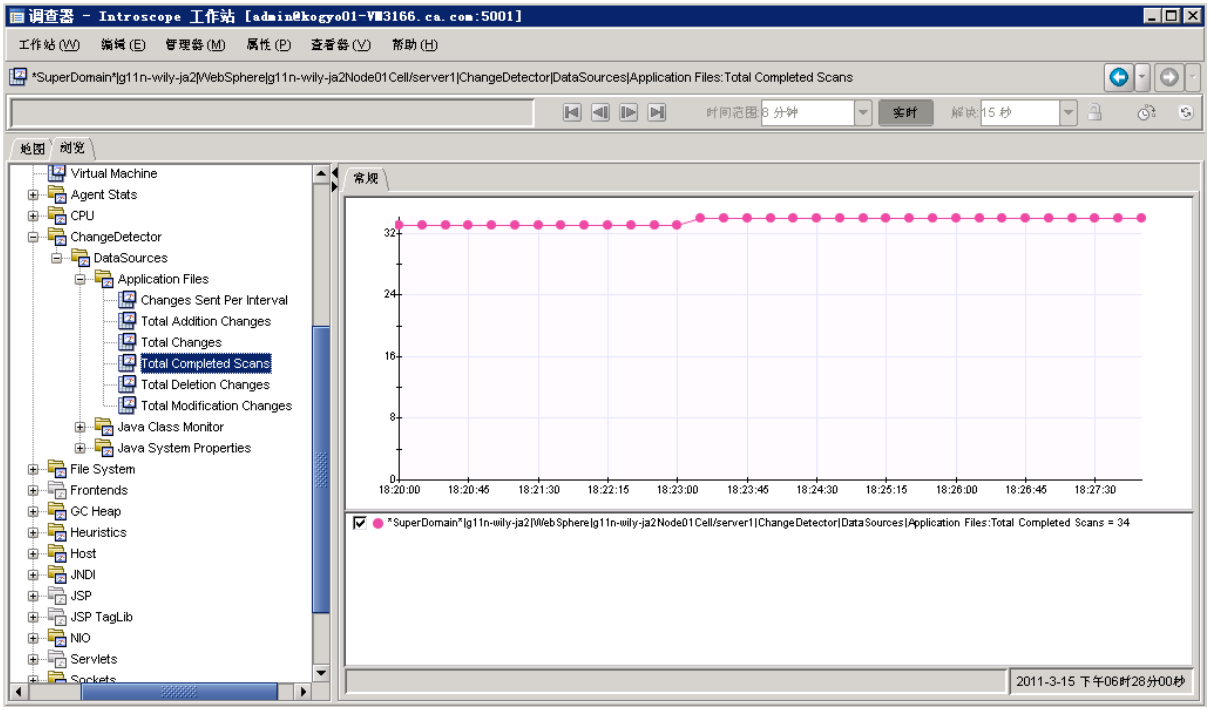


在图表和报表中查看更改数据

CA APM ChangeDetector 将有关更改数据的信息集成到 CA Introscope 图表中，并包括一个报表模板来显示过去 24 小时系统中发生的所有更改。

集成的 CA APM ChangeDetector 图表

CA APM ChangeDetector 将有关更改数据的信息集成到 CA Introscope 图表中。 更改数据在图表的 X 轴下方及工具提示中显示为图表注释，如下图所示：



查看 CA APM ChangeDetector 图表注释

CA APM ChangeDetector 图表注释使用与图例一致的符号和颜色，以便您可以轻松识别更改的起源。 下面是 ChangeDetector 图表注释的一些特征：

- 将鼠标指针悬停于注释上将打开显示详细信息的工具提示。

  由于多个图表注释可能会因相互靠得太近而不易识别，因此 CA APM ChangeDetector 显示最多 5 个注释的工具提示，并将其按从上到下的顺序摆放，以便您可以同时看到全部 5 个注释。 如果注释超过 5 个，窗口顶部将显示一条消息通知您总数目，并通知您只显示前 5 个注释。

- 双击图表注释将打开更改查看器，它列出了在该注释任意一侧的两条网格线之间发生的更改，并突出显示与该注释本身相对应的事件。
- 图表中显示的注释取决于图表中显示哪些度量标准。 例如，如果图表中的度量标准来自 CA Introscope 代理 A、B 和 C，而只有代理 A 和 B 启用了 CA APM ChangeDetector，将只显示指定时间段内来自代理 A 和 B 的更改注释。
- 如果指定时间段内没有更改，将不显示更改数据且图表底部的更改注释不可见。

有关 CA Introscope 图表的详细信息，请参阅 *《CA APM Workstation 用户指南》*。

指定显示哪些图表注释

可以指定在图表中显示哪些注释。 例如，您可能只需要查看来自特定数据源或资源的注释，或者需要将一个 CA Introscope 代理收集的更改与另一个 CA Introscope 代理收集的度量标准相关联。

可通过两种方法指定显示哪些更改：

- 在调查器中，导航到“属性”>“图表注释设置”。
- 右键单击该图表并选择“图表注释设置”。

指定图表中的注释：

1. 打开“图表注释设置”对话框。
2. 选择“默认设置”或“使用自定义设置”：

   默认设置

   仅为在图表中显示度量标准的启用了 CA APM ChangeDetector 的代理显示图表注释。

   例如，如果图表中的度量标准来自 CA Introscope 代理 A、B 和 C，而只有代理 A 和 B 启用了 CA APM ChangeDetector，将只显示指定时间段内来自代理 A 和 B 的更改注释。 图表中不显示来自代理 D 的更改（不论它是否启用了 CA APM ChangeDetector），因为该图表不包括来自该代理的度量标准。

   自定义设置

   选择要标注图表注释的 CA APM ChangeDetector 组件。

   注意：注释选项的更改仅在当前图表中应用。 如果离开了该图表，将不保留这些更改。

运行内置的 CA APM ChangeDetector 报表

CA APM ChangeDetector 包括一个报表模板，用于显示过去 24 小时系统中发生的所有更改。 此报表按 CA Introscope 代理分组，并包括每个代理的摘要和报表摘要。

运行*过去 24 个小时的更改*报告：

1. 选择“Workstation”>“生成报告”来打开“选择报告模板”对话框。

   列表中包括*过去 24 个小时的更改*报告模板。

2. 选择“过去 24 个小时的更改”并单击“选择”来打开“生成报告”对话框。
3. 指定报告的开始日期和结束日期。

   注意：报告的时间范围是根据生成报告的 Workstation 的时区来计算的。

4. 从启用了 CA APM ChangeDetector 的代理列表中选择一个代理，或指定其他代理表达式来覆盖代理表达式模板。
5. 单击“生成预览”。

   “预览”显示报告标题页。

6. 使用“预览”按钮可操作报告输出并保存报告。

将 CA APM ChangeDetector 元素添加到 CA Introscope 报表

除了内置的*过去 24 个小时的更改*报表模板之外，CA APM ChangeDetector 还包括 CA APM ChangeDetector 的报告功能，这样您便可以通过向报表中添加 CA APM ChangeDetector 元素来创建自定义报表。

创建自定义 ChangeDetector 报表：

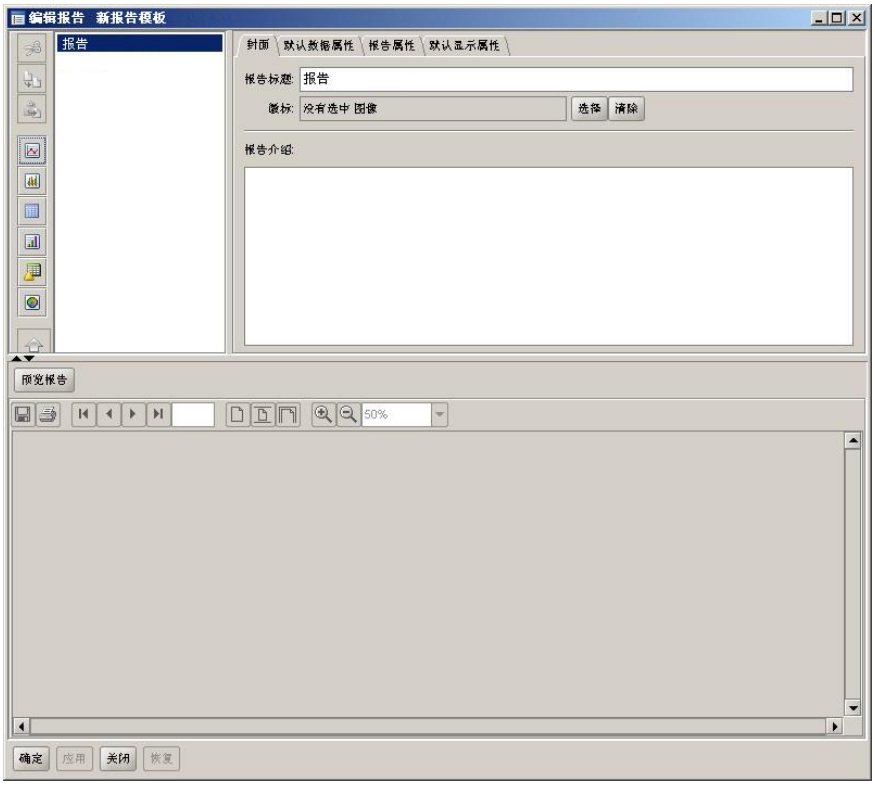
1. 在 Workstation 调查器中，选择“文件”>“新建管理模块编辑器”。
2. 导航到“元素”>“新建报告模板”。
3. 键入新报告模板的名称，并单击“强制唯一性”来确保报告名称是唯一的。

   选中“强制唯一性”后，如果您指定的名称不唯一，CA Introscope 将通过在此名称后追加一个数字将其改为唯一名称。创建报告模板之后，可在管理模块编辑器中看到这一追加数字。 如果未选中“强制唯一性”，当存在相同的报表模板名称时，CA Introscope 将显示一条错误消息但不会创建报表。
4. 选择要包括在报告中的管理模块，然后单击“确定”。

   此时将打开“新建报告模板”对话框。
5. 单击“活动”复选框以激活报表模板，并使其显示在 CA Introscope 控制台、调查器和管理模块编辑器的报表模板列表中。

6. 单击“打开模板编辑器”。

   报告模板打开，显示将出现在报告目录中的可用选项：



   注意：可以只包括更改数据的报告选项，也可以将该选项与度量标准图表一起添加到报告中，以显示该报告时限内发生的更改。

7. 单击“封面”选项卡指定报告的用途：

   报告标题

   为生成的报告键入一个标题；该标题与目录一起显示在标题页。

   徽标

   选择要与该报告相关联的徽标。

   报告介绍

   键入描述报告简要概述的文本。

8. 单击“默认数据属性”选项卡指定属性。

   默认情况下，数据属性不可修改。 但是，可以选择关闭“使用更改时间”来显示检测到的更改，而非文件上的实际更改时间。

   要修改默认的时间范围，请在“模板默认时间范围”下更改如下字段：

   开始时间和结束时间

   指定时间范围时，可指定特定开始日期和结束日期，也可以指定时间段，如*“24 小时”*。

可以采用以下方法之一指定报告的时间范围：

a. 键入特定开始和结束日期和时间，或单击日历图标选择开始和结束日期。

b. 将“开始时间”保留为空，而使用“持续时间”和“单位”参数来指定报告运行的时长。

c. 将“结束时间”保留为空，而使用“持续时间”和“单位”参数来指定报告运行的时长。

d. 在“结束时间”中键入立即，并使用“持续时间”和“单位”参数来指定要报告的历史长度。

Duration

键入一个数以指定报告运行的时长。此数字与“单位”值结合使用－例如，如果“单位”为 *小时*，您可能在“持续时间”中键入 *24*。

注意：有关“持续时间”和“单位”参数如何与“开始时间”和“结束时间”配合使用的说明，请参阅对“开始时间”和“结束时间”的解释。

单位

从下拉列表中选择一个时间单位。 设置包括 *分钟*、*小时*、*天*或 *周*。

9. 单击“显示属性”选项卡，通过设置相应的属性来决定报告中的图形和表格在报告生成后的外观：

排序依据

单击该行打开列表，然后从“时间戳”、“数据源”、“资源”或“更改类型”中进行选择。 报告中的所有更改最初都按代理 ID 进行分组。 “排序依据”字段是每个代理中的次级分组。

排列顺序

单击该行打开列表，然后选择“升序”或“降序”。

最大行数

键入可在报告中显示的最大行数。

有关创建和生成报告的详细信息，请参阅 *《CA APM Workstation 用户指南》*。

第 4 章： CA APM ChangeDetector 度量标准

CA APM ChangeDetector 报告企业管理器和 CA Introscope 代理的支持能力度量标准。

此部分包含以下主题：

企业管理器的 CA APM ChangeDetector 支持能力度量标准

在虚拟代理下的*企业管理器|ChangeDetector|DataStore* 节点下可以找到企业管理器的支持能力度量标准。

有关度量标准的详细信息，请参阅以下主题：

平均插入时间（毫秒）

将一个 CA APM ChangeDetector 数据点插入企业管理器端数据库的平均时间（毫秒）。

使用的数据存储 (%)

分配用于存储 CA APM ChangeDetector 数据的数据库空间百分比。 如果此值达到 100% 并且插入另一个数据点，则数据库需要更多空间。

插入数

已插入数据库的 CA APM ChangeDetector 更改数。

已知代理数

针对企业管理器报告其数据的 CA APM ChangeDetector 而启用的 CA Introscope 代理的数量。 此值因用户权限和代理连接情况而异。

更改数（数据库）

数据库中当前存储的 CA APM ChangeDetector 更改数。

数据存储大小

数据存储的大小（字节）。

注意：由于企业管理器弹出时度量标准会归零，因此*平均插入时间（毫秒）*和*插入数*度量标准值仅在企业管理器的生命周期内有效。

更改数 (CA Introscope)

从每个 CA Introscope 代理发送的 CA APM ChangeDetector 更改数。

CA Introscope 的 CA APM ChangeDetector 支持能力度量标准

如果扫描已完成或数据点已发送，CA APM ChangeDetector 会报告在配置期间定义的所有数据源的代理的 CA Introscope 代理支持能力度量标准。 所有度量标准总计只在 CA Introscope 代理的生命周期内有效。 因此，如果 CA Introscope 代理弹出，则所有计数器都会归零。

可在 Workstation 调查器树的 ChangeDetector|DataSources 节点中查看代理支持能力度量标准。

每个时间间隔发送的更改数

CA Introscope 在每个时间间隔发送的更改数。 每个计数器的每个时间间隔均归零。

添加更改总数

针对此数据源，CA Introscope 发送的添加更改总数。

完成扫描总数

针对此数据源已完成的扫描数。

更改总数

针对此数据源，代理发送的更改总数。

删除更改总数

针对此数据源，CA Introscope 发送的删除更改总数。

修改更改总数

针对此数据源，CA Introscope 发送的修改更改总数。

附录 A： 示例配置文件

CA APM ChangeDetector 安装包含一个示例标准配置文件。

通过使用配置向导，您可以轻松地根据环境来配置 (p. 13) CA APM ChangeDetector。

如果需要针对您的环境量身定制的配置文件，请基于 *ChangeDetector-config.xml* 或 *ChangeDetector-configDotnet.xml* 来创建自定义的 CA APM ChangeDetector 配置。

本章提供了有关如何创建自定义配置的详细说明，并包括一个自定义配置文件示例。

此部分包含以下主题：

示例 Java ChangeDetector-config.xml 文件

这是针对 Java 的一个可能的自定义 CA APM ChangeDetector 配置的示例。

注意：以下 XML 示例是数据示例。 它可能表示当前示例 Java ChangeDetector-config.xml 文件中的内容，或包含先前版本的 CA APM ChangeDetector 的内容。

```
<change-detector>
<!--
############################################################
# Introscope ChangeDetector Configuration
#
# CA Wily Introscope(R) ChangeDetector Version 8.1
# Copyright (c) 2008 CA. All Rights Reserved.
# Introscope(R) is a registered trademark of CA.
############################################################
-->

    <!-- ================================================== -->
    <!-- FILE CHANGE MONITORING                             -->
    <!-- ================================================== -->
    <!--
    扫描目录配置:
    必须修改的唯一配置元素是
     <scan-directory> 元素的 'name' 属性，该属性
    位于此 <datasource-instance> 元素的底部附近。
    必须为该 'name' 属性提供到
    您希望 ChangeDetector 扫描的目录的路径。 通常，
    这将是您的应用程序服务器的 Web 应用程序
    部署目录和配置目录。

    文件类型自定义:
    默认情况下，扫描
    由以下 <fileset> 元素定义的文件类型的当前工作目录。
    更改这些文件类型表示元素时，很可能
    会引起应用程序性能的改变。
    除非您需要扫描专用文件类型，
    否则无需修改这些元素。 -->
        <datasource-instance name="Application Files" type="file"
version="8.1">

                <property name="explodeArchiveFiles" value="false" />

                <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="3" unit="sec" />

                <property name="filesPerIteration" value="5" />

                <property name="delayBetweenArchiveIterations" value="10"
unit="sec" />
```

```
                <property name="archiveFilesPerIteration" value="1" />

                <!-- Accepted units are bytes, KBytes, MBytes -->
                <property name="maxFileSizeToUpload" value="50" unit="KB" />

                <property name="useDigest" value="needed" />

                <!-- Scans typical files used for web application config -->
                <fileset name="config">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)\.xml$"/>
                                <include pattern="(.*)\.cmd$"/>
                                <include pattern="(.*)\.sh$"/>
                                <include pattern="(.*)\.bat$"/>
                        </exclude>
                </fileset>

                <!-- Scans typical files that contain web application code -->
                <fileset name="webElements">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)\.jsp$"/>
                                <include pattern="(.*)\.cfm$"/>
                                <include pattern="(.*)\.js$"/>
                                <include pattern="(.*)\.html$"/>
                        </exclude>
                </fileset>

                <!-- Scans java archives -->
                <fileset name="archives">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)\.zip$"/>
                                <include pattern="(.*)\.jar$"/>
                                <include pattern="(.*)\.ear$"/>
                                <include pattern="(.*)\.war$"/>
                        </exclude>
                </fileset>

                <!-- Scans only the types of files defined in config, webElements,
and archive filesets -->
                <fileset name="default">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)\.xml$" />
                                <include pattern="(.*)\.cmd$" />
                                <include pattern="(.*)\.sh$" />
                                <include pattern="(.*)\.bat$" />
                                <include pattern="(.*)\.jsp$" />
                                <include pattern="(.*)\.cfm$" />
                                <include pattern="(.*)\.js$" />
                                <include pattern="(.*)\.html$" />
                                <include pattern="(.*)\.zip$" />
                                <include pattern="(.*)\.jar$" />
                                <include pattern="(.*)\.ear$" />
                                <include pattern="(.*)\.war$" />
                        </exclude>
```

```
                </fileset>

                <!-- Scans only the types of files defined in config and
webElements -->
                <fileset name="defaultNoArchives">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)\.xml$" />
                                <include pattern="(.*)\.cmd$" />
                                <include pattern="(.*)\.sh$" />
                                <include pattern="(.*)\.bat$" />
                                <include pattern="(.*)\.jsp$" />
                                <include pattern="(.*)\.cfm$" />
                                <include pattern="(.*)\.js$" />
                                <include pattern="(.*)\.html$" />
                        </exclude>
                </fileset>

                <!-- Scans everything except log files -->
                <fileset name="noLogs">
                        <exclude pattern="(.*)\.err$" />
                        <exclude pattern="(.*)\.log$" />
                        <exclude pattern="(.*)\.lok$" />
                        <exclude pattern="(.*)\.tlog$" />
                        <exclude pattern="(.*)\.log0(.*)" />
                </fileset>

        <!--
        更改 name 属性的值以指向您要监控的目录
        注意：开箱即用型配置使用“default”文件集。  如果您不想监控 java 存档文件，
        请改用 "defaultNoArchive" 文件集。  另外，您可以自定义一个文件集元素，
        来监控所选的文件，或替换以下 scan-directory 元素或添加新目录。
        也可以在 name 属性
        （或任何其他属性）值中使用 Java 系统属性或 Introscope 代理属性值（例如，
name="${MY_APP_HOME}/filesICareAbout/"-->
                <scan-directory recursive="true" name="." fileset="default"
                        enabled="true">
                </scan-directory>
        </datasource-instance>


    <!-- ================================================== -->
    <!-- DATABASE CHANGE MONITORING                         -->
    <!-- ================================================== -->
    <!--
    以下是 DB 监视器每 10 分钟从 Oracle v$parameter 表扫描一次名称/值
    对的示例步骤。

    注意：默认情况下已注释掉此数据源实例
    因为它需要特定于您的环境的
    连接参数。  请输入特定于您的数据库步骤
    的值。  -->
    <!--
```

```
    <datasource-instance name="Oracle DB" type="database"
driver="oracle.jdbc.driver.OracleDriver"
driverClasspath="C:\\somePathTo\\yourOracleDriver.zip"
                        url="jdbc:oracle:thin:@yourdbserver:1521:orcl"
username="username"
                        password="password" version="8.1">
                SQL Server
                        SELECT name, value FROM v$parameter
                </sql>
                <schedule type="repetitive" interval="10" unit="min"/>
        </datasource-instance>
        -->


    <!-- ======================================================== -->
    <!-- JAVA SYSTEM PROPERTIES MONITORING                        -->
    <!-- ======================================================== -->
    <!--
        默认情况下，
    Java 系统属性监视器会包括并监控所有属性。

    要排除属性，必须添加排除节点。 如果要
    覆盖任何排除，
    必须在特定 exclude 元素内嵌套 include 元素。  以下示例排除了
    具有以“java.”开头的属性异常的所有属性。
                 <exclude pattern=".*">
                        <include pattern="java\.*"/>
                 </exclude> -->
        <datasource-instance name="Java System Properties" type="javaenv"
version="8.1">
        </datasource-instance>


    <!-- ======================================================== -->
    <!-- JAVA CLASS MONITOR                                       -->
    <!-- ======================================================== -->
    <!--
    默认情况下，会使用以下 exclude 元素排除应用程序服务器类。
    exclude 元素的 pattern 属性定义
    用来匹配类名称（包括软件包名称）的正则表达式。

    如果要覆盖任何排除，
    必须在特定 exclude 元素内嵌套 include 元素。  以下示例排除了
     java 软件包内具有类异常的所有类。
                <exclude pattern=".*">
                        <include pattern="java\.*"/>
                </exclude>

        注意：ChangeDetector 当前对每个 JVM 仅支持 1 个 classmonitor
                        数据源实例。

        注意：
                        Java 类监视器将包括并监控与任何排除参数都不匹配的所有类
```

```
        -->
        <datasource-instance name="Java Class Monitor" type="classmonitor"
version="8.1">
                <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="2" unit="sec" />
                <property name="classesPerIteration" value="100" />

                <!-- exclude classes from Wily -->
                <exclude pattern="com\.wily\.(.*)"/>

                <!--
                        以下示例将从某些应用程序服务器跳过类。
                        包括可能导致跟踪大量
                        极少更改的类的类，以及可能与
                        应用程序性能无关的类。
                -->

                <!-- 从 BEA 排除类 -->
                <exclude pattern="weblogic\.(.*)"/>
                <exclude pattern="com\.bea\.(.*)"/>

                <!-- 从 IBM 排除类 -->
                <exclude pattern="com\.ibm\.(.*)">
                        <include pattern="com\.ibm\.(.*)jdbc(.*)"/>
                </exclude>

                <!-- 从 SAP 排除类 -->
                <exclude pattern="com\.sap\.(.*)"/>

                <!-- 从 Oracle 排除类 -->
                <exclude pattern="oracle\.*">
                        <include pattern="oracle\.jdbc\.(.*)"/>
                </exclude>

                <!-- 从 Sun 排除类 -->
                <exclude pattern="com\.sun\.enterprise\.(.*)"/>
        </datasource-instance>

        <!-- ========================================================= -->
    <!-- CONFIGURATION PROPERTIES—不要修改 datasource-type -->
    <!-- ========================================================= -->
        <datasource-type name="file"
class="com.wily.rave.agent.ds.file.FileDataSourceConfig" />
        <datasource-type name="database"
class="com.wily.rave.agent.ds.db.DBDataSourceConfig" />
        <datasource-type name="javaenv"
class="com.wily.rave.agent.ds.sysprop.SysPropDataSourceConfig" />
        <datasource-type name="classmonitor"
class="com.wily.rave.agent.ds.classmonitor.RuntimeClassMonitorConfig" />

</change-detector>
```

示例 .NET ChangeDetectorDotnet-config.xml 文件

这是针对 .NET 的一个可能的自定义 CA APM ChangeDetector 配置的示例。

注意：以下 XML 示例是数据示例。 它可能表示当前示例 .NET ChangeDetectorDotnet-config.xml 文件中的内容，或包含先前版本的 CA APM ChangeDetector 的内容。

```
<change-detector>
<!--
############################################################
# Introscope ChangeDetector Configuration
#
# CA Wily Introscope(R) ChangeDetector Version 8.1
# Copyright (c) 2008 CA. All Rights Reserved.
# Introscope(R) is a registered trademark of CA.
############################################################
-->
    <!-- ================================================== -->
    <!-- ENVIRONMENT VARIABLES MONITORING                   -->
    <!-- ================================================== -->
    <!--
         默认情况下，环境监视器会包括并监控
         所有环境变量。

    要排除变量，必须添加排除节点。 如果要
    覆盖任何排除，
    必须在特定 exclude 元素内嵌套 include 元素。  以下示例排除了
    具有以“windows.”开头的属性异常的所有属性。
                  <exclude pattern=".*">
                        <include pattern="windows\.*"/>
                  </exclude> -->

     <datasource-instance name="Environment Variables" type="javaenv"
version="8.1">
         <exclude pattern="foo" />
         <exclude pattern=".*bar.*">
           <include pattern="hello" />
           <include pattern=".*world.*" />
         </exclude>
     </datasource-instance>

    <!-- ================================================== -->
    <!-- FILE CHANGE MONITORING                             -->
    <!-- ================================================== -->
    <!--
    扫描目录配置：
    必须修改的唯一配置元素是
     <scan-directory> 元素的 'name' 属性，该属性
    位于此 <datasource-instance> 元素的底部附近。
    必须为该 'name' 属性提供到
    您希望 ChangeDetector 扫描的目录的路径。 通常，
```

```
        这将是您的应用程序服务器的 Web 应用程序
        部署目录和配置目录。

        文件类型自定义：
        默认情况下，扫描
        由以下 <fileset> 元素定义的文件类型的当前工作目录。
        更改这些文件类型表示元素时，很可能
        会引起应用程序性能的改变。
        除非您需要扫描专用文件类型，
        否则无需修改这些元素。 -->
            <datasource-instance name="Application Files" type="file"
version="8.1">

                    <property name="explodeArchiveFiles" value="false" />

                    <!-- Accepted units are hour, min, sec -->
                    <property name="delayBetweenIterations" value="3" unit="sec" />

                    <property name="filesPerIteration" value="5" />

                    <property name="delayBetweenArchiveIterations" value="10"
unit="sec" />

                    <property name="archiveFilesPerIteration" value="1" />

                    <!-- Accepted units are bytes, KBytes, MBytes -->
                    <property name="maxFileSizeToUpload" value="50" unit="KB" />

                    <property name="useDigest" value="needed" />

                    <!-- Scans typical files used for web application config -->
                    <fileset name="config">
                            <exclude pattern="(.*)">
                                    <include pattern="(.*)\.xml$"/>
                                    <include pattern="(.*)\.cmd$"/>
                                    <include pattern="(.*)\.sh$"/>
                                    <include pattern="(.*)\.bat$"/>
                            </exclude>
                    </fileset>

                    <!-- Scans typical files that contain web application code -->
                    <fileset name="webElements">
                            <exclude pattern="(.*)">
                                    <include pattern="(.*)\.asp$"/>
                                    <include pattern="(.*)\.asm$"/>
                                    <include pattern="(.*)\.js$"/>
                                    <include pattern="(.*)\.html$"/>
                            </exclude>
                    </fileset>

                    <!-- Scans archives -->
                    <fileset name="archives">
                            <exclude pattern="(.*)">
                                    <include pattern="(.*)\.zip$"/>
```

```
                </exclude>
        </fileset>

        <!-- Scans only the types of files defined in config, webElements,
and archive filesets -->
        <fileset name="default">
                <exclude pattern="(.*)">
                        <include pattern="(.*)\.xml$" />
                        <include pattern="(.*)\.cmd$" />
                        <include pattern="(.*)\.sh$" />
                        <include pattern="(.*)\.bat$" />
                        <include pattern="(.*)\.asp$" />
                        <include pattern="(.*)\.asm$" />
                        <include pattern="(.*)\.js$" />
                        <include pattern="(.*)\.html$" />
                        <include pattern="(.*)\.zip$" />
                        <include pattern="(.*)\.profile$" />
                </exclude>
        </fileset>

        <!-- Scans only the types of files defined in config and
webElements -->
        <fileset name="defaultNoArchives">
                <exclude pattern="(.*)">
                        <include pattern="(.*)\.xml$" />
                        <include pattern="(.*)\.cmd$" />
                        <include pattern="(.*)\.sh$" />
                        <include pattern="(.*)\.bat$" />
                        <include pattern="(.*)\.asp$" />
                        <include pattern="(.*)\.asm$" />
                        <include pattern="(.*)\.js$" />
                        <include pattern="(.*)\.html$" />
                </exclude>
        </fileset>

        <!-- Scans everything except log files -->
        <fileset name="noLogs">
                <exclude pattern="(.*)\.err$" />
                <exclude pattern="(.*)\.log$" />
                <exclude pattern="(.*)\.lok$" />
                <exclude pattern="(.*)\.tlog$" />
                <exclude pattern="(.*)\.log0(.*)" />
        </fileset>

    <!--
    scan-directory:
    更改 name 属性的值以指向您要监控的目录
    注意：开箱即用型配置使用“default”文件集。 另外，您可以自定义一个文件集元素，
    来监控所选的文件，或替换以下 scan-directory 元素或添加
    新目录。 也可以在 "name"
    属性（或任何其他属性）值中使用环境变量或 Introscope 代理属性值（例如，
name="${MY_APP_HOME}/filesICareAbout/"
    它可能是一个单独的点 (.)，表示相对于应用程序（非代理）的工作目录
    或者它需要完整路径。 以下示例指向安装代理时通常使用的
```

```
        代理安装
        -->

                <scan-directory recursive="true" name="C:/Program Files/CA
Wily/Introscope8.1/wily" fileset="default"
                        enabled="true">
                </scan-directory>
        </datasource-instance>


    <!-- ================================================== -->
    <!-- ASSEMBLY MONITOR                                   -->
    <!-- ================================================== -->
    <!--
    默认情况下，会使用以下 exclude 元素排除 windows 系统类。
    exclude 元素的 pattern 属性定义
    用来匹配类名称（包括命名空间）的正则表达式。

    如果要覆盖任何排除，
    必须在特定 exclude 元素内嵌套 include 元素。  以下示例排除了
    命名空间内具有类异常的所有类。
                <exclude pattern=".*">
                        <include pattern="system\.*"/>
                </exclude>

        注意：ChangeDetector 当前对每个 JVM 仅支持 1 个程序集监视器
                        数据源。

        注意：
                        程序集监视器将包括并监控与任何排除参数都不匹配的所有类

        -->



        <datasource-instance name="Assembly Monitor" type="classmonitor"
version="8.1">

                <!-- The initial wait on startup and when an assembly is loaded
before scanning -->
                 <!-- Accepted units are hour, min, sec -->
                <property name="initialWaitTime" value="30" unit="sec" />

                 <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="2" unit="min" />
                <property name="classesPerIteration" value="5" />

                <!--
                        以下示例将从系统程序集跳过类。
                        包括可能导致跟踪大量
                        极少更改的类的类，以及可能与
                        应用程序性能无关的类。
                        添加不需要
```

```
                        使用 <excludeassembly> 标记监控的程序集
                    -->

                    <!-- exclude assemblies -->
                    <excludeassembly pattern=".\\mscorlib\.dll"/>
                    <excludeassembly pattern=".\\System\.dll"/>
                    <excludeassembly pattern=".\\System\.Xml\.dll"/>
                    <excludeassembly pattern=".\\System\.Web\.dll"/>
                    <excludeassembly pattern=".\\System\.Configuration\.dll"/>
                    <excludeassembly pattern=".\\wily\.."/>
                    <excludeassembly pattern=".\\Microsoft\.JScript\.dll"/>
                    <excludeassembly pattern=".\\VJSharpCodeProvider\.dll"/>
                    <excludeassembly pattern=".\\System\.Data\.dll"/>
                    <excludeassembly pattern=".\\Oracle\.DataAccess\.dll"/>
                    <excludeassembly pattern=".\\System\.Web\.Mobile\.dll"/>
                    <excludeassembly pattern=".\\System\.ServiceModel\.dll"/>
                    <excludeassembly pattern=".\\SMDiagnostics\.dll"/>
                    <excludeassembly pattern=".\\System\.Drawing\.dll"/>
                    <excludeassembly
pattern=".\\System\.Web\.RegularExpressions\.dll"/>
                    <excludeassembly pattern=".\\Microsoft\.VisualBasic\.dll"/>
                    <excludeassembly pattern=".\\CppCodeProvider\.dll"/>
                    <excludeassembly
pattern=".\\System\.EnterpriseServices\.dll"/>
                    <excludeassembly pattern=".\\System\.Transactions\.dll"/>

                    <!-- exclude classes from Wily -->
                    <!-- Add classes you wish to exclude in exclude patterns -->
                    <exclude pattern="com\.wily\.(.*)"/>

            </datasource-instance>


        <!-- ======================================================== -->
        <!-- DATABASE CHANGE MONITORING                               -->
        <!-- ======================================================== -->
        <!--
        以下是 DB 监视器每 10 分钟从 Oracle v$parameter 表扫描一次名称/值
        对的示例步骤。

        注意：默认情况下已注释掉此数据源实例
        因为它需要特定于您的环境的
        连接参数。 请输入特定于您的数据库步骤
        的值。 -->
<!--
           <datasource-instance name="SQl Server DB" type="database"
                               url="Provider=SQLOLEDB;Data Source=localhost;
Integrated Security=SSPI;Initial Catalog=northwind" version="8.1">
                    SQL Server
                            SELECT name, value FROM sampletable
                    </sql>
                    <schedule type="repetitive" interval="10" unit="min"/>
            </datasource-instance> -->
```

```
        <!-- ================================================== -->
        <!-- CONFIGURATION PROPERTIES—不要修改 datasource-type -->
        <!-- ================================================== -->
        <datasource-type name="javaenv"
class="com.wily.rave.agent.ds.sysprop.SysPropDataSourceConfig" />
        <datasource-type name="file"
class="com.wily.rave.agent.ds.file.FileDataSourceConfig" />
        <datasource-type name="classmonitor"
class="com.wily.rave.agent.ds.classmonitor.RuntimeAssemblyMonitorConfig" />
        <datasource-type name="database"
class="com.wily.rave.agent.ds.db.DBDataSourceConfig" />

</change-detector>
```

附录 B： 常见问题

CA APM ChangeDetector 检测到一个 9MB 存档文件，并报告了它的更改，但是我已将 maxFileSizeToUpload 配置文件属性设置为 4MB。 这是一个缺陷吗？

这不是缺陷。 *maxFileSizeToUpload* 属性仅适用于文本文件，而不适用于存档文件。 CA APM ChangeDetector 打开存档文件并单独处理每个文件，而不是上传整个存档文件的内容。

我从应用程序中运行的代码中删除了一个类，但是 CA APM ChangeDetector 未检测到该删除操作。 为什么？

CA APM ChangeDetector 仅检测代码中的添加和修改操作，而不检测代码删除操作。

CA APM ChangeDetector 为什么没有显示我的受监控文件系统中的任何文件？

受监控的文件系统可能没有读权限，如果系统处于网络文件夹中，则可能是因为网络关闭。