CA Mediation Manager 和 CA Mediation Manager for Infrastructure Management

管理指南

CA Mediation Manager 版本 2.2.3/CA Mediation Manager for Infrastructure Management 2.0，版本 2.2.3

CA Technologies 产品引用

本文档引用以下 CA Technologies 产品：

- CA Mediation Manager
- CA Mediation Manager for Infrastructure Management 2.0

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： 简介

本指南针对 CA Mediation Manager 安装提供有关体系结构、安装、先决条件和要求的信息。

注意：此章的信息仅适用于 CA Mediation Manager。

此部分包含以下主题：

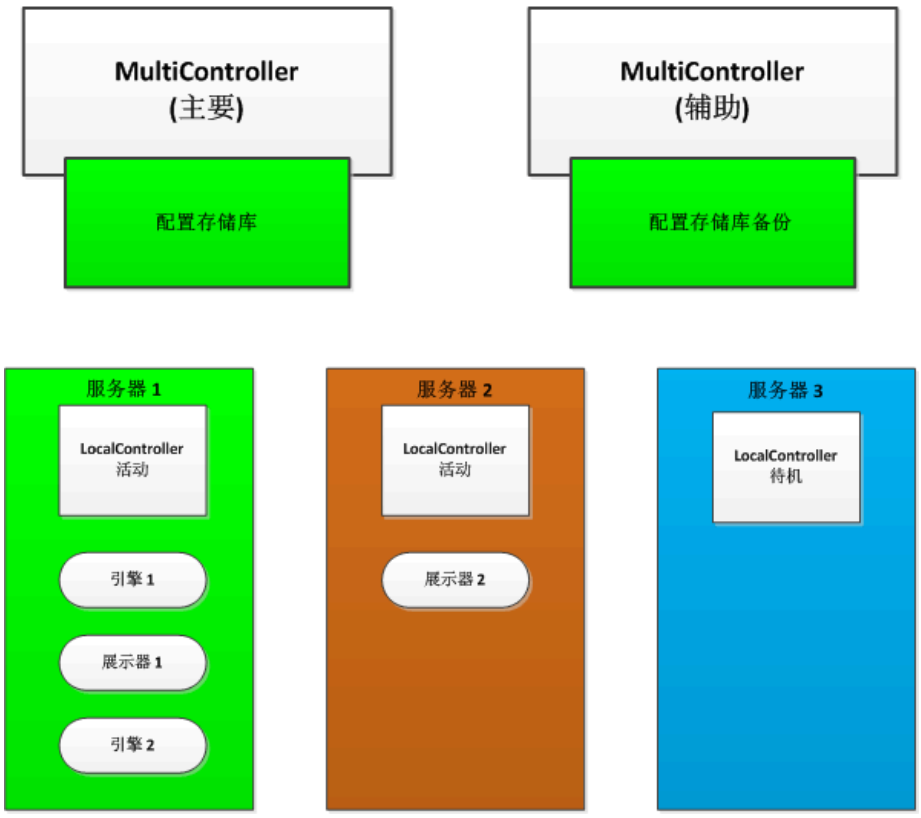
体系结构 (p. 7)

体系结构

CA Mediation Manager 包括两个主组件和两个子组件。主组件是 MultiController (MC) 和 LocalController (LC)。

子组件是引擎和展示器。

下图说明了常规体系结构：



体系结构还包括其他组件，名为“常规执行器”和“传递服务”。这些组件未显示在上图中，但稍后会在该指南中进行说明。

第 2 章： 概述

注意：此章的信息仅适用于 CA Mediation Manager。

此部分包含以下主题：

组件概述

CA Mediation Manager 安装程序将安装 MultiController、LocalController 和 Web 组件。

CA Mediation Manager 安装程序还将安装以下组件：

- 传递服务（LocalController 的一部分）
- 常规执行器

安装设备包时将安装引擎和展示器子组件。

注意：有关特定设备包的安装详细信息，请参阅 CAMM_HOME（安装 CA Mediation Manager 的目录）下 DpConfig 文件夹中的《设备包指南》。

主要组件

CA Mediation Manager 有三个主要组件：MultiController、LocalController 和 Web。

MultiController

您可以在群集中部署主要和辅助多达两个 MultiController。在每个群集中至少部署一个 MultiController。MultiController 将在您的群集环境中执行以下操作：

- 监视来自远程服务器上的 LocalController 组件的检测信号消息。
- 用作群集的集中式许可服务器。
- 存储群集中组件的集中式配置文件。

LocalController

在子组件（引擎或展示器）所在的群集中的每台物理服务器上都安装一个 LocalController。LocalController 将执行以下操作：

- 为安装在服务器上的子组件提供通信机制。
- 监视本地服务器上子组件的检测信号消息，并且在子组件发生故障时自动重新启动。
- 使用传递服务处理引擎子组件的输出。该服务会使用压缩和加密格式将 XML 文档传递到本地或远程展示器子组件。

Web

Web 组件允许您使用基于 Web 的界面，集中管理设备包部署。该界面显示以下信息：

- 运行设备包的状态。
- 安装有设备包的 LocalController 的状态。
- 主要和辅助 MultiController 的状态。

CA Mediation Manager 安装两个 Web 服务器：

- 主要 Web 服务器
- 辅助 Web 服务器

主要 Web 服务器在主要 MultiController 安装期间安装，辅助 Web 服务器在辅助 MultiController 安装期间安装。

- 要访问主要 Web 服务器，请启动 CA Mediation Manager Web UI：

  `http://<PrimaryMCMachineIP>:<web-port>/tim-web/index.htm`

  其中，<web-port> 是在 CA Mediation Manager 安装期间配置的端口号，<PrimaryMCMachineIP> 是*主要 MultiController* 系统的 IP 地址或主机名。

  如果主要 MultiController 没有响应，辅助 MultiController 则会自动启动辅助 Web 服务器。

  如果主要 MultiController 开始响应，辅助 MultiController 则会停止辅助 Web 服务器。

- 要访问辅助 Web 服务器，请启动 CA Mediation Manager Web UI：

  `http://<SecondaryMCMachineIP>:<web-port>/tim-web/index.htm`

  其中，<web-port> 是在 CA Mediation Manager 安装期间配置的端口号，<SecondaryMCMachineIP> 是*辅助 MultiController* 系统的 IP 地址或主机名。

以下部分说明 Web 组件选项。

默认选项

以下信息说明 CA Mediation Manager “管理”选项卡中的默认选项：

安装或删除

安装设备包或从现有存储库删除设备包。在安装期间使用的默认路径是 $CAMM_HOME/MC/repository/device packs。如果在安装期间使用了其他路径，您可以浏览并选择您的设备包路径。

升级

升级设备包版本。

高级选项

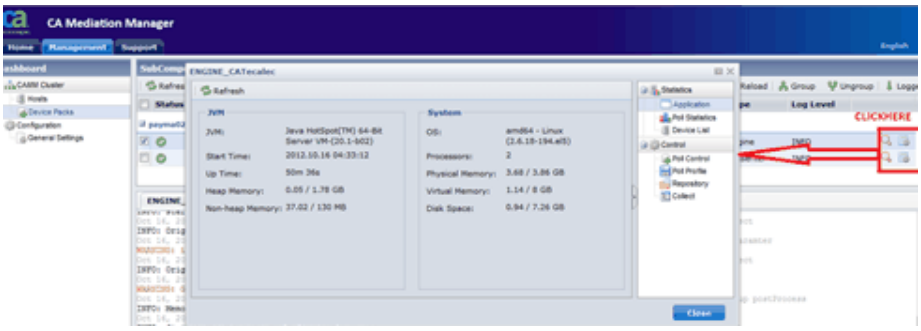
以下信息说明高级选项。

统计信息

提供下列选项来收集和了解统计信息：

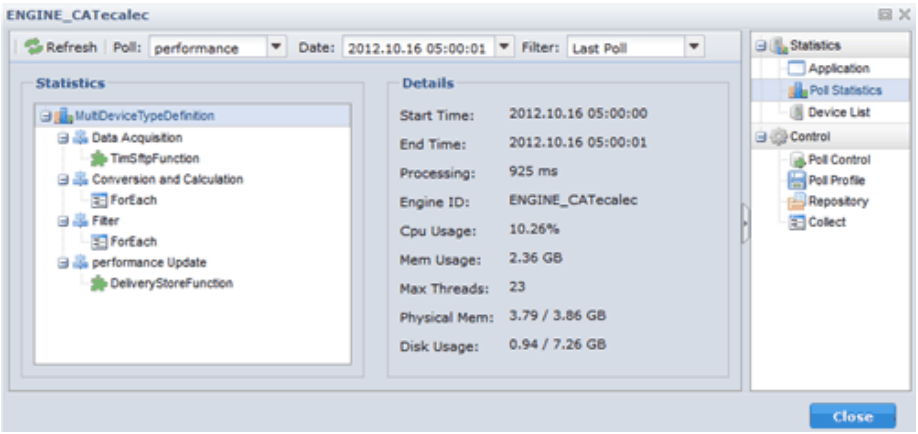
应用程序

提供 Java 虚拟机 (JVM) 和正在运行引擎或展示器的系统的性能度量标准。



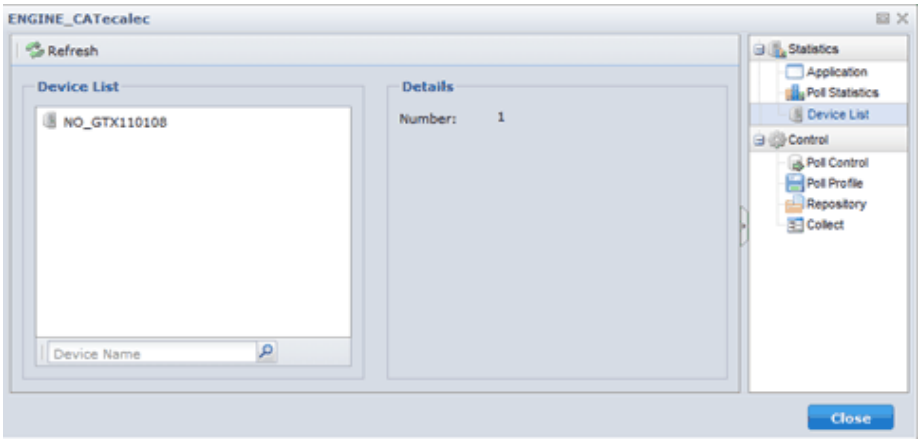
轮询统计

提供清单轮询状态和性能轮询状态。重新启动引擎以反映更改。



设备列表

使用位于 $CAMM_HOME/repository/work 文件夹下的清单设备列表文件，提供由清单轮询发现的设备列表。



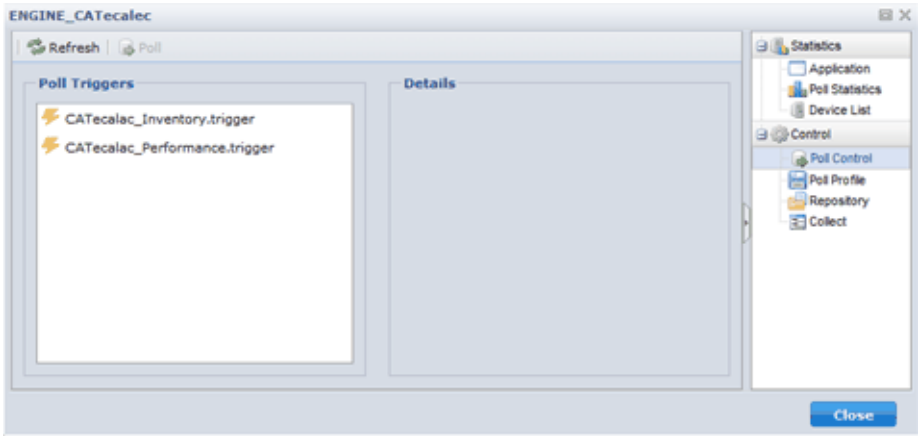
控制

提供以下界面，可用于：

- 触发手工轮询
- 查看和编辑设备包使用的外部全局变量值
- 查看和编辑设备包文件
- 收集日志文件

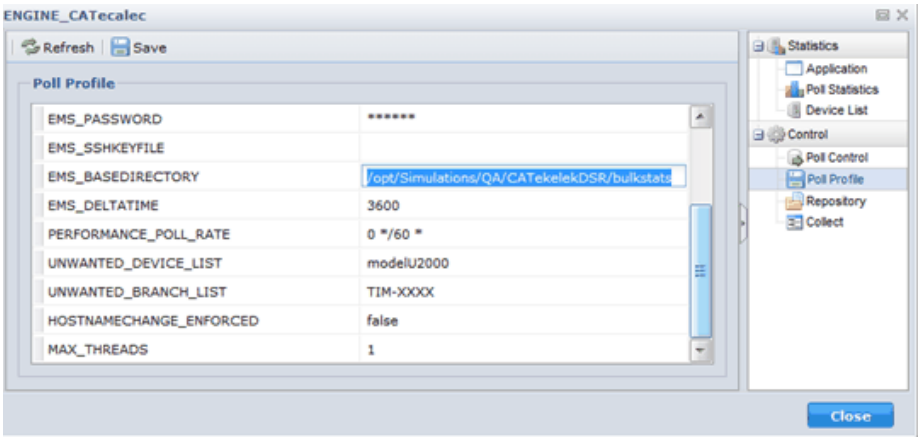
轮询控制

触发清单轮询和性能轮询。



轮询配置文件

包含您修改的外部全局变量的列表。重新启动引擎或展示器以反映您的更改。

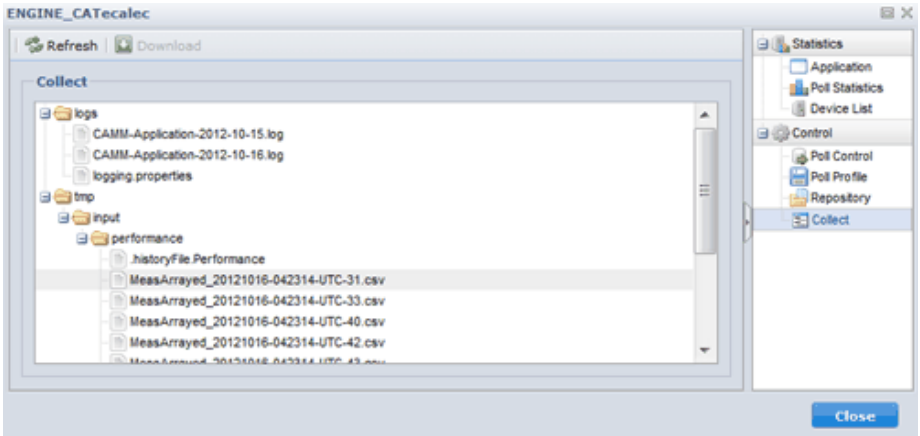


存储库

允许您查看和修改 MultiController 中的设备包组件的文件。重新启动引擎或展示器以反映您的更改。

收集

收集引擎和展示器日志。



常规设置

使用以下选项来指定常规设置：

Connection

指定 CA Mediation Manager 组件之间的连接重试次数和连接间隔的默认设置。

日志记录

为 CA Mediation Manager 组件指定日志记录选项，例如刷新速率、缓冲区大小以及最大行号。

子组件

CA Mediation Manager 中有两个子组件：引擎和展示器。

引擎

引擎是 CA Mediation Manager 中主要的线程轮询引擎。您可以以活动或待机模式部署引擎。引擎将执行以下操作：

- 使用 XML、CSV、Telnet、SSH 等从设备收集信息，并将数据处理到 CA Mediation Manager 标准 XML 文档中。
- 将 CA Mediation Manager 标准 XML 文档部署到队列中，由传递服务进行处理。

展示器

展示器子组件是执行以下操作的线程演示引擎：

- 从引擎接收 CA Mediation Manager 标准 XML 文档。
- 将数据格式设置为所需的输出格式，如 CSV、XML、SNMP、DDI。

其他组件

常规执行器和传递服务是 CA Mediation Manager 的另外两个组件。常规执行器启动其他组件。传递服务将 XML 文件输出发送到展示器。

常规执行器

群集中的所有组件共享一组通用的通信和执行功能。常规执行器启动引擎和展示器子组件，并清除临时和日志文件。

常规执行器在系统启动时启动，并侦听特定的 TCP 端口。要启动类似 MultiController 的组件，CA Mediation Manager 控制实用工具 cammCtrl 会将 MultiController XML 配置文件发送到常规执行器。常规执行器接收到此数据时，会使用配置文件中的信息识别和启动 MultiController 组件。

传递服务

引擎完成其轮询周期时，会将一个或多个 CA Mediation Manager 标准 XML 文档生成到队列目录中。传递服务可独立地监视队列目录，并将数据分发到一个或多个本地或远程展示器子组件。

如果本地或远程展示器子组件不可用，传递服务则不会处理队列，直到本地或远程展示器变得可用。

第 3 章：安装、卸载和升级 CA Mediation Manager

注意：此章的信息仅适用于 CA Mediation Manager。

此部分包含以下主题：

系统要求

CA Mediation Manager 需要 Java Runtime Environment (JRE) 1.7 或更高版本。

下表说明了每个受支持操作系统的最低硬件要求：

操作系统	体系结构	CPU	内存	磁盘
Solaris 9 或 10	SPARC（64 位）	1 x 1.4 GHz	4 GB	18 GB
Linux	x86（64 位）	1 x 2 GHz	4 GB	18 GB
Windows 2003	x86（64 位）	1 x 2 GHz	4 GB	18 GB
Windows 2008	x86（64 位）	1 x 2 GHz	4 GB	18 GB

注意：保持 JRE 和操作系统体系结构之间的一致性。例如，在 64 位操作系统上，您用来安装并运行 CA Mediation Manager 的 JRE 也必须是 64 位。CA Technologies 建议使用最新版本的 JRE，您可以从 Java 下载站点获得相应的最新版本。

安装和升级

要安装和升级 CA Mediation Manager 和设备包，请参阅《*CA Mediation Manager 安装指南*》。

启动和停止服务

以下信息说明了如何在用于 UNIX 和 Windows 的 CA Mediation Manager 中启动和停止服务。

UNIX

您可以执行 startall 或 stopall 脚本或 init.camm 脚本来启动/停止 CA Mediation Manager。init.camm 脚本在 CAMM 主目录中在工具目录中。

以 root 或 sudo su 身份执行以下 init.camm.install 脚本，以便在系统启动或停止时自动启动或停止 CA Mediation Manager：

```
shell# tools/init.camm.install
```

执行以下 init.camm.uninstall 脚本以便删除自动启动或停止 CA Mediation Manager 的设置：

```
shell# tools/init.camm.uninstall
```

Windows

在 Windows 上安装 CA Mediation Manager 过程中，常规执行器和 Web 组件将注册为 Windows 服务。服务名称为 CAMM-GE-{user}-{port} 和 CAMM-tomcat7-8880。

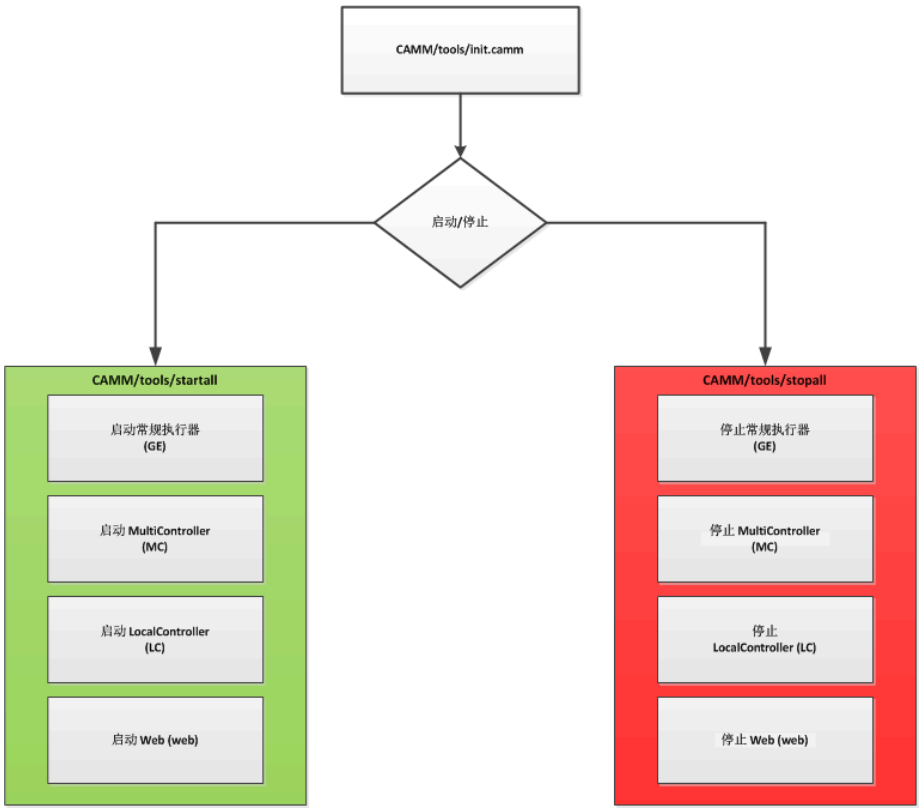
默认情况下，这些服务需要手工启动。与 Linux 类似，Windows 中，您可以通过执行 init.camm.install.bat 在系统启动或停止时自动启动或停止 CA Mediation Manager：

```
C:/CAMM/tools/init.camm.install.bat
```

执行 init.camm.uninstall 脚本以便删除自动启动或停止 CA Mediation Manager 的设置：

```
C:/CAMM/tools/init.camm.uninstall.bat
```

下图说明了启动和停止服务的流程：



第 4 章： 组件配置

以下部分说明如何配置 CA Mediation Manager 中的组件：

注意：此章的信息仅适用于 CA Mediation Manager。

此部分包含以下主题：

常规执行器配置

通常在初始安装之后，常规执行器不需要任何配置。除非您需要在不同用户 ID 下运行组件，否则每个服务器只需一个常规执行器。

常规执行器配置安装在常规执行器目录中，在 CAMM 主目录中名为 GE_<userid>。<userid> 是您在安装期间指定的用户名。

常规执行器目录中包含一个名为 LocalConfig-ge.xml 的文件。

常规执行器承担某个组件的角色时，它使用在 LocalConfig-ge.xml 中配置的相同用户 ID 来执行。

UNIX 的 LocalConfig-ge.xml 文件示例

```
<?xml version="1.0" encoding="UTF-8"?>
<AppDaemon>
    <Names>
    </Names>
    <Paths>
        <Path name="tim.base">/opt/CA/CAMM</Path>
        <Path name="appHome">${tim.base}/GE_camm</Path>
        <Path name="configBase">${appHome}/tmp</Path>
    </Paths>
    <Binding>
        CA Portal29560</Port>
        <UserId>camm</UserId>
    </Binding>

</AppDaemon>
```

Windows 的 LocalConfig-ge.xml 文件示例

对于 Windows 用户，主常规执行器配置文件略有不同：

```
<?xml version="1.0" encoding="UTF-8"?>
<AppDaemon>
    <Names>
    </Names>
    <Paths>
        <Path name="tim.base">/opt/CA/CAMM</Path>
        <Path name="appHome">${tim.base}/GE_camm</Path>
        <Path name="configBase">${appHome}/tmp</Path>
    </Paths>
    <Binding>
        CA Portal29560</Port>
        <UserId>camm</UserId>
    </Binding>
    <CompanyItems>
        <Item>
                <Name>MC</Name>
                <Config>${tim.base}/MC/LocalConfig-mc.xml</Config>
                CA Portal29599</Port>
```

```
        </Item>
        <Item>
                <Name>LC</Name>
                <Config>${tim.base}/LC/LocalConfig-lc.xml</Config>
                CA Portal29598</Port>
        </Item>
    </CompanyItems>
</AppDaemon>
```

添加到常规执行器配置文件中的 <CompanyItems> 部分定义 MultiController 和 LocalController 的位置与服务端口。定义 <CompanyItems> 之后，常规执行器会隔一分钟检查一次这些项，如果其未运行，常规执行器将自动启动这些项。通过此配置，MultiController 和 LocalController 无需外部干预即可启动。

默认情况下，安装后会禁用 <CompanyItems> 功能。您可以执行 camm.init.install 命令以启用该功能。

注意：只有主常规执行器需要 <CompanyItems> 部分。其他常规执行器不得包含此部分。

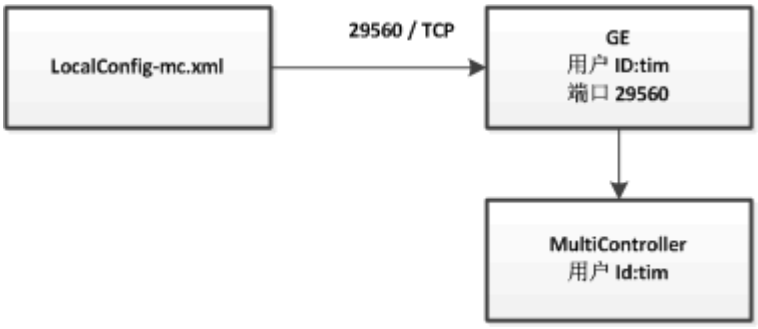
常规执行器工作方式

常规执行器是可重复使用的实体，是 CA Mediation Manager 组件的基础。服务器启动时，必须至少存在且启动一个常规执行器组件。

默认情况下，LocalConfig-ge.xml 文件包含 CAMM_USER 和 TCP 端口信息的用户 ID。下图说明 TCP 端口 29560 上的常规执行器，以及安装 CA Mediation Manager 的 CAMM_USER：

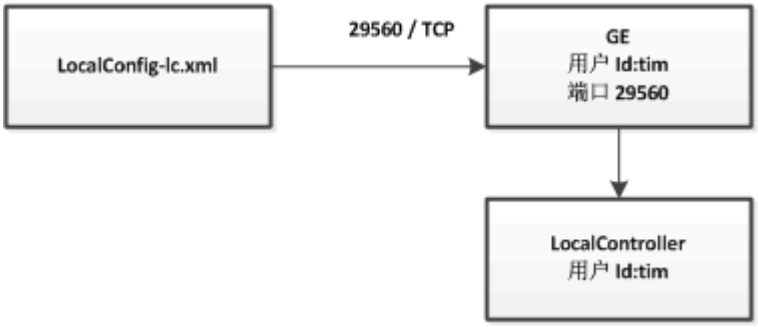
如果需要在此服务器上运行 MultiController，您需要 MultiController 的 LocalConfig-mc.xml 配置文件和常规执行器的 TCP 端口。

下图说明了如何将 LocalConfig-mc.xml 文件在 TCP 端口 29560 上发送给常规执行器，以及常规执行器如何为 MultiController 启动新的进程：



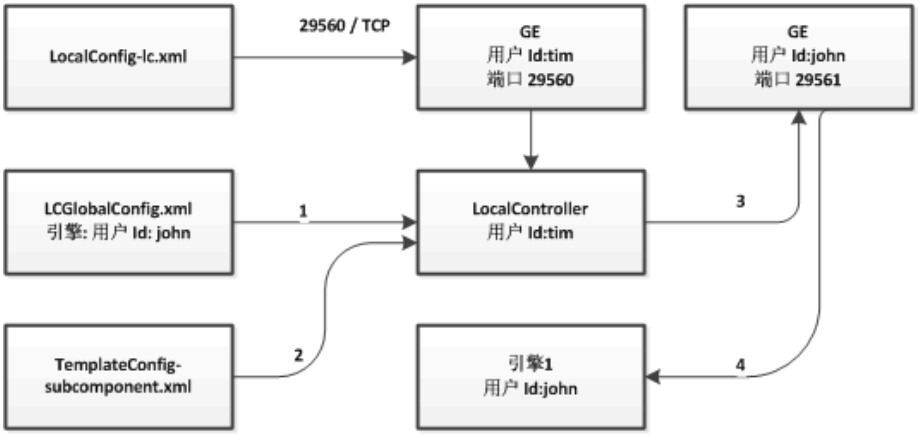
如果 LocalController 必须在服务器上运行，则仅需要 LocalController 的 LocalConfig-lc.xml 配置文件和常规执行器的 TCP 端口。

下图说明了如何将 LocalConfig-lc.xml 文件在 TCP 端口 29560 上发送给常规执行器，以及常规执行器如何为 LocalController 启动新的进程：



LocalController 启动时，会从其存储库中读取 LCGlobalConfig.xml 文件。LCGlobalConfig.xml 文件包含子组件和端口信息。

如果需要在此服务器上运行引擎，LocalController 使用 LCGlobalConfig.xml 文件来确定运行需要的用户 ID。LocalController 在 LocalController 目录中查找 ExecutorMap.xml 文件。此操作允许 LocalController 查找以所需用户 ID 运行的常规执行器。然后，TemplateConfig-subcomponent.xml 文件与 LCGlobalConfig.xml 文件中的特定配置进行组合。组合结果将发送到常规执行器的 TCP 端口。



常规执行器配置选项

路径 tim.base、appHome 和 configBase 是必需的，且由安装程序配置。

tim.base

指定 CA Mediation Manager 的基目录。

appHome

指定此 CA Mediation Manager 常规执行器的基目录。

configBase

指定临时目录，用于存储常规执行器当前正在执行的组件的配置。

Port

指定常规执行器侦听的 TCP 端口。

默认值：TCP 29560

用户 ID

指定拥有常规执行器进程及其子进程的用户 ID。

常规执行器启动顺序

您可以使用 startall 或 startall.bat 脚本，手工启动常规执行器进程（或 Windows 上的服务）。但是，如果执行 camm.init.install，常规执行器会在系统启动时自动启动。

添加另一个常规执行器 (UNIX)

如果您需要用不同的用户 ID 来执行某个组件，请添加另一个常规执行器。

遵循这些步骤:

1. 为需要其他用户 ID 的 LocalController 添加新用户。新用户必须是拥有 CAMM_HOME 目录的同一个组的成员。

   注意：为所有待机 LocalController 重复此过程。

2. 以新用户身份登录。
3. 执行以下命令以添加新的常规执行器：

   ```
   shell# /opt/CA/CAMM/tools/camm.ge.install
   ```

   已添加其他常规执行器。

   注意：已修改 LocalController 目录中名为 ExecutorMap.xml 的配置文件，以在所需用户 ID 和 TCP 端口之间提供映射。

引擎子组件启动时，LocalController 为该组件引用 LCGobalConfig 中提供的用户 ID。如果此用户 ID 不是默认用户 ID，则引用 ExecutorMap，并使用备用常规执行器的 TCP 端口启动子组件。

此引用在引擎将用户或基于主机的 RSA 密钥身份验证用于 SFTP 或 SCP 信息时十分有用。

MultiController 配置

MultiController 是群集最必不可少的部分。它包含集中式许可文件，并维护群集中的组件的检测信号。

群集中的 LocalController、引擎和展示器的配置还位于 MultiController 的存储库中。

MultiController 在 TCP 端口号 29599 上侦听来自群集成员的检测信号操作。

在 GUI 安装期间，您可以仅配置安装基本 MultiController 的必需选项。但是，您可以手工编辑 MC 或 LocalConfig-mc.xml 文件。

示例 LocalConfig-mc.xml 文件（基本配置）

```
<?xml version="1.0" encoding="UTF-8"?>
<LocalConfig>
      <Description>Configuration for Multi Controller</Description>
      <Names>
            <Name name="mainClass">com.torokina.tim.mc.Main</Name>
            <Name name="appName">CAMM-Multi-Controller</Name>
            <Name name="appShortName">MC</Name>
            <Name name="primaryMcAddress">127.0.0.1</Name>
            <Name name="secondaryMcAddress"></Name>
            <Name name="primaryMcPort">29599</Name>
            <Name name="secondaryMcPort">-1</Name><
            <Name name="myMode">primary</Name>
            <Name name="myAddress">127.0.0.1</Name>
            <Name name="mcPort">29599</Name>
            <Name name="otherMcAddress"></Name>
            <Name name="otherMcPort">-1</Name>
            <Name name="heartbeatFrequency">15</Name>
            <Name name="heartbeatTimeout">180</Name>
            <Name name="repositoryFrequency">15</Name>
      </Names>
      <Paths>
            <Path name="license">${tim.base}/license.lic</Path>
      </Paths>
<LocalConfig>
```

示例 LocalConfig-mc.xml 文件（隐藏的日志和清理配置）

```
    <Logging>
        <LogLevel>INFO</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
        <ObjectLogging>
          <ObjectToLog>
            <ObjectName>com.torokina.tim.config</ObjectName>
            <ObjectLogLevel>TRACE</ObjectLogLevel>
          </ObjectToLog>
        </ObjectLogging>
    </Logging>
```

```
    <CleanUps>
        <CleanUp>
            <CleanUpName>clean-temporary-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${tmp}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>archive-log-directory</CleanUpName>
            <CleanUpAction>archive</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>clean-log-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>7d</ParameterValue>
            </Parameter>
        </CleanUp>
    </CleanUps>
```

在 LocalConfig-mc.xml 文件正确的 XML 结构中指定任意这些字段将会覆盖默认内容。例如，以下配置会将默认日志级别更改为 FINEST。

将默认日志级别更改为 FINEST

```
<?xml version="1.0" encoding="UTF-8"?>
<LocalConfig>
      ... ...
      <Logging>
        <LogLevel>FINEST</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
      </Logging>
      ... ...
</LocalConfig>
```

示例 LocalConfig-mc.xml（MultiController 运行时）文件

```
<?xml version="1.0" encoding="UTF-8"?>
<Runtime>
   <Names>
      <Name name="mainClass">com.torokina.tim.mc.Main</Name>
      <Name name="appName">CAMM-Multi-Controller</Name>
      <Name name="appShortName">MC</Name>
      <Name name="primaryMcAddress">127.0.0.1</Name>
```

```
      <Name name="secondaryMcAddress"/>
      <Name name="primaryMcPort">29599</Name>
      <Name name="secondaryMcPort">-1</Name>
      <Name name="myMode">primary</Name>
      <Name name="myAddress">127.0.0.1</Name>
      <Name name="mcPort">29599</Name>
      <Name name="otherMcAddress"/>
      <Name name="otherMcPort">-1</Name>
      <Name name="heartbeatFrequency">15</Name>
      <Name name="heartbeatTimeout">180</Name>
      <Name name="repositoryFrequency">15</Name>
      <Name name="lcPort">29598</Name>
      <Name name="manageable">469</Name>
   </Names>
   <Paths>
      <Path name="license">${tim.base}/license.lic</Path>
      <Path name="apphome">${tim.base}/${appShortName}</Path>
      <Path name="runtimeConfig">${apphome}/runtime.xml</Path>
      <Path name="tmp">${apphome}/tmp</Path>
      <Path name="logbase">${apphome}/logs</Path>
      <Path name="basedir">${tim.base}</Path>
   </Paths>
</Runtime>
```

runtime.xml 文件会与隐藏的配置合并，然后用于启动 MultiController 组件。

MultiController 配置选项

以下信息说明您可以用来配置 MultiController 的选项。

路径

允许您为下列项指定路径信息：

basedir

指定 CA Mediation Manager 基目录。

apphome

指定 CA Mediation Manager MultiController 应用程序主目录。

tmp

指定 CA Mediation Manager MultiController 临时文件目录。

logbase

指定 CA Mediation Manager MultiController 日志目录。

runtimeConfig

指定由常规执行器提供的 CA Mediation Manager MultiController 运行时 XML 配置。

Java

允许您为使用 Java 指定选项。

CommandPath

指定常规执行器调用以启动 MultiController 的 Java 执行器的完整路径。

ClassPath/JarBase

允许您创建具有添加到 ClassPath 的一个或多个条目的列表。

Options

指定解析到 Java 的命令行选项。

Environment

指定执行 MultiController 组件的环境变量。

MainClass/Class

指定要执行的主 Java 类。

MainClass/Args

指定解析到 Java 类的参数。

运行时

允许您指定运行时选项。

Binding/BindAddress

指定 MultiController 组件绑定到的 IP 地址。

值：将 0.0.0.0 作为所有 MultiController 组件的 IP 地址。对于两个或更多个 IP 地址，使用逗号分隔列表。

MultiController 绑定到 TCP 端口 29599。

Binding/MyAddress

指定 MultiController 用来标识自身的 IP 地址。

注意：此 IP 地址必须是该主机上的有效 IP 地址。

MultiControllerConfig/Mode

指定 MultiController 的运行模式。

值：指定 Primary 或 Secondary。

MultiControllerConfig/MCAddresses/Other

指定群集中其他 MultiController 的 IP 地址。

MultiControllerConfig/Heartbeat/ParameterName == frequency

指定发送给其他 MultiController 的检测信号消息的频率。

MultiControllerConfig/Heartbeat/ParameterName == timeout

指定此 MultiController 等待来自 LocalController 组件的检测信号的时间段。如果 MultiController 180 秒没有接收到来自一个 LocalController 的检测信号，则 MultiController 会触发故障转移，转移到第一个可用的 LocalController。

日志记录

允许您指定日志记录选项。

LogLevel

指定输出日志文件中的日志记录级别。

值：指定 DEBUG、TRACE、INFO、WARNING 或 ERROR。

LogDirectory

指定输出日志文件中的日志记录级别。

值：指定 DEBUG、TRACE、INFO、WARNING 或 ERROR。

ObjectLogging/ObjectToLog/ObjectName

指定要启用日志记录的 Java 类名称。

ObjectLogging/ObjectToLog/ObjectLogLevel

指定 Java 类的日志记录级别。

清除

允许您指定清除选项。

CleanUpName

指定清除的描述性名称。

CleanUpAction

指定清除操作。

值：指定 Delete 或 Archive。

CleanupTarget

指定要清除的目录。

值：可以指定使用诸如 ${camm.variable} 等 CA Mediation Manager 变量。

Parameter/ParameterName—Parameter/ParameterValue

以 <n><unit> 格式指定要到期的参数及其到期时间。例如：

10d = 10 天

10h = 10 小时

10m = 10 分钟

手工启动和停止 MultiController

init.camm 脚本会自动启动 MultiController 进程。您可以使用 cammCtrl 实用工具独立地手工停止或启动 MultiController 组件。

遵循这些步骤:

1. 以 CAMM_USER 身份登录并转到 CAMM 主目录。
2. 执行以下命令以启动 MultiController：

   ```
   /opt/CA/CAMM# tools/startall -c mc
   ```

3. 执行以下命令以停止 MultiController：

   ```
   /opt/CA/CAMM # tools/stopall -c mc
   ```

LocalController 配置

LocalController 是安装在群集中每个服务器上的必要服务。

LocalController 执行下列主要功能：

- 促进运行在本地服务器上的子组件和远程 MultiController 之间的通信。
- 监视引擎和展示器的性能和可用性，并重新启动任何故障组件。
- 侦听来自群集成员的检测信号操作。默认情况下，LocalController 在端口 29598 或 TCP 上侦听。
- 将检测信号信息发送到 MultiController。
- 启动、停止和重新启动本地引擎和展示器。

在 GUI 安装期间，您可以仅配置安装基本 LocalController 的必需选项。但是，您可以手工编辑 LC 或 LocalConfig-lc.xml 文件。

示例 LocalConfig-lc.xml 文件（基本配置）

```
<LocalConfig>
      <Names>
            <Name name="mainClass">com.torokina.tim.lc.Main</Name>
            <Name name="primaryMcAddress">127.0.0.1</Name>
            <Name name="secondaryMcAddress"></Name>
            <Name name="primaryMcPort">29599</Name>
            <Name name="secondaryMcPort">-1</Name>
            <Name name="myAddress">127.0.0.1</Name>
            <Name name="appName">CAMM-Local-Controller</Name>
            <Name name="appShortName">LC</Name>
            <Name name="lcPort">29598</Name>
            <Name name="heartbeatFrequency">15</Name>
            <Name name="heartbeatTimeout">180</Name>
      </Names>
      <Paths>
            <Path name="dsLocalConfig">${basedir}/DS/LocalConfig-ds.xml</Path>
      </Paths>
</LocalConfig>
```

示例 LocalConfig-lc.xml 文件（隐藏的日志和清理配置）

```
    <Logging>
        <LogLevel>INFO</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
        <ObjectLogging>
          <ObjectToLog>
            <ObjectName>com.torokina.tim.config</ObjectName>
            <ObjectLogLevel>TRACE</ObjectLogLevel>
        </ObjectToLog>
        </ObjectLogging>
    </Logging>
    <CleanUps>
```

```
        <CleanUp>
            <CleanUpName>clean-temporary-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${tmp}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>archive-log-directory</CleanUpName>
            <CleanUpAction>archive</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>clean-log-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>7d</ParameterValue>
            </Parameter>
        </CleanUp>
    </CleanUps>
```

示例 LocalConfig-lc.xml 文件（LocalController 运行时）

```
<?xml version="1.0" encoding="UTF-8"?>
<Runtime>
   <Names>
      <Name name="mainClass">com.torokina.tim.lc.Main</Name>
      <Name name="primaryMcAddress">127.0.0.1</Name>
      <Name name="secondaryMcAddress"/>
      <Name name="primaryMcPort">29599</Name>
      <Name name="secondaryMcPort">-1</Name>
      <Name name="myAddress">127.0.0.1</Name>
      <Name name="appName">CAMM-Local-Controller</Name>
      <Name name="appShortName">LC</Name>
      <Name name="lcPort">29598</Name>
      <Name name="heartbeatFrequency">15</Name>
      <Name name="heartbeatTimeout">180</Name>
      <Name name="mcPort">29599</Name>
      <Name name="manageable">996</Name>
   </Names>
   <Paths>
      <Path name="dsLocalConfig">${basedir}/DS/LocalConfig-ds.xml</Path>
      <Path name="apphome">${tim.base}/${appShortName}</Path>
      <Path name="runtimeConfig">${apphome}/runtime.xml</Path>
      <Path name="tmp">${apphome}/tmp</Path>
      <Path name="logbase">${apphome}/logs</Path>
```

```
    <Path name="basedir">${tim.base}</Path>
  </Paths>
</Runtime>
```

LocalController 配置选项

以下信息说明您可以用来配置 LocalController 的选项。

路径

允许您为下列项指定路径信息：

basedir

指定 CA Mediation Manager 基目录。

apphome

指定 CA Mediation Manager LocalController 应用程序主目录。

tmp

指定 CA Mediation Manager LocalController 临时文件目录。

logbase

指定 CA Mediation Manager LocalController 日志目录。

runtimeConfig

指定由常规执行器提供的 CA Mediation Manager LocalController 运行时 XML 配置文件。

Java

允许您为使用 Java 指定选项。

CommandPath

指定常规执行器调用以启动 LocalController 的 Java 执行器的完整路径。

ClassPath/JarBase

允许您创建具有添加到 ClassPath 的一个或多个条目的列表。

Options

指定解析到 Java 的命令行选项。

Environment

指定执行 MultiController 组件的环境变量。

MainClass/Class

指定要执行的主 Java 类。

MainClass/Args

指定解析到 Java 类的参数。

运行时

允许您指定运行时选项。

Binding/BindAddress

指定 LocalController 组件绑定到的 IP 地址。

值：将 0.0.0.0 作为所有 LocalController 组件的 IP 地址。对于两个或更多个 IP 地址，使用逗号分隔列表。

Binding/MyAddress

指定 LocalController 用来标识自身的 IP 地址。

注意：此 IP 地址必须是该主机上的有效 IP 地址。

Binding/BindPort

指定 LocalController 用来标识自身的 IP 地址。

注意：此 IP 地址必须是该主机上的有效 IP 地址。默认情况下，LocalController 绑定到 TCP 端口 29598。

LocalControllerConfig/Mode

指定 LocalController 的运行模式。

值：指定 Active 或 Standby。

LocalControllerConfig/MCAddresses/Primary

指定群集中主要 MultiController 的 IP 地址。

LocalControllerConfig/MCAddresses/Secondary

指定群集中辅助 MultiController 的 IP 地址。

LocalControllerConfig/Heartbeat/ParameterName == frequency

指定发送给 MultiController 的检测信号消息的频率。

LocalControllerConfig/Heartbeat/ParameterName == timeout

指定此 LocalController 等待来自子组件（引擎和展示器）的检测信号的时间。如果 LocalController 180 秒没有从子组件收到检测信号，LocalController 将重新启动。

日志记录

允许您指定日志记录选项。

LogLevel

指定输出日志文件中的日志记录级别。

值：指定 DEBUG、TRACE、INFO、WARNING 或 ERROR。

LogDirectory

指定输出日志文件中的日志记录级别。

值：指定 DEBUG、TRACE、INFO、WARNING 或 ERROR。

ObjectLogging/ObjectToLog/ObjectName

指定要启用日志记录的 Java 类名称。

ObjectLogging/ObjectToLog/ObjectLogLevel

指定 Java 类的日志记录级别。

清除

允许您指定清除选项。

CleanUpName

指定清除的描述性名称。

CleanUpAction

指定清除操作。

值：指定 Delete 或 Archive。

CleanupTarget

指定要清除的目录。

值：可以指定使用诸如 ${camm.variable} 等 CA Mediation Manager 变量。

Parameter/ParameterName—Parameter/ParameterValue

以 <n><unit> 格式指定要到期的参数及其到期时间。例如：

10d = 10 天

10h = 10 小时

10m = 10 分钟

手工启动和停止 LocalController

init.camm 脚本会自动启动 LocalController 进程。您可以使用 cammCtrl 实用工具独立地手工停止或启动 LocalController 组件。

遵循这些步骤:

1. 以 CAMM_USER 身份登录并转到 CAMM 主目录。
2. 执行以下命令以启动 LocalController：

   ```
   /opt/CA/CAMM# tools/startall -c lc
   ```

3. 执行以下命令以停止 LocalController：

   ```
   /opt/CA/CAMM # tools/stopall -c lc
   ```

引擎和展示器配置

使用每个设备包随附的设备包安装程序来完成引擎和展示器的安装和配置。

故障转移操作

MultiController 以以下两种模式之一运行：主要或辅助。

MultiController 有三个主要通信功能：

- 检测信号
- 名称服务查找
- 存储库同步

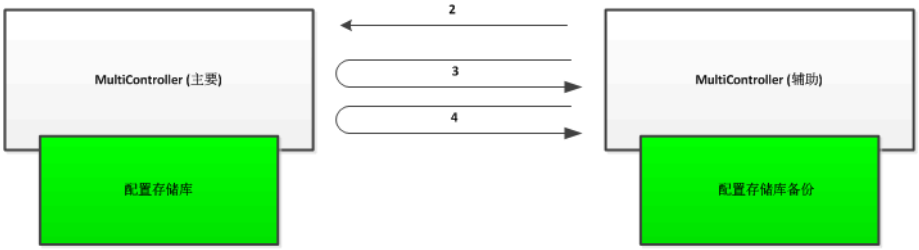
MultiController 故障

以下信息说明 CA Mediation Manager 如何管理 MultiController 故障。

MultiController 通信

下图显示了辅助 MultiController 如何与主要 MultiController 通信。

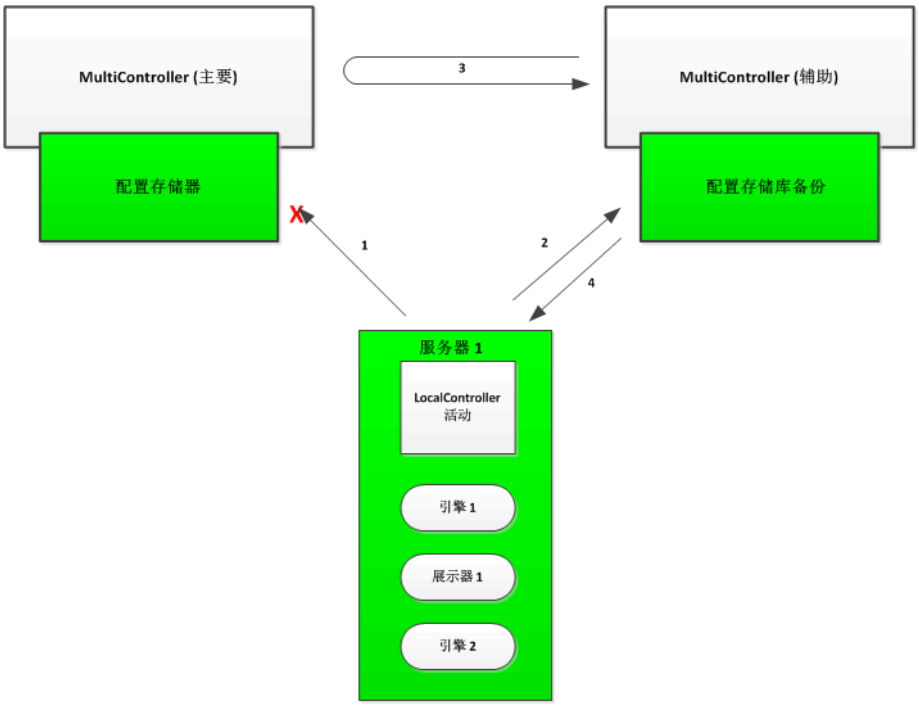
“主要”上有一个 watchdog，也用来关注“辅助”是否变为不可用。当“辅助”变为不可用时，将在日志文件中创建报警。



1. 辅助 MultiController 启动。
2. 辅助 MultiController 向主要 MultiController 发送第一个检测信号以注册为活动。
3. 辅助 MultiController 定期向主要 MultiController 发送检测信号。
4. 辅助 MultiController 定期同步存储库。

主要 MultiController 故障

下图显示了辅助 MultiController 如何处理主要 MultiController 的故障。在此进程期间，辅助 MultiController 将接管检测信号、名称服务和配置存储库请求，直到主要 MultiController 重新可用。



1. LocalController 向主要 MultiController 发送检测信号，但失败了。
2. LocalController 向辅助 MultiController 发送检测信号。
3. 因为辅助 MultiController 认为主要 MultiController 仍可用，所以它代理对主要 MultiController 的请求。
4. 主要 MultiController 通过辅助 MultiController 响应 LocalController 的检测信号。

MultiController 代理功能

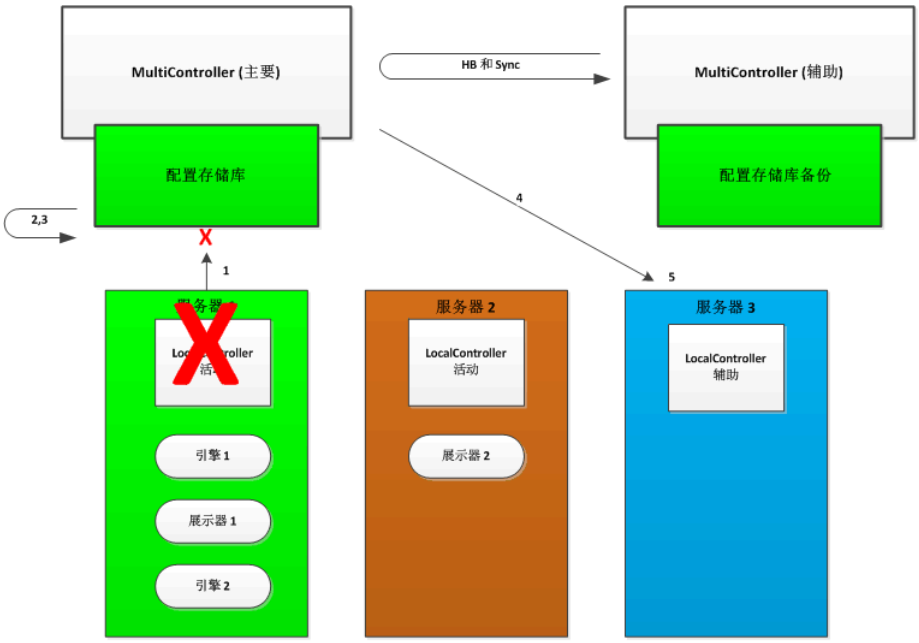
在某些网络环境下，LocalController 无法直接联系主要 MultiController。鉴于此，辅助 MultiController 具有代理功能，并尝试向主要 MultiController 发送检测信号。

但是，发送给辅助 MultiController 的配置存储库请求是由辅助 MultiController 处理的，以便同步配置存储库。

LocalController 故障

所有 LocalController 组件都会将检测信号信息发送到 MultiController。如果活动的 MultiController 检测到它没有收到来自 LocalController 的检测信号，将记录丢失的检测信号。如果此时间超过超时时段，那么 MultiController 将采用下图所述的过程触发故障转移，转移到第一个可用的待机 LocalController。

故障回复是手工过程。要实现故障转移，请强制服务器 3 上的 LocalController 发生故障。



1. MultiController 意识到在超过超时时段的时间内没有接收到检测信号。
2. MultiController 将所有引擎子组件的配置存储库从服务器 1 移动到服务器 3。
3. 如果服务器 1 没有安装任何子组件，那么 MultiController 将服务器 1 的配置标记为待机。
4. MultiController 将消息发送到服务器 3 的 LocalController，以便重新启动并重新读取其配置。
5. 服务器 3 上的 LocalController 变为活动状态。

子组件故障

LocalController 组件监视其正在运行的子组件（引擎和展示器）。如果某个子组件出现故障，将立即重新启动该子组件。

高可用性配置

以下部分说明如何为高可用性配置 CA Mediation Manager。

配置主要 MultiController

在主要 MultiController 安装期间，为高可用性配置主要 MultiController。

遵循这些步骤:

1. 在“MultiController 配置”面板中，选中“*此群集中是否将存在其他 MC*”复选框。
2. 提供主要 MultiController IP 地址，然后单击“下一步”。
3. 在“其他 MultiController 配置”面板中，提供辅助 MultiController 的 IP 地址。
4. 单击“下一步”并完成安装。

配置辅助 MultiController

在辅助 MultiController 安装期间，为高可用性配置辅助 MultiController。

遵循这些步骤:

1. 在“MultiController 配置”面板中，从“MultiController”下拉列表中选择“辅助”。
2. 提供辅助 MultiController IP 地址。
3. 选中“*此群集中是否将存在其他 MC*”复选框，然后单击“下一步”。
4. 在“其他 MultiController 配置”面板中，提供主要 MultiController 的 IP 地址。
5. 单击“下一步”并完成安装。

配置 LocalController

在 LocalController 安装期间，为高可用性配置 LocalController。

遵循这些步骤:

1. 在“LocalController 配置”面板中，提供主要 MultiController 和辅助 MultiController 的 IP 地址。
2. 单击“下一步”并完成安装。
3. 对每个 LocalController 安装重复这些步骤。

日志文件配置

以下部分说明如何配置 CA Mediation Manager 中的日志文件。

CA Mediation Manager 组件启动时，默认 log 目录会创建 logging.properties 文件，您可以使用该文件配置配置所有组件的日志记录属性。logging.properties 文件预配置为在默认 log 目录中生成日志文件。不过，编辑 logging.properties 文件，以将日志文件重定向到其他目录中。修改 logging.properties 文件后，重新启动组件，以加载修改的日志记录属性。所有日志（应用程序日志和 STD-ERROR/STD-OUTPUT 日志）会在 logging.properties 文件中指定的目录中生成。

logging.properties 文件—示例（按组件）

以下示例说明不同的 logging.properties 文件：

MultiController 示例 logging.properties 文件

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/MC/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Multi-Controller-
```

LocalController 示例 logging.properties 文件

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/LC/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Local-Controller-
```

传递系统示例 logging.properties 文件

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/DS/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Delivery-System-
```

ENGINE_CAMM 示例 logging.properties 文件

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/COMPONENTS/ENGINE_CAMM/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO
```

```
com.torokina.common.logging.apache.FileHandler.prefix=CAMM-ENGI
NE_CAMM-
```

常规执行器示例 logging.properties 文件

默认情况下，常规执行器不在其日志目录中创建 logging.properties 文件。所有日志都在 ~GE/logs 目录中生成。以下示例说明您可以如何在 logs 目录中创建 logging.properties 文件，以重定向常规执行器日志（仅非 Windows 平台）：

```
#Properties for Logger

#Tue May 07 04:08:45 EDT 2013

com.torokina.common.logging.apache.FileHandler.directory=/opt/C
A/CAMM/GE/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Gene
ric-Executor-
```

配置日志文件清除

默认情况下，每个组件的清除操作均配置为在 *logbase* 目录上运行。如果日志文件被重定向到其他目录，为了成功完成 *Archive* 或 *Delete* 操作，请修改清除配置。

在配置文件的 *LocalConfig* xml-element 中定义清除操作。针对所有 CAMM 组件，在各自的配置文件中定义清除操作：

- **MultiController**：LocalConfig-mc.xml
- **LocalController**：LocalConfig-lc.xml
- **传递服务**：LocalConfig-ds.xml
- **子组件（引擎/展示器）**：TemplateConfig-subcomponent.xml

  注意：您可以在 <camm.base>/LC 目录中找到 TemplateConfig-subcomponent.xml 文件。

以下示例说明了两个示例清除配置。

示例：清除传递服务组件的配置文件（*Delete* 操作）

```
<LocalConfig>
<Description>Configuration for Delivery Module</Description>
        …
        …
        <CleanUps>
                <!-- SAMPLE  DELETE  ACTION -->
                <CleanUp>
                        <CleanUpName>Delete</CleanUpName>
                        <CleanUpAction>delete</CleanUpAction>

        <CleanUpTarget>${apphome}/.local</CleanUpTarget> <!--
Directory Name -->
                        <Parameter>

        <ParameterName>expire</ParameterName>
                                <ParameterValue>7d</ParameterValue>
<!-- 1y0m3d1h -->
                        </Parameter>
                        <Parameter>

        <ParameterName>includeDir</ParameterName>

        <ParameterValue>true</ParameterValue><!-- true/false -->
                        </Parameter>
                        <Parameter>

        <ParameterName>recursive</ParameterName>

        <ParameterValue>true</ParameterValue><!-- true/false -->
                        </Parameter>
                        <Parameter>
                                <ParameterName>match</ParameterName>
```

```
                <ParameterValue>^[\d]+\.xml$</ParameterValue>
                                <!-- Regular Pattern -->
                        </Parameter>
                </CleanUp>
```

示例：清除传递服务组件的配置文件（*Archive* 操作）

```
<!-- SAMPLE  ARCHIVE  ACTION -->
<CleanUp>
<CleanUpName>Archive</CleanUpName>
<CleanUpAction>archive</CleanUpAction>
<CleanUpTarget>${logbase}</CleanUpTarget>
<!-- Directory Name -->
<Parameter>
<ParameterName>expire</ParameterName>
<ParameterValue>7d</ParameterValue>
<!-- 1y0m3d1h -->
</Parameter>
<Parameter>
<ParameterName>includeDir</ParameterName>
<ParameterValue>true</ParameterValue><!-- true/false -->
</Parameter>
<Parameter>
<ParameterName>recursive</ParameterName>
<ParameterValue>true</ParameterValue><!-- true/false -->
</Parameter>
<Parameter>
<ParameterName>match</ParameterName>
<ParameterValue>CAMM-.*\.log</ParameterValue>
<!-- Regular Pattern -->
</Parameter>
<Parameter>
<ParameterName>achiveHome</ParameterName>
```

```
      <ParameterValue>${logbase}</ParameterValue> <!-- folder
path -->
                  </Parameter>
                  <Parameter>

      <ParameterName>achivePrefix</ParameterName>

      <ParameterValue>Archive-</ParameterValue> <!-- prefix
string -->
                  </Parameter>
                  <Parameter>

      <ParameterName>achiveSuffix</ParameterName>

      <ParameterValue>.zip</ParameterValue> <!-- suffix string
-->
                  </Parameter>
            </CleanUp>
      </CleanUps>
<LocalConfig>
```

您可以包括组件配置文件中的任何清除操作。常规执行器执行所有清除操作。常规执行器将清除文件存储在 {camm.base}/GE_<User>/cleanup 目录中，其中 camm.base 是 CA Mediation Manager 安装目录。组件配置文件修改后，请重新启动相关组件，以使修改的配置生效。

第 5 章： 将 EMS 配置文件用于 CA Mediation Manager for Infrastructure Management 2.0 版本 2.2.3

注意：本章中的信息仅适用于 CA Mediation Manager for Infrastructure Management 2.0。

此部分包含以下主题：

EMS 集成配置文件

EMS 集成配置文件指定 EMS 清单发现在您的 Data Aggregator 环境中的运行方式。

使用 EMS 集成配置文件，您可以指定状态、Data Collector、设备包、EMS IP 以及备份 EMS IP。为您创建的每个 EMS 集成配置文件指定一个 IP 域。您指定的 IP 域适用于管理多个设备（通常同时管理 1000 个设备）的目标 EMS 服务器。Data Aggregator 始终同时处理来自 EMS 服务器的所有清单数据。（使用 SNMP 或 ICMP，按设备逐个轮询。）

注意：

- 您安装设备包之后，“EMS 集成配置文件”选项在用户界面中变得可见。
- 不要多次为同一 Data Collector 添加相同的 EMS 集成配置文件。

添加 EMS 集成配置文件

您可以创建 EMS 集成配置文件以指定 EMS 清单发现在您的 Data Aggregator 环境中的运行方式。

注意：您必须以管理员身份登录才能执行该任务。

没有事先将范围设置为某一承租方的情况下，创建 EMS 集成配置文件会将配置文件放入全局空间中，可供所有承租方访问。使用全局空间中的配置文件运行发现，允许任何人看到发现结果，无论他们是否将范围设置为某一承租方。

因此，在创建 EMS 集成配置文件*之前*，将范围设置为某一承租方，以使该配置文件仅供特定承租方访问。将范围设置为某一承租方之后，页面右上角将显示承租方指示器。之后，您可以手工将承租方与 CA Infrastructure Management 进行同步，或等待自动同步。承租方与 Data Aggregator 同步之前，您不能创建 EMS 集成配置文件。

注意：有关将范围设置为某一承租方以及同步承租方的详细信息，请参阅《*CA Performance Center 管理员指南*》。

遵循这些步骤:

1. 选择“管理”、“数据源设置”，然后单击 CA Performance Center 用户界面中的一个 Data Aggregator 数据源。
2. 在“监视配置”菜单中单击“EMS 集成配置文件”。

   “EMS 集成配置文件”页面将打开，其中显示可用的发现配置文件的列表。
3. 单击“新建”。

   此时将打开“添加 EMS 发现配置文件”对话框。
4. 在字段中输入必需的信息。显示的配置字段取决于您选择的设备包。每个设备包都具有您配置的唯一全局变量。

   注意：有关每个产品的唯一全局变量的信息，请参阅 CA 支持站点。
5. 单击“保存”。

   EMS 集成配置文件已创建。

   单击“保存”并选定“已启用”选项时，清单发现*不会*自动运行。仅当满足以下条件之一时，清单发现才会运行：

   - 达到清单轮询排定。
   - EMS 集成配置文件已手工启动。

详细信息：

查看 EMS 发现结果 (p. 54)
手工启动 EMS 发现 (p. 53)

手工启动 EMS 发现

EMS 集成配置文件发现网络中的设备及其组件。您可以手工启动 EMS 集成配置文件以开始发现。

注意：您也可以等待达到清单轮询排定，发现自动开始。

遵循这些步骤:

1. 选择“管理”、“数据源设置”，然后单击 CA Performance Center 用户界面中的一个 Data Aggregator 数据源。
2. 在“监视配置”菜单中单击“EMS 集成配置文件”。

   “EMS 集成配置文件”页面将打开，其中显示可用的发现配置文件的列表。
3. 选择要对其运行发现的一个或多个 EMS 集成配置文件，然后单击“启动”。

   注意：只能在状态为“就绪”的 EMS 集成配置文件上运行发现。

   此时将打开确认对话框。
4. 单击“是”。

   发现启动。选定发现配置文件的“状态”列指示“已启动”。

   此时将打开确认对话框。
5. 单击“确定”。

   将发现设备及其所有关联接口，轮询开始。此时将返回到“EMS 集成配置文件”页面。

   如果发现挂起 10 分钟以上，Data Aggregator 会将其停止。如果在 10 分钟之内未发现任何新设备，*并且*所选的发现配置文件的状态在 10 分钟之内未更改，则 Data Aggregator 会将发现视为挂起。将在发现实例项上生成一个审核事件。如果没有成功发现设备，选定发现配置文件的“状态”列指示“失败”。如果至少成功发现一台设备，而非所有设备，“状态”列指示“部分失败”。

   已发现的设备和组件需要最多 5 分钟来与 CA Performance Center 同步。同步完成后，已发现的设备和组件将显示在 CA Performance Center 的“清单”选项卡中。

详细信息：

查看 EMS 发现结果 (p. 54)

查看 EMS 发现结果

您可以查看发现的所有可管理 EMS 设备的数目汇总。

遵循这些步骤:

1. 选择“管理”、“数据源设置”，然后单击 CA Performance Center 用户界面中的一个 Data Aggregator 数据源。
2. 在“监视配置”菜单中单击“EMS 集成配置文件”。

   “EMS 集成配置文件”页面将打开，其中显示可用的发现配置文件的列表。
3. 选择要查看发现结果的 EMS 集成配置文件实例，然后单击“历史”。

   EMS 历史结果显示如下：
   - “设备”表显示监视的设备及其创建时间。
   - “元素”表显示监视的接口及其创建时间。

启动或停止 EMS 发现服务

“启动”服务用于 EMS 服务器上的发现以连续复查清单。发现可按排定运行，但您也可以根据需求手工启动、停止或重新启动服务。例如，您可以在 EMS 服务器发生故障并重新运行后，或者升级设备包安装后，重新启动发现。

停止服务将删除所有非活动数据轮询，但是会等待活动数据轮询完成，而不会将其中断。EMS 文件不会被删除。只要处于已停止状态，此操作还会禁用任何新的轮询。在您重新启动前，服务将保持已停止状态。

重新启动 Data Collector 对 EMS 集成配置文件的状态没有任何影响。

注意：您必须以管理员身份登录才能执行该任务。

遵循这些步骤:

1. 选择“管理”、“数据源设置”，然后单击 CA Performance Center 用户界面中的一个 Data Aggregator 数据源。
2. 在“监视配置”菜单中单击“EMS 集成配置文件”。

   “EMS 集成配置文件”页面将打开，其中显示可用的发现配置文件的列表。

3. 选择配置文件，然后单击“启动”或“停止”。

   此时将打开确认对话框。

4. 单击“是”确认该操作。

   服务将启动或停止（取决于您的选择）。在您手工进行更改之前，此服务将保持已启动或已停止状态。

添加事件规则

可通过 Data Aggregator 中的“监视配置文件”对话框添加事件规则。

注意：有关详细信息，请参阅《*Data Aggregator 管理员指南*》或联机帮助。