

# 第 7 章 管理 WebSphere Product Center 服务

## 服务类型

完整的 WebSphere Product Center 系统由下列并发运行的服务组成：

admin	管理服务器启动 / 停止远程机器上的模块
appsvr	应用程序服务器处理 JSP
eventprocessor	事件处理器在所有模块之间分派事件
queuemanager	队列管理器将文档发送到 WebSphere Product Center 之外
scheduler	调度程序运行后台作业
workflow	工作流引擎

### admin\_properties.xml 和集群

服务可以在工作站的集群中运行。集群中的不同机器是在 admin\_properties.xml 文件中定义的：

\$TOP/etc/default/admin\_properties.xml

**注意：** admin\_properties.xml 中提供了更多的信息。每个服务都可以在 admin\_properties.xml 文件中列示的任何机器上运行。

典型的 WebSphere Product Center 集群可以在 WebSphere Product Center 服务器上包含应用程序服务器和“RMI 注册程序”支持实用程序，并且在辅助服务器上包含其余 WebSphere Product Center 组件。

对于主服务器故障转移，可以通过最小的努力来让先前未在辅助服务器上运行的服务重新联机，从而最大程度地缩短了停机时间。

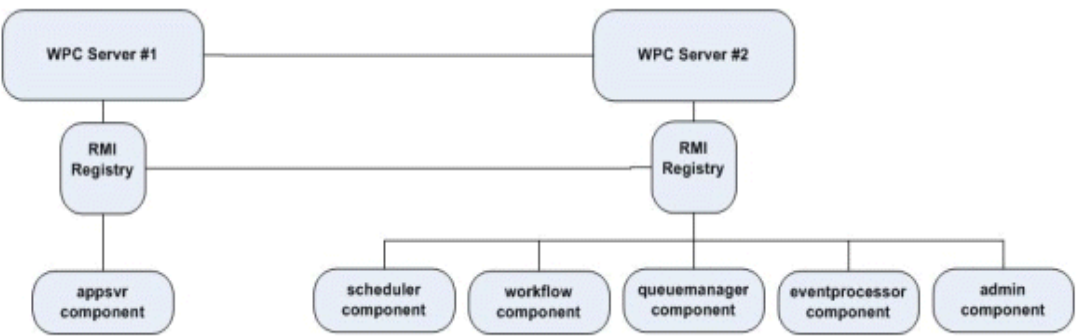


图 4 — 典型 WebSphere Product Center 集群

## 服务名称 — 长名称和短名称

每个服务都由服务名称唯一地标识。服务名称必须是唯一的（如果有另一个同名的服务正在集群中的某台机器上运行，则服务将拒绝启动。）

只要服务名称不相同，每个服务都可以在多台机器上运行。

“admin”和“appsvr”服务的名称已被系统固定。

对于 admin，服务名称是 admin\_<machine name>（例如“admin\_server1”）

对于 appsvr，服务名称是 appsvr\_<machine name>（例如“appsvr\_server1”）

对于其它服务，检取任意的名称。选择的名称实际上是服务的短名称。

在内部，使用这个短名称来构建长名称：

```
rmi://<machine name>:<rmi port>/<db user name>/<service type>/<service short name>
```

示例：

如果正在机器“server1”上使用 RMI 端口 17507 来运行“scheduler”服务，正在连接至数据库用户“pauadm”，并且服务名称是“sch1”，则长名称是：

```
rmi://server1:17507/pauadm/scheduler/sch1
```

如果在 server2 上为同一个用户和端口运行另一个调度程序（sch2），则长名称是：

```
rmi://server2:17507/pauadm/scheduler/sch2
```

## 为服务类型设置内存标志

各种 WebSphere Product Center 服务的内存标志是在 WebSphere Product Center 初始化脚本中设置的，该脚本位于 WebSphere Product Center 安装目录中。

```
<install location>/setup/init_ccd_vars.sh
```

建议您将下列内存标志设置用于 WebSphere Product Center 服务：

```
export ADMIN_MEMORY_FLAG='-Xmx64m -Xms48m'
```

```
export APPSVR_MEMORY_FLAG='-Xmx512m -Xms64m'
```

```
export EVENTPROCESSOR_MEMORY_FLAG='-Xmx64m -Xms48m'
```

```
export QUEUEMANAGER_MEMORY_FLAG='-Xmx64m -Xms48m'
```

```
export SCHEDULER_MEMORY_FLAG='-Xmx1024m -Xms48m'
```

```
export WORKFLOWENGINE_MEMORY_FLAG='-Xmx64m -Xms48m'
```

## RMI — 远程方法调用

服务注册是通过 RMI（Java 远程方法调用）完成的。在运行任何服务之前，确保已在机器上启动了 RMI。

### RMI 状态

要获取集群中所有正在运行的服务的列表，请执行以下脚本：

```
$TOP/bin/go/rmi_status.sh
```

此脚本与集群中的所有机器上的 RMI 守护进程联系并获取每台机器上的本地服务的列表。它返回长名称的列表。

### 日志文件

每个服务都将生成运行时日志文件

```
$TOP/logs/<service>/<service name>/svc.out
```

示例：

名为“sch1”的调度程序在 \$TOP/logs/scheduler/sch1 中生成运行时日志文件 svc.out

在启动服务之后，建议您检查该日志文件以确保所有内容都已启动并且没有发生任何问题。

### 启动服务

下列各节描述如何使用本地脚本来控制服务。在可以使用某个服务之前，必须在使用该服务的机器上启动 RMI 注册程序。

要启动 RMI，请运行以下脚本：

```
$TOP/bin/go/start/start_rmiregistry.sh
```

在本地机器上启动服务

在本地机器上启动服务的最简单方法是使用 \$TOP/bin/go/start/ 目录中

的脚本。

脚本	描述
start_admin.sh	启动 admin 服务
start_appsvr.sh	启动应用程序服务器
start_eventprocessor.sh	启动事件处理器
start_queuemanager.sh	启动队列管理器
start_rmiregistry.sh	启动 RMI 注册程序
start_scheduler.sh	启动调度程序
start_workflowengine.sh	启动工作流引擎

在这些脚本中，每个脚本（start\_admin.sh、start\_appsvr.sh 和 start\_rmiregistry.sh 除外）都可以接收服务名称来作为可选自变量：

`-svc_name=<service name>`

admin 和 appsvr 服务使用缺省名称（admin\_<machine name> 和 appsvr\_<machine name>）。指定另一个名称是不起任何作用的。

如果未指定任何服务名称，则它使用缺省名称：

对于调度程序，使用 “scheduler”

对于事件处理器，使用 “eventprocessor”

对于队列管理器，使用 “queuemanager”

对于工作流引擎，使用 “workflow”

**注意**，如果启动一个本地服务，并且它具有已在运行中的本地服务的名称，则先前的本地服务将首先异常终止。因此，这些脚本也可以用来 “重新启动” 服务（首先异常终止，然后重新启动）。

*示例：*

要启动具有名称 “sch1” 的调度程序：

```
$STOP/bin/go/start/start_scheduler.sh -svc_name=sch1
```

要启动具有缺省名称的调度程序：

```
$STOP/bin/go/start/start_scheduler.sh
```

## 异常终止服务

异常终止一个服务将关闭该服务并使该服务变为不可用。

例如，如果调度程序正在运行一个作业，则该作业将在进程的中间异常终止。

在本地机器上异常终止服务

此处的结构反映的是该起始结构。

使用 \$STOP/bin/go/abort/ 目录中的脚本。

脚本	描述
abort_admin.sh	异常终止 admin 服务
abort_appsvr.sh	异常终止应用程序服务器
abort_eventprocessor.sh	异常终止事件处理器
abort_queuemanager.sh	异常终止队列管理器
abort_rmiregistry.sh	异常终止 RMI 注册程序
abort_scheduler.sh	异常终止调度程序
abort_workflowengine.sh	异常终止工作流引擎

在这些脚本中，每个脚本（abort\_admin.sh、abort\_appsvr.sh 和 abort\_rmiregistry.sh 除外）都可以接收服务名称来作为可选自变量：

-svc\_name=<service name>

**注意：**异常终止 RMI 将导致不可能与远程机器上的服务联系。

## 停止服务

停止服务将请求平稳地关闭该服务。如果该服务“被阻塞”，则它可能完全不执行关闭过程。在调度程序完成执行所有当前正在运行的作业之前，它将不会停止。

在本地机器上停止服务

此处的结构反映的是起始结构。

使用 \$STOP/bin/go/stop/ 目录中的脚本。

脚本	描述
stop_admin.sh	停止 admin 服务
stop_appsvr.sh	停止应用程序服务器
stop_eventprocessor.sh	停止事件处理器
stop_queuemanager.sh	停止队列管理器
stop_scheduler.sh	停止调度程序
stop_workflowengine.sh	停止工作流引擎

在这些脚本中，每个脚本（abort\_admin.sh、abort\_appsvr.sh 和 abort\_rmiregistry.sh 除外）都可以接收服务名称来作为可选自变量：

-svc\_name=<service name>

## 有关异常终止和停止的重要说明

应该使用哪一项操作？停止还是异常终止？

异常终止	保证服务将被关闭，但它不能保证当前正在运行的任务不被中断。
停止	保证“如果”该服务被停止，则该服务将在当前正在运行的每项任务首先停止后平稳地停止。

## 启动所有 WebSphere Product Center 模块

在本地机器上启动 WebSphere Product Center。

运行脚本 `$TOP/bin/go/start/start_local.sh`

这将启动 RMI 注册程序以及下列服务：

- 名为“admin\_<machine name>”的 admin
- 名为“appsvr\_<machine name>”的应用程序服务器
- 名为“eventprocessor”的事件处理器
- 名为“queuemanager”的队列管理器
- 名为“scheduler”的调度程序
- 名为“workflow”的工作流

**注意：**在启动任何服务之前，此脚本将首先尝试关闭本地机器上的任何现有系统。

## 异常终止本地机器上的 WebSphere Product Center

运行脚本 `$TOP/bin/go/abort/abort_local.sh`

在本地机器上启动的每个服务都异常终止。RMI 注册程序也异常终止。

## 停止本地机器上的 WebSphere Product Center

运行脚本 `$TOP/bin/go/stop/stop_local.sh`

在本地机器上启动的每个服务都停止。缺省情况下，RMI 注册程序与其它服务一起停止。要保持 RMI 注册程序运行，请传递以下选项：

`--kill_rmi=no`

**注意：**在“kill\_rmi=no”前面有两个连字符。

## 服务状态

获取服务的短状态

要获取服务的短状态，请传递下列参数：

-cmd=check -svc=<service name>

示例:

要获取调度程序的状态:

```
rootadmin.sh -cmd=check -svc=scheduler
```

短状态可以是下列其中一种:

正在运行	该服务正在运行并且正在响应“脉动信号”功能。
找不到	找不到该服务。该服务可能尚未启动，或者可能已崩溃。
找到但未响应	找到该服务，并发现该服务已对 RMI 注册程序注册，但它不响应“脉动信号”功能。可能必须重新启动该服务。

## 获取服务的长状态

要获取服务的长状态，请传递下列参数:

-cmd=status -svc=<service name>

它将生成一个 html 文件，可以使用任何浏览器来查看该文件。在终端上，您可能想使用 lynx 来格式化输出。

示例:

要获取调度程序的状态:

```
rootadmin.sh -cmd=status -svc=scheduler  
> /tmp/sch_status.html; lynx /tmp/sch_status.html
```

或者

```
rootadmin.sh -cmd=status -svc=scheduler  
> /tmp/sch_status.html; lynx -dump /tmp/sch_status.html
```

**注意：**以上示例中使用的“>”将状态详细信息定向至一个文件输出位置。

此状态提供正在该服务中运行的不同线程的概述以及该服务当前所使用的数据库连接的状态。