Red Hat JBoss Enterprise Application Platform 7.3

配置指南

设置和维护红帽 JBoss 企业应用平台的说明，包括运行应用程序和服务.

Last Updated: 2023-03-27

Red Hat JBoss Enterprise Application Platform 7.3 配置指南

设置和维护红帽 JBoss 企业应用平台的说明，包括运行应用程序和服务.

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

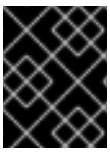
摘要

本指南的目的是涵盖设置和维护 JBoss EAP 以及在其上运行应用程序和其他服务所需的许多配置任务。

目录

第 1 章 简介

在使用本指南配置 JBoss EAP 之前，假定已下载和安装了 JBoss EAP 的最新版本。有关安装说明，请参阅 JBoss EAP *安装指南*。



重要

由于 JBoss EAP 的安装位置因主机计算机而异，本指南将安装位置称为 ***EAP_HOME***。在执行管理任务时，应使用 JBoss EAP 安装的实际位置，而非 ***EAP_HOME***。

第 2 章 启动和停止 JBOSS EAP

2.1. 启动 JBOSS EAP

JBoss EAP 在红帽企业 Linux、Windows Server 和 Oracle Solaris 上受支持，并且以单机服务器或受管域工作模式运行。启动 JBoss EAP 的具体命令取决于底层平台和所需的工作模式。

服务器最初以暂停状态启动，在启动所有必要的服务之前，将不会接受任何请求；达到此刻，服务器将置于正常运行状态并可以开始接受请求。

启动 **JBoss EAP 作为单机服务器**

```
$ EAP_HOME/bin/standalone.sh
```

注意

对于 Windows Server，请使用 ***EAP_HOME*\bin\standalone.bat** 脚本。

此启动脚本使用 ***EAP_HOME*/bin/standalone.conf** 文件或 **standalone.conf.bat** 用于 Windows Server 来设置一些默认首选项，如 JVM 选项。您可以自定义此文件中的设置。

JBoss EAP 默认使用 **standalone.xml** 配置文件，但可使用其他配置文件启动。有关可用独立配置文件以及如何使用它们的详情，请查看单机服务器配置文件部分。

有关所有可用启动脚本参数及其目的的完整列表，请使用 **--help** 参数或查看 服务器运行时参数 部分。

在受管域中启动 JBoss EAP
域控制器必须在域中任何服务器组的服务器之前启动。使用此脚本首先启动域控制器，然后为每个关联的主机控制器启动。

```
$ EAP_HOME/bin/domain.sh
```

注意

对于 Windows Server，请使用 ***EAP_HOME*\bin\domain.bat** 脚本。

此启动脚本使用 ***EAP_HOME*/bin/domain.conf** 文件或 **domain.conf.bat** 作为 Windows Server 来设置一些默认首选项，如 JVM 选项。您可以自定义此文件中的设置。

JBoss EAP 默认使用 **host.xml** 主机配置文件，但可使用其他文件启动。有关可用的受管域配置文件和使用方法的详细信息，请参阅受管域配置文件部分。

在设置受管域时，需要将其他参数传递到启动脚本。有关所有可用启动脚本参数及其目的的完整列表，请使用 **--help** 参数或查看 服务器运行时参数 部分。

2.2. 停止 JBOSS EAP

停止 JBoss EAP 的方式取决于它的启动方式。

停止 JBoss EAP 的交互实例
在启动 JBoss EAP 的终端中按 **Ctrl+C**。

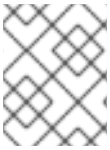
停止 JBoss EAP 的后台实例
使用管理 CLI 连接正在运行的实例并关闭服务器。

1. 启动管理 CLI。

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

2. 发出 **shutdown** 命令。

```
shutdown
```



注意

在受管域中运行时，您必须通过将 **--host** 参数与 **shutdown** 命令搭配使用来指定要关闭的主机名。

2.3. 在仅限管理员模式中运行 JBOSS EAP

JBoss EAP *能够以仅限管理员模式启动*。这使得 JBoss EAP 能够运行和接受管理请求，但不启动其他运行时服务或接受最终用户请求。仅管理模式可用于单机服务器和 受管域。

在仅限管理员模式中运行单机服务器
以管理员模式启动服务器
要在仅管理员模式下启动 JBoss EAP 实例，请在启动 JBoss EAP 实例时使用 **--start-mode=admin-only** runtime 参数。

```
$ EAP_HOME/bin/standalone.sh --start-mode=admin-only
```

检查服务器是否在仅限管理员模式下运行
使用以下命令检查服务器的运行模式：如果服务器以管理员模式运行，则结果为 **ADMIN_ONLY**。

```
:read-attribute(name=running-mode)
{
  "outcome" => "success",
  "result" => "ADMIN_ONLY"
}
```

注意

此外，您可以使用以下命令检查启动 JBoss EAP 的初始运行模式：

```
/core-service=server-environment:read-attribute(name=initial-running-mode)
```

从管理 CLI 以不同模式重新加载
除了停止和启动具有其他运行时交换机的 JBoss EAP 实例外，也可使用管理 CLI 以不同模式重新加载它。

以仅限管理员模式重新载入服务器：

```
reload --start-mode=admin-only
```

以正常模式重新载入服务器：

```
reload --start-mode=normal
```

请注意，如果服务器是以 admin-only 模式启动且没有为 **重新加载** 命令指定 **--start-mode** 参数，那么服务器将以正常模式启动。

在仅限管理员模式中运行受管域
在受管域中，如果域控制器以管理员模式启动，它将不接受来自从属主机控制器的传入连接。

以管理员模式启动主机控制器
传递 **--admin-only** runtime 参数，以仅 admin 模式启动主机控制器。

```
$ EAP_HOME/bin/domain.sh --admin-only
```

检查主机控制器是否在仅限管理员模式下运行
使用以下命令，检查主机控制器的运行模式：如果主机控制器以管理员模式运行，则结果为 **ADMIN_ONLY**。

```
/host=HOST_NAME:read-attribute(name=running-mode)
{
  "outcome" => "success",
  "result" => "ADMIN_ONLY"
}
```

从管理 CLI 以不同模式重新加载
除了停止和启动具有其他运行时交换机的主机控制器外，也可使用管理 CLI 以其他模式重新加载它。

以仅限管理员模式重新载入主机控制器：

```
reload --host=HOST_NAME --admin-only=true
```

以正常模式重新载入主机控制器：

```
reload --host=HOST_NAME --admin-only=false
```

请注意，如果主机控制器是以 admin-only 模式启动且没有为 **重新加载** 命令指定 **--admin-only** 参数，则主机控制器将以正常模式启动。

2.4. 正常暂停和关闭 JBOSS EAP

JBoss EAP 可以被暂停或正常关闭。这允许活动请求正常完成，而不接受任何新请求。超时值指定暂停或关机操作将等待活跃请求完成的时间。服务器暂停时，仍然处理管理请求。

安全关闭在服务器范围内协调，主要关注请求进入服务器的入口点。以下子系统支持安全关闭：

Undertow
: **undertow** 子系统将等待所有请求完成。

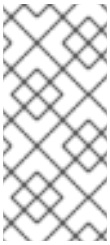
mod_cluster
: **modcluster** 子系统将通知负载平衡器，服务器正在 **PRE_SUSPEND** 阶段暂停。

EJB

ejb3 子系统将等待所有远程 EJB 请求和 MDB 消息发送完成。在 **PRE_SUSPEND** 阶段停止发送到 MDB。EJB 定时器会被暂停，在服务器恢复时将激活错过的计时器。

事务

暂停后，服务器将不会接受新请求，但内点事务和请求可以继续，直到它们完成或超时后过期。这也适用于与 XTS 事务关联的 Web 服务请求。



注意

默认情况下，对于 **ejb** 子系统禁用事务安全关闭。如果您希望服务器在暂停前等待 EJB 相关的事务完成，则必须启用事务正常关闭。例如：

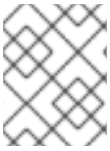
```
/subsystem=ejb3:write-attribute(name=enable-graceful-txn-shutdown,value=true)
```

EE 一致性

服务器将等待所有活动任务完成。将跳过所有排队的作业。目前，由于 EE Concurrency 没有持久性，被跳过的排队作业将丢失。
服务器处于暂停状态时，计划的任务将继续在计划的时间执行，但会抛出 **java.lang.IllegalStateException**。服务器恢复后，计划的任务将继续正常执行，而且在大多数情况下，不需要重新排期任务。

batch

服务器将在超时时间内停止所有运行的作业，并延迟所有计划的作业。



注意

目前正常关闭不会拒绝新的入站 JMS 消息。目前允许继续执行由 in-light 活动调度的 EE 批处理作业和 EE 并发任务；但是，提交的 EE 并发任务会在执行时传递超时窗口。

请求由 **request-controller** 子系统跟踪。如果没有此子系统，则暂停和恢复功能会受到限制，并且服务器不会在暂停或关闭前等待请求完成；但是，如果您不需要此功能，则可移除 **request-controller** 子系统以提高小的性能。

2.4.1. 挂起服务器

JBoss EAP 7 *引入了暂停模式*，可正常暂停服务器操作。这允许所有活动的请求正常完成，但不接受任何新请求。服务器暂停后，可以关机，返回到运行中状态，或保持暂停状态以执行维护。

注意

暂停服务器不会影响到管理接口。

可以使用管理控制台或管理 CLI 暂停和恢复服务器。

检查 Server Suspend 状态
可以通过下列管理 CLI 命令查看服务器暂停状态：生成的值为 **RUNNING**、**PRE_SUSPEND**、**SUSPEND ING 或 SUS PENDED** 之一。

- 检查单机服务器的暂停状态。

```
:read-attribute(name=suspend-state)
```

- 检查受管域中服务器的暂停状态。

```
/host=master/server=server-one:read-attribute(name=suspend-state)
```

suspend
使用以下管理 CLI 命令，通过指定超时值（以秒为单位）暂停服务器，以便服务器等待活动请求完成：默认值为 **0**，它会立即暂停。值为 **-1** 将导致服务器无限期等待所有活动的请求完成。

每个示例最多等待 60 秒以请求完成，然后再暂停。

- 暂停单机服务器.

```
:suspend(suspend-timeout=60)
```

- 暂停受管域中的所有服务器。

```
:suspend-servers(suspend-timeout=60)
```

- 在受管域中暂停一台服务器。

```
/host=master/server-config=server-one:suspend(suspend-timeout=60)
```

- 暂停服务器组中的所有服务器。

```
/server-group=main-server-group:suspend-servers(suspend-timeout=60)
```

- 在主机级别暂停所有服务器。

```
/host=master:suspend-servers(suspend-timeout=60)
```

恢复
restore 命令将服务器返回到正常运行状态，以接受新请求。您可以在主机、服务器、服务器组或域级别启动 命令。例如：

```
:resume
```

在 Suspended State 中启动 Server
您可以启动处于暂停状态的服务器，以便在服务器恢复之前不会接受任何请求。

- 若要以暂停状态启动单机服务器，可在启动 JBoss EAP 实例时使用 **--start-mode=suspend** 运行时参数。

```
$ EAP_HOME/bin/standalone.sh --start-mode=suspend
```

- 若要启动处于暂停状态的受管域服务器，可将 **start-mode=suspend** 参数传递到管理 CLI 命令中的 **启动** 操作。

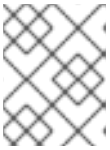
```
/host=HOST_NAME/server-config=SERVER_NAME:start(start-mode=suspend)
```

注意

您还可以将 **start-mode** 参数传递到服务器的 **重新加载** 和 **重新启动** 操作。

- 若要启动受管域服务器组中处于暂停状态的所有服务器，可将 **start-mode=suspend** 参数传递到管理 CLI 命令中的 **start-servers** 操作：

```
/server-group=SERVER_GROUP_NAME:start-servers(start-mode=suspend)
```



注意

您还可以将 **start-mode** 参数传递到 **reload-servers** 和 **restart-servers** 操作，供服务器组使用。

2.4.2. 使用管理 CLI 正常关闭服务器

如果在停止服务器时指定了适当的超时值，则会安全地关闭服务器。发出 命令后，服务器将被暂停，并将等待所有请求的指定超时完成，然后再关机。

使用以下管理 CLI 命令，安全地关闭服务器：指定服务器的超时值（以秒为单位），以便服务器等待活跃请求完成。默认值为 **0**，它将立即关闭服务器。值为 **-1** 将导致服务器无限期等待所有活动的请求完成，然后再关机。

每个示例最多等待 60 秒以请求完成，然后关机。

- 正常关闭单机服务器。

```
shutdown --suspend-timeout=60
```

- 正常停止受管域中的所有服务器。

```
:stop-servers(suspend-timeout=60)
```

- 正常停止受管域中的一台服务器。

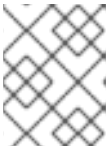
```
/host=master/server-config=server-one:stop(suspend-timeout=60)
```

- 正常停止服务器组中的所有服务器。

```
/server-group=main-server-group:stop-servers(suspend-timeout=60)
```

- 关闭主机控制器及其管理的所有服务器。

```
/host=master:shutdown(suspend-timeout=60)
```



注意

suspend-timeout 属性仅应用到主机控制器管理的服务器，而不应用到主机控制器本身。

2.4.3. 使用 OS 信号正常关闭服务器

可以通过发送 OS **TERM 信号** （如 **kill -15 *PID***）来安全地关闭服务器。默认情况下，这个值与管理 CLI 的 **shutdown --suspend-timeout=0** 命令相同，从而导致任何当前处理请求立即终止。超时可以通过 **org.wildfly.sigterm.suspend.timeout** 系统属性进行配置，这表示服务器关闭之前等待请求完成的最大秒数。值 **-1** 表示服务器将无限期等待。

重要

在受管域中，不应使用 OS 信号来关闭服务器。取而代之，应当使用管理 CLI 和管理主机控制器来关闭服务器。

如果使用 OS 信号正常关机，如果 JVM 配置为禁用信号处理，例如 **-Xrs** java 参数已传递到 JVM 选项，或者发送的信号不是进程可以响应的信号，例如发送 **KILL 信号** 时，则不会起作用。

2.5. 启动和停止 JBOSS EAP（RPM 安装）

与 ZIP 或安装程序安装相比，启动和停止 JBoss EAP 对于 RPM 安装不同。

2.5.1. 启动 JBoss EAP（RPM 安装）

启动 JBoss EAP RPM 安装的命令取决于您要启动的操作模式、单机服务器或受管域，以及您正在运行的红帽企业 Linux 版本。

启动 JBoss EAP 作为单机服务器（RPM 安装）

- Red Hat Enterprise Linux 6：

```
$ service eap7-standalone start
```

- Red Hat Enterprise Linux 7 及更新的版本：

```
$ systemctl start eap7-standalone.service
```

默认情况下，此操作将使用 **standalone.xml** 配置文件启动 JBoss EAP。您可以通过在 RPM 服务配置文件中设置属性，使用不同的单机服务器配置文件启动 JBoss EAP。如需更多信息，请参阅以下 Configure RPM Service Properties 部分。

在受管域中启动 JBoss EAP（RPM 安装）

- Red Hat Enterprise Linux 6：

```
$ service eap7-domain start
```

- Red Hat Enterprise Linux 7 及更新的版本：

```
$ systemctl start eap7-domain.service
```

默认情况下，此操作将使用 **host.xml** 配置文件启动 JBoss EAP。您可以通过在 RPM 服务配置文件中设置属性，使用不同的受管域配置文件启动 JBoss EAP。如需更多信息，请参阅以下 Configure RPM Service Properties 部分。

配置 RPM 服务属性
本节介绍如何为您的 JBoss EAP 安装配置 RPM 服务属性和其他启动选项。请注意，建议在进行修改前备份您的配置文件。

有关 RPM 安装的所有可用启动选项列表，请参阅 RPM 服务配置属性部分。

重要

对于 Red Hat Enterprise Linux 7 及更新的版本，使用 **systemd** 加载 RPM 服务配置文件，因此不会扩展变量表达式。

- 指定服务器配置文件。
  启动单机服务器时，默认情况下将使用 **standalone.xml** 文件。在受管域中运行时，默认情况下将使用 **host.xml** 文件。您可以通过在相应的 RPM 配置文件中设置 **WILDFLY_SERVER_CONFIG** 属性来使用其他配置文件启动 JBoss EAP， 如 **eap7-standalone.conf**。

```
WILDFLY_SERVER_CONFIG=standalone-full.xml
```

- 绑定到特定的 IP 地址。
  默认情况下，JBoss EAP RPM 安装绑定到 **0.0.0.0**。您可以通过在适当的 RPM 配置文件中 设置 **WILDFLY_BIND** 属性，将 JBoss EAP 绑定到特定的 IP 地址，如 **eap7-standalone.conf**。

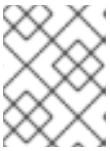
```
WILDFLY_BIND=192.168.0.1
```

注意

如果要将管理接口绑定到特定的 IP 地址，可以在 JBoss EAP 启动配置文件中进行配置，如下一个示例所示。

- 设置 JVM 选项或 Java 属性.
  您可以通过编辑 启动配置文件，指定要传递给 JBoss EAP 启动脚本的 JVM 选项或 Java 属性。此文件是单机服务器的 ***EAP_HOME*/bin/standalone.conf**，或用于受管域的 ***EAP_HOME*/bin/domain.conf**。以下示例配置了堆大小，并将 JBoss EAP 管理接口绑定到 IP 地址。

```
JAVA_OPTS="$JAVA_OPTS -Xms2048m -Xmx2048m"
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address.management=192.168.0.1"
```



注意

如果需要，必须使用 **WILDFLY_BIND** 属性来配置 JBoss EAP 绑定地址，而不在此处使用 **jboss.bind.address** 标准属性。

注意

如果在 RPM 服务配置文件中具有相同的名称（如 **/etc/sysconfig/eap7-standalone** ），并且在 JBoss EAP 启动配置文件中，如 ***EAP_HOME*/bin/standalone.conf**，则优先的值就是 JBoss EAP 启动配置文件中的值。个这样的属性是 **JAVA_HOME**。

2.5.2. 停止 JBoss EAP（RPM 安装）

用于停止 JBoss EAP RPM 安装的命令取决于启动的运营模式、单机服务器或受管域，以及您正在运行的红帽企业 Linux 版本。

停止 JBoss EAP 作为单机服务器（RPM 安装）

- Red Hat Enterprise Linux 6：

```
$ service eap7-standalone stop
```

- Red Hat Enterprise Linux 7 及更新的版本：

```
$ systemctl stop eap7-standalone.service
```

在受管域中停止 JBoss EAP（RPM 安装）

- Red Hat Enterprise Linux 6：

```
$ service eap7-domain stop
```

- Red Hat Enterprise Linux 7 及更新的版本：

```
$ systemctl stop eap7-domain.service
```

有关 RPM 安装的所有可用启动选项列表，请参阅 RPM 服务配置文件 部分。

2.6. POWERSHELL 脚本（WINDOWS 服务器）

重要

PowerShell 脚本的集合仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

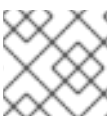
JBoss EAP 包含与大多数 JBoss EAP 管理脚本等效的 PowerShell 脚本。这包括在 Microsoft Windows Server 上启动 JBoss EAP 的 PowerShell 脚本。

JBoss EAP PowerShell 脚本设计为能与 PowerShell 版本 2 一起使用，并在经过测试的 Windows Server 版本上运行。

JBoss EAP PowerShell 脚本位于 ***EAP_HOME*\bin** 中，其使用方式与 JBoss EAP 批处理脚本基本相同。

例如，要启动带有 **standalone-full.xml** 配置文件的独立 JBoss EAP 服务器，请使用以下 PowerShell 命令：

```
.\standalone.ps1 "-c=standalone-full.xml"
```



注意

JBoss EAP PowerShell 脚本的参数必须放在引号内。

第 3 章 JBOSS EAP 管理

JBoss EAP 使用简化的配置，每个单机服务器或受管域有一个配置文件。单机服务器的默认配置存储在 ***EAP_HOME*/standalone/configuration/standalone.xml** 文件中，受管域的默认配置则存储在 ***EAP_HOME*/domain/configuration/domain.xml** 文件中。此外，主机控制器的默认配置存储在 ***EAP_HOME*/domain/configuration/host.xml** 文件中。

可以使用命令行管理 CLI、基于 Web 的管理控制台、Java API 或 HTTP API 来配置 JBoss EAP。使用这些管理接口所做的更改将自动保留，并且管理 API 将覆盖 XML 配置文件。管理 CLI 和管理控制台是首选的方法，不建议手动编辑 XML 配置文件。

3.1. 关于子系统、扩展和配置文件

在不同的子系统中配置 JBoss EAP 功能的不同方面。例如，应用和服务器日志记录在 **logging** 子系统中配置。

*扩展是扩展服务器核心功能的模块。*扩展会根据部署需要加载，并在不再需要时卸载。有关如何添加 和删除扩展的信息，请参阅 JBoss EAP *管理 CLI 指南*。

*子系统提供特定扩展的配置选项。*如需有关可用子系统的更多信息，请参阅 JBoss EAP 子系统概述。

*子系统配置的集合构成一个配置文件，*它配置为满足服务器的需求。单机服务器具有单个未命名的配置文件。受管域可以定义许多配置文件，供域中的服务器组使用。

使用管理控制台或管理 CLI

管理控制台和管理 CLI 均有效，受到支持的方式可以更新 JBoss EAP 实例的配置。在两者间进行选择是一个优先的问题。更喜欢使用基于 Web 的图形界面的用户应使用管理控制台。更喜欢命令行界面的人员应使用管理 CLI。

3.2. 管理用户

默认 JBoss EAP 配置提供本地身份验证，让用户可以访问本地主机上的管理 CLI，而无需身份验证。

但是，如果您要远程访问管理 CLI，则必须添加管理用户，或使用管理控制台，即使流量来自本地主机上也被视为远程访问。如果在添加管理用户之前尝试访问管理控制台，您会收到错误消息。

如果使用图形安装程序安装 JBoss EAP，则在安装过程中创建管理用户。

本指南介绍了使用 **add-user** 脚本对 JBoss EAP 进行简单的用户管理，此脚本可用于将新用户添加到用于开箱即用身份验证的属性文件中。

有关更高级的身份验证和授权选项，如 LDAP 或基于角色的访问控制(RBAC)，请参见 JBoss EAP 安全架构的核心管理身份验证部分 。

3.2.1. 添加管理用户

1. *运行 **add-user** 实用程序脚本并按照提示进行操作。*

```
$ EAP_HOME/bin/add-user.sh
```

注意

*对于 Windows Server，请使用 **EAP_HOME\bin\add-user.bat** 脚本。*

2. 按 **ENTER**，选择默认选项 **a** 以添加管理用户。
   此用户将添加到 ManagementRealm 中，并将获得使用管理控制台或管理控制台执行管理操作的授权。另一个选择 **b** 将用户添加到 ApplicationRealm 中，该应用程序用于应用程序，不提供任何特定权限。

3. 输入所需的用户名和密码。系统将提示您确认密码。

注意

用户名只能以任何数字和顺序包含以下字符：

- 字母数字字符（a-z、A-Z、0-9）
- 短划线(-)、句点(.)、逗号(@)
- 反斜杠(\)
- Equals (=)

默认情况下，JBoss EAP 允许弱密码，但会发出警告。

有关更改此默认行为的详情，请参阅设置附加用户实用程序密码限制。

4. 输入用户所属组的逗号分隔列表。如果您不希望用户属于任何组，请按 **ENTER** 将它留空。

5. 检查信息并输入 **yes** 进行确认。

6. 确定此用户是否代表远程 JBoss EAP 服务器实例。对于基本管理用户，请输入 **no**.
   可能需要添加到 ManagementRealm 的一种用户是代表另一个 JBoss EAP 实例的用户，它必须能够进行身份验证以作为群集成员加入。如果出现这种情况，则在此提示中回答 **yes**，系统会为您提供一个表示用户密码的散列化机密值，该值需要添加到其他配置文件中。

也可以通过向 **add-user** 脚本传递参数，以非交互方式创建用户。共享系统上不建议使用此方法，因为密码将在日志和历史记录文件中可见。如需更多信息，请参阅非活动运行 Add-User 实用程序。

3.2.2. 主动运行 Add-User 实用程序

您可以通过在命令行中传递参数，以非交互方式运行 **add-user** 脚本。必须至少提供用户名和密码。

警告

共享系统上不建议使用此方法，因为密码将在日志和历史记录文件中可见。

创建一个属于多个组的用户
以下命令添加了一个管理用户, **mgmtuser1**，它带有 **guest and mgmtgroup** 组。

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser1' -p 'password1!' -g 'guest,mgmtgroup'
```

指定替代属性文件
默认情况下，使用 **add-user** 脚本创建的用户和组信息存储在服务器配置目录中的属性文件中。

用户信息存储在以下属性文件中：

- **EAP_HOME/standalone/configuration/mgmt-users.properties**
- **EAP_HOME/domain/configuration/mgmt-users.properties**

组信息存储在以下属性文件中：

- **EAP_HOME/standalone/configuration/mgmt-groups.properties**
- **EAP_HOME/domain/configuration/mgmt-groups.properties**

这些默认目录和属性文件名可以被覆盖。以下命令添加新用户，为用户属性文件指定不同的名称和位置。

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser2' -p 'password1!' -sc '/path/to/standaloneconfig/' -dc '/path/to/domainconfig/' -up 'newname.properties'
```

新用户已添加到位于 **/path/to /standaloneconfig/newname.properties** 和 **/path/to /domainconfig/newname.properties** 的用户属性文件中。请注意，这些文件必须已经存在，否则您将看到错误。

有关所有可用 **add-user** 参数及其目的的完整列表，请使用 **--help** 参数或参阅 Add-User 实用程序参数 部分。

3.2.3. 附加用户实用程序密码限制

可以使用 **EAP_HOME/bin/add -user.properties** 文件配置 **add -user** 实用程序脚本的密码限制。

重要

add-user.properties 文件是一个未受保护的纯文本文件，必须加以保护，以避免对其内容进行不必要的访问。

要避免设置不需要的密码，请检查键盘的系统键映射是否正确。默认系统键映射为 **en-qwerty**。如果更改此默认设置并创建新密码，您必须检查密码是否满足类 **SimplePasswordStrengthChecker** 中的条件。

默认情况下，JBoss EAP 允许弱密码，但会发出警告。要拒绝不满足指定最低要求的密码，请将 **password.restriction** 属性设置为 **REJECT**。

下表描述了可在 **EAP_HOME/bin/add-user.properties** 文件中配置的额外密码要求设置：

表 3.1. 额外密码要求设置

属性	描述
minLength	密码的最少字符数.例如，password **.restriction.minLength=8** 将密码限制为最少 8 个字符。

属性	描述
strength	设置密码必须满足的阈值才有效。有效的阈值条目包括： * **VERY_WEAK** * **弱** * **中等** * **中** * **强** * **VERY_STRONG** * **例外** 默认值为 **MODERATE**。定义了阈值，并且定义了类 **SimplePasswordStrengthChecker** 中指定的默认值。 注意：如果您没有指定阈值，MODE **RATE** 将成为默认值。这个值在类 **SimplePasswordStrengthChecker 中指定。**
minAlpha	为密码设置的最少字母字符数。例如，**password.restriction.minAlpha=1** 将密码限制为至少包含一个字母字符。
minDigit	为密码设置的最小数字字符数.例如，**password.restriction.minDigit=1** 将密码限制为至少包含一个数字字符。
minSymbol	为密码设置的最少符号数。例如，**password.restriction.minSymbol=1** 将密码限制为至少包含一个符号。
forbiddenValue	限制用户设置易于确定的密码，如 *root*。例如，password **.restriction.forbidden=root、admin、administrators 限制** 设置 *root*、admin *或 admin* 作为密码。
mustNotMatchUsername	限制用户将其用户名设置为密码。例如，**password.restriction.mustNotMatchUsername=TRUE** 限制用户将其用户名设置为密码。如果设置为 **false**，则忽略此规则。

其它资源

请参阅红帽客户门户网站中的配置基本系统设置指南。

3.2.4. 更新管理用户

*您可以使用 **add-user** 实用程序脚本在提示时输入用户名来更新现有管理用户的设置。*

```
$ EAP_HOME/bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
```

```
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : test-user
User 'test-user' already exists and is enabled, would you like to...
 a) Update the existing user password and roles
 b) Disable the existing user
 c) Type a new username
(a):
```

当您输入已存在的用户名时，会显示几个选项：

- 键入 **a** 以更新现有用户的密码。
- 键入 **b** 以禁用现有用户。
- 按 **c** 键输入新用户名。

警告

以非交互方式使用 **add-user** 脚本更新用户时，用户会自动更新，无需确认提示。

3.3. 优化 JBOSS EAP 服务器配置

安装 JBoss EAP 服务器并且创建了管理用户后，红帽建议您优化服务器配置。

确保查看《性能调优指南》中的信息，了解如何优化服务器配置，以避免在生产环境中部署应用程序时出现常见问题。常见的优化包括设置 ulimits、启用垃圾回收、创建 Java 堆转储，以及调整线程池大小。

最好为产品发布应用任何现有的补丁。EAP 的每个补丁都包含大量漏洞修复。如需更多信息，请参阅 JBoss EAP 的补丁和升级指南中的 JBoss EAP 补丁和升级指南。

3.4. 管理接口

3.4.1. 管理 CLI

管理命令行界面(CLI)是 JBoss EAP 的命令行管理工具。

使用管理 CLI 启动和停止服务器、部署和取消部署应用、配置系统设置，以及执行其他管理任务。操作可以在批处理模式下执行，允许以组形式运行多个任务。

许多常见的终端命令可用，如 **ls**、**cd** 和 **pwd**。管理 CLI 也支持 tab 自动完成功能。

有关使用管理 CLI 的详细信息，包括命令和操作、语法以及批处理模式下运行，请参阅 JBoss EAP 管理 CLI 指南。

启动管理 **CLI**

```
$ EAP_HOME/bin/jboss-cli.sh
```

注意

对于 Windows Server，请使用 **EAP_HOME\bin\jboss-cli.bat** 脚本。

连接到正在运行的服务器

```
connect
```

或者，您可以使用 **EAP_HOME/bin/jboss-cli.sh --connect** 命令启动管理 CLI 并在一个步骤中进行连接。

显示帮助
使用以下命令获取一般帮助：

```
help
```

在命令中使用 **--help** 标志，以接收有关 使用该特定命令的说明。例如，若要使用 **deploy** 接收关于 的信息，请执行以下命令：

```
deploy --help
```

退出管理 CLI

```
quit
```

查看系统设置
以下命令使用 **read-attribute** 操作来显示是否启用了示例数据源：

```
/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
{
    "outcome" => "success",
    "result" => true
}
```

在受管域中运行时，您必须通过 **/profile=PROFILE_NAME** 命令指定要更新的配置集。

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
```

更新系统设置
以下命令使用 **write-attribute** 操作来禁用示例数据源。

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled,value=false)
```

启动服务器
管理 CLI 也可用于在受管域中运行时启动和停止服务器。

```
/host=HOST_NAME/server-config=server-one:start
```

3.4.2. 管理控制台

管理控制台是用于 JBoss EAP 的基于 Web 的管理工具。

使用管理控制台启动和停止服务器、部署和取消部署应用、调优系统设置，以及对服务器配置进行持续修改。管理控制台还具备执行管理任务的功能，当当前用户执行的任何更改都要求重新启动或重新加载服务器实例时，实时通知功能。

在受管域中，可以从域控制器的管理控制台集中管理同一域中的服务器实例和服务器组。

对于使用默认管理端口在本地主机上运行的 JBoss EAP 实例，可通过位于 http://localhost:9990/console/index.html 的 Web 浏览器访问管理控制台。您将需要使用具有访问管理控制台权限的用户进行身份验证。

管理控制台提供下列选项卡，用于浏览和管理 JBoss EAP 单机服务器或受管域。

Home

了解如何完成几个常见配置和管理任务。参加导览，熟悉 JBoss EAP 管理控制台。

部署

添加、移除和启用部署。在受管域中，将部署分配到服务器组。

Configuration

配置可用的子系统，提供 Web 服务、消息传递或高可用性等功能。在受管域中，管理包含不同子系统配置的配置文件。

Runtime

查看运行时信息，如服务器状态、JVM 使用量和服务器日志。在受管域中，管理您的主机、服务器组和服务器。

patching

将补丁应用到您的 JBoss EAP 实例。

访问控制

在使用基于角色的访问控制时，将角色分配给用户和组。

3.4.2.1. 在管理控制台中更新属性

导航到管理控制台中相应部分以用于要修改的资源后，只要您拥有适当的权限，即可编辑其属性。

1. 单击**编辑**链接。
2. 进行所需的更改。
   必填字段标有星号(*)。您可以通过单击 **Help** 链接来查看属性描述。

注意

根据属性类型，输入字段可以是文本字段、ON/OFF 字段，或者下拉字段。在某些文本字段中，正如您键入的，配置的其他位置中的值可能显示为建议。

3. 单击 **Save** 以保存更改。
4. 如有必要，重新加载服务器以使更改生效。
   保存需要重新加载才能生效的更改时会出现弹出窗口。若要重新加载单机服务器，可在弹出窗口中单击 **Reload** 链接。若要重新加载受管域中的服务器，可单击 **Topology** 链接，选择相应的服务器，然后单击 **Reload** 下拉列表。

要查看您最近执行的配置操作的历史记录，请单击管理控制台右上角的通知图标。

3.4.2.2. 启用/禁用管理控制台

您可以通过设置 **/core-service=management/management -interface=http-interface=http-interface=http-interface=http-interface** 属性来启用或禁用管理控制台。对于域模式中的 master 主机，请使用 **/host=master/core-service=management/management-interface=http-interface**。

例如，要启用：

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=true)
```

例如，要禁用：

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=false)
```

3.4.2.3. 更改管理控制台的语言

默认情况下，管理控制台的语言设置为英文。您可以选择使用以下语言之一：

- 德语(de)
- 简体中文(zh-Hans)
- 巴西葡萄牙语(pt-BR)
- 法语(fr)
- 西班牙语(es)
- 日语(Jab)

更改管理控制台的语言

1. 登录管理控制台。
2. 单击管理控制台右下角的 **Settings** 链接。
3. 从 **Locale** 选择框中选择所需的语言。
4. 选择 **Save**。确认框通知您需要重新加载应用。
5. 单击 **Yes**。系统会自动刷新 Web 浏览器以使用所选区域设置。

3.4.2.4. 自定义管理控制台标题

您可以自定义管理控制台标题，以便快速识别每个 JBoss EAP 实例。

自定义管理控制台标题：

1. 登录管理控制台。
2. 单击管理控制台右下角的 **Settings**。
3. 在 **Settings** 窗口中，修改标题字段中的标题。

4. 点 **Save**。
   确认框通知您必须重新加载管理控制台。

5. 单击 **Yes**。
   系统会自动刷新 Web 浏览器，新的标题会显示在标签页标头中。

3.5. 管理 API

3.5.1. HTTP API

HTTP API 端点是管理客户端的入口点，它依赖于 HTTP 协议与 JBoss EAP 管理层集成。

HTTP API 由 JBoss EAP 管理控制台使用，但也为其他客户端提供集成功能。默认情况下，HTTP API 可通过 **http://HOST_NAME: 9990/management** 访问。此 URL 将显示公开给 API 的原始属性和值。

读取资源
虽然您可以使用 HTTP **POST** 方法读取、写入或执行其他操作，但您可以使用 **GET** 请求 来执行一些读取操作。HTTP **GET** 方法使用以下 URL 格式：

```
http://HOST_NAME:9990/management/PATH_TO_RESOURCE?
operation=OPERATION&PARAMETER=VALUE
```

务必将所有可替换值替换为适合您的请求的值。以下值是 OPERATION 可替换 值 的可用选项：

值	描述
attribute	执行 **读-attribute** 操作。
operation-description	执行 **read-operation-description** 操作。
operation-names	执行 **read-operation-names** 操作。
resource	执行 **read-resource** 操作。
resource-description	执行 **read-resource-description** 操作。
快照	执行 **list-snapshots** 操作。

以下示例 URL 演示了如何使用 HTTP API 执行读取操作。

示例：阅读资源的所有属性和值

```
http://HOST_NAME:9990/management/subsystem/undertow/server/default-server/http-
listener/default
```

这将显示 默认 HTTP 侦听器的所有属性及其值。

注意

默认操作为 **read-resource**。

示例：阅读资源属性的值

```
http://HOST_NAME:9990/management/subsystem/datasources/data-source/ExampleDS?
operation=attribute&name=enabled
```

这会读取 **ExampleDS** 数据源的 **enabled** 属性的值。

更新资源
您可以使用 HTTP **POST** 方法更新配置值或使用 HTTP API 执行其他操作。您必须为这些操作提供身份验证。

以下示例演示了如何使用 HTTP API 更新资源。

示例：更新资源的属性值

```
$ curl --digest http://HOST_NAME:9990/management --header "Content-Type: application/json" -u
USERNAME:PASSWORD -d '{"operation":"write-attribute", "address":
["subsystem","datasources","data-source","ExampleDS"], "name":"enabled", "value":"false",
"json.pretty":"1"}'
```

这会将 **ExampleDS** 数据源的 **enabled** 属性值更新为 **false**。

示例：把操作到服务器

```
$ curl --digest http://localhost:9990/management --header "Content-Type: application/json" -u
USERNAME:PASSWORD -d '{"operation":"reload"}'
```

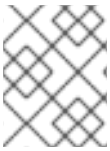
这将重新加载服务器。

如需有关如何使用 HTTP API 部署应用程序到 JBoss EAP 的信息，请参阅使用 HTTP API 部署应用 。

3.5.2. 原生 API

原生 API 端点是管理客户端的入口点，它依赖于原生协议与 JBoss EAP 管理层集成。原生 API 由 JBoss EAP 管理 CLI 使用，但也为其他客户端提供集成功能。

以下 Java 代码演示了如何使用原生 API 从 Java 代码执行管理操作的示例。



注意

您必须将 **EAP_HOME/bin/client/jboss-cli-client.jar** 文件中所需的 JBoss EAP 库添加到您的类路径中。

示例：使用原生 API 读取资源

```
// Create the management client
ModelControllerClient client = ModelControllerClient.Factory.create("localhost", 9990);

// Create the operation request
ModelNode op = new ModelNode();

// Set the operation
op.get("operation").set("read-resource");
```

```
// Set the address
ModelNode address = op.get("address");
address.add("subsystem", "undertow");
address.add("server", "default-server");
address.add("http-listener", "default");

// Execute the operation and manipulate the result
ModelNode returnVal = client.execute(op);
System.out.println("Outcome: " + returnVal.get("outcome").toString());
System.out.println("Result: " + returnVal.get("result").toString());

// Close the client
client.close();
```

3.6. 配置数据

3.6.1. 独立服务器配置文件

独立配置文件位于 **EAP_HOME/standalone/configuration/** 目录中。每个预定义配置文件（默认为 ha、full、full-ha、负载平衡器）均存在单独的 文件。

表 3.2. 独立配置文件

配置文件	用途
standalone.xml	此独立配置文件是启动单机服务器时使用的默认配置。它包含有关服务器的所有信息，包括子系统、网络、部署、套接字绑定和其他可配置的详细信息。它不提供消息传递或高可用性所需的子系统。
standalone-ha.xml	此单机配置文件包含所有默认子系统，并添加 **modcluster** 和 **jgroups** 子系统，以实现高可用性。它不提供消息传递所需的子系统。
standalone-full.xml	此单机配置文件包含所有默认子系统，并添加 **messaging-activemq** 和 **iiop-openjdk** 子系统。它不提供高可用性所需的子系统。
standalone-full-ha.xml	此独立配置文件包括对每一种可能子系统的支持，包括消息传递和高可用性方面的支持。
standalone-load-balancer.xml	此单机配置文件包含使用内置 mod_cluster 前端负载平衡器对其他 JBoss EAP 实例进行负载平衡所需的最小子系统。

默认情况下，将 JBoss EAP 作为单机服务器启动使用 **standalone.xml** 文件。若要使用其他配置启动 JBoss EAP，可使用 **--server-config** 参数：例如，

```
$ EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml
```

3.6.2. 受管域配置文件

受管域配置文件位于 **EAP_HOME/domain/configuration/** 目录中。

表 3.3. 受管域配置文件

配置文件	用途
domain.xml	这是受管域的主配置文件。只有域 master 会读取此文件。*此文件包含所有配置文件的配置（默认为 ha、full、full-ha、load-balancer）。*
host.xml	此文件包含特定于受管域中物理主机的配置详细信息，如网络接口、套接字绑定、主机名称和其他特定于主机的详细信息。**host.xml** 文件包含 **host-master.xml** 和 **host-slave.xml** 的所有功能，如下所述。
host-master.xml	此文件仅包含将服务器作为主域控制器运行所需的配置详细信息。
host-slave.xml	此文件仅包含作为受管域主机控制器运行服务器所需的配置详细信息。

*默认情况下，在受管域中启动 JBoss EAP 将使用 **host.xml** 文件。若要使用其他配置启动 JBoss EAP，可使用 **--host-config** 参数：例如，*

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml
```

3.6.3. 备份配置数据

为了稍后恢复 JBoss EAP 服务器配置，应备份以下位置中的项目：

- ***EAP_HOME/standalone/configuration/***
  - *备份整个目录，以保存单机服务器的用户数据、服务器配置和日志记录设置。*
- ***EAP_HOME/domain/configuration/***
  - *备份整个目录，以保存用户和配置文件数据、域和主机配置，以及受管域的日志记录设置。*
- ***EAP_HOME/modules/***
  - *备份任何自定义模块。*
- ***EAP_HOME/welcome-content/***
  - *备份任何自定义欢迎内容。*
- ***EAP_HOME/bin/***
  - *备份任何自定义脚本或启动配置文件。*

3.6.4. 配置文件快照

为了协助服务器维护和管理，JBoss EAP 在启动时创建原始配置文件的时间戳版本。管理操作的任何其他配置更改将导致自动备份原始文件，保留实例的工作副本供引用和回滚。此外，还可以生成配置快照，它们是当前服务器配置的即时副本。这些快照可由管理员保存和加载。

*以下示例使用 **standalone.xml** 文件，但同一进程适用于 **domain.xml** 和 **host.xml** 文件。*

拍摄快照

使用管理 CLI 为当前配置生成快照。

```
:take-snapshot
{
    "outcome" => "success",
    "result" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20151022-
133109702standalone.xml"
}
```

列出快照
使用管理 CLI 列出已执行的所有快照。

```
:list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
        "names" => [
            "20151022-133109702standalone.xml",
            "20151022-132715958standalone.xml"
        ]
    }
}
```

删除快照
使用管理 CLI 删除快照。

```
:delete-snapshot(name=20151022-133109702standalone.xml)
```

使用快照启动服务器
可以使用快照或自动保存的配置启动服务器。

1. 导航到 **EAP_HOME/standalone/configuration/standalone_xml_history** 目录，再识别要加载的快照或保存的配置文件。
2. 启动服务器并指向所选配置文件。传递与配置目录相关的文件路径 **EAP_HOME/standalone/configuration/**。

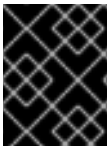
```
$ EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/snapshot/20151022-133109702standalone.xml
```

注意

在受管域中运行时，请使用 **--host-config** 参数来指定配置文件。

3.6.5. 查看配置更改

JBoss EAP 7 提供了跟踪对运行系统进行的配置更改的功能。这样，管理员可以查看其他授权用户所做的配置更改历史记录。



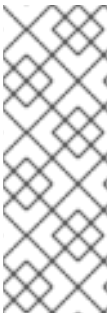
重要

更改存储在内存中，不会在服务器重新启动之间保留。此功能不能取代管理审计日志记录。

您可以从管理 CLI 或管理控制台启用跟踪和查看配置更改。

通过管理 **CLI** 跟踪和查看配置更改
若要启用跟踪配置更改，可使用以下管理 CLI 命令：您可以使用 **max-history** 属性指定要存储的条目数量。

```
/subsystem=core-management/service=configuration-changes:add(max-history=20)
```



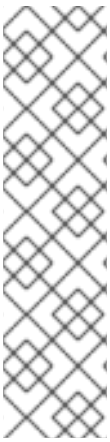
注意

在受管域中，配置更改在主机级别上跟踪，以进行主机和服务器相关的修改。为主机控制器启用配置更改可为其所有受管服务器启用它。您可以使用以下命令跟踪每个主机的配置更改：

```
/host=HOST_NAME/subsystem=core-management/service=configuration-changes:add(max-history=20)
```

要查看最近的配置更改列表，请使用以下管理 CLI 命令：

```
/subsystem=core-management/service=configuration-changes:list-changes
```



注意

在受管域中，您可以使用以下命令列出主机的配置更改：

```
/host=HOST_NAME/subsystem=core-management/service=configuration-changes:list-changes
```

您可以使用以下命令列出影响特定服务器的配置更改：

```
/host=HOST_NAME/server=SERVER_NAME/subsystem=core-management/service=configuration-changes:list-changes
```

这会列出所做的每个配置更改，包括日期、来源、结果和操作详情。例如，以下 **list-changes** 命令的输出显示配置更改，最近显示的第一个显示为第一个。

```
{
    "outcome" => "success",
    "result" => [
        {
            "operation-date" => "2016-02-12T18:37:00.354Z",
            "access-mechanism" => "NATIVE",
            "remote-address" => "127.0.0.1/127.0.0.1",
            "outcome" => "success",
            "operations" => [{
                "address" => [],
```

```
            "operation" => "reload",
            "operation-headers" => {
                "caller-type" => "user",
                "access-mechanism" => "NATIVE"
            }
        }]
    },
    {
        "operation-date" => "2016-02-12T18:34:16.859Z",
        "access-mechanism" => "NATIVE",
        "remote-address" => "127.0.0.1/127.0.0.1",
        "outcome" => "success",
        "operations" => [{
            "address" => [
                ("subsystem" => "datasources"),
                ("data-source" => "ExampleDS")
            ],
            "operation" => "write-attribute",
            "name" => "enabled",
            "value" => false,
            "operation-headers" => {
                "caller-type" => "user",
                "access-mechanism" => "NATIVE"
            }
        }]
    },
    {
        "operation-date" => "2016-02-12T18:24:11.670Z",
        "access-mechanism" => "HTTP",
        "remote-address" => "127.0.0.1/127.0.0.1",
        "outcome" => "success",
        "operations" => [{
            "operation" => "remove",
            "address" => [
                ("subsystem" => "messaging-activemq"),
                ("server" => "default"),
                ("jms-queue" => "ExpiryQueue")
            ],
            "operation-headers" => {"access-mechanism" => "HTTP"}
        }]
    }
  ]
}
```

这个示例列出了影响配置的三个操作详情：

- *从管理 CLI 重新加载服务器：*
- *从管理 CLI 禁用 **ExampleDS** 数据源.*
- *从管理控制台 移除 **ExpiryQueue** 队列。*

从管理控制台跟踪和查看配置更改
*若要启用从管理控制台跟踪配置更改，可选择到 **Runtime** 选项卡，前往服务器或主机以跟踪 的更改，并从下拉菜单中选择 **Configuration Changes**。单击 **Enable Configuration Changes**，并提供最大历史记录值。*

然后，此页面上的表列出进行的每个配置更改，包括日期、来源、结果和操作详情。

3.6.6. 属性替换

JBoss EAP 允许您使用表达式来定义可在配置中替换字面值的可替换属性。表达式使用 **${PARAMETER: DEFAULT_VALUE}** 格式。如果设置了指定参数，则将使用参数的值。否则，将使用提供的默认值。

解析表达式支持的源包括系统属性、环境变量和密码库。对于部署，源可以是部署存档中的 **META-INF/jboss.properties** 文件中的属性。对于支持子部署的部署类型，如果属性文件位于外部部署中，则解析范围仅限于所有子部署，如 EAR。如果属性文件在子部署中，则解析的范围仅限于该子部署。

以下 **standalone.xml** 配置文件的示例将 公共 接口的 **inet-address** 设为 **127.0.0.1**，除非设置了 **jboss.bind.address** 参数。

```
<interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
</interface>
```

将 EAP 启动为单机服务器时，可以使用以下命令设置 **jboss.bind.address** 参数：

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

嵌套表达式
表达式可以嵌套，允许更高级地使用表达式来代替固定值。嵌套表达式的格式类似于普通表达式的格式，但一个表达式被嵌入在另一个表达式中，例如：

```
${SYSTEM_VALUE_1${SYSTEM_VALUE_2}}
```

嵌套表达式是递归评估的，因此首先评估内嵌表达式，然后评估外部表达式。表达式也可能是递归的，其中表达式解析为其他表达式，然后解析。允许表达式的任何位置都允许嵌套表达式，但管理 CLI 命令除外。

例如，如果数据源定义中使用的密码被屏蔽，则可以使用嵌套表达式。数据源的配置可能包含以下行：

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

ds_ExampleDS 的值可以替换为使用嵌套表达式的系统属性(**datasource_name**)。数据源的配置可以改为有以下行：

```
<password>${VAULT::${datasource_name}::password::1}</password>
```

JBoss EAP 首先评估表达式 **${datasource_name}**，然后将其输入到更大的表达式并评估生成的表达式。此配置的优点在于数据源的名称是从固定配置中提取的。

基于描述符的特征替换
应用程序配置（如数据源连接参数）通常会因开发、测试和生产环境而异。构建系统脚本有时可以容纳这种差异，因为 Jakarta EE 规范不包含将这些配置外部化的方法。借助 JBoss EAP，您可以使用基于描述符的属性替换在外部管理配置。

基于描述符的属性替换基于描述符的属性，允许您从应用和构建链中删除对环境相关的假设。特定环境的配置可以在部署描述符中指定，而不是注释或构建系统脚本。您可以在文件中提供配置，或者作为参数在命令行中提供。

ee 子系统中有几个标记控制是否应用属性替换。

JBoss 特定的描述符替换由 **jboss-descriptor-property-replacement** 标志控制，默认情况下是 启用的。启用后，可以在以下部署描述符中替换属性：

- ***jboss-ejb3.xml***
- ***jboss-app.xml***
- ***jboss-web.xml***
- ****-jms.xml***
- ****-ds.xml***

以下管理 CLI 命令可用于启用或禁用特定于 JBoss 的描述符中的属性替换：

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-replacement",value=VALUE)
```

Jakarta EE 描述符替换由 **spec-descriptor-property-replacement** 标志控制，默认为 禁用。启用后，可以在以下部署描述符中替换属性：

- ***ejb-jar.xml***
- ***persistence.xml***
- ***application.xml***
- ***web.xml***

以下管理 CLI 命令可用于在 Jakarta EE 描述符中启用或禁用属性替换：

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-replacement",value=VALUE)
```

3.6.7. 使用 Git 管理配置数据

自 JBoss EAP 7.3 起，您可以使用 Git 来管理和持久保留您的服务器配置数据、属性文件和部署。这不仅允许您管理这些文件的版本历史记录，还允许您使用一个或多个 Git 存储库在多个服务器和节点间共享服务器和应用配置。此功能仅适用于使用默认配置目录布局的单机服务器。

您可以选择在本地 Git 存储库中使用配置数据，也可以从远程 Git 存储库中提取数据。Git 存储库在 **jboss.server.base.dir** 目录中配置，这是单机服务器内容的基础目录。在 **jboss.server.base.dir** 目录配置为使用 Git 后，JBoss EAP 将自动通过管理 CLI 或管理控制台提交每次更新到配置的更新。通过手动编辑配置文件在服务器外进行的任何更改都不会提交或保留，但您可以使用 Git CLI 添加和提交手动更改。您还可以使用 Git CLI 查看提交历史记录，管理分支和管理内容。

要使用此功能，在您启动服务器时在命令行中传递一个或多个参数。

表 3.4. Git 配置管理的服务器启动参数

参数	描述
--git-repo	用于管理和持久服务器配置数据的 Git 存储库的位置。如果要在本地存储，或者远程存储库的 URL，这可能是本地的。

参数	描述
--git-branch	Git 存储库中要使用的分支或标签名称。*此参数应命名现有的分支或标签名称，因为如果不存在，则不会创建该分支或标签名称*。如果使用标签名称，请将存储库置于分离的 HEAD 状态，这意味着以后的提交不会附加到任何分支。标签名称为只读，通常在多个节点之间复制配置时使用。
--git-auth	指向 Elytron 配置文件的 URL，该文件包含连接远程 Git 存储库时要使用的凭据。如果您的远程 Git 存储库需要身份验证，则需要此参数。虽然 Git 支持 SSH 身份验证，但 Elytron 不支持；因此，目前只能指定要通过 HTTP 或 HTTPS 进行身份验证的凭据，而不是通过 SSH 进行身份验证。此参数不与 **本地存储库** 一起使用。

*使用本地 **Git** 存储库*
*要使用本地 Git 存储库，请使用 **--git-repo=local** 参数启动服务器。您还可以在启动服务器时添加 **--git-branch=GIT_BRANCH_NAME** 参数，在远程存储库中指定可选分支或标签名称。此参数应命名现有的分支或标签名称，因为如果不存在，则不会 创建该分支或标签名称。如果使用标签名称，请将存储库置于分离的 HEAD 状态，这意味着以后的提交不会附加到任何分支。*

*以下是使用 本地存储库 的 **1.0.x** 分支启动服务器的命令示例：*

```
$ EAP_HOME/bin/standalone.sh --git-repo=local --git-branch=1.0.x
```

*如果您使用 参数启动服务器以使用 本地 Git 存储库，JBoss EAP 将检查 **jboss.server.base.dir** 目录是否已针对 Git 进行了配置。否则，JBoss EAP 会利用现有配置内容在 **jboss.server.base.dir** 目录中创建并初始化 Git 存储库。JBoss EAP 检查由 **--git-branch** 参数传递的分支名称。如果未传递该参数，它将检查 **master** 分支。初始化后，您应该会看到单机服务器内容的基础目录中的a **.git/** 目录和 **a.gitignore** 文件。*

*使用远程 **Git** 存储库*
*要使用远程 Git 存储库，请使用 **--git-repo=REMOTE_REPO** 参数启动服务器。参数的值可以是您手动添加到本地 Git 配置的 URL 或远程别名。*

*您还可以在启动服务器时添加 **--git-branch=GIT_BRANCH_NAME** 参数，在远程存储库中指定可选分支或标签名称。此参数应命名现有的分支或标签名称，因为如果不存在，则不会创建该分支或标签名称。如果使用标签名称，请将存储库置于分离的 HEAD 状态，这意味着以后的提交不会附加到任何分支。*

*如果您的 Git 存储库需要身份验证，那么在启动服务器时，还必须添加 **--git-auth=AUTH_FILE_URL** 参数。此参数应当是 Elytron 配置文件的 URL，其中包含连接 Git 存储库所需的凭据。以下是可用于身份验证的 Elytron 配置文件的示例：*

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <authentication-client xmlns="urn:elytron:client:1.2">
    <authentication-rules>
      <rule use-configuration="test-login">
      </rule>
    </authentication-rules>
    <authentication-configurations>
      <configuration name="test-login">
```

```
        <sasl-mechanism-selector selector="BASIC" />
        <set-user-name name="eap-user" />
        <credentials>
          <clear-password password="my_api_key" />
        </credentials>
        <set-mechanism-realm name="testRealm" />
      </configuration>
    </authentication-configurations>
  </authentication-client>
</configuration>
```

注意

虽然 Git 支持 SSH 身份验证，但 Elytron 不支持；因此，目前只能指定要通过 HTTP 或 HTTPS 进行身份验证的凭据，而不是通过 SSH 进行身份验证。

以下是使用 full 配置文件启动服务器的命令示例，它使用远程 **eap-configuration** 存储库的 **1.0.x** 分支，并将 URL 传递到包含身份验证凭据的 Elytron 配置文件。

```
$ EAP_HOME/bin/standalone.sh --git-repo=https://github.com/MY_GIT_ID/eap-configuration.git --git-branch=1.0.x --git-auth=file:///home/USER_NAME/github-wildfly-config.xml --server-config=standalone-full.xml
```

如果您使用 参数启动服务器以使用远程 Git 存储库，JBoss EAP 会检查 **jboss.server.base.dir** 目录是否已针对 Git 进行了配置。否则，JBoss EAP 将删除 **jboss.server.base.dir** 目录中的现有配置文件，并将它们替换为远程 Git 配置数据。JBoss EAP 检查由 **--git-branch** 参数传递的分支名称。如果未传递该参数，它将检查 **master** 分支。这个过程完成后，您应该会在单机服务器内容的基础目录中看到 a **.git/** 目录和 **a.gitignore** 文件。

警告

如果您稍后启动服务器传递与最初使用的 **--git-repo** URL 或 **--git-branch** 名称不同的服务器，您将在尝试启动服务器时看到错误消息 **java.lang.RuntimeException: WFLYSRV0268: Failed to pull repository GIT_REPO_NAME**。这是因为 JBoss EAP 尝试从与 **jboss.server.base.dir** 目录中当前配置的仓库和分支不同的存储库和分支中提取配置数据，而 Git 拉取会导致冲突。

使用 **Git** 时发布远程配置数据

您可以使用管理 CLI 发布配置 操作将您的 Git 存储库更改推送到远程存储库。由于当您启动服务器时，JBoss EAP 会在启动过程中从远程 Git 存储库提取配置，这允许您在多个服务器之间共享配置数据。您只能将此操作用于远程存储库。它不适用于本地存储库。

以下管理 CLI 操作将配置数据发布到远程 **eap-configuration** 存储库：

```
:publish-configuration(location="=https://github.com/MY_GIT_ID/eap-configuration.git")
{"outcome" => "success"}
```

通过 **Git** 使用快照

除了使用 Git 提交历史记录来跟踪配置更改外，您还可以生成快照以在特定时间点保留配置。您可以列出快照并删除快照。

使用 **Git** 时生成快照
快照作为标签存储在 Git 中。您可以在 **take-snapshot** 操作上将快照标签名称和提交消息指定为参数。

以下管理 CLI 操作将拍摄快照，并将其命名为"snapshot-01"标签。

```
:take-snapshot(name="snapshot-01", comment="1st snapshot")
{
    "outcome" => "success",
    "result" => "1st snapshot"
}
```

使用 **Git** 列出快照
您可以使用 **list-snapshots** 操作列出所有快照标签。

以下管理 CLI 操作列出了快照标签：

```
:list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" => "",
        "names" => [
            "snapshot : 1st snapshot",
            "refs/tags/snapshot-01",
            "snapshot2 : 2nd snapshot",
            "refs/tags/snapshot-02"
        ]
    }
}
```

使用 **Git** 删除快照
您可以通过在 **delete-snapshot** 操作上传递标签名称来删除特定的快照。

以下管理 CLI 操作将使用标签名称 "snapshot-01" 删除快照。

```
:delete-snapshot(name="snapshot-01")
{"outcome" => "success"}
```

3.7. 文件系统路径

JBoss EAP 将逻辑名称用于文件系统路径。然后，其他配置区域可以使用其逻辑名称引用路径，从而避免为每个实例使用绝对路径，并允许特定主机配置解析为通用逻辑名称。

例如，默认的 **logging** 子系统配置将 **jboss.server.log.dir** 声明为服务器日志目录的逻辑名称。

示例：服务器日志目录的相对路径示例

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

JBoss EAP 自动提供多个标准路径，用户无需在配置文件中进行配置。

表 3.5. 标准路径

属性	描述
java.home	Java 安装目录
jboss.controller.temp.dir	单机服务器和受管域的常用别名.用于临时文件存储的目录。等同于受管域中的 **jboss.domain.temp.dir** 和单机服务器上的 **jboss.server.temp.dir**。
jboss.domain.base.dir	域内容的基础目录。
jboss.domain.config.dir	包含域配置的目录。
jboss.domain.data.dir	域将用于持久数据存储的目录。
jboss.domain.log.dir	域将用于持久日志文件存储的目录。
jboss.domain.temp.dir	域将用于临时文件存储的目录。
jboss.domain.deployment.dir	域将用于存储已部署内容的目录。
jboss.domain.servers.dir	域将用于存储受管域实例输出的目录。
jboss.home.dir	JBoss EAP 分发的根目录.
jboss.server.base.dir	单机服务器内容的基础目录。
jboss.server.config.dir	包含单机服务器配置的目录。
jboss.server.data.dir	单机服务器将用于持久数据存储的目录。
jboss.server.log.dir	单机服务器将用于日志文件存储的目录。
jboss.server.temp.dir	单机服务器将用于临时文件存储的目录。
jboss.server.deploy.dir	单机服务器将用于存储已部署内容的目录。
user.dir	用户的当前工作目录.
user.home	用户主目录。

您可以覆盖标准路径或 添加自定义路径。

3.7.1. 查看文件系统路径

使用以下命令列出文件系统路径：

```
ls /path
```

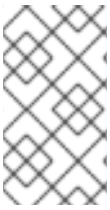
注意

在受管域中，您可以使用以下命令列出特定服务器的文件系统路径：

```
ls /host=HOST_NAME/server=SERVER_NAME/path
```

使用以下命令来读取文件系统路径的值：

```
/path=PATH_NAME:read-resource
```



注意

在受管域中，您可以使用以下命令读取特定服务器的文件系统路径值：

```
/host=HOST_NAME/server=SERVER_NAME/path=PATH_NAME:read-resource
```

3.7.2. 覆盖标准路径

您可以覆盖以 jboss. **server.*** 或 **jboss. domain.*** 开头的标准路径的默认位置。这可以通过以下两种方式之一完成：

- 启动服务器时传递命令行参数。例如：

```
$ EAP_HOME/bin/standalone.sh -Djboss.server.log.dir=/var/log
```

- 修改服务器配置文件中 **standalone.conf** 或 **domain.conf** 中的 **JAVA_OPTS** 变量，使其包含新位置。例如：

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.log.dir=/var/log"
```

覆盖受管域的标准路径
在本示例中，目标是将域文件存储在 **/opt/jboss_eap/domain_data** 目录中，并为每个顶级目录指定一个自定义名称。使用默认目录分组 **by-server**。

- 日志文件存储在 **all_logs** 子目录中
- 数据文件存储在 **all_data** 子目录中
- 临时文件存储在 **all_temp** 子目录中
- 服务器的文件存储在 **all_servers** 子目录中

为了实现此配置，启动 JBoss EAP 时将覆盖多个系统属性。

```
$ EAP_HOME/bin/domain.sh -Djboss.domain.temp.dir=/opt/jboss_eap/domain_data/all_temp -
Djboss.domain.log.dir=/opt/jboss_eap/domain_data/all_logs -
Djboss.domain.data.dir=/opt/jboss_eap/domain_data/all_data -
Djboss.domain.servers.dir=/opt/jboss_eap/domain_data/all_servers
```

生成的路径结构如下：

```
/opt/jboss_eap/domain_data/
            ├── all_data
            ├── all_logs
            ├── all_servers
            │   ├── server-one
            │   │   ├── data
            │   │   ├── log
            │   │   └── tmp
            │   └── server-two
            │       ├── data
            │       ├── log
            │       └── tmp
            └── all_temp
```

3.7.3. 添加自定义路径

您可以使用管理 CLI 或管理控制台添加自定义文件系统路径。

- 在管理 CLI 中，您可以使用以下管理 CLI 命令添加新路径：

```
/path=my.custom.path:add(path=/my/custom/path)
```

- 从管理控制台中，您可以通过导航到 **Configuration** 选项卡、选择路径并单击 **View** 来配置文件系统路径。在这里，您可以添加、修改和删除路径。

然后您可以在配置中使用这个自定义路径。例如，以下日志处理程序将自定义路径用于其相对路径：

```
<subsystem xmlns="urn:jboss:domain:logging:6.0">
  ...
  <periodic-rotating-file-handler name="FILE" autoflush="true">
    <formatter>
      <named-formatter name="PATTERN"/>
    </formatter>
    <file relative-to="my.custom.path" path="server.log"/>
    <suffix value=".yyyy-MM-dd"/>
    <append value="true"/>
  </periodic-rotating-file-handler>
  ...
</subsystem>
```

3.8. 目录分组

在受管域中，每一服务器的文件存储在 **EAP_HOME/domain** 目录中。您可以使用主机控制器的 **directory-grouping** 属性来指定如何组织服务器的子目录。目录可以按 服务器 或 类型 分组.默认情况下，目录按 服务器 分组。

通过服务器分组目录
默认情况下，目录按服务器分组。如果您的管理以 server 为中心的，则建议进行此配置。例如，它允许为每个服务器实例配置备份和日志文件处理。

如果使用 ZIP 安装方法安装 JBoss EAP，默认目录结构（由 server 进行分组）将如下所示：

```
EAP_HOME/domain
        └─ servers
          ├── server-one
          │   ├── data
          │   ├── tmp
          │   └── log
          └── server-two
              ├── data
              ├── tmp
              └── log
```

要按服务器对域目录进行分组，请输入以下管理 CLI 命令：

```
/host=HOST_NAME:write-attribute(name=directory-grouping,value=by-server)
```

这将更新主机控制器的 **host.xml** 配置文件：

```
<servers directory-grouping="by-server">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

按类型分组目录

您可以改为按文件类型分组目录，而不是按 server 对目录进行分组。如果您的管理是以文件类型为中心的，则建议采用此配置。例如，这样可以轻松备份仅备份 数据文件。

如果使用 ZIP 安装方法安装 JBoss EAP，并且域的文件按照类型分组，则目录结构将如下所示：

```
EAP_HOME/domain
        ├── data
        │   └── servers
        │       ├── server-one
        │       └── server-two
        ├── log
        │   └── servers
        │       ├── server-one
        │       └── server-two
        └── tmp
            └── servers
                ├── server-one
                └── server-two
```

要根据类型对域目录进行分组，请输入以下管理 CLI 命令：

```
/host=HOST_NAME:write-attribute(name=directory-grouping,value=by-type)
```

这将更新主机控制器的 **host.xml** 配置文件：

```
<servers directory-grouping="by-type">
  <server name="server-one" group="main-server-group"/>
  <server name="server-two" group="main-server-group" auto-start="true">
```

```
    <socket-bindings port-offset="150"/>
  </server>
</servers>
```

3.9. 系统属性

您可以使用 Java 系统属性配置许多 JBoss EAP 选项，并可设置任何名称值对供应用服务器内使用。

系统属性可用于覆盖 JBoss EAP 配置中的默认值。例如，公共接口绑定地址的以下 XML 配置显示它可以通过 **jboss.bind.address** 系统属性设置，但如果未提供 system 属性，则默认为 **127.0.0.1**。

```
<inet-address value="${jboss.bind.address:127.0.0.1}"/>
```

您可以通过几种方式在 JBoss EAP 中设置系统属性，包括：

- 将 system 属性传递给 JBoss EAP 启动脚本
- 使用管理 CLI
- 使用管理控制台
- 使用 **JAVA_OPTS** 环境变量

如果您使用 JBoss EAP 受管域，系统属性可以应用到整个域、特定服务器组、特定主机及其所有服务器实例，或者仅应用到一个特定的服务器实例。与大多数其他 JBoss EAP 域设置一样，在更具体的级别上设置的系统属性将覆盖更抽象化的级别。如需更多信息，请参阅域管理一章。

将系统属性传递给启动脚本

您可以使用 **-D** 参数将系统属性传递到 JBoss EAP 启动脚本。例如：

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=192.168.1.2
```

这种设置系统属性的方法对于 JBoss EAP 启动之前需要设置的 JBoss EAP 选项特别有用。

使用管理 CLI 设置系统属性

使用管理 CLI，您可以使用以下语法设置系统属性：

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例如：

```
/system-property=jboss.bind.address:add(value=192.168.1.2)
```

使用管理 CLI 设置系统属性时，某些 JBoss EAP 选项（包括上述 **jboss.bind.address** 示例）仅在下一次服务器重启后生效。

对于受管域，上例为整个域配置系统属性，但您也可以在更具体的域配置级别上设置或覆盖系统属性。

使用管理控制台设置系统属性

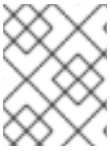
- 对于独立 JBoss EAP 服务器，您可以在管理控制台 **Configuration** 选项卡下配置系统属性。选择"系统属性"，然后单击"查看"按钮。
- 对于受管域：

- 域级系统属性可以在 Configuration 选项卡中设置。选择"系统属性"，然后单击"查看"按钮。
- 服务器组和服务器级系统属性可以在 Runtime 选项卡中设置。选择您要配置的服务器组或服务器，单击服务器组或服务器名称旁边的 View 按钮，然后选择系统属性选项卡。
- 主机级别系统属性可以在 Runtime 选项卡中设置。选择您要配置的主机，然后使用主机名旁边的下拉菜单选择属性。

使用 JAVA_OPTS 设置系统属性

系统属性也可以使用 **JAVA_OPTS** 环境变量进行配置。可以通过多种方式修改 **JAVA_OPTS**，但 JBoss EAP 提供了用于设置 JBoss EAP 流程使用的 **JAVA_OPTS** 的配置文件。

对于单机服务器，此文件为 **EAP_HOME/bin/standalone.conf**，或者对于受管域，它是 **EAP_HOME/bin/domain.conf**。对于 Microsoft Windows 系统，这些文件具有 **.bat** 扩展名。



注意

对于 RPM 安装，RPM 服务配置文件是修改 **JAVA_OPTS** 以配置系统属性的首选位置。如需更多信息，请参阅配置 RPM 服务属性。

将您的系统属性定义添加到相关配置文件中的 **JAVA_OPTS** 中。以下示例演示了在 Red Hat Enterprise Linux 系统中设置绑定地址。

- 对于 **standalone.conf**，请在文件的末尾添加 **JAVA_OPTS** 系统属性定义。例如：

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"
```

- 对于 **domain.conf**，必须在进程控制器 **JAVA_OPTS** 设置之前设置 **JAVA_OPTS**。例如：

```
...
# Set the bind address
JAVA_OPTS="$JAVA_OPTS -Djboss.bind.address=192.168.1.2"

# The ProcessController process uses its own set of java options
if [ "x$PROCESS_CONTROLLER_JAVA_OPTS" = "x" ]; then
...
```

3.10. 管理审计日志记录

您可以为管理接口启用审计日志，它会记录使用管理控制台、管理 CLI 或使用管理 API 的自定义应用执行的所有操作。审计日志条目以 JSON 格式存储。默认情况下禁用审计日志。

您可以将审计日志记录配置为输出到文件或 syslog 服务器。

注意

由于 JBoss EAP 中没有经过身份验证的会话，因此无法对登录和注销事件进行审计。相反，当从用户收到操作时，会记录审计消息。

独立服务器审计日志记录

虽然默认禁用，但默认的审计日志配置会写入到文件中。

```
<audit-log>
    <formatters>
        <json-formatter name="json-formatter"/>
    </formatters>
    <handlers>
        <file-handler name="file" formatter="json-formatter" path="audit-log.log" relative-
to="jboss.server.data.dir"/>
    </handlers>
    <logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="file"/>
        </handlers>
    </logger>
</audit-log>
```

此配置可以通过以下管理 CLI 命令读取：

```
/core-service=management/access=audit:read-resource(recursive=true)
```

请参阅启用审计日志记录，为单机服务器启用审计日志记录。

受管域审计日志记录

虽然默认为禁用，默认的审计日志记录配置会为每个主机和每台服务器写入一个文件。

```
<audit-log>
    <formatters>
        <json-formatter name="json-formatter"/>
    </formatters>
    <handlers>
        <file-handler name="host-file" formatter="json-formatter" relative-to="jboss.domain.data.dir"
path="audit-log.log"/>
        <file-handler name="server-file" formatter="json-formatter" relative-to="jboss.server.data.dir"
path="audit-log.log"/>
    </handlers>
    <logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="host-file"/>
        </handlers>
    </logger>
    <server-logger log-boot="true" log-read-only="false" enabled="false">
        <handlers>
            <handler name="server-file"/>
        </handlers>
    </server-logger>
</audit-log>
```

此配置可以通过以下管理 CLI 命令读取：

```
/host=HOST_NAME/core-service=management/access=audit:read-resource(recursive=true)
```

请参阅启用审计日志记录，为受管域启用审计日志记录。

3.10.1. 启用管理审计日志记录

JBoss EAP 预配置了审计日志记录的文件处理程序，但审计日志记录默认为禁用。启用审计日志记录的管理 CLI 命令取决于您是以单机服务器还是在受管域中运行。如需文件处理程序属性，请参阅管理审计日志记录属性。

以下说明启用 **NATIVE** 和 **HTTP** 审计日志记录。要配置 JMX 审计日志，请参阅启用 JMX 管理审计日志。

要设置 syslog 审计日志记录，请参阅发送管理审计日志记录到 Syslog 服务器。

启用单机服务器审计日志记录
可以使用以下命令启用审计日志记录：

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

默认情况下，这会将审计日志写入 **EAP_HOME/standalone/data/audit-log.log**。

启用受管域审计日志记录
受管域的默认审计日志记录配置已预先配置为为每个主机和每一服务器写入审计日志。

可以使用以下命令，为每个主机启用审计日志记录：

```
/host=HOST_NAME/core-service=management/access=audit/logger=audit-log:write-
attribute(name=enabled,value=true)
```

默认情况下，这会将审计日志写入 **EAP_HOME/domain/data/audit-log.log**。

可以使用以下命令，为每个服务器启用审计日志记录：

```
/host=HOST_NAME/core-service=management/access=audit/server-logger=audit-log:write-
attribute(name=enabled,value=true)
```

默认情况下，这会将审计日志写入 **EAP_HOME/domain/servers/SERVER_NAME/data/audit-log.log**。

3.10.2. 启用 JMX 管理审计日志记录

JBoss EAP 预配置了 JMX 审计日志的文件处理程序，尽管默认情况下这些日志是禁用的。启用审计日志记录的管理 CLI 命令取决于您是作为单机服务器还是受管域来运行。

要配置 **NATIVE** 或 **HTTP** 审计日志记录，请参阅启用管理审计日志记录。

启用单机服务器 JMX 审计日志记录
可以使用以下命令为单机服务器启用 JMX 审计日志记录：

```
/subsystem=jmx/configuration=audit-log:add()
/subsystem=jmx/configuration=audit-log/handler=file:add()
```

这将启用 JMX 审计日志记录，然后使用定义 的文件 处理程序将这些日志写入 **EAP_HOME/standalone/data/audit-log.log**。

启用受管域 JMX 审计日志记录
可以为受管域中的每个主机和配置文件启用 JMX 审计日志记录。

为主机启用 JMX 审计日志记录

1. 在主机的 **jmx** 子系统中启用审计日志记录。

```
/host=HOST_NAME/subsystem=jmx/configuration=audit-log:add()
```

2. 启用了 **jmx** 子系统的审计日志记录后，可以使用以下命令为主机定义处理程序：

```
/host=HOST_NAME/subsystem=jmx/configuration=audit-log/handler=host-file:add()
```

默认情况下，这会将 JMX 审计日志写入 **EAP_HOME/domain/data/audit-log.log**。

为配置集启用 JMX 审计日志记录

1. 在配置集的 **jmx** 子系统中启用审计日志记录。

```
/profile=PROFILE_NAME/subsystem=jmx/configuration=audit-log:add()
```

2. 启用了 **jmx** 子系统的审计日志记录后，可以使用以下命令为配置文件定义处理程序：

```
/profile=PROFILE_NAME/subsystem=jmx/configuration=audit-log/handler=server-file:add()
```

默认情况下，这会将 JMX 审计日志写入 **EAP_HOME/domain/servers/SERVER_NAME/data/audit-log.log**。

3.10.3. 将管理审计日志记录发送到 Syslog 服务器

syslog 处理程序指定审计日志条目发送到 **syslog** 服务器的参数，特别是 **syslog** 服务器的主机名和 **syslog** 服务器侦听的端口。将审计日志记录发送到 **syslog** 服务器比记录到本地文件或本地 **syslog** 服务器更安全的选项。可以定义多个 **syslog** 处理程序并同时处于活动状态。

默认情况下，审计日志在启用时预先配置为输出到文件。使用以下步骤将审计日志记录设置并启用到 **syslog** 服务器。如需 **syslog** 处理程序属性，请参阅管理审计日志记录属性。

1. 添加 **syslog** 处理程序。
通过指定 **syslog** 服务器的主机和端口，创建 **syslog** 处理程序。在受管域中，您必须在 **/core-service** 命令之前使用 **/host=HOST_NAME**。

```
batch
/core-service=management/access=audit/syslog-handler=SYSLOG_HANDLER_NAME:add(formatter=json-formatter)
/core-service=management/access=audit/syslog-handler=SYSLOG_HANDLER_NAME/protocol=udp:add(host=HOST_NAME,port=PORT)
run-batch
```

注意

要传递的参数因指定的协议而异。

要将处理器配置为使用 TLS 与 syslog 服务器安全地通信，还必须配置身份验证，例如：

```
/core-service=management/access=audit/syslog-handler=SYSLOG_HANDLER_NAME/protocol=tls/authentication=truststore:add(keystore-path=PATH_TO_TRUSTSTORE,keystore-password=TRUSTSTORE_PASSWORD)
```

2. 添加对 syslog 处理程序的引用。
在受管域中，您必须在此命令前加上 **/host=HOST_NAME**。

```
/core-service=management/access=audit/logger=audit-log/handler=SYSLOG_HANDLER_NAME:add
```

3. 启用审计日志记录。
请参阅 启用管理审计日志记录以启用审计日志记录。

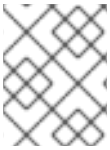
重要

在 JBoss EAP 中启用对 syslog 服务器的审计日志记录将不起作用，除非操作系统中也启用了日志记录。

有关 Red Hat Enterprise Linux 中的 **rsyslog** 配置的更多信息，请参阅 Red Hat Enterprise Linux 系统管理员指南的 Rsyslog 基本配置 部分，网址为 https://access.redhat.com/documentation/en/red-hat-enterprise-linux/。

3.10.4. 阅读审计日志条目

输出到文件的审计日志条目最好使用文本查看器查看，而 syslog 服务器的输出则最好使用 syslog viewer 应用程序查看。



注意

不建议使用文本编辑器查看日志文件，因为这样可能会导致进一步日志条目写入到日志文件中。

审计日志条目以 JSON 格式存储。每个日志条目都以可选时间戳开头，后跟下表中的字段。

表 3.6. 管理审计日志字段

字段名称	描述

字段名称	描述
访问	这可以有以下值之一： • **NATIVE** - 操作通过原生管理界面启动。 • **HTTP** - 操作通过域 HTTP 接口进入。 • **JMX** - 该操作通过 **jmx** 子系统进行。
引导	如果操作在启动过程中执行，则值为 **true** ；如果该操作在服务器启动并运行后执行，则为 **false**。
domainUUID	个 ID，用于链接所有操作，因为它们从域控制器传播到其服务器、从属主机控制器和从属主机控制器服务器。
ops	正在执行的操作。这是序列化为 JSON 的操作列表。在引导时，这是解析 XML 时生成的操作。引导后，列表通常包含单个条目。
r/o	如果操作不更改管理模式，则值为 **true** ；如果更改管理模式，则为 **false**。
remote-address	执行此操作的客户端地址。
success	如果操作成功，则值为 **true**，如果回滚，则为 **false**。
type	这可以具有值 **core**，即管理操作或 **jmx**，即来自 **jmx** 子系统。
user	经过身份验证的用户的用户名。如果使用管理 CLI 在与正在运行的服务器相同的计算机上发生，则会使用特殊用户 **$local**。
version	JBoss EAP 实例的版本号。

3.11. 服务器生命周期事件通知

您可以使用 JBoss EAP core-management 子系统 或 JMX 为服务器生命周期事件设置通知。服务器运行时配置状态或服务器运行状态的变化将触发通知。

JBoss EAP 的服务器运行时配置状态为 STARTING、RUNNING、RELOAD_REQUIRED、RESTART_REQUIRED、STOPPING 和 STOPPED。

JBoss EAP 的服务器运行状态为 STARTING、NORMAL、ADMIN_ONLY、PRE_SUSPENDING、SUSPENDING、SUSPENDED、STOPPING 和 STOPPED。

3.11.1. 使用核心管理系统监控服务器生命周期事件

您可以将侦听器注册到 JBoss EAP 核心管理 子系统，以监控服务器生命周期事件。下列步骤演示了如何创建和注册将事件记录到文件中的示例侦听器。

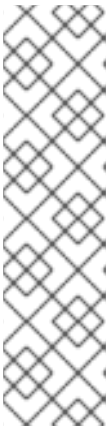
1. *创建监听器。*

*创建 **org.wildfly.extension.core.management.client.ProcessStateListener** 实施，如下例所示。*

示例：Listener Class

```
package org.simple.lifecycle.events.listener;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import org.wildfly.extension.core.management.client.ProcessStateListener;
import
org.wildfly.extension.core.management.client.ProcessStateListenerInitParameters;
import org.wildfly.extension.core.management.client.RunningStateChangeEvent;
import
org.wildfly.extension.core.management.client.RuntimeConfigurationStateChangeEven
t;

public class SimpleListener implements ProcessStateListener {

    private File file;
    private FileWriter fileWriter;
    private ProcessStateListenerInitParameters parameters;

    public void init(ProcessStateListenerInitParameters parameters) {
        this.parameters = parameters;
        this.file = new File(parameters.getInitProperties().get("file"));
        try {
            fileWriter = new FileWriter(file, true);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void cleanup() {
        try {
            fileWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            fileWriter = null;
        }
    }

    public void
runtimeConfigurationStateChanged(RuntimeConfigurationStateChangeEvent evt) {
        try {
            fileWriter.write(String.format("Runtime configuration state change for %s: %s
to %s\n", parameters.getProcessType(), evt.getOldState(), evt.getNewState()));
            fileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    public void runningStateChanged(RunningStateChangeEvent evt) {
        try {
            fileWriter.write(String.format("Running state change for %s: %s to %s\n",
parameters.getProcessType(), evt.getOldState(), evt.getNewState()));
            fileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



注意

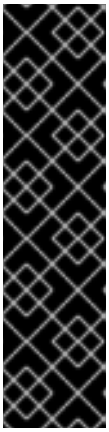
实施监听程序时请牢记以下几点：

- 在服务器重新加载时，侦听器在服务器尝试停止时停止侦听，并在服务器启动时重新加载侦听器。因此，实施必须确保它们可以在同一 JVM 中正确加载、初始化和删除多次。
- 通知监听器正在阻止，允许响应服务器状态变化。实施必须确保它们不会阻断或死锁。
- 每个侦听器实例在自己的线程中执行，而且顺序不受保证。

2. 编译该类，并将它打包为 JAR。
请注意，要编译，您需要依赖于 **org.wildfly.core:wildfly-core-management-client** Maven 模块。

3. 添加 JAR 作为 JBoss EAP 模块。
使用以下管理 CLI 命令，并提供模块名称和 JAR 路径：

```
module add --name=org.simple.lifecycle.events.listener --
dependencies=org.wildfly.extension.core-management-client --resources=/path/to/simple-
listener-0.0.1-SNAPSHOT.jar
```



重要

使用 模块 管理 CLI 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 CLI 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

4. 注册侦听器。
使用下列管理 CLI 命令，将侦听器添加到 **core-management** 子系统：指定记录服务器生命周期事件的类、模块和文件位置。

```
/subsystem=core-management/process-state-listener=my-simple-
listener:add(class=org.simple.lifecycle.events.listener.SimpleListener,
module=org.simple.lifecycle.events.listener,properties={file=/path/to/my-listener-output.txt})
```

现在，服务器生命周期事件会根据上面的 **SimpleList ener** 类记录到 **my-listener -output.txt** 文件中。例如，在管理 CLI 中发出 **:suspend** 命令会将以下内容输出到 **my-listener-output.txt** 文件：

```
Running state change for STANDALONE_SERVER: normal to suspending
Running state change for STANDALONE_SERVER: suspending to suspended
```

这表明运行状态已从 正常 更改为 暂停，然后从 暂停 变为 暂停。

3.11.2. 使用 JMX 通知监控服务器生命周期事件

您可以注册 JMX 通知监听程序来监控服务器生命周期事件。以下步骤演示了如何创建和添加将事件记录到文件中的示例侦听器。

1. 创建监听器。
   创建 **javax.management.NotificationListener** 实施，如下例所示：

   示例： Listener Class

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

import javax.management.AttributeChangeNotification;
import javax.management.Notification;
import javax.management.NotificationListener;

import org.jboss.logging.Logger;

public class StateNotificationListener implements NotificationListener {

  public static final String RUNTIME_CONFIGURATION_FILENAME = "runtime-
configuration-notifications.txt";
  public static final String RUNNING_FILENAME = "running-notifications.txt";
  private final Path targetFile;

  public StateNotificationListener() {
    this.targetFile = Paths.get("notifications/data").toAbsolutePath();
    init(targetFile);
  }

  protected Path getRuntimeConfigurationTargetFile() {
    return this.targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME);
  }

  protected Path getRunningConfigurationTargetFile() {
    return this.targetFile.resolve(RUNNING_FILENAME);
```

```
  }

  protected final void init(Path targetFile) {
    try {
      Files.createDirectories(targetFile);

      if (!Files.exists(targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME))) {
        Files.createFile(targetFile.resolve(RUNTIME_CONFIGURATION_FILENAME));
      }

      if (!Files.exists(targetFile.resolve(RUNNING_FILENAME))) {
        Files.createFile(targetFile.resolve(RUNNING_FILENAME));
      }
    } catch (IOException ex) {
        Logger.getLogger(StateNotificationListener.class).error("Problem handling JMX
Notification", ex);
    }
  }

  @Override
  public void handleNotification(Notification notification, Object handback) {
    AttributeChangeNotification attributeChangeNotification =
(AttributeChangeNotification) notification;
    if
("RuntimeConfigurationState".equals(attributeChangeNotification.getAttributeName()))
{
      writeNotification(attributeChangeNotification,
getRuntimeConfigurationTargetFile());
    } else {
      writeNotification(attributeChangeNotification,
getRunningConfigurationTargetFile());
    }
  }

  private void writeNotification(AttributeChangeNotification notification, Path path) {
    try (BufferedWriter in = Files.newBufferedWriter(path, StandardCharsets.UTF_8,
StandardOpenOption.APPEND)) {
      in.write(String.format("%s %s %s %s", notification.getType(),
notification.getSequenceNumber(), notification.getSource().toString(),
notification.getMessage()));
      in.newLine();
      in.flush();
    } catch (IOException ex) {
      Logger.getLogger(StateNotificationListener.class).error("Problem handling JMX
Notification", ex);
    }
  }
}
```

2. *注册通知监听程序。*
   将通知监听程序添加到 **MBeanServer**。

   示例：添加通知列表

```
MBeanServer server = ManagementFactory.getPlatformMBeanServer();
server.addNotificationListener(ObjectName.getInstance("jboss.root:type=state"), new
StateNotificationListener(), null, null);
```

3. 打包并部署到 JBoss EAP.

服务器生命周期事件现在根据上面的 **StateNotificationListener** 类记录到文件中。例如，在管理 CLI 中发出 **:suspend** 命令会将以下内容输出到 **running-notifications.txt** 文件：

```
jmx.attribute.change 5 jboss.root:type=state The attribute 'RunningState' has changed from 'normal'
to 'suspending'
jmx.attribute.change 6 jboss.root:type=state The attribute 'RunningState' has changed from
'suspending' to 'suspended'
```

这表明运行状态已从 正常 更改为 暂停，然后从 暂停 变为 暂停。

第 4 章 网络和端口配置

4.1. INTERFACES

JBoss EAP 在整个配置中引用了命名接口。这允许配置引用带有逻辑名称的独立接口声明，而不必要求每次使用的接口的完整详情。

这也便于在受管域中配置，其中网络接口详细信息可能因多台计算机而异。每个服务器实例可以对应一个逻辑名称组。

standalone.xml、**domain.xml** 和 **host.xml** 文件都包含接口声明。根据使用的默认配置，有几个预配置的接口名称。管理接口可用于需要管理层 的所有组件和服务，包括 HTTP 管理端点。公共 接口可用于所有应用相关的网络通信。无安全 接口用于标准配置中的 IIOP 套接字。专用 接口用于标准配置中的 JGroups 套接字。

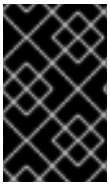
4.1.1. 默认接口配置

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="private">
    <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

默认情况下，JBoss EAP 将这些接口绑定到 **127.0.0.1**，但可以通过设置适当的 属性在运行时覆盖这些值。例如，通过以下命令将 JBoss EAP 启动为单机服务器时可以设置 公共 接口的**inet-address** ：

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

或者，您也可以在 **server start** 命令行上使用 **-b** 参数。有关服务器启动选项的更多信息，请参阅服务器运行时参数。



重要

如果您修改了 JBoss EAP 使用的默认网络接口或端口，您还必须记得更改使用修改后的接口或端口的任何脚本。其中包括 JBoss EAP 服务脚本，以及记得在访问管理控制台或管理控制台或 CLI 时指定正确的接口和端口。

4.1.2. 配置接口

通过为物理接口指定逻辑名称和选择条件来声明网络接口。选择条件可以引用通配符地址，或者指定接口或地址必须具有的一个或多个特征集，才能成为有效的匹配项。有关所有可用接口选择条件的列表，请参阅 Interface Attributes 部分。

接口可以使用管理控制台或管理 CLI 进行配置。以下是添加和更新接口的几个示例。首先显示管理 CLI 命令，后跟对应的配置 XML。

使用 NIC 值添加接口
添加一个 NIC 值为 **eth0** 的新接口。

```
/interface=external:add(nic=eth0)
```

```
<interface name="external">
  <nic name="eth0"/>
</interface>
```

使用 Several Conditional 值添加接口
添加一个与正确子网上任何接口/地址匹配的新接口（如果已启动，支持多播），且不是点对点的新接口。

```
/interface=default:add(subnet-match=192.168.0.0/16,up=true,multicast=true,not={point-to-point=true})
```

```
<interface name="default">
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
</interface>
```

更新接口属性
更新 公共 接口的默认 **inet-address** 值，保留 **jboss.bind.address** 属性，以允许在运行时设置此值。

```
/interface=public:write-attribute(name=inet-address,value="${jboss.bind.address:192.168.0.0}")
```

```
<interface name="public">
   <inet-address value="${jboss.bind.address:192.168.0.0}"/>
</interface>
```

向受管域中的服务器添加接口

```
/host=HOST_NAME/server-config=SERVER_NAME/interface=INTERFACE_NAME:add(inet-address=127.0.0.1)
```

```
<servers>
  <server name="SERVER_NAME" group="main-server-group">
    <interfaces>
      <interface name="INTERFACE_NAME">
        <inet-address value="127.0.0.1"/>
      </interface>
    </interfaces>
  </server>
</servers>
```

4.2. 套接字绑定

通过套接字绑定和套接字绑定组，您可以定义网络端口及其与 JBoss EAP 配置所需的网络接口的关系。套接字绑定是套接字的命名配置。套接字绑定组是套接字绑定声明的集合，这些声明按照逻辑名称分组。

这允许配置的其他部分根据其逻辑名称引用套接字绑定，而不必在每次使用套接字配置的完整详情。

这些指定配置的声明可以在 **standalone.xml** 和 **domain.xml** 配置文件中找到。单机服务器仅包含一个套接字绑定组，而受管域则可包含多个组。您可以为受管域中的每个服务器组创建一个套接字绑定组，或者在多个服务器组之间共享套接字绑定组。

JBoss EAP 默认使用的端口取决于使用的套接字绑定组以及您各个部署的要求。

JBoss EAP 配置的套接字绑定组中可以定义三种类型的套接字绑定：

入站套接字绑定

socket-binding 元素用于为 JBoss EAP 服务器配置入站套接字绑定。默认 JBoss EAP 配置提供多个预配置的 套接字绑定 元素，例如用于 HTTP 和 HTTPS 流量的元素。另一个示例是在为 JBoss EAP配置消息传递 https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/configuring_messaging/#broadcast_groups 的广播组小节。
此元素的属性可以在入站套接字绑定属性表中找到。

远程出站套接字绑定

remote-destination-outbound-socket-binding 元素用于为远程到 JBoss EAP 服务器的目的地配置出站套接字绑定。默认 JBoss EAP 配置提供一个示例远程目标套接字绑定，可用于邮件服务器。在为 JBoss EAP 配置消息传递的 https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/configuring_messaging/#use_provided_amq_adapter "为远程连接使用集成 Artemis 资源适配器"一节中，可找到另一个示例。
此元素的属性可以在 Remote Outbound Socket Binding Attributes 表中找到。

本地出站套接字绑定

local-destination-outbound-socket-binding 元素用于为属于 JBoss EAP 服务器本地的目的地配置出站套接字绑定。预计通常不会使用这种套接字绑定。
此元素的属性可以在 Local Outbound Socket Binding Attributes 表中找到。

4.2.1. 管理端口

JBoss EAP 7 中整合了管理端口。默认情况下，JBoss EAP 7 将端口 **9990** 用于本地管理（由管理 CLI 使用）和 HTTP 管理（由基于 Web 的管理控制台使用）。用作 JBoss EAP 6 中的原生管理端口的端口 **9999** 不再使用，但在需要时仍可启用。

如果为管理控制台启用了 HTTPS，则默认使用端口 **9993**。

4.2.2. 默认套接字绑定

JBoss EAP随附一个套接字绑定组，适用于这五个预定义配置文件（默认为 **ha**、**full**、**full-ha**、负载均衡）。

有关默认套接字绑定的详细信息，如默认端口和描述，请参阅 Default Socket Bindings 部分。

重要

如果您修改了 JBoss EAP 使用的默认网络接口或端口，您还必须记得更改使用修改后的接口或端口的任何脚本。其中包括 JBoss EAP 服务脚本，以及记得在访问管理控制台或管理控制台或 CLI 时指定正确的接口和端口。

独立服务器
作为单机服务器运行时，每个配置文件仅定义一个套接字绑定组。每个独立配置文件（**standalone.xml**、**standalone-ha.xml**、**standalone-full.xml**、**standalone-full-ha.xml**、**standalone-load-balancer.xml**）定义其对应配置集所使用的技术的套接字绑定。

例如，默认的单机配置文件**(standalone.xml)**指定以下套接字绑定：

```
<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

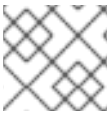
受管域
在受管域中运行时，所有套接字绑定组都在 **domain.xml** 文件中定义。有五个预定义的套接字绑定组：

- 标准套接字
- **ha-sockets**
- 全套接字
- **full-ha-sockets**
- **load-balancer-sockets**

每个套接字绑定组都指定其对应配置集所使用的技术套接字绑定。例如，**full-ha-sockets** 套接字绑定组定义几个 **jgroups** 套接字绑定，供 full-ha 配置文件用于高可用性。

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-interface="public">
    <!-- Needed for server groups using the 'default' profile  -->
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
```

```
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'ha' profile  -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full' profile  -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile  -->
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" interface="private" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" interface="private" port="7600"/>
    <socket-binding name="jgroups-udp" interface="private" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-
port="23364"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="load-balancer-sockets" default-interface="public">
    <!-- Needed for server groups using the 'load-balancer' profile  -->
    ...
  </socket-binding-group>
</socket-binding-groups>
```



注意

管理接口的套接字配置在域控制器的 **host.xml** 文件中定义。

4.2.3. 配置套接字绑定

在定义套接字绑定时，您可以配置 端口 和 接口 属性，以及多播设置，如 多播地址和多 播端口。有关所有可用套接字绑定属性的详情，请查看 **Socket Binding Attributes** 部分。

可以使用管理控制台或管理 CLI 配置套接字绑定。下列步骤介绍了添加套接字绑定组、添加套接字绑定和使用管理 CLI 配置套接字绑定设置。

1. 添加新套接字绑定组。请注意，作为单机服务器运行时，无法执行此步骤。

```
/socket-binding-group=new-sockets:add(default-interface=public)
```

2. 添加套接字绑定。

```
/socket-binding-group=new-sockets/socket-binding=new-socket-binding:add(port=1234)
```

3. 将套接字绑定更改为使用默认接口，由套接字绑定组设置。

```
/socket-binding-group=new-sockets/socket-binding=new-socket-binding:write-attribute(name=interface,value=unsecure)
```

以下示例显示了在上述步骤完成后 XML 配置可以如何进行。

```
<socket-binding-groups>
  ...
  <socket-binding-group name="new-sockets" default-interface="public">
    <socket-binding name="new-socket-binding" interface="unsecure" port="1234"/>
  </socket-binding-group>
</socket-binding-groups>
```

4.2.4. 查看服务器的套接字绑定和开放端口

您可以从管理控制台查看服务器的套接字绑定名称和开放端口。当服务器处于以下状态时，可看到该信息：

- **running**
- **reload-required**
- **restart-required**

查看服务器的套接字绑定和开放端口：

1. 访问管理控制台并导航到 **Runtime**。
2. 单击服务器，以查看右侧窗格中的套接字绑定名称和打开的端口。

4.2.5. 端口偏移

端口偏移是一个数字偏移值，添加到该服务器的套接字绑定组中指定的所有端口值中。这使得服务器能够继承其套接字绑定组中定义的端口值，并提供偏移以确保它不与同一主机上的任何其他服务器冲突。例如，如果套接字绑定组的 HTTP 端口为 **8080**，并且服务器使用端口偏移 **100**，则其 HTTP 端口为 **8180**。

以下是使用管理 CLI 为受管域中的服务器设置端口偏移 250 的示例。

```
/host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)
```

端口偏移可用于受管域中的服务器和在同一主机上运行多个单机服务器。

使用 **jboss.socket.binding.port-offset** 属性启动单机服务器时，您可以传递端口偏移。

```
$ EAP_HOME/bin/standalone.sh -Djboss.socket.binding.port-offset=100
```

4.3. IPV6 地址

默认情况下，JBoss EAP 配置为使用 IPv4 地址运行。以下步骤演示了如何配置 JBoss EAP 以使用 IPv6 地址运行。

为 IPv6 地址配置 JVM 堆栈

更新启动配置，以首选 IPv6 地址。

1. 打开 启动配置文件。
   - 作为单机服务器运行时，编辑 **EAP_HOME/bin/standalone.conf** 文件（或 **standalone.conf.bat** for Windows Server）。
   - 在受管域中运行时，编辑 **EAP_HOME/bin/domain.conf** 文件（或 **domain.conf.bat** for Windows Server）。
2. 将 **java.net.preferIPv4Stack** 属性设置为 **false**。

   ```
   -Djava.net.preferIPv4Stack=false
   ```

3. 附加 **java.net.preferIPv6Addresses** 属性，并将它设为 **true**。

   ```
   -Djava.net.preferIPv6Addresses=true
   ```

下例演示了在进行上述更改后，启动配置文件中的 JVM 选项如何显示。

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms1303m -Xmx1303m -Djava.net.preferIPv4Stack=false"
  JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
Djava.awt.headless=true"
  JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv6Addresses=true"
else
```

更新 IPv6 地址的接口声明

配置中的默认接口值可以更改为 IPv6 地址。例如，以下管理 CLI 命令将 管理 接口设置为 IPv6 环回地址 (**::1**)。

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:[::1]}")
```

以下示例演示了在运行上述命令后 XML 配置可以如何进行。

```
<interfaces>
   <interface name="management">
      <inet-address value="${jboss.bind.address.management:[::1]}"/>
   </interface>
   ....
</interfaces>
```

第 5 章 JBOSS EAP 安全

JBoss EAP 提供为其自己的接口和服务配置安全性的功能，并为其上运行的应用程序提供安全性。

- 有关常规安全概念以及 JBoss EAP 特定安全概念的概述，请参阅安全架构指南。
- 有关保护 JBoss EAP 本身的信息，请参阅如何配置服务器安全性。
- 如需有关为部署至 JBoss EAP 的应用提供安全性的信息，请参阅如何配置身份管理。
- 有关使用Kerberos 为 JBoss EAP 配置单点登录的信息，请参阅如何使用 Kerberos 设置 SSO。
- 如需有关使用 SAML v2 配置 JBoss EAP 单点登录的信息，请参阅如何使用 SAML v2 设置 SSO。

第 6 章 JBOSS EAP 类加载

JBoss EAP 使用模块化类加载系统来控制已部署应用的类路径。这个系统比传统的分层类加载器系统提供更多灵活性和控制力。开发人员对其应用可用的类有精细的控制，并且可以配置部署来忽略应用服务器提供的类，而非自有的类。

模块类加载器将所有 Java 类划分为称为模块的逻辑组。每个模块可以定义对其他模块的依赖关系，以便将该模块中的类添加到其自己的类路径中。由于每个部署的 JAR 和 WAR 文件都被视为模块，因此开发人员可以通过在其应用中添加模块配置来控制其应用的类路径的内容。

6.1. 模块

模块是用于类加载和依赖关系管理的逻辑类分组。JBoss EAP 识别两种不同类型的模块：静态 和动态。这两者之间的主要区别在于如何打包它们。

静态模块

静态模块在应用服务器的 **EAP_HOME/modules/** 目录中定义。每个模块都作为子目录存在，例如 **EAP_HOME/modules/com/mysql/**。每个模块目录随后包含一个插槽子目录，默认为 **main**，它包含 **module.xml** 配置文件以及任何所需的 JAR 文件。所有应用程序服务器提供的 API 都作为静态模块提供，包括 Jakarta EE API 以及其他 API。

示例：MySQL JDBC Driver module.xml 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

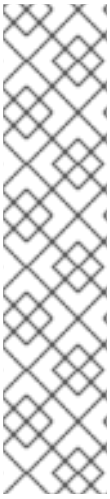
模块名称 **com.mysql** 必须与模块的目录结构匹配，不包括插槽名称 **main**。

如果许多应用部署在使用相同的第三方库的同一服务器上，创建自定义静态模块非常有用。管理员可以创建和安装包含这些库的模块，而不是将这些库与各个应用捆绑在一起。然后，应用可以声明对自定义静态模块的显式依赖关系。

JBoss EAP 分发中提供的模块位于 **EAP_HOME/modules** 目录中的 系统 目录中。这使它们与第三方提供的任何模块相隔离。任何红帽都在 JBoss EAP 基础上提供的产品也会将其模块安装到 系统 目录中。

用户必须确保自定义模块安装至 **EAP_HOME/modules** 目录中，每个模块使用一个目录。这可确保加载 系统 目录中已存在的模块自定义版本，而不是所附带的版本。这样，用户提供的模块优先于系统模块。

如果您使用 **JBOSS_MODULEPATH** 环境变量更改 JBoss EAP 搜索模块的位置，则产品将在指定的位置之一内查找 系统 子目录结构。系统 结构必须存在于通过 **JBOSS_MODULEPATH** 指定的位置。



注意

从 JBoss EAP 7.1 开始，还支持在 **module.xml** 文件的**resource-root path** 元素中使用绝对路径。这样，就可以访问您的资源库，而无需将它们移到 **EAP_HOME/modules** 目录。

示例： module.xml 文件中的 Absolute Path

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="oracle.jdbc">
   <resources>
      <resource-root path="/home/redhat/test.jar"/>
   </resources>
</module>
```

动态模块

动态模块由应用服务器为每个 JAR 或 WAR 部署创建和加载，或者针对 EAR 中的每个子部署创建和加载。动态模块的名称派生自部署存档的名称。由于部署是作为模块加载的，所以它们可以配置依赖项，供其他部署用作依赖项。

只有在需要时才加载模块。这通常仅在部署具有显式或隐式依赖项的应用时才发生。

预定义模块

从 JBoss EAP 7.2 开始，当您使用默认模块加载程序时，可以使用一组预定义模块。包括所有 JBoss 模块 API 的特殊模块 **org.jboss.modules** 始终可用，并由 JBoss 模块提供。标准 Java 平台模块系统 (JPMS) 模块（Java 9 及更高版本中提供）也可通过其标准名称提供。使用 JDK 8 时，JDK 9 模块由 JBoss 模块模拟。

有关 Java 9 或更高版本中可用平台模块列表，请参阅相应的 JDK 文档。

有关为 Java 8 提供的平台模块列表，请参阅为 Java 8 提供的平台模块。

6.2. 模块依赖项

模块依赖项是一种声明，一个模块需要一个或多个其他模块的类才能正常工作。当 JBoss EAP 加载模块时，模块类加载程序会解析该模块的依赖性，并将各个依赖项中的类添加到其类路径中。如果无法找到指定的依赖项，模块将无法加载。

注意

有关模块和模块化类加载系统的完整详情，请参阅模块部分。

部署的应用（如 JAR 或 WAR）作为动态模块加载，并利用依赖项来访问 JBoss EAP 提供的 API。

有两种类型的依赖项：显式和隐式。

显式依赖项

开发人员在配置文件中声明显式依赖项。静态模块可以在其 **module.xml** 文件中声明依赖项。动态模块可以声明部署的 **MANIFEST.MF** 或 **jboss-deployment-structure.xml** 部署描述符中的依赖关系。

隐式依赖项

当部署中发现特定条件或元数据时，JBoss EAP 会自动添加隐式依赖关系。JBoss EAP 提供的 Jakarta EE API 是通过检测部署中隐式依赖项而添加的模块示例。
也可以使用 **jboss-deployment-structure.xml** 部署描述符文件将部署配置为排除特定的隐式依赖项。当应用捆绑了 JBoss EAP 将尝试添加 作为隐式依赖项的库的特定版本时，这非常有用。

可选依赖项
明确的依赖项可以作为可选指定。如果加载可选依赖项，则不会导致模块加载失败。但是，如果稍后依赖项可用，它不会添加到模块的类路径中。加载模块时，依赖项必须可用。

导出依赖性
模块的类路径仅包含其自身的类和直接依赖项的类。模块无法访问其中一个依赖项的依赖项类别。不过，模块可以指定导出明确的依赖项。导出的依赖关系会提供给依赖于导出它的模块的任何模块。

例如，模块A 依赖于模块B，而模块B 则依赖于模块C。模块A 可以访问模块B 的类，而模块B 则可访问模块C 类。模块A 无法访问模块C 类，除非：

- 模块A 声明了明确依赖模块C 或
- 模块B 导出其对模块C 的依赖。

全局模块
全局模块是 JBoss EAP 提供的一个模块，作为每个应用的依赖性。任何模块均可通过添加至 JBoss EAP 的全局模块列表来实现全局。它不需要更改模块。

详情请查看 Define Global Modules 部分。

6.3. 创建自定义模块

可以添加自定义静态模块，以便为在 JBoss EAP 上运行的部署提供资源。您可以手动创建模块，也可以使用管理 CLI 来创建模块。

创建模块后，如果需要将其资源提供给应用，则必须将模块添加为依赖项。

手动创建自定义模块
您可以按照以下步骤手动创建自定义模块。

1. 在 **EAP_HOME/modules/** 目录中创建适当的目录结构。

   示例：创建 MySQL JDBC Driver Directory 结构

   ```
   $ cd EAP_HOME/modules/
   $ mkdir -p com/mysql/main
   ```

2. 将 JAR 文件复制到 **main/** 子目录中。

   示例：复制 MySQL JDBC 驱动程序 JAR

   ```
   $ cp /path/to/mysql-connector-java-8.0.12.jar EAP_HOME/modules/com/mysql/main/
   ```

3. 在 **main/** 子目录中创建 **module.xml** 文件，并在 文件中指定适当的资源和依赖项。

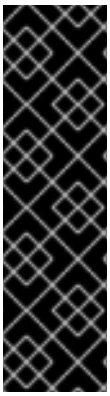
   示例：MySQL JDBC Driver module.xml 文件

   ```
   <?xml version="1.0" ?>
   <module xmlns="urn:jboss:module:1.1" name="com.mysql">
     <resources>
       <resource-root path="mysql-connector-java-8.0.12.jar"/>
     </resources>
     <dependencies>
   ```

```
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

使用管理 CLI 创建自定义模块

您可以使用模块 **add management CLI** 命令来创建自定义模块。



重要

使用 模块 管理 CLI 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 CLI 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

1. 启动 JBoss EAP 服务器。
2. 启动管理 CLI。

```
$ EAP_HOME/bin/jboss-cli.sh
```

3. 使用 模块 **add management CLI** 命令，添加新的核心模块。

```
module add --name=MODULE_NAME --resources=PATH_TO_RESOURCE --
dependencies=DEPENDENCIES
```

示例：创建一个 MySQL 模块

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

请参阅 模块命令参数 以了解可用于自定义此命令的参数，例如，使用外部模块目录提供您自己的 **module.xml** 文件，或者为模块指定备选插槽。您还可以执行 模块 **--help** 来了解有关使用此命令添加和删除模块的更多详细信息。

添加模块作为依赖性

为了使您的应用能够访问此模块的资源，您必须将模块添加为依赖项。

- 如需使用部署描述符添加特定于应用程序的依赖项，请参阅 JBoss EAP 开发指南中的将 Explicit 模块依赖性添加到部署部分。
- 有关将模块添加为依赖性到所有应用程序的说明，请参阅 Define Global Modules 部分。

例如，下列步骤添加一个 JAR 文件，该文件包含多个属性文件作为模块，再定义全局模块，以便应用随后可以加载这些属性。

1. 将 JAR 文件添加为核心模块。

```
module add --name=myprops --resources=/path/to/properties.jar
```

2. 将此模块定义为全局模块，使它可供所有部署使用。

```
/subsystem=ee:list-add(name=global-modules,value={name=myprops})
```

3. 然后，应用可以从 JAR 中包含的其中一个属性文件中检索属性。

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

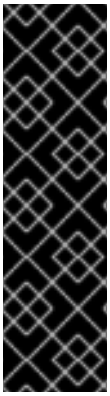
6.4. 删除自定义模块

可以手动或使用管理 CLI 删除自定义静态模块。

手动删除自定义模块
在手动删除模块之前，请确保部署的应用或服务器配置中的其他位置不需要该模块，如数据源。

若要移除自定义模块，可删除 **EAP_HOME/modules/** 下的模块目录，其中包括模块的 **module.xml** 文件以及关联的 JAR 文件或其他资源。例如，删除 **EAP_HOME/modules/com/mysql/main/** 目录，以删除主插槽中的自定义 MySQL JDBC 驱动程序模块。

使用管理 CLI 移除自定义模块
您可以使用模块删除管理 **CLI** 命令删除自定义模块。



重要

使用 模块 管理 CLI 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 CLI 时使用。在生产环境中，应当手动添加和删除模块。

技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

1. 启动 JBoss EAP 服务器。

2. 启动管理 CLI。

```
$ EAP_HOME/bin/jboss-cli.sh
```

3. 使用 模块 **remove** management CLI 命令，以删除自定义模块。

```
module remove --name=MODULE_NAME
```

- 如果要删除的模块位于 **main** 以外的插槽中，请使用 **--slot** 参数。

示例：删除 MySQL 模块

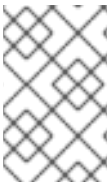
```
module remove --name=com.mysql
```

-

执行 模块 **--help** 获取关于使用此命令添加和删除模块的更多详细信息。

6.5. 定义全局模块

可以为 **JBoss EAP** 定义全局模块列表，它们将模块添加为依赖项到所有部署。



注意

您必须知道要配置为全局模块的模块名称。有关所含模块的完整列表以及是否被支持，请参阅红帽客户门户网站中的 **Red Hat JBoss Enterprise Application Platform 7** 包含的模块。有关部署中模块的命名约定，请参阅 **Dynamic** 模块命名部分。

使用以下管理 **CLI** 命令，定义全局模块列表：

```
/subsystem=ee:write-attribute(name=global-modules,value=[{name=MODULE_NAME_1},
{name=MODULE_NAME_2}]
```

使用以下管理 **CLI** 命令，将单个模块添加到现有全局模块列表中：

```
/subsystem=ee:list-add(name=global-modules,value={name=MODULE_NAME})
```

也可以使用管理控制台从 **Configuration** 选项卡导航到 **EE** 子系统并选择 **Global Modules** 部分来添加和删除全局模块。

如果您希望全局模块可由外部依赖项访问，您必须明确使其可用。以下选项可用于从外部获取全局模块中的服务：

- 在 **jboss-deployment-structure.xml** 中，将 **services="import"** 添加到模块中
- 在全局模块定义中添加 **services="true"**。

  ```
  /subsystem=ee:write-attribute(name=global-modules,value=[{name=module1,services=true}]
  ```

  或者，在添加多个模块时：

  ```
  /subsystem=ee:write-attribute(name=global-modules,value=[{name=module1,services=true},
  {name=module2,services=false}]
  ```

  将新模块添加到现有列表中：

  ```
  /subsystem=ee:list-add(name=global-modules,value={name=module1,services=true})
  ```

- 使用管理控制台定义全局模块时，请确保 **Services** 属性的值为 **On**。

6.6. 配置子部署隔离

企业存档(EAR)中的每个子部署是一个具有自己的类加载器的动态模块。**Subdeployments** 始终对父模块具有隐式依赖关系，这使得它们有权访问 **EAR/lib** 中的类。默认情况下，子部署可以访问该 **EAR** 内其他子部署的资源。

如果您不希望子部署被允许访问属于其他子部署的类，则可以在 JBoss EAP 中启用严格的子部署隔离。此设置将影响所有部署。

为所有部署启用子部署模块隔离

可以使用管理控制台或来自 **ee** 子系统的管理 CLI 来启用或禁用子部署隔离。默认情况下，子部署隔离设为 **false**，允许子部署访问 EAR 部署中其他子部署的资源。

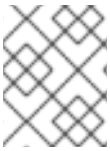
使用以下管理 CLI 启用 EAR 子部署隔离：

```
/subsystem=ee:write-attribute(name=ear-subdeployments-isolated,value=true)
```

EAR 中的子部署将不再能够从其他子部署访问资源。

6.7. 定义外部 JBOSS EAP 模块目录

JBoss EAP 模块的默认目录是 **EAP_HOME/modules**。您可以使用 **JBOSS_MODULEPATH** 变量为 JBoss EAP 模块指定不同的目录。按照以下步骤，在 JBoss EAP 启动配置文件中设置此变量。



注意

您还可以将 **JBOSS_MODULEPATH** 设置为环境变量，而不是在 JBoss EAP 启动配置文件中设置它。

1. 编辑 启动配置文件。
   - 作为单机服务器运行时，编辑 **EAP_HOME/bin/standalone.conf** 文件或 **standalone.conf.bat** (Windows Server)。
   - 在受管域中运行时，编辑 **EAP_HOME/bin/domain.conf** 文件或 **domain.conf.bat** (Windows Server)。
2. 设置 **JBOSS_MODULEPATH** 变量，例如：

   ```
   JBOSS_MODULEPATH="/path/to/modules/directory/"
   ```

   若要指定目录列表，可使用冒号 **(:)** 来分隔目录列表。

注意

对于 Windows Server，使用以下语法设置 **JBOSS_MODULEPATH** 变量：

```
set "JBOSS_MODULEPATH /path/to/modules/directory/"
```

要指定目录列表，请使用分号 **(;)** 来分隔目录列表。

6.8. 动态模块命名约定

JBoss EAP 将所有部署作为模块加载，这些模块按照以下惯例命名：

- WAR 和 JAR 文件的部署使用以下格式命名：

  ```
  deployment.DEPLOYMENT_NAME
  ```

例如，**inventory.war** 和 **store.jar** 将分别具有 **deployment.inventory.war** 和 **deployment.store.jar** 的模块名称。

- Enterprise Archive(EAR) 中的子部署使用以下格式命名：

  ```
  deployment.EAR_NAME.SUBDEPLOYMENT_NAME
  ```

  例如，在企业存档 帐户内部署 **report .war** 的子 部署将具有 模块名称 **deployment.accounts.ear.reports.war**。

第 7 章 部署应用程序

JBoss EAP 具有一系列应用部署和配置选项，可同时满足管理员和开发人员的需求。对于管理员而言，管理控制台 和管理 CLI 提供了理想的图形和命令行界面来管理生产环境中的应用部署。对于开发人员，应用部署测试选项的范围包括可配置的文件系统部署扫描程序 、HTTP API、红帽 CodeReady Studio 和 Maven 等 IDE。

在部署应用时，您可能希望通过将 **org.jboss.metadata.parser.validate** system 属性设置为 **true** 来启用对部署描述符的验证。这可以通过以下方法之一完成：

- 启动服务器时：

```
$ EAP_HOME/bin/standalone.sh -Dorg.jboss.metadata.parser.validate=true
```

- 使用以下管理 CLI 命令将其添加到服务器配置中：

```
/system-property=org.jboss.metadata.parser.validate:add(value=true)
```

7.1. 使用管理 CLI 部署应用

使用管理 CLI 部署应用可让您可以利用单一命令行界面来创建和运行部署脚本。您可以使用此脚本编写功能来配置特定的应用部署和管理场景。作为单机服务器运行时，您可以管理单个服务器的部署，或者受管域中运行的整个服务器网络。

7.1.1. 使用管理 CLI 将应用部署到单机服务器

部署应用程序
在管理 CLI 中，使用 部署 **deploy-file** 命令并指定应用部署的路径。

```
deployment deploy-file /path/to/test-application.war
```

成功部署不会向管理 CLI 生成任何输出，但服务器日志会显示部署消息。

```
WFLYSRV0027: Starting deployment of "test-application.war" (runtime-name: "test-application.war")
WFLYUT0021: Registered web context: /test-application
WFLYSRV0010: Deployed "test-application.war" (runtime-name : "test-application.war")
```

同样，您可以使用以下 部署 命令：

- 使用 部署 **deploy-cli-archive** 从 a **.cli** 存档文件中部署内容。CLI 部署存档是扩展名为 **.cli** 的 **JAR** 文件。它包含应当部署的应用存档，以及包含命令和操作的 CLI 脚本文件 **deploy .scr** 和 **undeploy**.scr。个脚本文件 **deploy.scr** 包含部署应用存档和设置环境的命令和操作；另一个脚本文件 **undeploy.scr** 包含用于取消部署应用存档并清理环境的命令。
- 使用 部署 **deploy-url** 来部署 URL 引用的内容。

您还可以使用 **deployment enable** 命令启用 禁用 的应用。

取消部署应用
在管理 CLI 中，使用 部署 **undeploy** 命令并指定部署名称。这将从存储库删除部署内容。如需在取消部署时保留部署内容，请参阅禁用应用程序。

```
deployment undeploy test-application.war
```

成功卸载不会向管理 CLI 生成任何输出，但服务器日志会显示未部署消息。

```
WFLYUT0022: Unregistered web context: /test-application
WFLYSRV0028: Stopped deployment test-application.war (runtime-name: test-application.war) in 62ms
WFLYSRV0009: Undeployed "test-application.war" (runtime-name: "test-application.war")
```

类似地，您可以使用 部署 **undeploy-cli-archive** 从 a **.cli** 存档文件中取消部署内容。您还可以使用通配符 **(*)**取消部署所有部署。

```
deployment undeploy *
```

禁用应用程序
禁用部署的应用，但不从存储库中删除部署内容。

```
deployment disable test-application.war
```

您可以使用 **deployment disable-all** 命令禁用所有部署。

```
deployment disable-all
```

启用应用程序
启用禁用的部署应用。

```
deployment enable test-application.war
```

您可以使用 **deployment enable-all** 命令启用所有部署。

```
deployment enable-all
```

列出部署
在管理 CLI 中，使用 **deployment info** 命令列出部署信息。

```
deployment info
```

输出中将显示每个部署的详细信息，如运行时名称、状态以及是否启用。

```
NAME                RUNTIME-NAME        PERSISTENT ENABLED STATUS
helloworld.war      helloworld.war      true       true    OK
test-application.war test-application.war true       true    OK
```

按名称显示部署信息：

```
deployment info helloworld.war
```

您还可以使用 **deployment list** 命令列出所有 部署。

7.1.2. 使用管理 CLI 在受管域中部署应用

部署应用程序
在管理 CLI 中，使用 部署 **deploy-file** 命令并指定应用部署的路径。您还必须指定应用应部署到的服务器组。

- 将应用部署到所有服务器组：

```
deployment deploy-file /path/to/test-application.war --all-server-groups
```

- 将应用部署到特定的服务器组：

```
deployment deploy-file /path/to/test-application.war --server-groups=main-server-group,other-server-group
```

成功部署不会向管理 CLI 生成任何输出，但服务器日志显示每个受影响的服务器的部署消息。

```
[Server:server-one] WFLYSRV0027: Starting deployment of "test-application.war" (runtime-name: "test-application.war")
[Server:server-one] WFLYUT0021: Registered web context: /test-application
[Server:server-one] WFLYSRV0010: Deployed "test-application.war" (runtime-name : "test-application.war")
```

同样，您可以使用以下 部署 命令：

- 使用 **deployment deploy-cli-archive** 命令，从 a **.cli** 存档文件中部署内容。CLI 部署存档是扩展名为 **.cli** 的 **JAR** 文件。它包含应当部署的应用存档，以及包含命令和操作的 CLI 脚本文件 **deploy.scr** 和 **undeploy**.scr。个脚本文件 **deploy.scr** 包含部署应用存档和设置环境的命令和操作；另一个脚本文件 **undeploy.scr** 包含用于取消部署应用存档并清理环境的命令。

- 使用 **deployment deploy-url** 命令来部署 URL 引用的内容。

您还可以使用 **deployment enable** 命令启用 禁用 的应用。

取消部署应用
在管理 CLI 中，使用 部署 **undeploy** 命令并指定部署名称。您还必须指定应当从中取消部署应用的服务器组。如需从特定服务器组取消部署，请参阅禁用应用程序。

利用该部署，从所有服务器组取消部署应用。

```
deployment undeploy test-application.war --all-relevant-server-groups
```

成功卸载不会向管理 CLI 生成任何输出，但服务器日志显示每个受影响的服务器的未部署消息。

```
[Server:server-one] WFLYUT0022: Unregistered web context: /test-application
[Server:server-one] WFLYSRV0028: Stopped deployment test-application.war (runtime-name: test-application.war) in 74ms
[Server:server-one] WFLYSRV0009: Undeployed "test-application.war" (runtime-name: "test-application.war")
```

类似地，您可以使用 部署 **undeploy-cli-archive** 命令从 a **.cli** 存档文件中取消部署内容。您还可以使用通配符 **(*)** 取消部署所有部署。

```
deployment undeploy * --all-relevant-server-groups
```

禁用应用程序
从特定的服务器组禁用部署的应用，并将其内容保留在存储库中，以用于使用该部署的其他服务器组。

```
deployment disable test-application.war --server-groups=other-server-group
```

您可以使用 **deployment disable-all** 命令禁用所有部署。

```
deployment disable-all --server-groups=other-server-group
```

启用应用程序
启用禁用的部署应用。

```
deployment enable test-application.war
```

您可以使用 **deployment enable-all** 命令启用所有部署。

```
deployment enable-all --server-groups=other-server-group
```

列出部署
在管理 CLI 中，使用 **deployment info** 命令列出部署信息。您可以按照部署名称或服务器组列出部署信息。

按名称显示部署信息：

```
deployment info helloworld.war
```

输出中将列出各个服务器组中的部署及其状态。

```
NAME            RUNTIME-NAME
helloworld.war  helloworld.war

SERVER-GROUP    STATE
main-server-group  enabled
other-server-group added
```

显示服务器组的部署信息：

```
deployment info --server-group=other-server-group
```

输出将列出指定服务器组的部署及其状态。

```
NAME                RUNTIME-NAME        STATE
helloworld.war      helloworld.war      added
test-application.war test-application.war enabled
```

您还可以使用 **deployment list** 命令列出域中的所有部署 。

7.2. 使用管理控制台部署应用

使用管理控制台部署应用程序可为您提供易于使用的图形界面。您可以在概览中看到哪些应用部署到您的服务器或服务器组，您可以根据需要从内容存储库中启用、禁用或删除应用。

7.2.1. 使用管理控制台将应用程序部署到单机服务器

可以从 JBoss EAP 管理控制台的 **Deployments** 选项卡查看和管理部署。

部署应用程序

单击添加(+)按钮。您可以选择通过上传部署、添加非受管部署或 创建空部署 来部署应用。部署默认是启用的。

- 上传部署
  上传将复制到服务器的内容存储库并由 JBoss EAP 管理的应用。
- 添加非受管部署
  指定部署的位置。此部署不会复制到服务器的内容存储库中，也不会由 JBoss EAP 管理。
- 创建空部署
  创建一个展开式空部署。您可以在创建后将文件添加到部署中。

取消部署应用
选择部署，再选择 Undeploy 选项。JBoss EAP 从内容存储库中移除部署。

禁用应用程序
选择部署并选择 Disable 选项来禁用应用。这会取消部署部署，但不会将它从内容存储库中移除。

替换应用程序
选择部署并选择 Replace 选项。选择部署的新版本，其名称必须与原始版本相同，然后单击 Finish。这会取消部署和删除部署的原始版本，然后部署新版本。

7.2.2. 使用管理控制台在受管域中部署应用

在 JBoss EAP 管理控制台的 Deployments 选项卡中，可以通过以下方法查看和管理部署：

- 内容存储库
  所有受管和非受管部署都列在 Content Repository 部分中。可以在此处添加部署并将其部署到服务器组。
- 服务器组
  Server Groups 部分中列出了已部署到一个或多个服务器组的部署。可以在此处启用部署并直接添加到服务器组中。

添加应用程序

1. 从内容存储库，单击添加按钮。
2. 选择上传部署或添加非受管部署 来添加应用。
3. 按照提示部署应用。
   请注意，部署必须部署到服务器组，然后才能启用。

将应用程序部署到服务器组

1. 从 Content Repository，选择一个部署，再单击 Deploy 按钮。
2. 选择应当要部署此部署的一个或多个服务器组。
3. （可选）选择 选项，以启用在选定服务器组上的部署。

从服务器组取消部署应用

1. 从服务器组中，选择相应的服务器组。
2. 选择所需的部署，再单击取消部署按钮。

也可以选择 Content Repository 中为部署的 Undeploy 按钮，一次性从多个服务器组取消部署部署。

删除应用程序

1. 如果部署仍然部署到任何服务器组，请务必取消部署该部署。
2. 从 Content Repository，选择部署并单击删除按钮。

这会从内容存储库删除部署。

禁用应用程序

1. 从服务器组中，选择相应的服务器组。
2. 选择所需的部署，再单击禁用按钮。

这会取消部署部署，但不会将它从内容存储库中移除。

替换应用程序

1. 从 Content Repository，选择部署并单击替换按钮。
2. 选择部署的新版本，其名称必须与原始版本相同，然后单击 Replace。

这会取消部署和删除部署的原始版本，然后部署新版本。

7.3. 使用部署扫描器部署应用

部署扫描器监控要部署的应用的部署目录。默认情况下，部署扫描器将每五秒扫描 **EAP_HOME/standalone/deployments/** 目录以进行更改。标记文件用于指示部署的状态，以及触发针对部署的操作，如取消部署或重新部署。

虽然建议使用管理控制台或管理 CLI 来在生产环境中应用部署，但为开发人员提供了使用部署扫描程序进行部署。这允许用户以适合快速开发周期的方式构建和测试应用程序。此外，部署扫描程序不应与其他部署方法一起使用。

部署扫描程序仅可在将 JBoss EAP 作为单机服务器运行时使用。

7.3.1. 使用 Deployment Scanner 将应用程序部署到单机服务器

部署扫描器可以配置为允许或禁止自动部署 XML、压缩和展开的内容。如果禁用自动部署，您必须手动创建标志文件来触发部署操作。如需有关可用标志文件类型及其用途的更多信息，请参阅 Deployment Scanner Marker Files 部分。

默认情况下，启用 XML 和压缩的内容的自动部署。有关为每种内容类型配置自动部署的详情，请参阅配置 Deployment Scanner。

警告

为方便开发人员，我们不建议在生产环境中使用部署扫描器进行部署。它也不应与其他部署方法一起使用。

部署应用程序
将内容复制到部署文件夹。

```
$ cp /path/to/test-application.war EAP_HOME/standalone/deployments/
```

如果启用自动部署，将自动提取此文件，并创建 a **.deployed** 标记文件。如果未启用自动部署，您将需要手动添加 a **.dodeploy** 标记文件来触发部署。

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

取消部署应用
通过删除 **.deployed** 标记文件来触发未部署。

```
$ rm EAP_HOME/standalone/deployments/test-application.war.deployed
```

如果启用了自动部署，您也可以删除 **test-application.war** 文件，该文件将触发取消部署。请注意，这不适用于展开式部署。

重新部署应用
创建 a **.dodeploy** 标记文件以启动重新部署。

```
$ touch EAP_HOME/standalone/deployments/test-application.war.dodeploy
```

7.3.2. 配置部署扫描器

部署扫描程序可以使用管理控制台或管理 CLI 进行配置。您可以配置部署扫描器的行为，如扫描间隔、部署文件夹位置和自动部署特定应用文件类型。您还可以完全禁用部署扫描程序。

有关所有可用部署扫描器属性的详情，请参阅 Deployment Scanner Attributes 部分。

使用以下管理 CLI 命令配置默认部署扫描程序：

禁用 Deployment Scanner

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
```

这将禁用 默认的 部署扫描程序。

更改扫描间隔

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-interval,value=10000)
```

这会将扫描间隔时间从 **5000** 毫秒（五秒）更新为 **10000** 毫秒（十秒）。

更改 Deployment Folder

```
/subsystem=deployment-scanner/scanner=default:write-
attribute(name=path,value=/path/to/deployments)
```

这会将部署文件夹的位置从 **EAP_HOME/standalone/deployments** 的默认位置更改为 **/path/to/deployments**。

除非指定了 **relative-to** 属性，否则路径 值 将被视为绝对路径，在这种情况下，它将相对于该路径。

启用自动部署展开的内容

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-exploded,value=true)
```

这可实现自动部署展开式内容，这在默认情况下是禁用的。

禁用 Zipped 内容的自动部署

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-zipped,value=false)
```

这会禁用默认启用的 zipped 内容的自动部署。

禁用 XML 内容的自动部署

```
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-xml,value=false)
```

这禁用了 XML 内容的自动部署，默认为启用。

7.3.3. 定义自定义 Deployment Scanner

可以使用管理 CLI 或从管理控制台的 Configuration 选项卡导航到 Deployment Scanners 子系统来添加新的部署扫描程序。这将定义用于扫描部署的新目录。默认部署扫描器监控 **EAP_HOME/standalone/deployments**。有关 配置现有部署扫描程序的详情，请参阅配置 Deployment Scanner。

以下管理 CLI 命令添加新的部署扫描器，它将每隔五秒检查 **EAP_HOME/standalone/new_deployment_dir** 以进行部署：

```
/subsystem=deployment-scanner/scanner=new-scanner:add(path=new_deployment_dir,relative-to=jboss.server.base.dir,scan-interval=5000)
```

注意

指定的目录必须已经存在，此命令将失败并显示错误。

已定义一个新的部署扫描器，并将监控指定的目录以监控部署。

7.4. 使用 MAVEN 部署应用程序

使用 Apache Maven 部署应用后，您可以将部署至 JBoss EAP 轻松整合到现有的开发工作流中。

您可以使用 Maven 使用 WildFly Maven 插件将应用部署到 JBoss EAP，它提供简单操作来部署和取消部署应用到应用服务器。

7.4.1. 使用 Maven 将应用部署到单机服务器

以下说明演示了如何使用 Maven 部署和取消部署 JBoss EAP **helloworld** 快速入门。

如需有关 JBoss EAP 快速入门的更多信息，请参阅 JBoss EAP 快速入门指南中的快速入门示例。

部署应用程序

在 Maven **pom.xml** 文件中初始化 **WildFly Maven** 插件。这应在 **JBoss EAP faststartpom.xml** 文件中配置。

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
</plugin>
```

从 **helloworld quickstart** 目录，执行以下 **Maven** 命令：

```
$ mvn clean install wildfly:deploy
```

在发出要部署的 **Maven** 命令后，终端窗口将显示以下输出表示部署成功：

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.981 s
[INFO] Finished at: 2015-12-23T15:06:13-05:00
[INFO] Final Memory: 21M/231M
[INFO] ------------------------------------------------------------------------
```

也可以通过查看活动服务器实例的服务器日志来确认部署。

```
WFLYSRV0027: Starting deployment of "helloworld.war" (runtime-name: "helloworld.war")
WFLYUT0021: Registered web context: /helloworld
WFLYSRV0010: Deployed "helloworld.war" (runtime-name : "helloworld.war")
```

取消部署应用
从 **helloworld quickstart** 目录，执行以下 **Maven** 命令：

```
$ mvn wildfly:undeploy
```

在发出 **Maven** 命令取消部署后，终端窗口将显示以下输出，指示成功取消部署：

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.237 s
[INFO] Finished at: 2015-12-23T15:09:10-05:00
[INFO] Final Memory: 10M/183M
[INFO] ------------------------------------------------------------------------
```

也可以通过查看活动服务器实例的服务器日志来确认取消部署。

```
WFLYUT0022: Unregistered web context: /helloworld
WFLYSRV0028: Stopped deployment helloworld.war (runtime-name: helloworld.war) in 27ms
WFLYSRV0009: Undeployed "helloworld.war" (runtime-name: "helloworld.war")
```

7.4.2. 使用 Maven 在受管域中部署应用程序

以下说明演示了如何使用 **Maven** 在受管域中部署和取消部署 **JBoss EAP helloworld** 快速启动。

如需有关 JBoss EAP 快速入门的更多信息，请参阅 JBoss EAP 快速入门指南中的快速入门示例。

部署应用程序
在受管域中部署应用时，您必须指定应当要将应用部署到的服务器组。这在 Maven **pom.xml** 文件中配置。

pom.xml 中的以下配置初始化 WildFly Maven 插件，并将**main-server-group** 指定为应用应部署到的服务器组。

```
<plugin>
  <groupId>org.wildfly.plugins</groupId>
  <artifactId>wildfly-maven-plugin</artifactId>
  <version>${version.wildfly.maven.plugin}</version>
  <configuration>
    <domain>
      <server-groups>
        <server-group>main-server-group</server-group>
      </server-groups>
    </domain>
  </configuration>
</plugin>
```

从 **helloworld** quickstart 目录，执行以下 Maven 命令：

```
$ mvn clean install wildfly:deploy
```

在发出要部署的 Maven 命令后，终端窗口将显示以下输出表示部署成功：

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.005 s
[INFO] Finished at: 2016-09-02T14:36:17-04:00
[INFO] Final Memory: 21M/226M
[INFO] ------------------------------------------------------------------------
```

也可以通过查看活动服务器实例的服务器日志来确认部署。

```
WFLYSRV0027: Starting deployment of "helloworld.war" (runtime-name: "helloworld.war")
WFLYUT0021: Registered web context: /helloworld
WFLYSRV0010: Deployed "helloworld.war" (runtime-name : "helloworld.war")
```

取消部署应用
从 **helloworld** quickstart 目录，执行以下 Maven 命令：

```
$ mvn wildfly:undeploy
```

在发出 Maven 命令取消部署后，终端窗口将显示以下输出，指示成功取消部署：

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.750 s
```

```
[INFO] Finished at: 2016-09-02T14:45:10-04:00
[INFO] Final Memory: 10M/184M
[INFO] ------------------------------------------------------------------------
```

也可以通过查看活动服务器实例的服务器日志来确认取消部署。

```
WFLYUT0022: Unregistered web context: /helloworld
WFLYSRV0028: Stopped deployment helloworld.war (runtime-name: helloworld.war) in 106ms
WFLYSRV0009: Undeployed "helloworld.war" (runtime-name: "helloworld.war")
```

7.5. 使用 HTTP API 部署应用程序

可以通过 **curl** 命令使用 **HTTP API** 将应用部署到 **JBoss EAP**。有关使用 **HTTP API** 的更多信息，请参阅 **HTTP API** 部分。

7.5.1. 使用 HTTP API 将应用程序部署到单机服务器

默认情况下，**HTTP API** 可通过 **http://HOST: PORT/management** 访问，例如 **http://localhost:9990/management**。

部署应用程序

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type: application/json" -u
USER:PASSWORD -d '{"operation" : "composite", "address" : [], "steps" : [{"operation" : "add",
"address" : {"deployment" : "test-application.war"}, "content" : [{"url" : "file:/path/to/test-
application.war"}]},{"operation" : "deploy", "address" : {"deployment" : "test-
application.war"}}],"json.pretty":1}'
```

取消部署应用

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type: application/json" -u
USER:PASSWORD -d '{"operation" : "composite", "address" : [], "steps" : [{"operation" : "undeploy",
"address" : {"deployment" : "test-application.war"}},{"operation" : "remove", "address" : {"deployment" :
"test-application.war"}}],"json.pretty":1}'
```

请参阅本红帽知识库文章，以编程方式生成 **JSON** 请求。

7.5.2. 使用 HTTP API 在受管域中部署应用程序

默认情况下，**HTTP API** 可通过 **http://HOST: PORT/management** 访问，例如 **http://localhost:9990/management**。

部署应用程序

1. 将部署添加到内容存储库。

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation" : "add", "address" : {"deployment" :
"test-application.war"}, "content" : [{"url" : "file:/path/to/test-application.war"}],"json.pretty":1}'
```

2. 将部署添加到所需的服务器组。

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation" : "add", "address" : {"server-group" :
"main-server-group","deployment":"test-application.war"},"json.pretty":1}'
```

3. 将应用部署到服务器组。

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation" : "deploy", "address" : {"server-
group" : "main-server-group","deployment":"test-application.war"},"json.pretty":1}'
```

取消部署应用

1. 从分配至的所有服务器组移除部署。

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation" : "remove", "address" : {"server-
group" : "main-server-group","deployment":"test-application.war"},"json.pretty":1}'
```

2. 从内容存储库删除部署。

```
$ curl --digest -L -D - http://HOST:PORT/management --header "Content-Type:
application/json" -u USER:PASSWORD -d '{"operation" : "remove", "address" : {"deployment"
: "test-application.war"}, "json.pretty":1}'
```

7.6. 自定义部署行为

7.6.1. 为部署内容定义自定义目录

您可以为 **JBoss EAP** 定义自定义位置，以存储已部署的内容。

为单机服务器定义自定义目录
默认情况下，单机服务器部署的内容存储在 **EAP_HOME/standalone/data/content** 目录中。可以通过在启动服务器时传递 **-Djboss.server.deploy.dir** 参数来更改此位置。

```
$ EAP_HOME/bin/standalone.sh -Djboss.server.deploy.dir=/path/to/new_deployed_content
```

在 **JBoss EAP** 实例中，所选位置应当是唯一的。

注意

jboss.server.deploy.dir 属性指定用于存储已使用管理控制台或管理 **CLI** 部署的内容的目录。要定义部署扫描程序要监控的自定义部署目录，请参阅 *配置 Deployment Scanner*。

为受管域定义自定义目录
默认情况下，受管域部署的内容存储在 **EAP_HOME/domain/data/content** 目录中。可以通过在启动域时传递 **-Djboss.domain.deployment.dir** 参数来更改此位置。

```
$ EAP_HOME/bin/domain.sh -Djboss.domain.deployment.dir=/path/to/new_deployed_content
```

在 **JBoss EAP** 实例中，所选位置应当是唯一的。

7.6.2. 控制部署顺序

JBoss EAP 在服务器启动时提供精细的控制部署顺序。可以指定多个 EAR 文件中应用的部署顺序以及重启后顺序的持久性。

您可以使用 **jboss-all.xml** 部署描述符来声明顶级部署之间的依赖关系。

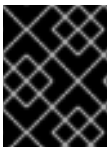
例如，如果您的 **app.ear** 依赖于 Framework **.ear**，则您可以创建一个 **app.ear/META-INF/jboss-all.xml** 文件，如下所示。

```
<jboss xmlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>
```

注意

您可以将部署的运行时名称用作 **jboss-all.xml** 文件中的依赖项名称。

这可确保在 **app.ear** 之前部署 **Framework.ear**。



重要

如果您在 **app.ear** 中创建 **jboss-all.xml** 文件，并且您没有部署 框架 **.ear**，则服务器将尝试部署 **app.ear** 和 失败。

7.6.3. 覆盖部署内容

部署覆盖可用于将内容覆盖到现有的部署中，而无需物理修改部署存档的内容。它允许您在运行时覆盖部署描述符、库 JAR 文件、类、JSP 页面和其他文件，而无需重新构建存档。

如果您需要为需要不同配置或设置的不同环境调整部署，这将非常有用。例如，当将部署从开发到测试、到暂存再移到生产环境时，您可能需要交换部署描述符，修改静态 Web 资源以更改应用的品牌，或者根据目标环境将 JAR 库替换为不同的版本。对于需要更改配置但因为策略或安全限制而无法修改或破解归档的安装，此功能也非常有用。

在定义部署覆盖时，您将指定将替换部署存档中的 文件的文件系统中的 文件。您还必须指定哪些部署应受到部署覆盖的影响。必须重新部署任何受影响的部署，才能使更改生效。

参数

您可以使用以下任一参数来配置部署覆盖：

- 名称 ：部署覆盖的名称。
- 内容:: 以逗号分隔的列表，将文件系统上的 文件映射到所替换的存档中的 文件。每个条目的格式为 **ARCHIVE_PATH=FILESYSTEM_PATH**。
- 部署 ：此覆盖所链接到的部署组合。
- 重新部署受影响:: 重新部署所有受影响的部署。

有关完整使用详情，请执行 **deployment-overlay --help**。

流程

1. 使用 **deployment-overlay add** management CLI 命令添加部署覆盖：

```
deployment-overlay add --name=new-deployment-overlay --content=WEB-
INF/web.xml=/path/to/other/web.xml --deployments=test-application.war --redeploy-affected
```

注意

在受管域中，使用 **--server-groups** 指定适用的服务器组，或者通过 **--all-server-groups** 指定所有服务器组。

2. 创建部署覆盖后，您可以将内容添加到现有覆盖中，将覆盖链接到部署，或删除覆盖。

3. 可选：您可以在 **<overlay>** 元素中指定一个覆盖配置来链接包含静态 Web 资源（如 HTML、镜像或视频）的外部目录。**<overlay>** 元素位于应用程序 **jboss-web.xml** 文件中。使用这种配置时，您不需要重新打包应用程序。
以下示例显示了 **<overlay>** 元素中的系统属性替换，其中 **${example.path.to.overlay}** 定义 /PATH/TO/STATIC/WEB/CONTENT 位置。

示例： jboss-web.xml 文件中的 <overlay> 元素

```
<jboss-web>
      <overlay>${example.path.to.overlay}</overlay>
</jboss-web>
```

如果 **jboss-descriptor-property-replacement** 设为 **true**，则您可以在 **<overlay>** 元素 中指定系统属性，这是 descriptor 属性的默认值。

要配置 **jboss-descriptor-property-replacement**，请使用以下命令：

```
/subsystem=ee:write-attribute(name=jboss-descriptor-property-replacement,value=true)
```

此命令在 JBoss EAP 配置的 **ee** 子系统中添加以下 XML 内容：

```
<subsystem xmlns="urn:jboss:domain:ee:4.0">
    <jboss-descriptor-property-replacement>true</jboss-descriptor-property-replacement>
</subsystem>
```

7.6.4. 使用 Rollout Plans

关于 Rollout 计划
在受管域中，针对域或主机资源的操作可能会影响多个服务器。此类操作可能包括详细介绍操作应用到服务器的顺序的推出计划，以及详细描述是否可以在部分服务器上成功执行该操作的策略。如果没有指定推出计划，则会使用默认的推出部署计划。

以下是涉及五个服务器组的推出计划示例：操作可以按序列、序列 或 并发组应用到服务器组。Rollout Plan Syntax 中对语法进行了更为详细的说明。

```
{"my-rollout-plan" => {"rollout-plan" => {
    "in-series" => [
        {"concurrent-groups" => {
            "group-A" => {
```

```
            "max-failure-percentage" => "20",
            "rolling-to-servers" => "true"
        },
        "group-B" => undefined
    }},
    {"server-group" => {"group-C" => {
        "rolling-to-servers" => "false",
        "max-failed-servers" => "1"
    }}},
    {"concurrent-groups" => {
        "group-D" => {
            "max-failure-percentage" => "20",
            "rolling-to-servers" => "true"
        },
        "group-E" => undefined
    }}
  ],
  "rollback-across-groups" => "true"
}}}
```

查看上面的示例，将操作应用到域中的服务器时要分三个阶段完成。如果任何服务器组的策略触发在服务器组中回滚该操作，则所有其他服务器组也将回滚。

1. 服务器组 **group-A** 和 **group-B** 将同时应用操作。该操作将连续应用到 **group-A** 中的服务器，而 **group-B** 中的所有服务器将同时处理该操作。如果 **group-A** 中超过 20% 的服务器未能应用该操作，它将在整个组中回滚。如果 **group-B** 中的任何服务器都无法应用该操作，它将在该组间回滚。

2. 完成 **group-A** 和 **group-B** 中的所有服务器后，该操作将应用到 **group-C** 中的服务器。这些服务器将同时处理操作。如果 **group-C** 中的多个服务器无法应用该操作，它将在该组中回滚。

3. 完成 **group-C** 中的所有服务器后，服务器组 **group-D** 和 **group-E** 将同时应用该操作。该操作将按顺序应用到 **group-D** 中的服务器，而 **group-E** 中的所有服务器将同时处理该操作。如果 **group-D** 中超过 20% 的服务器无法应用该操作，它将在整个组中回滚。如果 **group-E** 中的任何服务器都无法应用此操作，它将在该组间回滚。

rollout Plan Syntax
您可以通过以下任一方式指定推出部署计划。

- 在 **deploy** 命令操作标头中定义推出部署计划。详情请参阅使用 **Rollout Plan** 部署。
- 使用 **rollout-plan** 命令存储推出计划，然后在 **deploy** 命令操作标头中引用计划名称。详情请参阅使用 **Stored Rollout** 计划部署。

虽然每种方法都有不同的初始命令，但这两种方法都使用 **rollout** 操作标头来定义推出计划。这使用以下语法：

```
rollout (id=PLAN_NAME | SERVER_GROUP_LIST) [rollback-across-groups]
```

- ***PLAN_NAME*** 是使用 **rollout-plan** 命令存储的推出计划的名称。
- ***SERVER_GROUP_LIST*** 是服务器组的列表。使用逗号分隔(**,**)来分隔多个服务器组，以指示应当按顺序对每一服务器组执行操作。使用脱字符(**^**)分隔符表示应同时对每个服务器组执行操作。
  - 对于每一服务器组，请在括号中设置以下任何策略：使用逗号分隔多个策略。

- **Rolling-to-servers** ：一个布尔值，如果设为 **true**，则按顺序将操作应用到组中的每一服务器。如果值为 **false** 或未指定，操作将同时应用到组中的服务器。
- **max-failed-servers** ：整数，它取组中的最大服务器数，无法应用该操作，然后再将它恢复到组中的所有服务器上。如果没有指定，则默认值为 **0**，表示任何服务器上的失败都会在组中触发回滚。
- **max-failure-percentage** ：0 到 **100** 之间的整数，表示组中服务器总数的最大百分比，在应在该组中的所有服务器上恢复操作之前无法应用该操作。如果没有指定，则默认值为 **0**，表示任何服务器上的失败都会在组中触发回滚。

注意

如果 **max-failed-servers** 和 **max-failure-percentage** 都被设置为非零值，则 **max-failure-percentage** 将具有优先权。

- **rollback-acros-groups** ：一个布尔值，指示一个服务器组中的所有服务器上是否需要回滚操作。默认值为 **false**。

使用 **Rollout** 计划部署

您可以通过将推出(**rollout**) 设置 传递到 **headers** 参数，将部署计划的完整详情直接提供给 **deploy** 命令。有关格式的更多信息，请参阅 **Rollout Plan Syntax**。

以下管理 **CLI** 命令利用为串行部署指定 **rolling -to-servers=true** 的部署计划将应用部署到 **main-server-group** 服务器组：

```
deploy /path/to/test-application.war --server-groups=main-server-group --headers={rollout main-server-group(rolling-to-servers=true)}
```

使用 **Stored Rollout** 计划部署

由于推出计划可能比较复杂，您可以选择存储推出计划的详细信息。这可让您在您要使用部署计划名称时引用推出计划的名称，而不必每次都需要部署计划的完整详情。

1. 使用 **rollout-plan** 管理 **CLI** 命令来存储推出计划。有关格式的更多信息，请参阅 **Rollout Plan Syntax**。

```
rollout-plan add --name=my-rollout-plan --content={rollout main-server-group(rolling-to-servers=false,max-failed-servers=1),other-server-group(rolling-to-servers=true,max-failure-percentage=20) rollback-across-groups=true}
```

这会创建以下部署计划：

```
"rollout-plan" => {
    "in-series" => [
        {"server-group" => {"main-server-group" => {
            "rolling-to-servers" => false,
            "max-failed-servers" => 1
        }}},
        {"server-group" => {"other-server-group" => {
            "rolling-to-servers" => true,
            "max-failure-percentage" => 20
        }}}
    ],
    "rollback-across-groups" => true
}
```

2. 在部署应用时指定存储的 **rollout** 计划名称。

以下管理 **CLI** 命令使用 **my-rollout-plan** 存储的推出计划将应用部署到所有服务器组：

```
deploy /path/to/test-application.war --all-server-groups --headers={rollout id=my-rollout-plan}
```

删除 **Stored Rollout Plan**

您可以通过指定要删除的 **rollout-plan** 管理 **CLI** 命令，移除存储的推出部署计划。

```
rollout-plan remove --name=my-rollout-plan
```

默认 **Rollout Plan**

所有影响多个服务器的操作都将通过推出计划来执行。如果没有在操作请求中指定推出部署计划，则会生成默认的推出部署计划。计划将具有以下特征：

- 只有一个高级别阶段。受操作影响的所有服务器组将同时应用操作。
- 在每个服务器组中，操作将同时应用到所有服务器。
- 如果服务器组中的任何服务器上发生故障，则会导致在该组内回滚。
- 任何服务器组的故障将导致所有其他服务器组回滚。

7.7. 管理扩展部署

在 **JBoss EAP 7.1** 之前，您只能通过操作文件系统中的文件来管理爆炸式部署。从 **JBoss EAP 7.1** 开始，您可以使用管理界面管理爆炸式部署。这可让您更改展开式应用的内容，而无需部署新版本的应用。

注意

对部署中的静态文件（如 **JavaScript** 和 **CSS** 文件）的更新会立即生效。更改其他文件（如 **Java** 类）可能需要重新部署应用才能使更改生效。

您可以以空部署开始，或者展开现有的存档部署，然后添加或删除内容。

请参阅查看部署内容，以浏览部署中的文件或读取文件的内容。

创建 Empty Exploded Deployment

您可以创建一个展开式部署，稍后根据需要添加内容。使用以下管理 **CLI** 命令，创建一个空的展开式部署：

```
/deployment=DEPLOYMENT_NAME.war:add(content=[{empty=true}])
```

需要 **empty=true** 选项来确认您是否打算创建空部署。

展开现有归档部署

您可以展开现有的存档部署，以便能更新其内容。请注意，必须先禁用部署，然后才能展开部署。使用以下管理 **CLI** 命令，展开部署：

```
/deployment=ARCHIVE_DEPLOYMENT_NAME.ear:explode
```

现在，您可以添加或删除此部署中的内容。

注意

您也可以从管理控制台展开现有的存档部署。从 **Deployments** 选项卡中，选择部署并选择 **Explode** 下拉菜单。

将内容添加到扩展的部署中

若要向部署添加内容，可使用 **add-content** 管理 **CLI** 操作。提供添加内容的部署中的位置的路径，并提供要上传的内容。要上传的内容可以作为本地文件流、**URL**、**JBoss EAP** 内容存储库中已存在的内容哈希或内容字节数组提供。以下管理 **CLI** 命令使用 **input-stream-index** 选项将本地文件的内容上传到部署：

```
/deployment=DEPLOYMENT_NAME.war:add-content(content=[{target-
path=/path/to/FILE_IN_DEPLOYMENT, input-stream-index=/path/to/LOCAL_FILE_TO_UPLOAD}]
```

注意

使用 **add-content** 操作将内容添加到部署中时，默认情况下会覆盖部署中的内容。您可以通过将 **override** 选项设置为 **false** 来更改此行为。

从展开的部署中删除内容

若要从部署中删除内容，可使用 **remove-content** 管理 **CLI** 操作，并提供要移除的部署中内容的路径。

```
/deployment=DEPLOYMENT_NAME.war:remove-content(paths=[/path/to/FILE_1, /path/to/FILE_2])
```

7.8. 查看部署内容

您可以使用 **JBoss EAP** 管理界面浏览受管部署中文件的信息，并读取文件的内容。

7.8.1. 浏览部署中的文件

使用 **browse-content** 操作来查看受管部署中的文件和目录。不提供任何参数来返回整个部署结构，或使用 **path** 参数提供特定目录的路径。

注意

您还可以从管理控制台浏览部署内容，导航到 **Deployments** 选项卡，选择部署，然后从下拉菜单中选择 **View**。

```
/deployment=helloworld.war:browse-content(path=META-INF/)
```

这将显示 **helloworld.war** 部署的 **META-INF/** 目录中的文件和目录。

```
{
    "outcome" => "success",
    "result" => [
        {
            "path" => "MANIFEST.MF",
            "directory" => false,
            "file-size" => 827L
        },
        {
            "path" => "maven/org.jboss.eap.quickstarts/helloworld/pom.properties",
            "directory" => false,
            "file-size" => 106L
        },
        {
            "path" => "maven/org.jboss.eap.quickstarts/helloworld/pom.xml",
            "directory" => false,
            "file-size" => 2713L
        },
        {
            "path" => "maven/org.jboss.eap.quickstarts/helloworld/",
            "directory" => true
        },
        {
            "path" => "maven/org.jboss.eap.quickstarts/",
            "directory" => true
        },
        {
            "path" => "maven/",
            "directory" => true
        },
        {
            "path" => "INDEX.LIST",
            "directory" => false,
            "file-size" => 251L
        }
    ]
}
```

*您还可以为 **browse-content** 操作指定以下参数：*

Archive

是否仅返回存档文件.

深度

指定要返回的文件深度.

7.8.2. *读取部署内容*

您可以使用 **read-content** 操作读取受管部署中文件的内容。不提供任何参数来返回整个部署，或使用 **path** 参数提供特定文件的路径。例如：

```
/deployment=helloworld.war:read-content(path=META-INF/MANIFEST.MF)
```

这会返回一个文件流，它可显示在管理 **CLI** 中或保存到文件系统中。

```
{
    "outcome" => "success",
    "result" => {"uuid" => "24ba8e06-21bd-4505-b4d4-bdfb16451b95"},
    "response-headers" => {"attached-streams" => [{
        "uuid" => "24ba8e06-21bd-4505-b4d4-bdfb16451b95",
        "mime-type" => "text/plain"
    }]}
}
```

7.8.2.1. 显示文件的内容

使用 附加 **display** 命令读取 **MANIFEST.MF** 文件的内容。

```
attachment display --operation=/deployment=helloworld.war:read-content(path=META-
INF/MANIFEST.MF)
```

这会将 **MANIFEST.MF** 文件的内容从 **helloworld.war** 部署 显示到管理 **CLI**。

```
ATTACHMENT 8af87836-2abd-423a-8e44-e731cc57bd80:
Manifest-Version: 1.0
Implementation-Title: Quickstart: helloworld
Implementation-Version: 7.3.0.GA
Java-Version: 1.8.0_131
Built-By: username
Scm-Connection: scm:git:git@github.com:jboss/jboss-parent-pom.git/quic
 kstart-parent/helloworld
Specification-Vendor: JBoss by Red Hat
...
```

7.8.2.2. 保存文件的内容

使用 附加 **save** 命令将 **MANIFEST.MF** 文件的内容保存到文件系统中。

```
attachment save --operation=/deployment=helloworld.war:read-content(path=META-
INF/MANIFEST.MF) --file=/path/to/MANIFEST.MF
```

*这会将 **helloworld .war** 中的 **MANIFEST.MF** 文件保存在 路径/到 /**MANIFEST.MF** 的路径/到 文件系统上。如果不使用 **--file** 参数指定文件路径，该文件将使用其唯一附加 **ID** 进行命名，并保存在管理 **CLI** 的工作目录中，默认为 **EAP_HOME/bin/**。*

第 8 章 域管理

本节讨论特定于受管域工作模式的概念和配置。

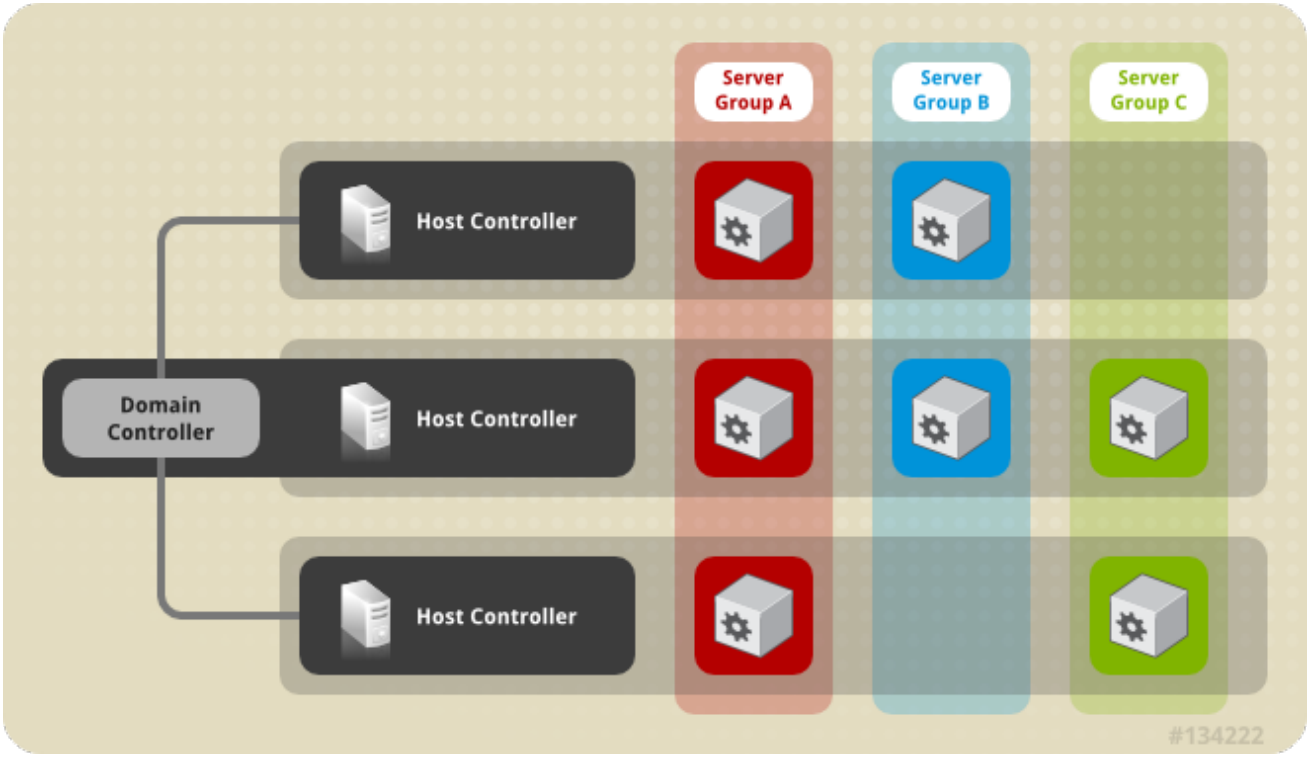
有关保护受管域的信息，请参阅 **JBoss EAP** 如何配置服务器安全性的安全受管域部分。

8.1. 关于受管域

受管域工作模式允许从一个控制点管理多个 **JBoss EAP** 实例。

集中管理的 **JBoss EAP** 服务器集合称为域的成员。域中的所有 **JBoss EAP** 实例共享共同的管理策略。

域包含一个域控制器、一个或多个主机控制器，以及每个主机的零个或多个服务器组。



域控制器是控制域的核心点。它确保每台服务器按照域的管理策略进行配置。域控制器也是主机控制器。

主机控制器是一种物理或虚拟主机，与域控制器交互，以控制其主机上运行的应用服务器实例的生命周期并协助域控制器管理它们。每一主机可以包含多个服务器组。

服务器组是一组 服务器实例，它们上已安装了 **JBoss EAP**，并且可以作为一个服务器实例进行管理和配置。域控制器管理部署到服务器组的 和 应用的配置。因此，服务器组中的每个服务器共享相同的配置和部署。

主机控制器与特定的物理或虚拟机绑定。如果您使用不同的配置，则可以在同一硬件上运行多个主机控制器，确保它们的端口和其他资源不会冲突。域控制器、单一主机控制器和多个服务器可能在同一物理系统上在同一 **JBoss EAP** 实例内运行。

8.1.1. 关于域控制器

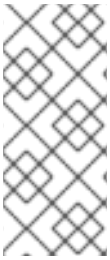
域控制器是 **JBoss EAP** 服务器实例，充当域的中央管理点。个主机控制器实例配置为充当域控制器。

域控制器的主要职责是：

- 维护域的中央管理策略.
- 确保所有主机控制器都已知晓其当前内容。
- 协助主机控制器确保所有正在运行的 **JBoss EAP** 服务器实例都按照此策略进行配置。

默认情况下，中央管理策略存储在 **EAP_HOME/domain/configuration/domain.xml** 文件中。主机控制器的此目录中需要此文件，此目录设置为作为域控制器运行。

domain.xml 文件包含可供域中服务器使用的配置集配置。配置文件包含该配置文件中提供的各种子系统的详细设置。域配置还包括套接字组的定义和服务器组定义。



注意

只要主机和服务器正在运行 **JBoss EAP 6.2** 或更高版本，**JBoss EAP 7** 域控制器就可以管理 **JBoss EAP 6** 主机和服务器。如需更多信息，请参阅将 **JBoss EAP 7.x** 域控制器配置为管理 **JBoss EAP 6** 实例。

如需更多信息，请参阅启动受管域和 域控制器配置部分。

8.1.2. 关于主机控制器

主机控制器的主要职责是服务器管理。它委派域管理任务，并负责启动和停止在其主机上运行的单个应用服务器进程。

它与域控制器交互，以帮助管理服务器和域控制器之间的通信。个域的多个主机控制器只能与一个域控制器交互。因此，在单一域模式中运行的所有主机控制器和服务器实例都有一个域控制器，必须属于同一域。

默认情况下，每一主机控制器从其主机文件系统上解压缩的 **JBoss EAP** 安装文件中的 **EAP_HOME/domain/configuration/host.xml** 文件中读取其配置。**host.xml** 文件包含以下特定于特定主机的配置信息：

- 用于从此安装运行服务器实例的名称。
- 特定于本地物理安装的配置。例如，**domain .xml** 中声明的 命名接口定义可映射到 **host.xml** 中实际的机器特定 **IP** 地址。和 **domain.xml** 中的抽象路径名称可映射到 **host.xml** 中的实际文件系统路径。
- 以下任何配置：
  - 主机控制器如何联系域控制器注册自己并访问域配置。
  - 如何查找和联系远程域控制器。
  - 主机控制器是否充当域控制器

如需更多信息，请参阅 **Start a Managed Domain** 和 **Host Controller Configuration** 部分。

8.1.3. 关于进程控制器

进程控制器是一个轻量级小进程，负责生成主机控制器进程并监控其生命周期。如果主机控制器崩溃，进程控制器将重启它。它还会按照主机控制器的指示启动服务器进程；但是，它不会自动重新启动崩溃的服务器进程。

进程控制器将日志记录到 **EAP_HOME/domain/log/process-controller.log** 文件。您可以使用 **PROCESS_ CONTROLLER_JAVA _OPTS** 变量，在 **EAP_HOME /bin/domain.conf** 文件中为进程控制器设置 **JVM** 选项。

8.1.4. 关于服务器组

服务器组是服务器实例的集合，可以作为一个服务器实例进行管理和配置。在受管域中，每个应用服务器实例属于服务器组，即使它是唯一的成员。个组中的服务器实例共享相同的配置文件配置和部署的内容。

域控制器和主机控制器在其域中每一服务器组的所有服务器实例上强制执行标准配置。

个域可以包含多个服务器组。不同的服务器组可以配置不同的配置文件和部署。例如，一个域可以配置不同的服务器层，提供不同的服务。

不同的服务器组也可以具有相同的配置文件和部署。例如，这可以实现滚动应用升级，从而在一个服务器组中升级应用，然后在第二个服务器组中更新，从而避免彻底的服务中断。

如需更多信息，请参阅 *配置服务器组部分*。

8.1.5. 关于服务器

服务器代表应用服务器实例。在受管域中，所有服务器实例都是服务器组的成员。主机控制器在其自己的 **JVM** 进程中启动每个服务器实例。

如需更多信息，请参阅 *配置服务器部分*。

8.2. 浏览域配置

JBoss EAP 提供可扩展的管理接口，以支持小型和大型受管域。

管理控制台

JBoss EAP 管理控制台提供了域的图形视图，可让您轻松管理域的主机、服务器、部署和配置文件。

Configuration

从 **Configuration** 选项卡，您可以配置域中使用的每个配置文件的子系统。您域中的不同服务器组可能会根据所需的功能使用不同的配置文件。

选择所需配置文件后，将列出该配置文件的所有可用子系统。有关配置配置文件的更多信息，请参阅管理 **JBoss EAP** 配置文件。

Runtime

从 **Runtime** 选项卡中，您可以管理服务器和服务器组以及主机配置。您可以按主机或服务器组浏览域。

从主机，您可以配置主机属性和 **JVM** 设置，并为该主机添加和配置服务器。

从服务器组中，您可以添加新的服务器组，配置属性和 **JVM** 设置，并为该服务器组添加和配置服务器。您可以执行各种操作，如启动、停止、暂停和重新加载选定服务器组中的所有服务器。

从主机或 服务器组中，您可以添加新的服务器并配置服务器属性和 **JVM** 设置。您可以执行各种操作，如启动、停止、暂停和重新加载选定的服务器。您还可以查看运行时信息，如 **JVM** 使用量、服务器日志和特定于子系统的信息。

从 **Topology**，您可以看到一个概述并查看域中主机、服务器组和服务器的详细信息。您可以对它们各自执行操作，例如重新加载或暂停。

部署

从 **Deployments** 选项卡中，您可以添加部署并部署到服务器组。您可以查看内容存储库中的所有部署，或查看部署到特定服务器组的部署。

有关使用管理控制台部署应用的更多信息，请参阅在受管域中部署应用程序

管理 CLI

JBoss EAP 管理 **CLI** 提供了一个命令行界面来管理域的主机、服务器、部署和配置文件。

且选择了适当的配置文件，即可访问子系统配置。

```
/profile=PROFILE_NAME/subsystem=SUBSYSTEM_NAME:read-resource(recursive=true)
```

注意

本指南中的说明和示例可能包含用于作为单机服务器运行时应用的子系统配置的管理 **CLI** 命令，例如：

```
/subsystem=datasources/data-source=ExampleDS:read-resource
```

要将这些管理 **CLI** 命令调整为在受管域中运行，您必须指定要配置的适当配置集，例如：

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-resource
```

指定适当的主机后，您可以配置主机设置并在该主机上的服务器上执行操作。

```
/host=HOST_NAME/server=SERVER_NAME:read-resource
```

指定适当的主机后，您可以为该主机配置服务器。

```
/host=HOST_NAME/server-config=SERVER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

指定适当的服务器组后，您可以配置服务器组设置，并对选定服务器组中的所有服务器执行操作。

```
/server-group=SERVER_GROUP_NAME:read-resource
```

您可以使用部署 管理 **CLI** 命令和指定适当的服务器组，在受管域中部署应用。具体步骤，请参阅在受管域中部署应用程序。

8.3. 启动受管域

8.3.1. 启动受管域

可以使用 **JBoss EAP** 提供的 **domain .sh** 或 **domain. bat** 脚本启动域控制器和主机控制器。有关所有可用启动脚本参数及其目的的完整列表，请使用 **--help** 参数或查看 服务器运行时参数 部分。

域控制器必须在域中任何服务器组的任何从属服务器之前启动。首先启动域控制器，然后在域中启动任何其他关联的主机控制器。

启动域控制器

使用 **host-master.xml** 配置文件启动域控制器，它已针对专用域控制器进行了预配置。

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml
```

根据域设置，您需要进行额外的配置，以允许主机控制器进行连接。另请参阅以下域设置示例：

- 在单一机器中设置受管域
- 在两个机器中设置受管域

启动主机控制器

使用 **host-slave.xml** 配置文件启动主机控制器，它已为从属主机控制器进行了预配置。

```
$ EAP_HOME/bin/domain.sh  --host-config=host-slave.xml
```

根据域设置，您将需要连接额外的配置，而不与域控制器冲突。另请参阅以下域设置示例：

- 在单一机器中设置受管域
- 在两个机器中设置受管域

8.3.2. 域控制器配置

您必须将域中的一个主机配置为域控制器。

重要

在使用 **RPM** 安装方法安装 **JBoss EAP** 时，不支持在同一计算机上配置多个域控制器或主机控制器。

通过在 **< domain-controller >** 声明中添加 **<local/ >** 元素，将主机配置为域控制器。**<domain-controller>** 应该不包含任何其他内容。

```
<domain-controller>
  <local/>
</domain-controller>
```

充当域控制器的主机必须公开一个可供域中其他主机访问的管理接口。**HTTP** 接口是标准的管理接口。

```
<management-interfaces>
  <http-interface security-realm="ManagementRealm" http-upgrade-enabled="true">
    <socket interface="management" port="${jboss.management.http.port:9990}"/>
  </http-interface>
</management-interfaces>
```

示例最小域控制器配置文件 **EAP_HOME/domain/configuration/host-master.xml** 包含这些配置设置。

8.3.3. 主机控制器配置

主机控制器必须配置为连接到域控制器，以便主机控制器能够将自身注册到域。

重要

在使用 **RPM** 安装方法安装 **JBoss EAP** 时，不支持在同一计算机上配置多个域控制器或主机控制器。

使用配置的 **<domain-controller>** 元素 来配置与域控制器的连接。

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9990}"/>
```

```
    </discovery-options>
  </remote>
</domain-controller>
```

示例最小主机控制器配置文件 **EAP_HOME/domain/configuration/host-slave.xml** 包含用于连接域控制器的配置设置。该配置假定您在启动主机控制器 时提供 **jboss.domain.master.address** 属性。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Djboss.domain.master.address=IP_ADDRESS
```

有关域控制器发现的更多信息，请参阅域控制器发现和故障转移部分。

根据您的域设置，您可能还需要为主机控制器提供身份验证，以通过域控制器进行身份验证。如需了解有关生成具有 **secret** 值的管理用户以及使用该值更新主机控制器配置的详细信息，请参阅在两个机器中设置受管域。

8.3.4. 配置主机的名称

受管域中运行的每个主机必须具有唯一的主机名。为简化管理并允许在多个主机上使用相同的主机配置文件，服务器在确定主机名时要使用以下优先级：

1. 如果设置，**host .xml** 配置文件中的 **host** 元素 **name** 属性。
2. **jboss.host.name** 系统属性的值。
3. **jboss.qualified.host.name** 系统属性中最后一个句点(.)字符后面的值；如果没有最终句点(.)字符，则为整个值。
4. 基于 **POSIX** 的操作系统的 **HOSTNAME** 环境变量中的句点(.)字符后面的值、**Microsoft Windows** 的 **COMPUTERNAME** 环境变量，或者整个值（如果没有最终句点(.)字符）。

主机控制器的名称在相关 **host .xml** 配置文件顶部的主机元素中配置，例如：

```
<host xmlns="urn:jboss:domain:8.0" name="host1">
```

使用以下步骤，通过管理 **CLI** 更新主机名。

1. 启动 **JBoss EAP** 主机控制器。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

2. 启动管理 **CLI**，连接域控制器。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --
controller=DOMAIN_CONTROLLER_IP_ADDRESS
```

3. 使用以下命令来设置新主机名：

```
/host=EXISTING_HOST_NAME:write-attribute(name=name,value=NEW_HOST_NAME)
```

这会修改 **host-slave.xml** 文件中的主机名属性，如下所示：

```
<host name="NEW_HOST_NAME" xmlns="urn:jboss:domain:8.0">
```

4. 重新加载主机控制器，使更改生效。

```
reload --host=EXISTING_HOST_NAME
```

如果主机控制器配置文件中没有设置名称，您也可以在运行时传递主机名。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml  -Djboss.host.name=HOST_NAME
```

8.4. 管理服务器

8.4.1. 配置服务器组

以下是服务器组定义的示例：

```
<server-group name="main-server-group" profile="full">
  <jvm name="default">
    <heap size="64m" max-size="512m"/>
```

```
    </jvm>
    <socket-binding-group ref="full-sockets"/>
    <deployments>
      <deployment name="test-application.war" runtime-name="test-application.war"/>
      <deployment name="helloworld.war" runtime-name="helloworld.war" enabled="false"/>
    </deployments>
  </server-group>
```

可以使用管理 **CLI** 或管理控制台 **Runtime** 选项卡来配置服务器组。

添加服务器组

以下管理 **CLI** 命令可用于添加服务器组：

```
/server-group=SERVER_GROUP_NAME:add(profile=PROFILE_NAME,socket-binding-
group=SOCKET_BINDING_GROUP_NAME)
```

更新服务器组

以下管理 **CLI** 命令可用于更新服务器组属性：

```
/server-group=SERVER_GROUP_NAME:write-attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

删除服务器组

以下管理 **CLI** 命令可用于移除服务器组：

```
/server-group=SERVER_GROUP_NAME:remove
```

服务器组属性

服务器组需要以下属性：

- **Name** ：服务器组名称。
- **profile** ：服务器组配置文件名称。
- **socket-binding-group** ：用于组中服务器的默认套接字绑定组。这可以逐个服务器覆盖。

服务器组包括以下可选属性：

- ***management-subsystem-endpoint*** *：设置为* ***true****，使属于服务器组的服务器使用来自其远程 子系统的端点重新 连接到主机控制器。必须存在远程 子系统 才能正常工作。*

- ***socket-binding-default-interface*** *：此服务器的套接字绑定组默认接口。*

- ***socket-binding-port-offset*** *：添加到套接字绑定组给出的端口值的默认偏移。*

- ***Deployment***： *要部署到组中服务器上的部署内容。*

- ***JVM*** *：组中所有服务器的默认* ***JVM*** *设置。主机控制器将这些设置与* ***host.xml*** *中提供的任何其他配置合并，以派生用于启动服务器的* ***JVM*** *的设置。*

- ***deployment-overlays*** *：此服务器组中定义的部署覆盖和部署之间的链接。*

- ***system-properties*** *：要在组中的服务器上设置的系统属性。*

8.4.2. 配置服务器

默认 ***host.xml*** *配置文件定义三个服务器：*

```
<servers>
  <server name="server-one" group="main-server-group">
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
    <socket-bindings port-offset="150"/>
  </server>
  <server name="server-three" group="other-server-group" auto-start="false">
    <socket-bindings port-offset="250"/>
  </server>
</servers>
```

名为 ***server-one*** *的服务器实例与* ***main-server-group*** *关联，并且继承该服务器组指定的子系统配置和套接字绑定。名为* ***server-two*** *的服务器实例也与* ***main-server-group*** *关联，但也定义了套接字绑定* ***port-offset*** *值，因此不会与* ***server-one*** *使用的端口值冲突。名为* ***server-three*** *的服务器实例与* ***other-server-group*** *关联并使用该组的配置。它还定义* ***port-offset*** *值，并将* ***auto-start*** *设置为* ***false****，以便此服务器在主机控制器启动时不启动。*

可以使用管理 **CLI** 或管理控制台运行时选项卡来配置服务器。

添加服务器

以下管理 **CLI** 命令可用于添加服务器：

```
/host=HOST_NAME/server-config=SERVER_NAME:add(group=SERVER_GROUP_NAME)
```

更新服务器

以下管理 **CLI** 命令可用于更新服务器属性：

```
/host=HOST_NAME/server-config=SERVER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=VALUE)
```

删除服务器

以下管理 **CLI** 命令可用于移除服务器：

```
/host=HOST_NAME/server-config=SERVER_NAME:remove
```

服务器属性

服务器需要以下属性：

- **name** ：服务器的名称。
- **Group** ：来自域模型的服务器组名称。

服务器包括以下可选属性：

- **auto-start** ：主机控制器启动时是否应启动此服务器。
- **socket-binding-group** ：此服务器所属的套接字绑定组。
- **socket-binding-port-offset** ：添加到此服务器的套接字绑定组给出的端口值的偏移。

- ***update-auto-start-with-server-status*** ： 将 **auto-start** 属性更新为服务器的状态。
- 接口 ：可在服务器上使用的完全指定的命名网络接口的列表。
- ***JVM*** ：此服务器的 **JVM** 设置。如果未声明，则从父服务器组或主机继承设置。
- ***path*** ：指定文件系统路径列表。
- ***system- property***：在此服务器上要设置的系统属性列表。

8.4.3. 启动和停止服务器

您可以通过导航到 **Runtime** 选项卡并选择适当的主机或服务器组，从管理控制台对服务器执行各种操作，如启动、停止和重新加载。

以下命令可以使用管理 **CLI** 执行这些操作。

启动服务器

您可以在特定主机上启动单个服务器。

```
/host=HOST_NAME/server-config=SERVER_NAME:start
```

您可以启动指定服务器组中的所有服务器。

```
/server-group=SERVER_GROUP_NAME:start-servers
```

停止服务器

您可以在特定主机上停止单个服务器。

```
/host=HOST_NAME/server-config=SERVER_NAME:stop
```

您可以停止指定服务器组中的所有服务器。

```
/server-group=SERVER_GROUP_NAME:stop-servers
```

重新载入服务器

您可以在特定主机上重新加载单一服务器。

```
/host=HOST_NAME/server-config=SERVER_NAME:reload
```

您可以重新加载指定服务器组中的所有服务器。

```
/server-group=SERVER_GROUP_NAME:reload-servers
```

终止服务器

您可以中断指定服务器组中的所有服务器进程。

```
/server-group=SERVER_GROUP_NAME:kill-servers
```

8.5. 域控制器发现和故障转移

在设置受管域时，每一主机控制器必须配置必要的信息以联系域控制器。在 **JBoss EAP** 中，每一主机控制器可以配置多个选项，以查找域控制器。主机控制器迭代选项列表，直到成功为止。

如果主域控制器出现问题，备份主机控制器可以提升为域聚合器。这使得主机控制器在新的域控制器提升后自动故障转移到它。

8.5.1. 配置域发现选项

下例中演示了如何使用查找域控制器的多个选项配置主机控制器：

示例：具有多个域控制器选项的主机控制器

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="172.16.81.100" port="${jboss.domain.master.port:9990}"/>
      <static-discovery name="backup" protocol="${jboss.domain.master.protocol:remote}"
host="172.16.81.101" port="${jboss.domain.master.port:9990}"/>
```

```
    </discovery-options>
  </remote>
</domain-controller>
```

静态发现选项包括以下所需属性：

name

此域控制器发现选项的名称。

主机

远程域控制器的主机名。

port

远程域控制器的端口。

在上例中，第一个发现选项是预期成功的选项。第二个选项可用于故障转移情况。

8.5.2. 使用缓存域配置启动主机控制器

可以使用 **--cached-dc** 选项在不连接域控制器的情况下启动主机控制器；不过，主机控制器必须之前已从域控制器在本地缓存其域控制器的域控制器配置。使用这个 **--cached-dc** 选项启动主机控制器将缓存主机控制器的域控制器配置。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --cached-dc
```

这会在 **EAP_HOME/domain/configuration/** 目录中创建一个 **domain.cached-remote.xml** 文件，该文件包含此主机控制器临时管理其当前服务器而无需域控制器连接所需的信息。

注意

默认情况下，使用 **--cached-dc** 选项仅缓存此主机控制器使用的配置，这意味着无法提升到整个域的域控制器。有关缓存整个域配置的信息，请参阅缓存域配置，以允许主机控制器充当域控制器。

如果在使用 **--cached-dc** 启动此主机控制器时域控制器不可用，主机控制器将开始使用 **domain.cached-remote.xml** 文件中保存的缓存配置。注意此文件必须存在，或者主机控制器将无法启动。

在这种状态下，主机控制器无法修改域配置，但可以启动服务器和管理部署。

在使用缓存的配置启动后，主机控制器将继续尝试重新连接到域控制器。域控制器变为可用后，主机控制器将自动重新连接它并同步域配置。请注意，一些配置更改可能要求您重新加载主机控制器才能生效。如果发生此情况，主机控制器上将记录警告。

8.5.3. 提升主机控制器以作为域控制器

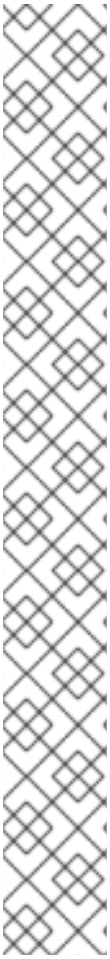
如果主域控制器出现问题，您可以提升主机控制器以充当域控制器。主机控制器必须先将域配置从域控制器本地缓存，然后才能提升。

缓存域配置

将 **--backup** 选项用于您可能希望成为域控制器的任何主机控制器。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --backup
```

这会在 **EAP_HOME/domain/configuration/** 目录中创建一个 **domain.cached-remote.xml** 文件，其中包含整个域配置的副本。如果主机控制器重新配置为充当域控制器，则将使用此配置。



注意

ignore-unused-configuration 属性用于确定要缓存的特定主机的配置量。值 **true** 表示仅缓存与此主机控制器相关的配置，这不允许它接管域控制器。值 **false** 表示将缓存整个域配置。

--backup 参数将此属性默认为 **false**，以缓存整个域。但是，如果您在 **host.xml** 文件中设置了此属性，则会使用该值。

您也可以单独使用 **--cached-dc** 选项来创建域配置的副本，但必须在 **host.xml** 中将 **ignore-unused-configuration** 设置为 **false** 来缓存整个域。例如：

```
<domain-controller>
  <remote username="$local" security-realm="ManagementRealm" ignore-unused-
configuration="false">
    <discovery-options>
      ...
    </discovery-options>
  </remote>
</domain-controller>
```

将主机控制器提升为域控制器

1. 确保原先的域控制器已停止。
2. 使用管理 **CLI** 连接主机控制器，该主机控制器将成为新的域控制器。
3. 执行以下命令，将主机控制器配置为充当新的域控制器：

```
/host=backup:write-attribute(name=domain-controller.local, value={})
```

4. 执行以下命令，以重新加载主机控制器：

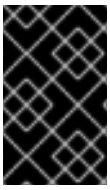
```
reload --host=HOST_NAME
```

此主机控制器现在将充当域控制器。

8.6. 受管域设置

8.6.1. 在单一机器中设置受管域

您可以使用 **jboss.domain.base.dir** 属性在一台计算机上运行多个主机控制器。



重要

不支持将多个 **JBoss EAP** 主机控制器配置为单一计算机上的系统服务。

1. 复制域控制器的 **EAP_HOME/domain** 目录。

```
$ cp -r EAP_HOME/domain /path/to/domain1
```

2. 复制主机控制器的 **EAP_HOME/domain** 目录。

```
$ cp -r EAP_HOME/domain /path/to/host1
```

3. 使用 **/path/to /domain1** 启动域控制器。

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml -
Djboss.domain.base.dir=/path/to/domain1
```

4. 使用 **/path/to /host1** 启动主机控制器。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Djboss.domain.base.dir=/path/to/host1 -Djboss.domain.master.address=IP_ADDRESS -
Djboss.management.http.port=PORT
```

注意

在启动主机控制器时，您必须使用 **jboss.domain.master.address** 属性指定域控制器的地址。

此外，由于此主机控制器在与域控制器相同的计算机上运行，因此您必须更改管理接口，使它不会与域控制器的管理接口冲突。此命令设置 **jboss.management.http.port** 属性。

以这种方式启动的每个实例将共享基础安装目录中的其余资源，如 **EAP_HOME/modules/**，但使用 **jboss.domain.base.dir** 指定的目录中的域配置。

8.6.2. 在两个机器中设置受管域

注意

您可能需要配置防火墙来运行此示例。

您可以在两台计算机上创建受管域，其中一台计算机是域控制器，另一台计算机是主机。如需更多信息，请参阅关于域控制器。

- **ip1 =** 域控制器的 **IP** 地址**(Machine 1)**
- **ip2 =** 主机的 **IP** 地址**(Machine 2)**

在两个机器中创建受管域

1. 在机器 **1** 上

   a. 添加管理用户，以便主机可由域控制器进行身份验证。

   使用 **add-user.sh** 脚本，为主机控制器 **HOST_NAME** 添加管理用户。确保对最后一个提示回答 **yes**，并注意提供的机密值。此 **secret** 值将在主机控制器配置中使用，其显示类似于下面这一行：

   ```
   <secret value="SECRET_VALUE" />
   ```

   b. 启动域控制器。

   指定 **host-master.xml** 配置文件，它为专用域控制器进行了预配置。此外，还要设置 **jboss.bind.address.management** 属性，使域控制器对其他计算机可见。

   ```
   $ EAP_HOME/bin/domain.sh --host-config=host-master.xml -
   Djboss.bind.address.management=IP1
   ```

2.

2. 在 **Machine 2** 上

a. 使用用户凭据更新主机配置。

编辑 **EAP_HOME/domain/configuration/host-slave.xml**，并设置主机名 **HOST_NAME** 和 **secret** 值 **SECRET_VALUE**。

```
<host xmlns="urn:jboss:domain:8.0" name="HOST_NAME">
 <management>
  <security-realms>
   <security-realm name="ManagementRealm">
    <server-identities>
     <secret value="SECRET_VALUE" />
    </server-identities>
    ...
```

b. 启动主机控制器。

指定 **host-slave.xml** 配置文件，它为从属主机控制器进行了预配置。此外，设置 **jboss.domain.master.address** 属性以连接域控制器，再设置 **jboss.bind.address** 属性来设置主机控制器绑定地址。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml  -
Djboss.domain.master.address=IP1  -Djboss.bind.address=IP2
```

现在，您可以在启动时使用 **--controller** 参数指定域控制器地址，从而通过管理 **CLI** 管理域。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=IP1
```

或者，您可以从位于 **http://IP1:9990** 的 管理控制台管理该域。

8.7. 管理多个 JBOSS EAP 版本

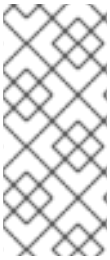
最新版本的 **JBoss EAP** 可以管理运行较早版本的 **JBoss EAP** 服务器和主机。根据您需要管理的 **JBoss EAP** 版本，请参见相应的部分。

- 将 **JBoss EAP 7.x** 域控制器配置为管理员 **JBoss EAP 6** 实例

- 配置 JBoss EAP 7.3 域控制器，以管理以前的 JBoss EAP 次要版本

8.7.1. 将 JBoss EAP 7.x 域控制器配置为管理员 JBoss EAP 6 实例

JBoss EAP 7.1 域控制器可以管理运行 JBoss EAP 6 的主机和服务器，只要它们是 JBoss EAP 6.2 或更高版本。



注意

对于管理不同补丁版本的 JBoss EAP 7.0 域控制器，无需任何配置更改。但是，JBoss EAP 7.0 域控制器必须运行一个等于或大于其管理的主机控制器上的版本的补丁版本。

完成以下任务，在 JBoss EAP 7 受管域中成功管理 JBoss EAP 6 实例。

1. 将 JBoss EAP 6 配置添加到 JBoss EAP 7 域控制器。
2. 更新 JBoss EAP 6 配置文件的行为。
3. 设置 JBoss EAP 6 服务器的服务器组。
4. 阻止 JBoss EAP 6 实例接收 JBoss EAP 7 更新。

完成这些任务后，您可以使用管理 CLI 从 JBoss EAP 7 域控制器管理 JBoss EAP 6 服务器和配置。请注意，JBoss EAP 6 主机将无法利用新的 JBoss EAP 7 功能，如批处理。

警告

*由于管理控制台针对 **JBoss EAP** 的最新版本进行了优化，因此您不应用它来更新 **JBoss EAP 6** 主机、服务器和配置文件。在从 **JBoss EAP 7** 受管域管理 **JBoss EAP 6** 配置时，应改为使用管理 **CLI**。*

8.7.1.1. 将 JBoss EAP 6 配置添加到 JBoss EAP 7 域控制器

*若要让域控制器管理 **JBoss EAP 6** 服务器，您必须在 **JBoss EAP 7** 域配置中提供 **JBoss EAP 6** 配置详细信息。您可以通过将 **JBoss EAP 6** 配置文件、套接字绑定组和服务器组复制到 **JBoss EAP 7** **domain.xml** 配置文件来实现 此操作。*

*如果与 **JBoss EAP 7** 配置中的现有名称有任何冲突，您需要重命名资源。为确保正确行为，还需要做一些额外的调整。*

*以下步骤使用 **JBoss EAP 6** 默认 配置文件、**standard -sockets** 套接字绑定组和 **main-server-group** 服务器组。*

1. *编辑 **JBoss EAP 7 domain.xml** 配置文件。建议先备份此文件，然后再编辑。*

2. *将适用的 **JBoss EAP 6** 配置文件复制到 **JBoss EAP 7 domain.xml** 文件。*

   *此流程假定 **JBoss EAP 6** 默认 配置文件已复制并重命名为 **eap6-default**。*

JBoss EAP 7 domain.xml

```
<profiles>
  ...
  <profile name="eap6-default">
    ...
  </profile>
</profiles>
```

3. 添加此配置集使用的必要扩展。

如果您的 **JBoss EAP 6** 配置文件使用 **JBoss EAP 7** 中不再存在的子系统，您必须将适当的扩展添加到 **JBoss EAP** 域配置中。

JBoss EAP 7 domain.xml

```
<extensions>
  ...
  <extension module="org.jboss.as.configadmin"/>
  <extension module="org.jboss.as.threads"/>
  <extension module="org.jboss.as.web"/>
<extensions>
```

4. 将适用的 **JBoss EAP 6** 套接字绑定组复制到 **JBoss EAP 7 domain.xml** 文件。

此流程假定 **JBoss EAP 6** 标准套接字绑定 组已复制并重命名为 **eap6-standard-sockets**。

JBoss EAP 7 domain.xml

```
<socket-binding-groups>
  ...
  <socket-binding-group name="eap6-standard-sockets" default-interface="public">
    ...
  </socket-binding-group>
</socket-binding-groups>
```

5. 将适用的 **JBoss EAP 6** 服务器组复制到 **JBoss EAP 7 domain.xml** 文件。

此流程假定 **JBoss EAP 6 main-server-group** 服务器组已复制并重命名为 **eap6-main-**

server-group。您还必须更新此服务器组，以使用 JBoss EAP 6 配置文件、eap6-default 和 JBoss EAP 6 套接字绑定组 eap6-standard-sockets。

JBoss EAP 7 domain.xml

```
<server-groups>
  ...
  <server-group name="eap6-main-server-group" profile="eap6-default">
    ...
    <socket-binding-group ref="eap6-standard-sockets"/>
  </server-group>
</server-groups>
```

8.7.1.2. 更新 JBoss EAP 6 配置文件的行为

根据 JBoss EAP 版本和所需行为，需要对 JBoss EAP 6 实例使用的配置集进行其他更新。根据现有 JBoss EAP 6 实例使用的子系统和配置，您可能需要进行其他更改。

启动 JBoss EAP 7 域控制器并启动其管理 CLI，以执行下列更新：这些示例假定 JBoss EAP 6 配置文件为 eap6-default。

- *移除 bean-validation 子系统：*

  JBoss EAP 7 将 bean 验证功能从 ee 子系统移到其自身的子系统 bean-validation。如果 JBoss EAP 7 域控制器看到 legacy ee 子系统，它将添加新的 bean-validation 子系统。但是，JBoss EAP 6 主机将不会识别此子系统，因此必须移除它。

  JBoss EAP 7 域控制器 CLI

  ```
  /profile=eap6-default/subsystem=bean-validation:remove
  ```

- *设置 CDI 1.0 行为。*

仅当您想要用于 **JBoss EAP 6** 服务器的 **CDI 1.0** 行为时，这才是必需的，而不是 **JBoss EAP 7** 中之后使用的 **CDI** 版本的行为。如果您希望 **CDI 1.0** 的行为，请对 **weld** 子系统进行以下更新：

JBoss EAP 7 域控制器 CLI

```
/profile=eap6-default/subsystem=weld:write-attribute(name=require-bean-descriptor,value=true)

/profile=eap6-default/subsystem=weld:write-attribute(name=non-portable-mode,value=true)
```

- 启用 **JBoss EAP 6.2** 的数据源统计信息.

仅当您的配置文件供 **JBoss EAP 6.2** 服务器使用时才需要。**JBoss EAP 6.3** 引入了 支持 **statistics** 的属性，默认为 **false**，不收集统计数据；但是，**JBoss EAP 6.2** 的行为是收集统计数据。如果 **JBoss EAP 6.2** 主机和运行较新的 **JBoss EAP** 版本的主机使用此配置文件，则主机之间的行为会不一致，这是不允许的。因此，供 **JBoss EAP 6.2** 主机使用的配置文件应对其数据源进行以下更改：

JBoss EAP 7 域控制器 CLI

```
/profile=eap6-default/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-enabled,value=true)
```

8.7.1.3. 设置 JBoss EAP 6 服务器的服务器组

如果您重命名了服务器组，您需要更新 **JBoss EAP 6** 主机配置，以使用 **JBoss EAP 7** 配置中指定的新服务器组。本例使用 **JBoss EAP 7 domain.xml** 中指定的 **eap6-main-server-group** 服务器组。

JBoss EAP 6 host-slave.xml

```
<servers>
```

```
    <server name="server-one" group="eap6-main-server-group"/>
    <server name="server-two" group="eap6-main-server-group">
        <socket-bindings port-offset="150"/>
    </server>
</servers>
```

注意

与主机正在运行的版本相比，主机无法使用较新版本的 **JBoss EAP** 中引入的功能或配置设置。

8.7.1.4. 阻止 JBoss EAP 6 实例接收 JBoss EAP 7 更新

受管域中的域控制器将配置更新转发到其主机控制器。您必须使用 **host-exclude** 配置来指定应在特定版本中隐藏的资源。为您的 **JBoss EAP 6** 版本选择适当的预配置 **host-exclude** 选项：**EAP62**、**EAP63**、**EAP64** 或 **EAP64z**。

host -exclude 配置的 **active- server-groups** 属性指定特定版本使用的服务器组的列表。这些服务器组及其关联的配置文件、套接字绑定组和部署资源将提供给此版本的主机，但所有其他内容都将从这些主机中隐藏。

本例假定版本为 **JBoss EAP 6.4.z**，并将 **JBoss EAP 6** 服务器组 **eap6-main-server-group** 添加为活动服务器组。

JBoss EAP 7 域控制器 CLI

```
/host-exclude=EAP64z:write-attribute(name=active-server-groups,value=[eap6-main-server-group])
```

如果需要，您可以使用 **active-socket-binding-groups** 属性指定供服务器使用的额外套接字绑定组。这只适用于没有与 **active-server-groups** 中指定的服务器组关联的套接字绑定组。

8.7.2. 将 JBoss EAP 7.3 域控制器配置为 JBoss EAP 的管理员次要版本

JBoss EAP 7.3 域控制器可以管理从之前的 JBoss EAP 次要版本运行的主机和服务器。

注意

对于管理位于不同补丁版本的 JBoss EAP 7.2 主机的 JBoss EAP 7.3 域控制器，您无需进行任何配置更改。但是，您必须在等于或大于其管理的主机控制器上的版本补丁版本上运行 JBoss EAP 7.2 域控制器。

完成以下任务，在 JBoss EAP 7.3 受管域中成功管理 JBoss EAP 7.2 实例。

1. 将 JBoss EAP 7.2 配置添加到 JBoss EAP 7.3 域控制器。
2. 设置 JBoss EAP 7.2 服务器的服务器组。
3. 阻止 JBoss EAP 7.2 实例接收 JBoss EAP 7.3 更新。

完成这些任务后，您可以使用管理 CLI 从 JBoss EAP 7.3 域控制器管理 JBoss EAP 7.2 服务器和配置。

警告

由于管理控制台针对 JBoss EAP 的最新版本进行了优化，因此您必须使用 CLI 从 JBoss EAP 7.3 受管域管理 JBoss EAP 7.2 配置。不要使用管理控制台更新 JBoss EAP 7.2 主机、服务器和配置文件。

8.7.2.1. 将 JBoss EAP 7.2 配置添加到 JBoss EAP 7.3 域控制器

若要让域控制器管理 JBoss EAP 7.2 服务器，您必须在 JBoss EAP 7.3 域配置中提供 JBoss EAP 7.2 配置详细信息。您可以通过将 JBoss EAP 7.2 配置文件、套接字绑定组和服务器组复制到 JBoss EAP 7.3 domain.xml 配置文件来实现 此操作。

如果资源的名称与 **JBoss EAP 7.3** 配置中的资源名称冲突，则必须重命名资源。

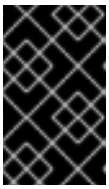
以下步骤使用 **JBoss EAP 7.2** 默认 配置文件、**standard -sockets** 套接字绑定组和 **main-server-group** 服务器组。

先决条件

- 您已将 **JBoss EAP 7.2** 默认配置 文件复制并重命名为 **eap72-default**。
- 您已将 **JBoss EAP 7.2** 标准套接字绑定 组复制并重命名为 **eap72-standard-sockets**。
- 您已将 **JBoss EAP 7.2 main-server-group** 服务器组复制并重命名为 **eap72-main-server-group**。
  - 您已更新了服务器组，以使用 **JBoss EAP 7.2** 配置文件 **eap72-default**，再使用 **JBoss EAP 7.2** 套接字绑定组 **eap72-standard-sockets**。

流程

1. 编辑 **JBoss EAP 7.3 domain.xml** 配置文件。



重要

在编辑 文件之前，备份 **JBoss EAP 7.3 domain.xml** 配置文件。

2. 将适用的 **JBoss EAP 7.2** 配置文件复制到 **JBoss EAP 7.3 domain.xml** 文件。例如：

```
<profiles>
 ...
 <profile name="eap72-default">
  ...
 </profile>
</profiles>
```

3. 将适用的 **JBoss EAP 7.2** 套接字绑定组复制到 **JBoss EAP 7.3 domain.xml** 文件。例如：

```
<socket-binding-groups>
 ...
 <socket-binding-group name="eap72-standard-sockets" default-interface="public">
  ...
 </socket-binding-group>
</socket-binding-groups>
```

4. 将适用的 **JBoss EAP 7.2** 服务器组复制到 **JBoss EAP 7.3 domain.xml** 文件。例如：

```
<server-groups>
 ...
 <server-group name="eap72-main-server-group" profile="eap72-default">
  ...
  <socket-binding-group ref="eap72-standard-sockets"/>
 </server-group>
</server-groups>
```

8.7.2.2. 设置 JBoss EAP7.2 服务器的服务器组

如果重命名了服务器组，您需要更新 **JBoss EAP 7.2** 主机配置，以使用 **JBoss EAP 7.3** 配置中指定的新服务器组。本例使用 **JBoss EAP 7.3 domain.xml** 中指定的 **eap72-main-server-group** 服务器组。

JBoss EAP 7.2 host-slave.xml

```
<servers>
   <server name="server-one" group="eap72-main-server-group"/>
   <server name="server-two" group="eap72-main-server-group">
      <socket-bindings port-offset="150"/>
   </server>
</servers>
```

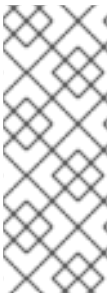
注意

与主机正在运行的版本相比，主机无法使用较新版本的 **JBoss EAP** 中引入的功能或配置设置。

8.7.2.3. 阻止 JBoss EAP 7.2 实例接收 JBoss EAP 7.3 更新

受管域控制器将配置更新转发到其主机控制器，这样 **JBoss EAP 7.2** 主机不会接收专用于 **JBoss EAP 7.3** 的配置和资源。您必须配置 **JBoss EAP 7.2** 主机来忽略这些资源。您可以通过在 **JBoss EAP**

7.2 主机上设置 **ignore-unused-configuration** 属性来完成此操作。



注意

您也可以使用 **host-exclude** 配置来指示域控制器从运行某些 **JBoss EAP** 版本的主机中隐藏某些资源。有关如何使用 **host-exclude** 配置的示例，请参阅 从接收 **JBoss EAP 7** 更新中防止 **JBoss EAP 6** 实例。对于 **JBoss EAP 7.2**，您将使用 **EAP72 host-exclude** 选项。

在 **JBoss EAP 7.2** 主机控制器连接配置中，将 **ignore-unused-configuration** 属性设置为 **true**。

JBoss EAP 7.2 host-slave.xml

```
<domain-controller>
    <remote security-realm="ManagementRealm" ignore-unused-configuration="true">
        <discovery-options>
            <static-discovery name="primary" protocol="${jboss.domain.master.protocol:remote}"
host="${jboss.domain.master.address}" port="${jboss.domain.master.port:9990}"/>
        </discovery-options>
    </remote>
</domain-controller>
```

通过此设置，仅此主机使用的服务器组及其关联的配置文件、套接字绑定组和部署资源可供主机使用。所有其他资源都会被忽略。

8.8. 管理 JBOSS EAP 配置文件

8.8.1. 关于配置集

JBoss EAP 使用配置文件来组织服务器可用的子系统。配置文件由一组可用的子系统以及各个子系统的特定配置组成。具有大量子系统的配置文件将产生具有大量功能的服务器。具有一组小、重点的子系统的配置集将获得较少的功能，但占用空间比较小。

JBoss EAP 附带五个预定义配置文件，应满足大多数用例的要求：

default

包括常用的子系统，如 日志记录、安全性、数据源、 **infinispan**、**webservices**、**ee**、**ejb3**、事务等。

ha

包含 默认配置 集中提供的子系统，同时添加了 **jgroups** 和 **modcluster** 子系统以实现高可用性

full

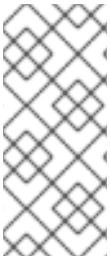
包括 **default** 配置文件中提供的子系统，同时添加了 **messaging-activemq** 和 **iiop-openjdk** 子系统

full-ha

包含 完整 配置集中提供的子系统，同时添加了 **jgroups** 和 **modcluster** 子系统以实现高可用性

load-balancer

包括使用内置 **mod_cluster** 前端负载平衡器对其他 **JBoss EAP** 实例进行负载平衡所需的最小子系统。



注意

JBoss EAP 通过从现有配置文件的配置中删除子系统，提供手动禁用扩展或卸载驱动程序和其他服务的功能。然而，在大多数情况下，这是不必要的。由于 **JBoss EAP** 根据需要动态加载子系统，如果服务器或应用从未使用子系统，它就不会加载。

在现有配置文件不提供必要的功能的情况下，**JBoss EAP** 也提供定义自定义配置文件的功能。

8.8.2. 克隆配置集

JBoss EAP 允许您通过克隆现有的配置文件在受管域中创建新配置文件。这将创建原始配置文件的配置和子系统的副本。

可以使用管理 **CLI** 对所需配置集的 克隆操作进行克隆，以克隆配置集。

```
/profile=full-ha:clone(to-profile=cloned-profile)
```

您也可以从管理控制台克隆配置集，方法是选择要克隆的所需配置集并单击 **Clone**。

8.8.3. 创建层次结构配置集

在受管域中，您可以创建配置文件层次结构。这可让您使用其他配置集可以继承的通用扩展来创建基础配置文件。

受管域在 **domain.xml** 中定义多个配置文件。如果多个配置集将相同的配置用于特定的子系统，您只需在一个位置进行配置，而不必配置不同的配置文件。父配置文件中的值不能被覆盖。

此外，每个配置文件都必须可自助使用。如果引用了元素或子系统，则必须在引用它的配置文件中定义它。

配置集可以使用 **management CLI** 在层次结构中包含其他配置集，方法是使用 **list-add** 操作并提供要包含的配置集。

```
/profile=new-profile:list-add(name=includes, value=PROFILE_NAME)
```

第 9 章 配置 JVM 设置

Java 虚拟机(JVM)设置的配置与独立 JBoss EAP 服务器或受管域中的 JBoss EAP 服务器不同。

对于独立 JBoss EAP 服务器实例，服务器启动进程在启动时将 JVM 设置传递到 JBoss EAP 服务器。可以在启动 JBoss EAP 之前从命令行声明它们，或者使用管理控制台中 Configuration 下的 System Properties 页面。

在受管域中，JVM 设置在 host.xml 和 domain.xml 配置文件中声明，可以在主机、服务器组或服务器级别上配置。

注意

系统属性必须在 JAVA_OPTS 中配置，以便在启动期间由 JBoss EAP 模块（如日志记录管理器）使用。

9.1. 为单机服务器配置 JVM 设置

单机 JBoss EAP 服务器实例的 JVM 设置可以在运行时声明，方法是在启动服务器之前设置 JAVA_OPTS 环境变量。

在 Linux 上设置 JAVA_OPTS 环境变量的示例如下所示：

```
$ export JAVA_OPTS="-Xmx1024M"
```

同一设置可用于 Microsoft Windows 环境：

```
set JAVA_OPTS="Xmx1024M"
```

或者，可以在 EAP_HOME/bin 文件夹中添加 JVM 设置到 standalone.conf 文件或 Windows Server 的 standalone.conf.bat，其中包含要传递给 JVM 的选项示例。

警告

*设置 **JAVA_OPTS** 环境变量将覆盖 **standalone.conf** 中的默认值，这可能会造成 **JBoss EAP** 启动问题。*

9.2. 为受管域配置 JVM 设置

*在 **JBoss EAP** 受管域中，您可以在多个级别上定义 **JVM** 设置。您可以在特定主机上定义自定义 **JVM** 设置，然后将这些设置应用到服务器组或个别服务器实例。*

*默认情况下，服务器组和个别服务器将继承其父级的 **JVM** 设置，但您可以选择覆盖每个级别的 **JVM** 设置。*

注意

*__domain .conf__ 或 __domain. conf.bat__ 中的 **JVM** 设置应用于 **JBoss EAP** 主机控制器的 **Java** 进程，而非由该主机控制器控制的单个 **JBoss EAP** 服务器实例。*

9.2.1. 在主机控制器上定义 JVM 设置

*您可以在主机控制器上定义 **JVM** 设置，并将这些设置应用到服务器组或个别服务器上。**JBoss EAP** 附带 默认的 **JVM** 设置，但以下管理 **CLI** 命令演示了使用一些自定义 **JVM** 设置和选项创建名为 **production_jvm** 的新 **JVM** 设置：*

```
/host=HOST_NAME/jvm=production_jvm:add(heap-size=2048m, max-heap-size=2048m, max-permgen-size=512m, stack-size=1024k, jvm-options=["-XX:-UseParallelGC"])
```

*有关所有可用选项的说明，请参阅受管域 **JVM** 配置属性。*

*您还可以导航到 **Runtime** → **Hosts**，选择主机并单击 **View**，然后选择 **JVM** 选项卡，在 **JBoss EAP** 管理控制台中创建和编辑 **JVM** 设置。*

*这些设置存储在 **host.xml** 中的 **<jvm>** 标签内。*

9.2.2. 将 JVM 设置应用到服务器组

在创建服务器组时，您可以指定组中的所有服务器都将使用的 JVM 配置。以下管理 CLI 命令演示了使用上例中显示的 production_jvm JVM 设置创建服务器组名称 groupA ：

```
/server-group=groupA:add(profile=default, socket-binding-group=standard-sockets)
/server-group=groupA/jvm=production_jvm:add
```

服务器组中的所有服务器都将从 production_jvm 继承 JVM 设置。

您也可以覆盖服务器组级别上的特定 JVM 设置。例如，要设置不同的堆大小，您可以使用以下命令：

```
/server-group=groupA/jvm=production_jvm:write-attribute(name=heap-size,value="1024m")
```

应用上述命令后，服务器组 groupA 将从 production_jvm 继承 JVM 设置，但堆大小除外，其值为 1024m。

有关所有可用选项的说明，请参阅受管域 JVM 配置属性。

您还可以导航到 Runtime → Server Groups，选择服务器组并单击 View，再选择 JVM 选项卡，以编辑 JBoss EAP 管理控制台中的服务器组 JVM 设置。

服务器组的这些设置存储在 domain.xml 中。

9.2.3. 将 JVM 设置应用到单个服务器

默认情况下，单个 JBoss EAP 服务器实例将继承它所属的服务器组的 JVM 设置。不过，您可以选择使用来自主机控制器的另一个完整的 JVM 设置定义覆盖继承的设置，或者选择覆盖特定的 JVM 设置。

例如，以下命令覆盖 上例中服务器组的 JVM 定义，并将 server-one 的 JVM 设置 设置为默认 JVM 定义：

```
/host=HOST_NAME/server-config=server-one/jvm=default:add
```

此外，与服务器组类似，您可以在服务器级别上覆盖特定的 JVM 设置。例如，要设置不同的堆大小，您可以使用以下命令：

```
/host=HOST_NAME/server-config=server-one/jvm=default:write-attribute(name=heap-
size,value="1024m")
```

有关所有可用选项的说明，请参阅受管域 **JVM** 配置属性。

您还可以导航到 **Runtime** → **Hosts**，选择主机，单击服务器上的 **View**，然后选择 **JVM** 选项卡，在 **JBoss EAP** 管理控制台中编辑服务器 **JVM** 设置。

单个服务器的这些设置存储在 **host.xml** 中。

9.3. 显示 JVM 状态

您可以从管理控制台查看单机或受管域服务器的 **JVM** 资源的状态，如堆和线程使用情况。虽然实时不显示统计数据，但您可以单击 **Refresh** 以提供 **JVM** 资源的最新概述。

显示独立 **JBoss EAP** 服务器的 **JVM** 状态：

- 导航到 **Runtime** 选项卡，选择服务器，然后选择 **Status**。

显示受管域中 **JBoss EAP** 服务器的 **JVM** 状态：

- 导航到 **Runtime** → **Hosts**，选择主机和服务器，然后选择 **Status**。

这显示了以下堆使用情况信息：

Max

内存管理的最大内存量。

Used

已用内存量。

已提交

提交用于 **Java** 虚拟机的内存量。

也提供其他信息，如 **JVM** 正常运行时间和线程使用情况。

9.4. 调优 JVM

有关优化 **JVM** 性能的提示，请参阅性能调优指南中的 **JVM** 调优部分。

第 10 章 邮件子系统

10.1. 配置邮件子系统

邮件 子系统允许您在 JBoss EAP 中配置邮件会话，然后使用 JNDI 将这些会话注入到应用中。它还支持使用 Jakarta EE @MailSessionDefinitions 注释和 @MailSessionDefinitions 注释进行配置。

配置要在应用程序中使用的 SMTP 服务器

1. 使用以下 CLI 命令配置 SMTP 服务器和出站套接字绑定，例如：

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp:add(host=localhost, port=25)
```

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp, username=user, password=pass, tls=true)
```

2. 在应用程序中调用配置的邮件会话

```
@Resource(lookup="java:jboss/mail/MySession")
private Session session;
```

有关可用于配置邮件会话和服务器的属性的完整列表，请参阅邮件子系统属性。

10.2. 配置自定义传输

使用标准邮件服务器（如 POP3 或 IMAP）时，邮件服务器具有一组可以定义的属性。其中一些属性是必需的。其中最重要的是 出站-socket-binding-ref，它是对出站邮件套接字绑定的引用，通过主机地址和端口号定义。

对于使用多个主机进行负载平衡的用户，定义 出站-socket-binding-ref 可能不是最有效的解决方案。标准 Jakarta Mail 不支持使用多个主机进行负载平衡的主机配置。因此，使用多个主机进行此配置的用户需要实施自定义邮件传输。这些自定义邮件传输不需要 出站-socket-binding-ref，也允许自定义主机属性格式。

您可以从管理 CLI 配置自定义邮件传输。

1. 添加新邮件会话并指定 **JNDI** 名称。

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. 添加出站套接字绑定并指定主机和端口。

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. 添加 **SMTP** 服务器并指定出站套接字绑定、用户名和密码。

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp-binding, username=user, password=pass, tls=true)
```

注意

您可以使用类似的步骤配置 **POP3** 或 **IMAP** 服务器。

POP3 服务器

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-pop3-binding:add(host=localhost, port=110)
/subsystem=mail/mail-session=mySession/server=pop3:add(outbound-socket-binding-ref=my-pop3-binding, username=user, password=pass)
```

IMAP 服务器

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-imap-binding:add(host=localhost, port=143)
/subsystem=mail/mail-session=mySession/server=imap:add(outbound-socket-binding-ref=my-imap-binding, username=user, password=pass)
```

要使用自定义服务器，请创建不含出站套接字绑定的自定义邮件服务器。您可以在自定义邮件服务器的属性定义中指定主机信息。例如：

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=pass, properties=
{"host" => "myhost", "my-property" =>"value"})
```

如果您定义了自定义协议，则包含句点(.)的任何属性名都将被视为完全限定名称，并直接传递。任何其他格式，如 **my-property**，都使用以下格式进行转换：**mail .server-name.my-property**。

以下 **XML** 是包含自定义服务器的邮件配置示例：

```
<subsystem xmlns="urn:jboss:domain:mail:3.0">
  <mail-session name="default" jndi-name="java:jboss/mail/Default">
      <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
   <mail-session name="myMail" from="user.name@domain.org" jndi-name="java:/Mail">
      <smtp-server password="password" username="user" tls="true" outbound-socket-binding-
ref="mail-smtp"/>
      <pop3-server outbound-socket-binding-ref="mail-pop3"/>
      <imap-server password="password" username="nobody" outbound-socket-binding-ref="mail-
imap"/>
   </mail-session>
   <mail-session name="custom" jndi-name="java:jboss/mail/Custom" debug="true">
      <custom-server name="smtp" password="password" username="username">
         <property name="host" value="mail.example.com"/>
      </custom-server>
   </mail-session>
   <mail-session name="custom2" jndi-name="java:jboss/mail/Custom2" debug="true">
      <custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
         <property name="custom-prop" value="some-custom-prop-value"/>
      </custom-server>
   </mail-session>
</subsystem>
```

10.3. 为密码使用凭证存储

除了在 **mail** 子系统中提供明文密码外，您还可以使用凭据存储来提供密码。**elytron** 子系统提供创建凭据存储的功能，以便在整个 **JBoss EAP** 中安全存放和使用您的密码。您可以在 **How to Configure Server Security** 的 **Credential Store** 部分中找到有关创建和使用凭证存储的更多详细信息。

使用管理 **CLI** 使用凭证存储

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref=my-smtp-binding, username=user, credential-reference={store=exampleCS, alias=mail-session-pw}, tls=true)
```

注意

*以下是如何指定使用明文密码 的 **credential-reference** 属性的示例：*

```
credential-reference={clear-text="MASK-Ewcyuqd/nP9;A1B2C3D4;351"}
```

使用管理控制台使用凭证存储

1. *访问管理控制台。如需更多信息，请参阅管理控制台。*

2. *导航到 **Configuration** → **Subsystems** → **Mail**。*

3. *选择适当的邮件会话，再单击 **View**。*

4. *选择**"服务器"**并选择相应的邮件服务器。您可以在 **Credential Reference** 选项卡中配置凭据引用，并在 **Attributes** 选项卡中编辑其他属性。*

第 11 章 使用 JBOSS EAP 记录

JBoss EAP 提供高度可配置的日志记录设施，供其自身的内部使用和部署的应用使用。logging 子系统基于 JBoss LogManager，除了 JBoss Logging 外还支持多个第三方应用日志框架。

11.1. 关于服务器日志记录

11.1.1. 服务器日志记录

默认情况下，所有 JBoss EAP 日志条目都写入 server.log 文件。该文件的位置取决于您的操作模式。

- 单机服务器： EAP_HOME/standalone/log/server.log
- 受管域： EAP_HOME/domain/servers/SERVER_NAME/log/server.log

此文件通常称为服务器日志。如需更多信息，请参阅 Root Logger 部分。

11.1.2. 引导日志记录

在启动期间，JBoss EAP 会记录有关 Java 环境和每个服务的启动信息。此日志在故障排除时非常有用。默认情况下，所有日志条目写入服务器日志中。

启动日志记录配置在 logging.properties 配置文件中指定，该文件处于活动状态，直到 JBoss EAP 日志记录 子系统启动并接管为止。该文件的位置取决于您的操作模式。

- 单机服务器： EAP_HOME/standalone/configuration/logging.properties
- 受管域：

  域控制器和每一服务器都有一个 logging.properties 文件。

  - 域控制器： EAP_HOME/domain/configuration/logging.properties

- **Server: EAP_HOME/domain/servers/SERVER_NAME/data/logging.properties**

警告

*建议您不要直接编辑 **logging.properties** 文件，除非您了解需要它的特定用例。在进行此操作前，建议您从红帽客户门户网站创建一个支持问题单。*

*在启动时会覆盖对 **logging.properties** 文件手动进行的更改。*

11.1.2.1. 查看引导错误

*对 **JBoss EAP** 进行故障排除时，首先要执行的步骤之一是检查启动过程中发生的错误。然后，您可以使用提供的信息来诊断和解决其原因。在对启动错误进行故障排除时，打开支持案例。*

*有两种方法可以查看引导错误，每种方法都各有优点。您可以使用 **read-boot-errors** 管理 CLI 命令检查 **server.log** 文件或读取引导错误。*

检查服务器日志文件

*您可以打开 **server.log** 文件，以查看启动期间发生的任何错误。*

通过这个方法，您可以查看每个错误消息以及可能的相关信息，以便您获取有关为何发生错误的更多信息。它还允许您以纯文本格式查看错误消息。

1. *在文件查看器中打开 **server.log** 文件。*
2. *导航到文件的末尾。*
3. *向后搜索 **WFLYSRV0049** 消息标识符，该标识符标记最新启动序列的开始。*
4. *从该刻搜索日志，以查找 **for ERROR** 实例。每个实例都将包含错误的描述并列出涉及的模*

块。

以下是 **server.log** 日志文件中的示例错误描述：

```
2016-03-16 14:32:01,627 ERROR [org.jboss.msc.service.fail] (MSC service thread 1-7) MSC000001:
Failed to start service jboss.undertow.listener.default: org.jboss.msc.service.StartException in service
jboss.undertow.listener.default: Could not start http listener
        at org.wildfly.extension.undertow.ListenerService.start(ListenerService.java:142)
        at
org.jboss.msc.service.ServiceControllerImpl$StartTask.startService(ServiceControllerImpl.java:1948)
        at org.jboss.msc.service.ServiceControllerImpl$StartTask.run(ServiceControllerImpl.java:1881)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:745)
Caused by: java.net.BindException: Address already in use
        ...
```

从管理 **CLI** 中读取引导错误

您可以在服务器启动时使用 **read-boot-errors** 管理 **CLI** 命令查看错误。

此方法不需要访问服务器文件系统，这对于负责监控没有文件系统访问权限的错误的任何人都有用。由于它是管理 **CLI** 命令，因此可以在 脚本中使用。例如，您可以编写一个启动多个 **JBoss EAP** 实例的脚本，然后检查启动时发生的错误。

运行以下管理 **CLI** 命令：

```
/core-service=management:read-boot-errors
```

将列出启动过程中出现的任何错误。

```
{
    "outcome" => "success",
    "result" => [
        {
            "failed-operation" => {
                "operation" => "add",
                "address" => [
                    ("subsystem" => "undertow"),
                    ("server" => "default-server"),
                    ("http-listener" => "default")
                ]
            },
            "failure-description" => "{\"WFLYCTL0080: Failed services\" =>
{\"jboss.undertow.listener.default\" => \"org.jboss.msc.service.StartException in service
```

```
jboss.undertow.listener.default: Could not start http listener
    Caused by: java.net.BindException: Address already in use\"}}",
          "failed-services" => {"jboss.undertow.listener.default" =>
"org.jboss.msc.service.StartException in service jboss.undertow.listener.default: Could not start http
listener
    Caused by: java.net.BindException: Address already in use"}
       }
       ...
    ]
}
```

11.1.3. 垃圾收集器日志记录

垃圾回收日志记录将所有垃圾回收活动记录到纯文本日志文件。这些日志文件可用于诊断目的。默认情况下，除 **IBM Java** 开发工具包外，所有支持的配置上的 **JBoss EAP** 单机服务器都启用了垃圾回收日志记录。

垃圾收集日志的位置为 **EAP_HOME/standalone/log/gc.log.DIGIT.current**。垃圾回收日志的收集日志限制为 **3 MB**，最多五个文件会被轮转。

强烈建议启用垃圾回收日志记录，因为垃圾回收日志记录在故障排除中非常有用，它的开销应最小。但是，您可以在启动服务器前将 **GC_LOG** 变量设置为 **false**，为单机服务器禁用垃圾回收日志。例如：

```
$ export GC_LOG=false
$ EAP_HOME/bin/standalone.sh
```

11.1.4. 默认日志文件位置

为默认日志记录配置创建以下日志文件：默认配置使用定期日志处理程序写入服务器日志文件。

*表 **11.1.** 单机服务器的默认日志文件*

日志文件	描述
EAP_HOME/standalone/log/server.log	包含服务器日志消息，包括服务器启动消息。
EAP_HOME/standalone/log/gc.log.DIGIT.current	包含垃圾回收详细信息。

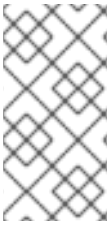
*表 **11.2.** 受管域的默认日志文件*

日志文件	描述
EAP_HOME/domain/log/host-controller.log	包含与主机控制器启动相关的日志消息。

日志文件	描述
EAP_HOME/domain/log/process-controller.log	包含与进程控制器启动相关的日志消息。
EAP_HOME/domain/servers/*SERVER_NAME*/log/server.log	包含指定服务器的日志消息，包括服务器启动消息。

11.1.5. 设置服务器的默认位置

*您可以通过在相应的启动配置文件中设置 **JVM** 属性来配置 **JBoss EAP** 的默认区域设置。启动配置文件是单机服务器的 **EAP_HOME/bin/standalone.conf**，或者受管域的 **EAP_HOME/bin/domain.conf**。*



注意

*对于 **Windows Server**，**JBoss EAP** 启动配置文件为 **standalone.conf.bat** 和 **domain.conf.bat**。*

*已国际化和本地化的日志消息将使用此默认区域设置。有关创建国际化日志消息的信息，请参见 **JBoss EAP** 开发指南。*

设置语言

*通过使用 **JAVA_OPTS** 变量设置 **user.language** 属性，以指定语言。例如，向启动配置文件添加以下行以设置法语区域设置：*

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=fr"
```

已国际化和本地化的日志消息现在将以法语输出。

设置语言和地区

除了语言之外，还可能需要通过设置 用户.国家属性来指定国家/ 地区。例如，在启动配置文件中添加以下行，为巴西设置葡萄牙区域设置：

```
JAVA_OPTS="$JAVA_OPTS -Duser.language=pt -Duser.country=BR"
```

已国际化和本地化的日志消息现在将在巴西葡萄牙语中输出。

*使用 **org.jboss.logging.locale** 属性设置服务器位置*

您可以配置 **org.jboss.logging.locale** 属性，以覆盖使用 **JBoss Logging** 记录的消息的区域设置，包括来自 **JBoss EAP** 的所有消息及其拥有的依赖关系。其他依赖项（如 **JSF**）无法获得覆盖的区域设置。

若要使用不同于系统默认值的区域设置启动 **JBoss EAP** 服务器，您可以编辑 **EAP_HOME/bin/standalone.conf** 或 **EAP_HOME/bin/domain.conf** 文件，具体取决于您的操作模式，并附加以下命令来为所需的区域设置 **JVM** 参数：属性的值必须以 **BCP 47** 格式指定。例如，要设置巴西葡萄牙语，请使用 **pt-BR**。

```
JAVA_OPTS="$JAVA_OPTS -Dorg.jboss.logging.locale=pt-BR"
```

11.2. 查看日志文件

查看服务器和应用程序日志非常重要，以帮助诊断错误、性能问题和其他问题。有些用户可能更喜欢直接在服务器文件系统中查看日志。对于无法直接访问文件系统或更喜欢图形界面的用户，**JBoss EAP** 允许您从管理控制台查看日志。您还可以使用管理 **CLI** 查看日志。

若要从其中一个管理接口访问日志，它必须位于服务器的 **jboss.server.log.dir** 属性指定的目录中，并且定义为文件、定期轮转、大小轮转或定期轮转日志处理程序。**RBAC** 角色分配也会被遵守，因此登录管理控制台或 **CLI** 的用户只能查看被授权访问的日志。

从管理控制台查看日志

您可以直接从管理控制台查看日志。

1. 选择 **Runtime** 选项卡，然后选择相应的服务器。
2. 选择 **Log Files**，然后从列表中选择一个日志文件。
3. 单击 **View** 查看和搜索日志内容，或者从下拉菜单中选择 **Download** 以将日志文件下载到本地文件系统。

警告

*管理控制台日志查看器不应成为文本编辑器，用于查看非常大的日志文件，例如大于 **100MB**。系统会提示您确认您是否尝试打开大于 **15MB** 的日志文件。在管理控制台中打开一个非常大的文件可能会使浏览器崩溃，因此您应始终在本地下载大型日志文件，并在文本编辑器中打开这些文件。*

*通过管理 **CLI** 查看日志*

*您可以使用 **read- log-file** 命令从管理 **CLI** 中读取 日志文件的内容。默认情况下，这会显示指定日志文件的最后 **10** 行。*

```
/subsystem=logging/log-file=LOG_FILE_NAME:read-log-file
```

注意

*在受管域中，在此命令前加上 **/host=HOST_NAME/server=SERVER_NAME**。*

您可以使用以下参数自定义日志输出：

编码

用于读取文件的字符编码。

行

*要从文件中读取的行数。值 **-1** 将读取所有日志行。默认值为 **10**。*

skip

*阅读前要跳过的行数。默认为 **0**。*

tail

*是否从文件末尾读取 **.**默认值为 **true**。*

*例如，以下管理 **CLI** 命令从 **server.log** 日志文件顶部读取前 **5** 行：*

```
/subsystem=logging/log-file=server.log:read-log-file(lines=5,tail=false)
```

这会生成以下输出：

```
{
    "outcome" => "success",
    "result" => [
        "2016-03-24 08:49:26,612 INFO  [org.jboss.modules] (main) JBoss Modules version 1.5.1.Final-redhat-1",
        "2016-03-24 08:49:26,788 INFO  [org.jboss.msc] (main) JBoss MSC version 1.2.6.Final-redhat-1",
        "2016-03-24 08:49:26,863 INFO  [org.jboss.as] (MSC service thread 1-7) WFLYSRV0049: JBoss EAP 7.0.0.GA (WildFly Core 2.0.13.Final-redhat-1) starting",
        "2016-03-24 08:49:27,973 INFO  [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating http management service using socket-binding (management-http)",
        "2016-03-24 08:49:27,994 INFO  [org.xnio] (MSC service thread 1-1) XNIO version 3.3.4.Final-redhat-1"
    ]
}
```

11.3. 关于 LOGGING 子系统

JBoss EAP logging 子系统使用 日志类别和日志 处理程序 系统进行配置。日志类别定义要捕获的消息。日志处理程序定义如何处理这些消息，例如写入磁盘或发送到控制台。

日志记录配置集允许创建唯一的日志配置集合，并将其分配给独立于任何其他日志配置的应用程序。记录配置文件的配置与主要的 **logging** 子系统几乎相同。

11.3.1. 根日志记录器

JBoss EAP root 日志记录器捕获指定日志级别或更高级别的所有日志消息，发送到未被日志类别捕获的服务器。

默认情况下，根日志记录器配置为使用控制台和定期日志处理程序。定期日志处理程序配置为写入 **server.log** 文件。此文件通常称为服务器日志。

如需更多信息，请参阅配置 **Root** 日志器。

11.3.2. 日志类别

日志类别定义要捕获的一组日志消息，以及一个或多个处理消息的日志处理程序。

要捕获的日志消息由指定的 **Java** 原始软件包和日志级别定义。来自该软件包及日志级别或更高级别的类的消息将被日志类别捕获并发送到指定的日志处理程序。

注意

虽然日志类别通常是 **Java** 软件包和类名称，但可以是 **Logger.getLogger(LOGGER_NAME)** 方法指定的任何名称。

日志类别可以选择使用根日志记录器的日志处理程序，而不使用自己的处理程序。

如需更多信息，请参阅配置日志类别。

11.3.3. 日志处理程序

日志处理程序定义如何记录捕获的日志消息。可用的日志处理程序类型有 **console**、**file**、**periodic**、**size**、**periodic size**、**syslog**、**custom** 和 **async**。

注意

日志处理程序必须添加到至少一个日志记录器才能处于活动状态。

日志处理程序类型

控制台（**Console**）

控制台日志处理程序将日志消息写入主机操作系统的标准输出、**stdout** 或标准 错误、**stderr**、流。在通过命令行提示符运行 **JBoss EAP** 时，将显示这些消息。除非操作系统配置为捕获标准输出或标准错误流，否则不会保存来自控制台日志处理程序的消息。

File

文件日志处理程序将日志消息写入到指定的文件中。

periodic

定期日志处理程序将日志消息写入到指定的文件，直到指定的时间段已过。过了时间段后，通过附加指定的时间戳来重命名文件，处理程序将继续写入到具有原始名称的新创建的日志文件中。

Size

大小日志处理程序将日志消息写入到指定的文件，直到文件达到指定的大小。当文件达到指定的大小时，将使用数字后缀对其进行重命名，处理程序将继续写入到具有原始名称的新创建的日志文件中。每个大小日志处理程序都必须以这种方式指定要保留的最大文件数。

定期大小

定期大小日志处理程序将日志消息写入到指定文件，直到文件达到指定的大小或指定的时间段已过。然后，文件重命名为，处理程序继续写入到具有原始名称的新创建的日志文件。

这是定期日志处理程序和大小日志处理程序的组合，并支持它们的组合属性。

Syslog

系统日志处理程序可用于发送消息到远程记录服务器。这使得多个应用可以发送其日志消息到同一服务器，它们都可以在上面一起解析。

socket

*套接字日志处理程序可用于通过套接字发送日志消息到远程记录服务器。这可以是 **TCP** 或 **UDP** 套接字。*

Custom

*通过自定义日志处理程序，您可以配置已实施的新类型的日志处理程序。自定义处理程序必须作为 **Java** 类实施，扩展 **java.util.logging.Handler** 并包含在模块中。您还可以使用 **Log4J** 附加程序作为自定义日志处理程序。*

Async

异步日志处理程序是一种打包程序日志处理程序，可为一个或多个其他日志处理程序提供异步行为。这可用于日志处理程序，它们可能具有较高的延迟或其他性能问题，例如将日志文件写入网络文件系统。

有关配置每个日志处理程序的详情，请参考 配置日志处理程序 部分。

11.3.4. 日志级别

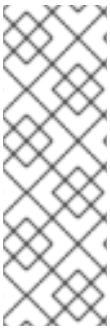
日志级别是一个枚举的值，指示日志消息的性质及严重性。作为开发人员，您可以使用所选日志记录框架的适当方法指定给定日志消息级别，以发送消息。

JBoss EAP 支持受支持的应用程序日志记录框架使用的所有日志级别。最常使用的从最低到最高日志级别为 TRACE、DEBUG、INFO、WARN、ERROR 和 FATAL。

日志类别和处理程序使用日志级别来限制它们所负责的消息。每个日志级别都有一个已分配的数字值，表示它相对于其他日志级别的顺序。日志级别和处理程序分配有日志级别，它们仅处理该级别或更高级别的日志消息。例如，级别为WARN的日志处理程序将仅记录 levels WARN 、ERROR 和 FATAL 的消息。

支持的日志级别

日志级别	值	描述
ALL	Integer.MIN_VALUE	提供所有日志消息.
FINEST	300	-
FINER	400	-
TRACE	400	**TRACE** 级别日志消息提供关于应用的运行状态的详细信息，通常仅在调试期间捕获。
DEBUG	500	**DEBUG** 级别的日志消息指示各个请求或应用活动的进度，通常仅在调试期间捕获。
OK	500	-
CONFIG	700	-
INFO	800	**INFO** 级别的日志消息指示应用的整体进度。通常用于应用程序启动、关闭和其他主要生命周期事件。
WARN	900	**WARN** 级别的日志消息指出的情况是不出错，但不被视为理想情况。**WARN** 日志消息可能会指示将来可能导致错误的情况。
WARNING	900	-
ERROR	1000	**ERROR** 级别日志消息指出发生了阻止当前活动或请求完成但不会阻止应用运行的错误。
严重性	1000	-
FATAL	1100	**FATAL** 级别的日志消息指示可能导致关键服务故障和应用关闭的事件，并可能导致 JBoss EAP 关闭。
OFF	Integer.MAX_VALUE	不显示任何日志消息。



注意

ALL 是最低日志级别，包含所有日志级别的消息。这可提供最多的日志量。

FATAL 是最高级别，仅包含该级别的消息。这可提供最小的日志量。

11.3.5. 日志格式

格式器用于格式化日志消息。可以使用 **named-formatter** 属性将格式化器分配到日志记录处理程序。如需有关日志记录处理程序配置的更多信息，请参阅 配置日志处理程序。

logging 子系统包括四种格式：

- **Pattern Formatter**
- **JSON Formatter**
- **XML Formatter**
- 自定义格式

Pattern Formatter

模式格式器用于以纯文本格式记录日志消息。除了将 **formatter** 用作日志处理程序 的 **named-formatter** 属性外，它还可用作 **formatter** 属性，而无需先创建 **formatter** 资源。有关模式语法的更多信息，请参阅 **Pattern Formatter** 的格式字符。

有关如何配置模式格式化器的详情，请参阅配置模式格式器。

JSON Formatter

JSON 格式器用于在 **JSON** 中格式化日志消息。

如需有关如何配置 **JSON** 格式器的信息，请参阅配置 **JSON** 日志格式器。

XML Formatter

XML 日志格式器用于在 XML 中格式化日志消息。

有关如何配置 XML 日志格式器的详情，请参阅配置 XML 日志格式器。

自定义格式

用于处理程序的自定义格式器。请注意，大多数日志记录都采用 printf 格式格式化。格式器可能需要调用 org.jboss.logmanager.ExtLogRecord#getFormattedMessage（）才能正确格式化消息。

有关如何配置自定义日志格式器的详情，请参阅配置自定义日志格式器。

11.3.6. filter Expressions

使用 filter-spec 属性配置的过滤器表达式用于根据各种条件记录日志消息。过滤器检查始终在原始未格式化的消息上完成。您可以为日志记录器或处理程序包含过滤器，但日志记录器过滤器优先于处理程序上放置的过滤器。

注意

为根日志记录器指定的 filter-spec 不会被其他日志记录器继承。相反，每个处理程序都必须指定 filter-spec。

表 11.3. Logging 的 filter Expressions

filter Expression	描述
accept	接受所有日志消息。
deny	拒绝所有日志消息。
不 [filter 表达式]	返回单个过滤器表达式倒置的值。例如： **not(match("WFLY"))**
all[filter 表达式]	从以逗号分隔的过滤器表达式列表中返回串联值。例如： **all(match("WFLY"),match("WELD"))**
任意[filter 表达式]	从以逗号分隔的过滤器表达式列表中返回一个值。例如： **any(match("WFLY"),match("WELD"))**

filter Expression	描述
levelChange[level]	使用指定的级别更新日志记录。例如： **levelChange(WARN)**
levels[levels]	使用逗号分隔级别列表中列出的级别过滤日志消息。例如： **级别（DEBUG、INFO、WARN、ERROR）**
levelRange[minLevel,maxLevel]	过滤指定级别的日志消息。**[** 和 **]** 字符用于表示包含的级别。（ 和 ） 字符用于表示专用级别。例如： ● **levelRange[INFO,ERROR]** ○ 最小级别必须大于或等于 **INFO**，最高级别必须小于或等于 **ERROR**。 ● **levelRange[DEBUG,ERROR)** ○ 最小级别必须大于或等于 **DEBUG**，最高级别必须小于 **ERROR**。
match["pattern"]	使用提供的正则表达式过滤日志消息。例如： **match("WFLY\d+")**
substitute["pattern","replacement value"]	使用替换文本（第二个参数）替换第一个匹配模式的过滤器（第一个参数）。例如： **substitute("WFLY","EAP")**
substituteAll["pattern","replacement value"]	将模式的所有匹配项（第一个参数）替换为替换文本（第二个参数）的过滤器。例如： **substituteAll("WFLY","EAP")**

注意

*在使用管理 **CLI** 配置过滤器表达式时，请确保在过滤器文本中转义逗号和引号，以便该值作为字符串正确处理。您必须在逗号和引号前加上反斜杠(\)，并将整个表达式括在引号中。以下是正确转义 **replace All**（**"WFLY","YLFW"**的示例。*

```
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=filter-spec,
value="substituteAll(\"WFLY\"\,\"YLFW\")")
```

11.3.7. 隐式日志记录依赖项

*默认情况下，**JBoss EAP** 日志记录 子系统为部署添加隐式日志记录 **API** 依赖项。您可以使用 **add-logging-api-dependencies** 属性（默认为 **true** ）来控制这些隐式依赖项是否添加到部署中。*

使用管理 **CLI**，您可以将 **add-logging-api-dependencies** 属性设置为 **false**，从而不会将隐式日志记录 **API** 依赖项添加到部署中。

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies, value=false)
```

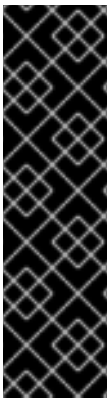
有关 **logging** 子系统隐式依赖项的信息，请参阅 **JBoss EAP** 开发指南 中的 **Implicit Module** 依赖项部分。

11.4. 配置日志类别

本节介绍如何使用管理 **CLI** 配置日志类别。您还可以使用管理控制台配置日志类别，方法是导航到 **Configuration** → **Subsystems** → **Logging** → **Configuration**，点 **View**，然后选择 **Categories**。

在配置日志类别时要执行的主要任务有：

- 添加新日志类别。
- 配置日志类别设置。
- 将日志处理程序分配到日志类别。



重要

如果您要为日志记录配置集配置此日志类别，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加日志类别

日志类别名称由 **Java** 原始软件包定义。来自该软件包中类的消息将捕获，只要它们遵循其他设置，例如日志级别。

```
/subsystem=logging/logger=LOG_CATEGORY:add
```

配置日志类别设置

根据您的需要，您可能需要设置一个或多个以下日志类别属性：有关可用日志类别属性及其描述的完整列表，请参阅 ***Log Category Attributes***。

- *设置日志级别。*

  为日志级别设置适当的日志级别。默认值为 ***ALL***。*有关所有可用选项，请参阅日志级别。*

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=level,value=LEVEL)
  ```

- *设置此类别是否应使用根日志记录器的日志处理程序。*

  默认情况下，日志类别除使用根日志记录器的处理程序外，还会使用自己的处理程序。如果日志类别应仅使用其分配的处理程序，则将 ***use-parent-handlers*** *属性设为* ***false***。

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=use-parent-
  handlers,value=USE_PARENT_HANDLERS)
  ```

- *设置过滤器表达式。*

  设置表达式，以过滤日志类别的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 ***not （match("WFLY")***，*需要将以下* ***FILTER_EXPRESSION*** *可替换变量替换为****"not（match(\" WFLY\")"***。

  ```
  /subsystem=logging/logger=LOG_CATEGORY:write-attribute(name=filter-spec,
  value=FILTER_EXPRESSION)
  ```

  有关可用过滤器表达式的更多信息，请参阅 ***Filter Expressions*** *部分。*

分配处理程序

将日志处理程序分配到日志类别。

```
/subsystem=logging/logger=LOG_CATEGORY:add-handler(name=LOG_HANDLER_NAME)
```

删除日志类别

可以使用 **remove** 操作删除日志类别。

```
/subsystem=logging/logger=LOG_CATEGORY:remove
```

11.5. 配置日志处理程序

日志处理程序定义如何记录捕获的日志消息。有关配置您需要的日志处理程序类型，请参见相应的部分。

- 控制台日志处理程序
- 文件日志处理程序
- 定期轮转日志处理程序
- 大小轮转日志处理程序
- 定期轮转日志处理程序
- **syslog** 处理程序
- 套接字日志处理程序
- 自定义日志处理程序
- **async** 日志处理程序

11.5.1. 配置控制台日志处理程序

本节介绍如何使用管理 **CLI** 配置控制台日志处理程序。您还可以使用管理控制台配置控制台日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选

择 **Handler → Console Handler**。

在配置控制台日志处理程序时要执行的主要任务有：

- 添加新的控制台日志处理程序。
- 配置控制台日志处理程序设置。
- 将控制台日志处理程序分配到日志记录器。

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加控制台日志处理程序

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:add
```

配置控制台日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个控制台日志处理程序属性：有关可用控制台日志处理程序属性及其描述的完整列表，请参阅 **Console Log Handler Attributes**。

- 设置日志级别。

  为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

  ```
  /subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
  ```

- 设置目标。

设置处理程序的目标，可以是 **System.out**、**System.err** 或 **console** 之一。默认值为 **System.out**。

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=target,value=TARGET)
```

- 设置编码.

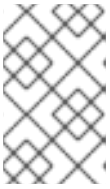
  设置处理程序的编码，如 **utf-8**。

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- 设置日志格式器。

  设置处理程序的格式字符串。例如，默认格式字符串为 **%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n**。务必在引号中包含 **FORMAT** 值。

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



注意

如果要引用 保存的格式器，请使用 **named-formatter** 属性。

- 设置自动刷新.

  设置是否在每次写入后自动清空。默认值为 **true**。

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")**，需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

-

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:write-
attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将控制台日志处理程序分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将控制台日志处理程序分配到根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CONSOLE_HANDLER_NAME)
```

以下管理 **CLI** 命令将控制台日志处理程序分配到其名称由 **CATEGORY** 指定的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CONSOLE_HANDLER_NAME)
```

删除控制台日志处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。

```
/subsystem=logging/console-handler=CONSOLE_HANDLER_NAME:remove
```

11.5.2. 配置文件日志处理程序

本节介绍如何使用管理 **CLI** 配置文件日志处理程序。您还可以使用管理控制台配置文件日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → File Handler**。

在配置文件日志处理程序时要执行的主要任务有：

- 添加新的文件日志处理程序。
- 配置文件日志处理程序设置。

- 将文件日志处理程序分配到日志记录器。

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加文件日志处理程序

在添加文件日志处理程序时，您必须使用 **file** 属性指定文件路径，该属性由 **path** 和 **relative -to** 属性组成。使用 **path** 属性设置日志的文件路径，包括名称，如 **my-log.log**。（可选）使用 **relative-to** 属性设置 路径 相对于指定路径，如 **jboss.server.log.dir**。

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

配置文件日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个文件日志处理程序属性：有关可用文件日志处理程序属性及其描述的完整列表，请参阅 **File Log Handler Attributes**。

- 设置日志级别。

为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- 设置附加行为。

默认情况下，**JBoss EAP** 将在服务器重启时将日志消息附加到同一文件中。您可以将 **append** 属性设置为 **false**，使其在服务器重启时覆盖文件。

```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- *设置编码.*

  设置处理程序的编码，如 ***utf-8****。*

  ```
  /subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
  ```

- *设置日志格式器。*

  设置处理程序的格式字符串。例如，默认格式字符串为 ***%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n****。务必在引号中包含* ***FORMAT*** *值。*

  ```
  /subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
  ```

  注意

  如果要引用 保存的格式器，请使用 ***named-formatter*** *属性。*

- *设置自动刷新.*

  设置是否在每次写入后自动清空。默认值为 ***true****。*

  ```
  /subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
  ```

- *设置过滤器表达式。*

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 ***not （match("WFLY")，*** *需要将以下* ***FILTER_EXPRESSION*** *可替换变量替换为****"not（match(\" WFLY\")"****。*

  ```
  /subsystem=logging/file-handler=FILE_HANDLER_NAME:write-attribute(name=filter-spec,value=FILTER_EXPRESSION)
  ```

  有关可用过滤器表达式的更多信息，请参阅 ***Filter Expressions*** *部分。*

将文件日志处理程序分配给日志器

要激活日志处理程序，您必须将其分配到日志记录器。

*以下管理 **CLI** 命令将文件日志处理程序分配到根日志记录器：*

```
/subsystem=logging/root-logger=ROOT:add-handler(name=FILE_HANDLER_NAME)
```

*以下管理 **CLI** 命令将文件日志处理程序分配到其名称由 **CATEGORY** 指定的日志记录器：*

```
/subsystem=logging/logger=CATEGORY:add-handler(name=FILE_HANDLER_NAME)
```

删除文件日志处理程序

*可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。*

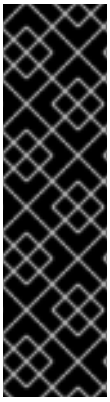
```
/subsystem=logging/file-handler=FILE_HANDLER_NAME:remove
```

11.5.3. 配置定期轮转日志处理程序

*本节介绍如何使用管理 **CLI** 配置定期轮转日志处理程序。您还可以使用管理控制台配置定期日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Periodic Handler**。*

用于配置定期日志处理器的主要任务有：

- *添加新的定期日志处理程序。*
- *配置定期日志处理程序设置。*
- *将定期日志处理程序分配到日志记录器。*



重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加定期日志处理程序

在添加定期日志处理程序时，您必须使用 **file** 属性指定文件路径，该属性由 **path** 和 **relative -to** 属性组成。使用 **path** 属性设置日志的文件路径，包括名称，如 **my-log.log**。（可选）使用 **relative-to** 属性设置 路径 相对于指定路径，如 **jboss.server.log.dir**。

您还必须使用 **suffix** 属性为轮转日志设置后缀。这的格式必须可以被 **java.text.SimpleDateFormat**（如 **.yyyyyy-MM-dd-HH**） 理解。轮转期间会基于此后缀自动计算。

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:add(file=
{path=FILE_PATH,relative-to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

配置定期日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个定期日志处理程序属性：有关可用定期日志处理程序属性及其描述的完整列表，请参阅 **Periodic Log Handler Attributes**。

- 设置日志级别。

  为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- 设置附加行为。

  默认情况下，**JBoss EAP** 将在服务器重启时将日志消息附加到同一文件中。您可以将 **append** 属性设置为 **false**，使其在服务器重启时覆盖文件。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-
  attribute(name=append,value=APPEND)
  ```

- 设置编码.

  设置处理程序的编码，如 **utf-8**。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
  ```

- 设置日志格式器。

  设置处理程序的格式字符串。例如，默认格式字符串为 **%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n**。务必在引号中包含 **FORMAT** 值。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
  ```

  注意

  如果要引用 保存的格式器，请使用 **named-formatter** 属性。

- 设置自动刷新.

  设置是否在每次写入后自动清空。默认值为 **true**。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
  ```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")**，需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

  ```
  /subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
  ```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将定期日志处理程序分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

*以下管理 **CLI** 命令将定期日志处理程序分配到根日志记录器：*

```
/subsystem=logging/root-logger=ROOT:add-handler(name=PERIODIC_HANDLER_NAME)
```

*以下管理 **CLI** 命令将定期日志处理程序分配到其名称由 **CATEGORY** 指定的日志记录器：*

```
/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_HANDLER_NAME)
```

删除定期日志处理程序

*可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。*

```
/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:remove
```

11.5.4. 配置大小轮转日志处理程序

*本节介绍如何使用管理 **CLI** 配置大小轮转日志处理程序。您还可以使用管理控制台配置大小日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Size Handler**。*

在配置大小日志处理器时要执行的主要任务有：

- *添加新的大小日志处理程序。*
- *配置大小日志处理程序设置。*
- *将大小日志处理程序分配到日志记录器。*

重要

*如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。*

*此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。*

添加大小日志处理程序

*在添加大小日志处理程序时，您必须使用 **file** 属性指定文件路径，该属性由 **path** 和 **relative -to** 属性组成。使用 **path** 属性设置日志的文件路径，包括名称，如 **my-log.log**。（可选）使用 **relative-to** 属性设置 路径 相对于指定路径，如 **jboss.server.log.dir**。*

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:add(file=
{path=FILE_PATH,relative-to=RELATIVE_TO_PATH})
```

配置大小日志处理程序设置

*根据您的需要，您可能需要设置以下一个或多个大小日志处理程序属性：有关可用大小日志处理程序属性及其描述的完整列表，请参阅 **Size Log Handler Attributes**。*

- *设置日志级别。*

  *为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。*

  ```
  /subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- *设置轮转日志的后缀。*

  *设置后缀字符串，格式为 **java.text.SimpleDateFormat**，如 **.yyyyy-MM-dd-HH**。轮转期间会基于此后缀自动计算。*

  ```
  /subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-
  attribute(name=suffix, value=SUFFIX)
  ```

- *设置轮转大小。*

设置在轮转之前文件能够达到的最大大小。默认值为 **2m (2 MB)**。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-size, value=ROTATE_SIZE)
```

- 设置要保留的最大备份日志数。

  设置要保留的备份数。默认值为 **1**。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=max-backup-index, value=MAX_BACKUPS)
```

- 设置是否在引导时轮转日志。

  默认情况下，服务器重启时不会创建新的日志文件。您可以将此项设置为 **true**，以在服务器重启时轮转日志。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot, value=ROTATE_ON_BOOT)
```

- 设置附加行为。

  默认情况下，**JBoss EAP** 将在服务器重启时将日志消息附加到同一文件中。您可以将 **append** 属性设置为 **false**，使其在服务器重启时覆盖文件。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- 设置编码.

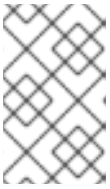
  设置处理程序的编码，如 **utf-8**。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- 设置日志格式器。

设置处理程序的格式字符串。例如，默认格式字符串为 **%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n**。务必在引号中包含 **FORMAT** 值。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



注意

如果要引用 保存的格式器，请使用 **named-formatter** 属性。

- 设置自动刷新.

  设置是否在每次写入后自动清空。默认值为 **true**。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")，** 需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将 **Size Log Handler** 分配给日志器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将大小日志处理程序分配到根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SIZE_HANDLER_NAME)
```

以下管理 **CLI** 命令将大小日志处理程序分配给名称由 **CATEGORY** 指定的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SIZE_HANDLER_NAME)
```

删除大小日志处理程序

*可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。*

```
/subsystem=logging/size-rotating-file-handler=SIZE_HANDLER_NAME:remove
```

11.5.5. 配置 Periodic 大小轮转日志处理程序

*本节介绍如何使用管理 **CLI** 配置定期大小轮转日志处理程序。您还可以通过使用管理控制台配置定期大小日志处理程序，导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Periodic Size Handler**。*

用于配置定期大小日志处理器的主要任务有：

- *添加新的定期大小日志处理程序。*
- *配置定期日志处理程序设置。*
- *将定期大小日志处理程序分配给日志记录器。*

重要

*如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。*

*此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。*

添加定期大小日志处理程序

*在添加定期大小日志处理程序时，您必须使用 **file** 属性指定文件路径，该属性由 **path** 和 **relative -to** 属性组成。使用 **path** 属性设置日志的文件路径，包括名称，如 **my-log.log**。（可选）使用 **relative-to** 属性设置 路径 相对于指定路径，如 **jboss.server.log.dir**。*

您还必须使用 **suffix** 属性为轮转日志设置后缀。这的格式必须可以被 **java.text.SimpleDateFormat**（如 **.yyyyyy-MM-dd-HH**）理解。轮转期间会基于此后缀自动计算。

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH},suffix=SUFFIX)
```

配置定期大小日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个定期大小日志处理程序属性：有关可用定期大小日志处理程序属性及其描述的完整列表，请参阅 **Periodic Sandler Attributes**。

- 设置日志级别。

  为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

  ```
  /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
  ```

- 设置轮转大小。

  设置在轮转之前文件能够达到的最大大小。默认值为 **2m (2 MB)**。

  ```
  /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-size,value=ROTATE_SIZE)
  ```

- 设置要保留的最大备份日志数。

  设置要保留的备份数。默认值为 **1**。

  ```
  /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=max-backup-index,value=MAX_BACKUPS)
  ```

- 设置是否在引导时轮转日志。

  默认情况下，服务器重启时不会创建新的日志文件。您可以将此项设置为 **true**，以在服务器重启时轮转日志。

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=rotate-on-boot, value=ROTATE_ON_BOOT)
```

- *设置附加行为。*

  *默认情况下，**JBoss EAP** 将在服务器重启时将日志消息附加到同一文件中。您可以将 **append** 属性设置为 **false**，使其在服务器重启时覆盖文件。*

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=append,value=APPEND)
```

- *设置编码.*

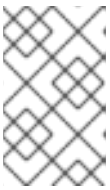
  *设置处理程序的编码，如 **utf-8**。*

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- *设置日志格式器。*

  *设置处理程序的格式字符串。例如，默认格式字符串为 **%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n**。务必在引号中包含 **FORMAT** 值。*

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```



注意

*如果要引用 保存的格式器，请使用 **named-formatter** 属性。*

- *设置自动刷新.*

  *设置是否在每次写入后自动清空。默认值为 **true**。*

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")**，需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:write-attribute(name=filter-spec,value=FILTER_EXPRESSION)
```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将 **Periodic Size Log Handler** 分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将定期大小日志处理程序分配给根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

以下管理 **CLI** 命令将定期大小日志处理程序分配给由 **CATEGORY** 指定名称的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_SIZE_HANDLER_NAME)
```

删除定期大小日志处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。

```
/subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE_HANDLER_NAME:remove
```

11.5.6. 配置 Syslog 处理程序

本节介绍如何使用管理 **CLI** 配置 **syslog** 处理程序，该 **CLI** 可用于将消息发送到支持 **Syslog** 协议的远程记录服务器，可以是 **RFC-3164** 或 **RFC-5424**。您还可以使用管理控制台配置 **syslog** 处理程序，方法

是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Syslog Handler**。

在配置 **syslog** 处理程序时要执行的主要任务有：

- 添加新的 **syslog** 处理程序。
- 配置 **syslog** 处理程序设置。
- 将 **syslog** 处理程序分配给日志记录器。

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加 **Syslog** 处理程序

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:add
```

配置 **Syslog** 处理程序设置

根据您的需要，您可能需要设置以下一个或多个 **syslog** 处理程序属性：有关可用 **syslog** 处理程序属性及其描述的完整列表，请参阅 **Syslog Handler Attributes**。

- 设置处理程序的日志级别。默认级别为 **ALL**。有关所有可用选项，请参阅日志级别。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
```

- 设置正在记录的应用的名称。默认名称为 **java**。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=app-name,value=APP_NAME)
```

- 设置 **syslog** 服务器的地址。默认地址为 **localhost**。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=server-address,value=SERVER_ADDRESS)
```

- 设置 **syslog** 服务器的端口。默认端口为 **514**。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=port,value=PORT)
```

- 设置 **syslog** 格式，如 **RFC** 规范所定义。默认格式为 **RFC5424**。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=syslog-format,value=SYSLOG_FORMAT)
```

- 指定 **named-formatter** 属性，以格式化 **syslog** 有效负载的消息。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:write-attribute(name=named-formatter, value=FORMATTER_NAME)
```

将 **Syslog Handler** 分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将 **syslog** 处理程序分配给根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SYSLOG_HANDLER_NAME)
```

以下管理 **CLI** 命令将 **syslog** 处理程序分配给名称由 **CATEGORY** 指定的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SYSLOG_HANDLER_NAME)
```

删除 **Syslog** 处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。

```
/subsystem=logging/syslog-handler=SYSLOG_HANDLER_NAME:remove
```

11.5.7. 配置套接字日志处理程序

本节介绍如何使用管理 **CLI** 配置套接字日志处理程序，它可用于通过套接字发送消息。这可以是 **TCP** 或 **UDP** 套接字。您还可以使用管理控制台配置套接字日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Socket Handler**。

注意

如果服务器以 **admin-only** 模式启动，则会丢弃日志消息。

配置套接字日志处理器时要执行的主要任务有：

- 添加套接字绑定。
- 添加日志格式器。
- 添加 **socket** 日志处理程序 并配置其设置。
- 将套接字日志处理程序分配到日志记录器。

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加 **Socket Binding**

定义 **remote-destination-outbound-socket-binding** 或 **local-destination-outbound-socket-binding**，作为 要使用的套接字绑定。

```
/socket-binding-group=SOCKET_BINDING_GROUP/remote-destination-outbound-socket-binding=SOCKET_BINDING_NAME:add(host=HOST, port=PORT)
```

添加日志格式

*定义要使用的日志格式器，如 **JSON** 格式器。*

```
/subsystem=logging/json-formatter=FORMATTER:add
```

添加套接字日志处理程序

在添加套接字日志处理程序时，您必须指定要使用的套接字绑定和格式化器。

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:add(outbound-socket-binding-ref=SOCKET_BINDING_NAME,named-formatter=FORMATTER)
```

配置套接字日志处理程序设置

*根据您的需要，您可能需要设置以下一个或多个套接字日志处理程序属性：有关可用套接字日志处理程序属性及其描述的完整列表，请参阅 **Socket Log Handler Attributes**。*

- *设置协议.*

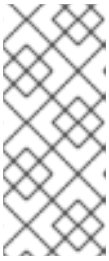
  *设置要使用的协议。默认值为 **TCP**。*

  ```
  /subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=protocol,value=PROTOCOL)
  ```

- *设置日志级别。*

  *为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。*

  ```
  /subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=level,value=LEVEL)
  ```

  注意

  *在服务器启动期间，套接字日志处理程序处理的日志消息排入队列，直到配置了套接字绑定并且初始化 **logging** 子系统。如果日志级别设置为低级别，如 **TRACE** 或 **DEBUG**，这可能会导致启动时消耗大量内存。*

- *设置编码.*

设置处理程序的编码，如 **utf-8**。

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=encoding,value=ENCODING)
```

- 设置自动刷新.

  设置是否在每次写入后自动清空。默认值为 **true**。

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=autoflush,value=AUTO_FLUSH)
```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")**，需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将 **Socket Log Handler** 分配给 **Logger**

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将套接字日志处理程序分配到根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=SOCKET_HANDLER_NAME)
```

以下管理 **CLI** 命令将套接字日志处理程序分配给名称由 **CATEGORY** 指定的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=SOCKET_HANDLER_NAME)
```

删除套接字日志处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。

```
/subsystem=logging/socket-handler=SOCKET_HANDLER_NAME:remove
```

使用 **SSL/TLS** 通过套接字发送日志消息

下列步骤演示了如何设置套接字日志处理程序以使用 **SSL_TCP** 协议通过套接字发送日志消息的示例。本例在 **elytron** 子系统中配置密钥存储、信任管理器和客户端 **SSL** 上下文，供套接字日志处理程序使用。来自根日志记录器的日志消息通过指定的套接字发送，格式为 **JSON** 格式。

有关配置 **Elytron** 组件的更多信息，请参阅如何为 **JBoss EAP** 配置服务器安全性的 **Elytron Subsystem**。

1. 配置 **Elytron** 设置。

   a. 添加 密钥存储。

   ```
   /subsystem=elytron/key-store=log-server-ks:add(path=/path/to/keystore.jks, type=JKS, credential-reference={clear-text=mypassword})
   ```

   b. 添加信任管理器。

   ```
   /subsystem=elytron/trust-manager=log-server-tm:add(key-store=log-server-ks)
   ```

   c. 添加客户端 **SSL** 上下文。

   ```
   /subsystem=elytron/client-ssl-context=log-server-context:add(trust-manager=log-server-tm, protocols=["TLSv1.2"])
   ```

2. 添加套接字绑定。

   ```
   /socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=log-server:add(host=localhost, port=4560)
   ```

3. 添加 **JSON** 格式器。

   ```
   /subsystem=logging/json-formatter=json:add
   ```

4. 添加 **socket** 日志处理程序。

```
/subsystem=logging/socket-handler=log-server-handler:add(named-formatter=json,
level=INFO, outbound-socket-binding-ref=log-server, protocol=SSL_TCP, ssl-context=log-
server-context)
```

5. 将日志处理程序分配到根日志记录器。

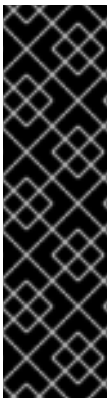
```
/subsystem=logging/root-logger=ROOT:add-handler(name=log-server-handler)
```

11.5.8. 配置自定义日志处理程序

本节介绍如何使用管理 **CLI** 配置自定义日志处理程序。您还可以使用管理控制台配置自定义日志处理程序，方法是导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Handler → Custom Handler**。

用于配置自定义日志处理器的主要任务有：

- 添加一个新的自定义日志处理程序。
- 配置自定义日志处理程序设置。
- 将自定义日志处理程序分配到日志记录器。



重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加自定义日志处理程序

在添加自定义日志处理程序时，您必须指定处理程序的 **Java** 类以及包含它的 **JBoss EAP** 模块。该类必须扩展 **java.util.logging.Handler**。

注意

您必须已创建了包含自定义日志记录器的模块，否则此命令将失败。

```
/subsystem=logging/custom-
handler=CUSTOM_HANDLER_NAME:add(class=CLASS_NAME,module=MODULE_NAME)
```

配置自定义日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个自定义日志处理程序属性：有关可用自定义日志处理程序属性及其描述的完整列表，请参阅自定义日志处理程序属性。

- 设置日志级别。

  为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

  ```
  /subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- 设置属性.

  设置日志处理程序所需的属性。属性必须能够通过 **setter** 方法访问。

  ```
  /subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-
  attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
  ```

- 设置编码.

  设置处理程序的编码，如 **utf-8**。

  ```
  /subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-
  attribute(name=encoding,value=ENCODING)
  ```

- 设置日志格式器。

  设置处理程序的格式字符串。例如，默认格式字符串为 **%d{HH:mm:ss,SSS} %-5p [%c] (%t)%s%e%n**。务必在引号中包含 **FORMAT** 值。

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=formatter,value=FORMAT)
```

注意

如果要引用保存的格式器，请使用 **named-formatter** 属性。

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")，** 需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:write-attribute(name=filter-spec, value=FILTER_EXPRESSION)
```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将自定义日志处理程序分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将自定义日志处理程序分配到根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=CUSTOM_HANDLER_NAME)
```

以下管理 **CLI** 命令将自定义日志处理程序分配到其名称由 **CATEGORY** 指定的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=CUSTOM_HANDLER_NAME)
```

删除自定义日志处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器或异步日志处理程序，则无法移除日志处理程序。

```
/subsystem=logging/custom-handler=CUSTOM_HANDLER_NAME:remove
```

11.5.9. 配置 Async 日志处理程序

本节介绍如何使用管理 CLI 配置 async 日志处理程序。您还可以通过使用管理控制台配置 async 日志处理程序，导航到 Configuration → Subsystems → Logging → Configuration，点 View，然后选择 Handler → Async Handler。

在配置 async 日志处理程序时要执行的主要任务是：

- 添加新的 async 日志处理程序。
- 将子处理程序添加到 async 日志处理程序。
- 配置 async 日志处理程序设置。
- 将 async 日志处理程序分配到日志记录器。

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 /subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/ 而不是 /subsystem=logging/。

此外，如果您在受管域中运行，请在命令前加上 /profile=PROFILE_NAME。

添加 Async 日志处理程序

在添加 async 日志处理程序时，您必须指定队列长度。这是队列中可以保留的最大日志请求数。

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add(queue-
length=QUEUE_LENGTH)
```

添加 Sub-handler

您可以添加一个或多个处理程序作为此 async 日志处理程序的子句柄。请注意，处理程序必须已存在于配置中，否则此命令将失败。

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:add-
handler(name=HANDLER_NAME)
```

配置 **Async** 日志处理程序设置

根据您的需要，您可能需要设置以下一个或多个 **async** 日志处理程序属性：有关可用 **async** 日志处理程序属性及其描述的完整列表，请参阅 **Async Log Handler Attributes**。

- 设置日志级别。

  为处理程序设置适当的日志级别。默认值为 **ALL**。有关所有可用选项，请参阅日志级别。

  ```
  /subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-
  attribute(name=level,value=LEVEL)
  ```

- 设置溢出操作。

  设置溢出时要执行的操作。默认值为 **BLOCK**，即当有完整队列时线程会阻止。您可以将此值更改为 **DISCARD**，这意味着如果已满队列，日志消息将被丢弃。

  ```
  /subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-
  attribute(name=overflow-action,value=OVERFLOW_ACTION)
  ```

- 设置过滤器表达式。

  设置表达式，以过滤处理程序的日志消息。务必用引号转义任何逗号和引号。例如，对于过滤器表达式 **not （match("WFLY")**，需要将以下 **FILTER_EXPRESSION** 可替换变量替换为**"not（match(\" WFLY\")"**。

  ```
  /subsystem=logging/async-handler=ASYNC_HANDLER_NAME:write-attribute(name=filter-
  spec, value=FILTER_EXPRESSION)
  ```

  有关可用过滤器表达式的更多信息，请参阅 **Filter Expressions** 部分。

将 **Async** 日志处理程序分配给日志记录器

要激活日志处理程序，您必须将其分配到日志记录器。

以下管理 **CLI** 命令将 **async** 日志处理程序分配给根日志记录器：

```
/subsystem=logging/root-logger=ROOT:add-handler(name=ASYNC_HANDLER_NAME)
```

以下管理 **CLI** 命令将 **async** 日志处理程序分配给由 **CATEGORY** 指定名称的日志记录器：

```
/subsystem=logging/logger=CATEGORY:add-handler(name=ASYNC_HANDLER_NAME)
```

删除 **Async** 日志处理程序

可以使用 **remove** 操作移除日志处理程序。如果当前分配给日志记录器，则无法移除日志处理程序。

```
/subsystem=logging/async-handler=ASYNC_HANDLER_NAME:remove
```

11.6. 配置根日志记录器

root 日志记录器捕获指定日志级别或更高级别的所有日志消息，发送到未被日志类别捕获的服务器。

本节介绍如何使用管理 **CLI** 配置根日志记录器。您还可以通过导航到 **Configuration → Subsystems → Logging → Configuration**，点 **View**，然后选择 **Root Logger**，使用管理控制台配置根日志记录器。

配置 **Root** 日志器

重要

如果您要为日志记录配置集配置此日志处理程序，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

1. 将日志处理程序分配到根日志记录器。

   添加日志处理程序。

   ```
   /subsystem=logging/root-logger=ROOT:add-handler(name=LOG_HANDLER_NAME)
   ```

   移除日志处理程序。

-

```
/subsystem=logging/root-logger=ROOT:remove-handler(name=LOG_HANDLER_NAME)
```

2. 设置日志级别。

```
/subsystem=logging/root-logger=ROOT:write-attribute(name=level,value=LEVEL)
```

有关可用根日志记录器属性及其描述的完整列表，请参阅根日志记录器属性。

11.7. 配置日志格式

日志格式**ter** 定义来自该处理程序的日志消息的外观。日志记录 子系统允许您配置以下类型的日志格式器：

- **Pattern Formatter**
- **JSON Log Formatter**
- **XML** 日志格式
- 自定义日志格式

11.7.1. 配置模式格式

您可以创建名为模式格式器，可在日志处理程序之间用于格式化日志消息。

重要

如果您要为日志记录配置集配置此日志格式器，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

创建模式格式

在定义模式格式器时，您可以提供一个用于格式化日志消息的模式字符串。如需有关模式语法的更多信息，请参阅 **Pattern Formatter** 的格式字符。

```
/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:add(pattern=PATTERN)
```

例如，默认配置使用以下日志格式器字符串将消息记录到服务器日志： **%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c](%t)%s%s%e%e%n**。这会创建类似于下方格式的日志消息。

```
2016-03-18 15:49:32,075 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
```

您还可以定义颜色映射，为不同的日志级别分配颜色。格式为以逗号分隔的 **LEVEL** ：**COLOR** 列表。

- 有效的级别： **ter ter**、 **fine**、**config**、**trace**、**debug**、**info**、**warning** 、**error**、 **fatal**、严重
- 有效颜色：黑、绿色、红色、黄色、蓝色、**mag enta**、**c yan**、白色、亮色、亮度、 亮色、亮色、亮色、 亮色、亮色、亮色、亮色、 亮度、精英代理、 亮度、 亮度、 亮度

```
/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:write-attribute(name=color-map,value="LEVEL:COLOR,LEVEL:COLOR")
```

您还可以使用管理控制台配置模式日志格式器：

1. 在浏览器中打开管理控制台。
2. 选择 **Configuration** → **Subsystems** → **Logging**。
3. 选择 **Configuration**，然后单击 **View**。
4. 选择 **Formatter**，然后选择 **Pattern Formatter** 选项。

11.7.2. 配置 JSON 日志格式器

您可以创建一个 **JSON** 日志格式器，以将日志消息格式化为 **JSON**。

重要

如果您要为日志记录配置集配置此日志格式器，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

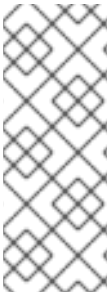
添加 **JSON** 日志格式

```
/subsystem=logging/json-formatter=JSON_FORMATTER_NAME:add(pretty-print=true, exception-output-type=formatted)
```

这会创建类似于下方格式的日志消息。

```
{
    "timestamp": "2018-10-18T13:53:43.031-04:00",
    "sequence": 62,
    "loggerClassName": "org.jboss.as.server.logging.ServerLogger_$logger",
    "loggerName": "org.jboss.as",
    "level": "INFO",
    "message": "WFLYSRV0025: JBoss EAP 7.3.0.GA (WildFly Core 10.0.0.Final-redhat-20190924) started in 5672ms - Started 495 of 679 services (331 services are lazy, passive or on-demand)",
    "threadName": "Controller Boot Thread",
    "threadId": 22,
    "mdc": {
    },
    "ndc": "",
    "hostName": "localhost.localdomain",
    "processName": "jboss-modules.jar",
    "processId": 7461
}
```

添加 **Logstash JSON** 日志格式**ter**



注意

您可以修改 **JSON** 日志格式输出键并添加静态元数据。**JSON** 日志格式器的主要用途是在 **JSON** 中格式化日志消息。**logstash** 会消耗这个 **JSON** 输出，并搜索字段 **@timestamp** 和 **@version**。以下示例创建了 **JSON** 日志格式，用于格式化 **Logstash** 的信息。

```
/subsystem=logging/json-formatter=logstash:add(exception-output-type=formatted, key-overrides=
[timestamp="@timestamp"], meta-data=[@version=1])
```

您可以使用 **JSON** 格式属性，如下所述：

- **key-overrides** 属性可用于覆盖定义的键的名称。
- 例外可以通过将 **exception-output-type** 属性设置为 格式化来格式化 为对象。
- 可以通过将 **exception-output-type** 属性设置为 **detailed** 来包含异常堆栈追踪。
- 通过将 **exception-output-type** 设置为 详细和格式，可以包含例外，作为对象和堆栈追踪。
- 可以使用 **meta-data** 属性将元数据添加到日志记录中。

如需有关 **JSON** 格式器属性的更多信息，请参阅 **JSON 日志格式ter Attributes**。

您还可以使用管理控制台配置 **JSON** 日志格式器：

1. 在浏览器中打开管理控制台。
2. 选择 **Configuration** → **Subsystems** → **Logging**。
3. 选择 **Configuration**，然后单击 **View**。
4. 选择 **Formatter**，然后选择 **JSON Formatter** 选项。

11.7.3. 配置 XML 日志格式

您可以创建一个 **XML** 日志格式器，以将日志消息格式化为 **XML**。

重要

如果您要为日志记录配置集配置此日志格式器，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

添加 **XML** 日志格式

```
/subsystem=logging/xml-formatter=XML_FORMATTER_NAME:add(pretty-print=true, exception-output-type=detailed-and-formatted)
```

这会创建类似于下方所示的日志消息。

```
<record>
  <timestamp>2018-10-18T13:55:53.419-04:00</timestamp>
  <sequence>62</sequence>
  <loggerClassName>org.jboss.as.server.logging.ServerLogger_$logger</loggerClassName>
  <loggerName>org.jboss.as</loggerName>
  <level>INFO</level>
  <message>WFLYSRV0025: JBoss EAP 7.3.0.GA (WildFly Core 10.0.0.Final-redhat-20190924) started in 6271ms - Started 495 of 679 services (331 services are lazy, passive or on-demand)</message>
  <threadName>Controller Boot Thread</threadName>
  <threadId>22</threadId>
  <mdc>
  </mdc>
  <ndc></ndc>
  <hostName>localhost.localdomain</hostName>
  <processName>jboss-modules.jar</processName>
  <processId>7790</processId>
</record>
```

添加密钥覆盖 **XML** 日志格式

```
/subsystem=logging/xml-formatter=XML_FORMATTER_NAME:add(pretty-print=true, print-namespace=true, namespace-uri="urn:custom:1.0", key-overrides={message=msg, record=logRecord, timestamp=date}, print-details=true)
```

您可以使用 **XML** 格式属性，如下所述：

- **key-overrides** 属性可用于覆盖定义的键的名称。

- 例外可以通过将 **exception-output-type** 属性设置为 格式化来格式化 为对象。
- 可以通过将 **exception-output-type** 属性设置为 **detailed** 来包含异常堆栈追踪。
- 通过将 **exception-output-type** 设置为 详细和格式，可以包含例外，作为对象和堆栈追踪。
- 可以使用 **meta-data** 属性将元数据添加到日志记录中。

有关 **XML** 格式器属性的更多信息，请参阅 **XML** 日志格式条件属性。

您还可以使用管理控制台配置 **XML** 日志格式器：

1. 在浏览器中打开管理控制台。
2. 选择 **Configuration → Subsystems → Logging**。
3. 选择 **Configuration**，然后单击 **View**。
4. 选择 **Formatter**，然后选择 **XML Formatter** 选项。

11.7.4. 配置自定义日志格式器

您可以创建自定义日志格式器，供日志处理程序用于格式化日志消息。

本节介绍如何使用管理 **CLI** 配置自定义日志格式器。

配置自定义日志格式器

重要

如果您要为日志记录配置集配置此日志格式器，则命令的开头为 **/subsystem=logging/logging-profile=LOGGING_PROFILE_NAME/** 而不是 **/subsystem=logging/**。

此外，如果您在受管域中运行，请在命令前加上 **/profile=PROFILE_NAME**。

1. 添加自定义日志格式器。

在添加自定义日志格式器时，您必须指定格式器的 **Java** 类以及包含它的 **JBoss EAP** 模块。该类必须扩展 **java.util.logging.Formatter**。

注意

您必须已创建了包含自定义格式的模块，否则此命令将失败。

```
/subsystem=logging/custom-
formatter=CUSTOM_FORMATTER_NAME:add(class=CLASS_NAME,
module=MODULE_NAME)
```

2. 设置日志格式器所需的属性。

属性必须能够通过 **setter** 方法访问。

```
/subsystem=logging/custom-formatter=CUSTOM_FORMATTER_NAME:write-
attribute(name=properties.PROPERTY_NAME,value=PROPERTY_VALUE)
```

3. 将自定义格式器分配到日志处理程序。

以下管理 **CLI** 命令分配可由定期轮转文件处理程序使用的自定义格式器：

```
/subsystem=logging/periodic-rotating-file-handler=FILE_HANDLER_NAME:write-
attribute(name=named-formatter, value=CUSTOM_FORMATTER_NAME)
```

自定义 **XML** 格式示例

以下示例配置了自定义 **XML** 格式。它使用 **org.jboss .logmanager** 模块中提供的 **java.util.logging.**

XMLFormatter 类，并将它分配给控制台日志处理程序。

```
/subsystem=logging/custom-formatter=custom-xml-
formatter:add(class=java.util.logging.XMLFormatter, module=org.jboss.logmanager)
/subsystem=logging/console-handler=CONSOLE:write-attribute(name=named-formatter,
value=custom-xml-formatter)
```

使用此格式器的日志消息将如下格式：

```
<record>
  <date>2016-03-23T12:58:13</date>
  <millis>1458752293091</millis>
  <sequence>93963</sequence>
  <logger>org.jboss.as</logger>
  <level>INFO</level>
  <class>org.jboss.as.server.BootstrapListener</class>
  <method>logAdminConsole</method>
  <thread>22</thread>
  <message>WFLYSRV0051: Admin console listening on http://%s:%d</message>
  <param>127.0.0.1</param>
  <param>9990</param>
</record>
```

使用管理控制台配置自定义日志格式器

您还可以使用管理控制台配置日志格式器。

1. 在浏览器中打开管理控制台。
2. 选择 **Configuration → Subsystems → Logging**。
3. 选择 **Configuration**，然后单击 **View**。
4. 选择 **Formatter**，然后选择 **Custom Formatter** 选项。

11.8. 关于应用程序日志

可以使用 **JBoss EAP logging** 子系统或每一部署来配置应用的日志记录。

有关日志记录子系统，请参阅使用 **JBoss EAP** 日志类别和处理程序来获取日志消息。

有关应用日志记录的更多信息，如受支持的应用日志框架和部署日志配置，请参阅 **JBoss EAP** 开发指南中的 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#logging_for_developers** 日志记录章节。

11.8.1. 按部署日志记录

通过按部署日志记录，开发人员可以提前为其应用配置日志配置。部署应用时，根据定义的配置开始日志记录。通过此配置创建的日志文件仅包含有关应用程序行为的信息。

注意

如果未执行按部署的日志配置，则所有应用和服务器都使用来自 **logging** 子系统的配置。

与使用整个系统日志记录相比，这种方法具有优缺点。优点是 **JBoss EAP** 实例的管理员不需要配置服务器日志记录之外的任何其他日志记录。个缺点是每个部署的日志配置仅在服务器启动时读取，因此在运行时无法更改。

有关在应用程序中使用按部署登录的说明，请参阅 **JBoss EAP** 开发指南中的 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#add_per_deployment_logging_to_an_application** 将 **Per-deployment Logging** 添加至应用。

11.8.1.1. 禁用 Per-deployment Logging

您可以使用以下方法之一禁用每个部署日志：

- 将 **use-deployment-logging-config** 属性设置为 **false**。

  use-deployment-logging-config 属性控制您的部署是否被扫描每个部署日志。默认默认为 **true**。您可以将此属性设置为 **false**，以禁用每个部署日志记录。

```
/subsystem=logging:write-attribute(name=use-deployment-logging-config,value=false)
```

- 使用 **jboss-deployment-structure.xml** 文件排除 **logging** 子系统。

  具体步骤，请参阅 **JBoss EAP** 开发指南中的 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#exclude_a_subsystem_from_a_deployment** 从部署中排除子系统。

11.8.2. 日志记录配置集

日志记录配置集是独立的日志配置集合，可以分配给已部署的应用。与常规的 **logging** 子系统一样，日志记录配置文件可以定义处理程序、类别和根日志记录器，但它不能引用其他配置文件或主要 日志记录子系统中的配置。日志配置文件的设计模仿 **logging** 子系统以方便配置。

通过日志记录配置文件，管理员可以创建专用于一个或多个应用的日志记录配置，而不影响任何其他日志配置。由于每个配置集都在服务器配置中定义，因此可以更改日志记录配置，而无需重新部署受影响的应用。

每个日志记录配置集都可以有：

- 唯一的名称。此值是必需的。
- 任意数量的日志处理程序.
- 任何数量的日志类别。
- 最多一个根日志记录器.

应用可以使用 **Logging-Profile** 属性在其 **MANIFEST.MF** 文件中指定要使用的日志记录配置文件。

11.8.2.1. 配置日志记录配置集

日志记录配置文件可以使用日志处理程序、类别和根日志记录器进行配置。配置日志记录配置集使用与配置 日志记录 子系统相同的语法，但以下不同之处除外：

- 根配置路径为 **/subsystem=logging/logging-profile=NAME**。
- 日志配置集无法包含其他日志配置集。
- **logging** 子系统具有以下无法用于日志记录配置集的属性：
  - **add-logging-api-dependencies**
  - **use-deployment-logging-config**

创建和配置日志配置集

以下流程使用管理 **CLI** 创建日志记录配置集并设置文件处理程序和日志记录器类别。也可以使用管理控制台配置日志记录配置集，方法是导航到 **Configuration → Subsystems → Logging → Logging Profiles**。

1. 创建日志记录配置文件。

```
/subsystem=logging/logging-profile=PROFILE_NAME:add
```

2. 创建 文件处理程序。

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-handler=FILE_HANDLER_NAME:add(file={path=>"LOG_NAME.log", "relative-to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=PROFILE_NAME/file-handler=FILE_HANDLER_NAME:write-attribute(name="level", value="DEBUG")
```

有关文件处理程序属性列表，请参阅文件日志处理程序属性列表。

3. 创建日志记录器类别。

```
/subsystem=logging/logging-profile=PROFILE_NAME/logger=CATEGORY_NAME:add(level=TRACE)
```

有关日志类别属性列表，请参阅 **Log Category Attributes**。

4. 将文件处理程序分配到该类别。

```
/subsystem=logging/logging-profile=PROFILE_NAME/logger=CATEGORY_NAME:add-handler(name="FILE_HANDLER_NAME")
```

然后，您可以将日志记录配置集设置为由应用在其 **MANIFEST.MF** 文件中使用。如需更多信息，请参阅《**JBoss EAP** 开发指南 》中的应用指定日志配置文件。

11.8.2.2. 日志配置集配置示例

本例显示了日志记录配置文件和使用它的应用的配置。它显示管理 **CLI** 命令、生成的 **XML** 和应用 **MANIFEST.MF** 文件。

日志记录配置集示例有以下特征：

- 名称为 **accounts-app-profile**。
- 日志类别为 **com.company.accounts.ejbs**。
- 日志级别 **TRACE.**
- 日志处理程序是使用文件 **ejb-trace.log** 的文件处理程序。

管理 **CLI** 会话

```
/subsystem=logging/logging-profile=accounts-app-profile:add

/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})

/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:write-attribute(name="level", value="DEBUG")
```

```
/subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)

/subsystem=logging/logging-profile=accounts-app-profile/logger=com.company.accounts.ejbs:add-
handler(name="ejb-trace-file")
```

XML 配置

```
<logging-profiles>
  <logging-profile name="accounts-app-profile">
    <file-handler name="ejb-trace-file">
      <level name="DEBUG"/>
      <file relative-to="jboss.server.log.dir" path="ejb-trace.log"/>
    </file-handler>
    <logger category="com.company.accounts.ejbs">
      <level name="TRACE"/>
      <handlers>
        <handler name="ejb-trace-file"/>
      </handlers>
    </logger>
  </logging-profile>
</logging-profiles>
```

应用程序 **MANIFEST.MF** 文件

```
Manifest-Version: 1.0
Logging-Profile: accounts-app-profile
```

11.8.3. 查看部署配置

您可以使用以下管理 **CLI** 命令获取有关特定部署的日志记录配置的信息。

```
/deployment=DEPLOYMENT_NAME/subsystem=logging/configuration=CONFIG:read-resource
```

部署的日志记录配置值 **CONFIG** 可以是以下三个值之一：

- 默认情况下，如果部署使用 **logging** 子系统 ：这将输出 **logging** 子系统配置。
- **profile-PROFILE_NAME**，如果部署使用 **logging** 子系统中定义的日志记录配置文件。这将输出日志记录配置文件配置。
- 使用的配置文件的路径，例如 **myear.ear/META-INF/logging.properties**。

例如，以下管理 **CLI** 命令显示 **MYPROFILE** 日志配置文件的配置，该配置集由指定部署使用。

```
/deployment=mydeployment.war/subsystem=logging/configuration=profile-MYPROFILE:read-resource(recursive=true,include-runtime=true)
```

这将输出以下信息：

```
{
    "outcome" => "success",
    "result" => {
        "error-manager" => undefined,
        "filter" => undefined,
        "formatter" => {
            "MYFORMATTER" => {
                "class-name" => "org.jboss.logmanager.formatters.PatternFormatter",
                "module" => undefined,
                "properties" => {"pattern" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n"}
            }
        },
        "handler" => {
            "MYPERIODIC" => {
                "class-name" => "org.jboss.logmanager.handlers.PeriodicRotatingFileHandler",
                "encoding" => undefined,
                "error-manager" => undefined,
                "filter" => undefined,
                "formatter" => "MYFORMATTER",
                "handlers" => [],
                "level" => "ALL",
                "module" => undefined,
                "properties" => {
                    "append" => "true",
                    "autoFlush" => "true",
                    "enabled" => "true",
                    "suffix" => ".yyyy-MM-dd",
                    "fileName" => "EAP_HOME/standalone/log/deployment.log"
```

```
                }
            }
        },
        "logger" => {"MYCATEGORY" => {
            "filter" => undefined,
            "handlers" => [],
            "level" => "DEBUG",
            "use-parent-handlers" => true
        }},
        "pojo" => undefined
    }
}
```

您还可以使用递归 **read-resource** 操作来读取日志记录配置和有关部署的其他信息。

```
/deployment=DEPLOYMENT_NAME/subsystem=logging:read-resource(include-runtime=true,
recursive=true)
```

11.9. 调整 LOGGING 子系统

有关监控 日志 子系统性能并优化性能的提示，请参阅 性能调优指南中 的日志子系统 调优部分。

第 12 章 数据源管理

12.1. 关于 JBOSS EAP 数据源

关于 JDBC

JDBC API 是定义 Java 应用如何访问数据库的标准。应用配置引用 JDBC 驱动程序的数据源。然后，可以针对驱动程序而非数据库编写应用程序代码。驱动程序将代码转换为数据库语言。这意味着，如果安装了正确的驱动程序，应用可以与任何受支持的数据库一起使用。

如需更多信息，请参见 JDBC 4.0 规范。

支持的数据库

如需 JBoss EAP 7 支持的 JDBC 兼容数据库列表，请参阅 JBoss EAP 支持的配置。

数据源类型

两种常规类型的资源称为数据源和 XA 数据源。

非 XA 数据源

用于不使用事务的应用程序，或者使用与单个数据库进行事务的应用程序。

XA 数据源

由将多个数据库或其他 XA 资源作为 XA 事务一部分的应用使用.XA 数据源会带来额外的开销。

您可以指定在使用 JBoss EAP 管理接口创建数据源时要使用的数据源类型。

ExampleDS 数据源

JBoss EAP 附带了数据源配置示例 ExampleDS，用于演示如何定义数据源。此数据源使用 H2 数据库，这是一个轻量级关系数据库管理系统，可为开发人员提供快速构建应用的能力。

警告

在生产环境中不应该使用 **ExampleDS** 数据源和 **H2** 数据库。这是一个非常小、自包含的数据源，支持测试和构建应用程序所需的所有标准，但安全性不足以用于生产用途。

12.2. JDBC DRIVERS

在 **JBoss EAP** 中定义数据源以供您的应用使用之前，必须先安装相应的 **JDBC** 驱动程序。

12.2.1. 将 JDBC 驱动程序作为核心模块安装

若要将 **JDBC** 驱动程序作为核心模块安装，您必须首先 将 **JDBC** 驱动程序添加为核心模块，然后在 **datasources** 子系统中 注册 **JDBC** 驱动程序。

12.2.1.1. 将 JDBC 驱动程序添加为核心模块

JDBC 驱动程序可以作为核心模块安装，使用管理 **CLI** 可以执行下列步骤：

1. 下载 **JDBC** 驱动程序。

   从您的数据库供应商下载适当的 **JDBC** 驱动程序。有关常见数据库的 **JDBC** 驱动程序的标准下载位置，请参阅 **JDBC** 驱动程序下载位置。

   如果 **JDBC** 驱动程序 **JAR** 文件包含在 **ZIP** 或 **TAR** 存档中，请确保提取存档。

2. 启动 **JBoss EAP** 服务器。

3. 启动管理 **CLI**。

```
$ EAP_HOME/bin/jboss-cli.sh
```

4. 使用 模块 **add management CLI** 命令，添加新的核心模块。

```
module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --
dependencies=DEPENDENCIES
```

例如，以下命令添加 **MySQL JDBC** 驱动程序模块：

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

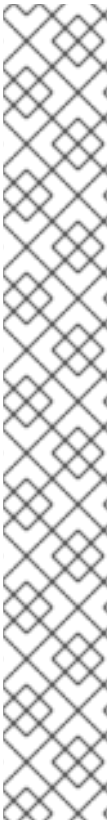
执行 模块 **--help** 获取关于使用此命令添加和删除模块的更多详细信息。

接下来，您必须将它注册为 **JDBC** 驱动程序，供应用数据源引用。

12.2.1.2. 注册 JDBC 驱动程序

驱动程序作为核心模块安装之后，您必须使用以下管理 **CLI** 命令将它注册为 **JDBC** 驱动程序：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME,driver-
module-name=MODULE_NAME,driver-xa-datasource-class-
name=XA_DATASOURCE_CLASS_NAME, driver-class-name=DRIVER_CLASS_NAME)
```



注意

只有 JDBC 驱动程序 jar 在 /META-INF/services/java.sql.Driver 文件中定义了多个类，才需要 driver- class-name 参数。

例如，MySQL 5.1.36 JDBC 驱动程序 JAR 中的 /META-INF/services/java.sql.Driver 文件定义了两个类：

- com.mysql.cj.jdbc.Driver
- com.mysql.fabric.jdbc.FabricMySQLDriver

在本例中，您将传递 driver-class-name=com.mysql.cj.jdbc.Driver。

例如，以下命令注册了 MySQL JDBC 驱动程序：

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MysqlXADataSource, driver-class-name=com.mysql.cj.jdbc.Driver)
```

JDBC 驱动程序现在可供应用数据源引用。

12.2.2. 将 JDBC 驱动程序安装为 JAR 部署

JDBC 驱动程序可以使用管理 CLI 或管理控制台作为 JAR 部署进行安装。只要驱动程序符合 JDBC 4 规范，部署时将自动识别并作为 JDBC 驱动程序安装。

下列步骤介绍了如何使用管理 CLI 安装 JDBC 驱动程序。

注意

JDBC 驱动程序的建议安装方法是将它们安装为核心模块。

1. 下载 **JDBC** 驱动程序。

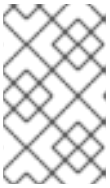
   从您的数据库供应商下载适当的 **JDBC** 驱动程序。有关常见数据库的 **JDBC** 驱动程序的标准下载位置，请参阅 **JDBC** 驱动程序下载位置。

   如果 **JDBC** 驱动程序 **JAR** 文件包含在 **ZIP** 或 **TAR** 存档中，请确保提取存档。

2. 如果 **JDBC** 驱动程序不兼容 **JDBC 4**，请参阅将 **JDBC** 驱动程序 **JAR** 更新为 **JDBC 4-Compliant** 的步骤。

3. 将 **JAR** 部署到 **JBoss EAP.**

   ```
   deploy PATH_TO_JDBC_JAR
   ```

   注意

   在受管域中，指定适当的服务器组。

   例如，以下命令将部署 **MySQL JDBC** 驱动程序：

   ```
   deploy /path/to/mysql-connector-java-8.0.12.jar
   ```

   JBoss EAP 服务器日志中将显示一条消息，显示部署的驱动程序名称，该名称将在定义数据源时使用。

   ```
   WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-8.0.12.jar
   ```

JDBC 驱动程序现在可供应用数据源引用。

将 JDBC 驱动程序 JAR 更新为 JDBC 4-Compliant

如果 **JDBC** 驱动程序 **JAR** 不兼容 **JDBC 4**，可以通过下列步骤使它可部署：

1. 创建一个空临时目录。

2. 创建 **META-INF** 子目录。

3. 创建 **META-INF/services** 子目录。

4. 创建 **META-INF/services/java.sql.Driver** 文件，再添加一行来指示 **JDBC** 驱动程序的完全限定类名称。

   例如，**MySQL JDBC** 驱动程序将添加下面这一行：

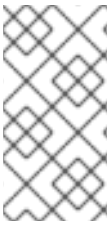
   ```
   com.mysql.cj.jdbc.Driver
   ```

5. 使用 **JAR** 命令行工具将此新文件添加到 **JAR**。

   ```
   jar \-uf jdbc-driver.jar META-INF/services/java.sql.Driver
   ```

12.2.3. JDBC 驱动程序下载位置

下表提供了用于 **JBoss EAP** 的常见数据库的 **JDBC** 驱动程序的标准下载位置。



注意

这些链接指向不受红帽控制或主动监控的第三方网站。有关数据库的最新驱动程序，请查看您的数据库供应商文档和网站。

表 12.1. JDBC 驱动程序下载位置

vendor	下载位置
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
IBM	http://www-01.ibm.com/support/docview.wss?uid=swg21363866

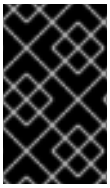
vendor	下载位置
Sybase	jConnect JDBC 驱动程序是用于 SAP ASE 安装的 SDK 的一部分。目前此驱动程序本身没有单独的下载网站。
Microsoft	http://msdn.microsoft.com/data/jdbc/

12.2.4. 访问供应商特定类

*在某些情况下，应用需要使用不属于 **JDBC API** 的供应商特定功能。在这些情况下，您可以通过在该应用程序中声明依赖项来访问特定于供应商的 **API**。*

警告

*这是高级使用。只有 **JDBC API** 中没有功能的应用才应实施此过程。*



重要

在使用重新身份验证机制和访问特定于供应商的类时，此过程是必需的。

*您可以使用 **MANIFEST.MF** 文件或 **jboss-deployment-structure.xml** 文件定义应用的依赖项。*

*如果您还没有这样做，请将 **JDBC** 驱动程序作为核心模块安装。*

*使用 **MANIFEST.MF** 文件*

1. *编辑应用的 **META-INF/MANIFEST.MF** 文件。*
2. *添加 **Dependencies** 行，并指定模块名称。*

例如，下面这一行将 **com.mysql** 模块声明为依赖项：

```
Dependencies: com.mysql
```

使用 **jboss-deployment-structure.xml** 文件

1. 在应用的 **META -INF/** 或 **WEB-INF/** 文件夹中创建名为 **jboss-deployment- structure.xml** 的文件。

2. 使用 **dependencies** 元素指定模块。

   例如，以下示例 **jboss-deployment-structure.xml** 文件将 **com.mysql** 模块声明为依赖项：

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

以下示例代码将访问 **MySQL API**：

```
import java.sql.Connection;
...
Connection c = ds.getConnection();
if (c.isWrapperFor(com.mysql.jdbc.Connection.class)) {
    com.mysql.jdbc.Connection mc = c.unwrap(com.mysql.jdbc.Connection.class);
}
```

重要

随着连接由 **IronJacamar** 容器控制，请严格遵循特定于供应商的 **API** 准则。

12.3. 创建数据源

可以使用管理控制台或管理 **CLI** 创建数据源。

JBoss EAP 7 允许您在数据源属性值中使用表达式，如 **enabled** 属性。有关在配置中使用表达式的详情，请参阅属性替换部分。

12.3.1. 创建非 XA 数据源

您可以使用管理 CLI 或管理控制台创建非 XA 数据源。

使用管理控制台定义非 XA 数据源

1. 导航到 Configuration → Subsystems → Datasources & Drivers → Datasources。
2. 单击添加(+)按钮，然后选择 Add Datasource。
3. 它将打开 Add Datasource 向导，您可以在其中选择数据源类型并点击 Next。这会为您的数据库创建一个模板。向导的以下页面预先填充了特定于所选数据源的值。这使得数据源创建过程变得简单。
4. 您可以在 Test Connection 页面上测试您的连接，然后完成数据源创建过程。
5. 检查详情，再单击 Finish 以创建数据源。

使用管理 CLI 定义非 XA 数据源

可以使用 数据源添加管理 CLI 命令来定义非 X A 数据源。

1. 如果您还没有这样做，请安装相应的 JDBC 驱动程序并将其注册为核心模块。
2. 使用 data-source add 命令定义数据源，并指定适当的参数值。

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME  --connection-url=CONNECTION_URL
```

注意

在受管域中，您必须指定 --profile=PROFILE_NAME 参数。

有关这些参数值的提示，请参见下面的 **Datasource Parameters** 部分。

具体示例，请参阅支持的数据库的 **Datasource** 配置示例。

数据源参数

jndi-name

数据源的 **JNDI** 名称必须以 **java:/** 或 **java :jboss/** 开头。例如，**java:jboss/datasources/ExampleDS.**

driver-name

驱动程序名称值取决于 **JDBC** 驱动程序是作为核心模块安装的还是 **JAR** 部署。

1. 对于核心模块，驱动程序名称值将是其注册时为 **JDBC** 驱动程序提供的名称。

2. 对于 **JAR** 部署，如果其 **/META-INF/services/java.sql.Driver** 文件中仅列出一个类，则驱动程序名称是 **JAR** 的名称。如果列出多个类，则值为 **JAR_NAME + "_ " + DRIVER_CLASS_NAME + "_ " + MAJOR_VERSION + "_" + MINOR_VERSION** （例如，） **mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1**）.

   部署 **JDBC JAR** 时，您还可以查看 **JBoss EAP** 服务器日志中的驱动程序名称。

   > *WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1*

connection-url

有关支持的数据库的连接 **URL** 格式的详情，请查看 **Datasource Connection URL** 列表。

有关所有可用数据源属性的完整列表，请参阅 **Datasource Attributes** 部分。

12.3.2. 创建 XA 数据源

您可以使用管理 **CLI** 或管理控制台创建 **XA** 数据源。

使用管理控制台定义 **XA** 数据源

1. 导航到 **Configuration → Subsystems → Datasources & Drivers → Datasources**。

2. 单击添加**(+)**按钮，然后选择 **Add XA Datasource**。

3. 它打开 **Add XA Datasource** 向导，您可以在其中选择数据源类型并点击 **Next**。这会为您的数据库创建一个模板。向导的以下页面预先填充了特定于所选数据源的值。这使得数据源创建过程变得简单。

4. 您可以在 **Test Connection** 页面上测试您的连接，然后完成数据源创建过程。

5. 检查详情，再单击 **Finish** 以创建数据源。

使用管理 **CLI** 定义 **XA** 数据源

XA 数据源可以使用 **the xa-data-source add management CLI** 命令来定义。

注意

在受管域中，您将需要指定要使用的配置文件。根据管理 **CLI** 命令的格式，您将在命令前面使用 **/profile=PROFILE_NAME**，或者传递 **--profile=PROFILE_NAME** 参数。

1. 如果您还没有这样做，请安装相应的 **JDBC** 驱动程序并将其注册为核心模块。

2. 使用 **the xa-data-source add** 命令定义数据源，并指定适当的参数值。

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS --xa-datasource-properties={"ServerName"=>"HOST_NAME","DatabaseName"=>"DATABASE_NAME"}
```

有关这些参数值的提示，请参见下面的 **Datasource Parameters** 部分。

3. 设置 **XA** 数据源属性.

定义 **XA** 数据源时至少需要一个 **XA** 数据源属性，或者在上一步中添加数据源时会收到错

误。在定义 **XA** 数据源时未设置的任何属性都可以在之后单独设置。

a. 设置服务器名称。

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=ServerName:add(value=HOST_NAME)
```

b. 设置数据库名称。

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=DatabaseName:add(value=DATABASE_NAME)
```

具体示例，请参阅支持的数据库的 **Datasource** 配置示例。

数据源参数

jndi-name

数据源的 **JNDI** 名称必须以 **java:/** 或 **java :jboss/** 开头。例如，**java:jboss/datasources/ExampleDS.**

driver-name

驱动程序名称值取决于 **JDBC** 驱动程序是作为核心模块安装的还是 **JAR** 部署。

1. 对于核心模块，驱动程序名称值将是其注册时为 **JDBC** 驱动程序提供的名称。

2. 对于 **JAR** 部署，如果其 **/META-INF/services/java.sql.Driver** 文件中仅列出一个类，则驱动程序名称是 **JAR** 的名称。如果列出了多个类，则值为 **JAR_NAME + "_" + DRIVER_CLASS_NAME + "_" + MAJOR_VERSION + "_" + MINOR_VERSION**，例如：**mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1.**

   部署 **JDBC JAR** 时，您还可以查看 **JBoss EAP** 服务器日志中的驱动程序名称。

```
WFLYJCA0018: Started Driver service with driver-name = mysql-connector-java-5.1.36-bin.jar_com.mysql.cj.jdbc.Driver_5_1
```

xa-datasource-class

指定 **javax.sql.XADataSource** 类的 **JDBC** 驱动程序实施的 **XA** 数据源类。

xa-datasource-properties

定义 **XA** 数据源时至少需要一个 **XA** 数据源属性，否则在尝试添加时会收到错误。其他属性也可以在定义后添加到 **XA** 数据源中。

有关所有可用数据源属性的完整列表，请参阅 **Datasource Attributes** 部分。

12.4. 修改数据源

可以使用管理控制台或管理 **CLI** 配置数据源设置。

JBoss EAP 7 允许您在数据源属性值中使用表达式，如 **enabled** 属性。有关在配置中使用表达式的详情，请参阅属性替换部分。

12.4.1. 修改非 XA 数据源

可以使用数据源 管理 **CLI** 命令更新非 **X A** 数据源设置。您还可以导航到 **Configuration** → **Subsystems** → **Datasources** → **Datasources** → **Datasources** → **Datasources**，从管理控制台更新数据源属性。

注意

非 **XA** 数据源可与 **JTA** 事务集成。要将数据源与 **JTA** 集成，请确保 **jta** 参数设为 **true**。

可以通过以下管理 **CLI** 命令更新数据源的设置：

```
data-source --name=DATASOURCE_NAME --ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```

注意

在受管域中，您必须指定 **--profile=PROFILE_NAME** 参数。

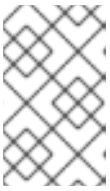
可能需要重新加载服务器才能使更改生效。

12.4.2. 修改 XA 数据源

XA 数据源设置可以使用 **xa-data-source** 管理 **CLI** 命令更新。您还可以导航到 **Configuration → Subsystems → Datasources → Datasources → Datasources → Datasources**，从管理控制台更新数据源属性。

- 可以使用以下管理 **CLI** 命令更新 **XA** 数据源的设置：

```
xa-data-source --name=XA_DATASOURCE_NAME -
-ATTRIBUTE_NAME=ATTRIBUTE_VALUE
```

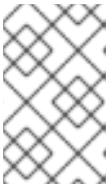


注意

在受管域中，您必须指定 **--profile=PROFILE_NAME** 参数。

- **XA** 数据源属性可以通过以下管理 **CLI** 命令添加：

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-
properties=PROPERTY:add(value=VALUE)
```



注意

在受管域中，您必须在此命令前加上 **/profile=PROFILE_NAME**。

可能需要重新加载服务器才能使更改生效。

12.5. 删除数据源

可以使用管理控制台或管理 **CLI** 删除数据源。

12.5.1. 删除非 XA 数据源

可以使用 数据源删除管理 **CLI** 命令来删除非 **X A** 数据源。您还可以通过导航到 **Configuration →**

Subsystems → Datasources → Datasources → Datasources 来使用管理控制台删除数据源 。

```
data-source remove --name=DATASOURCE_NAME
```

注意

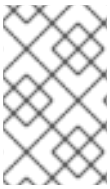
在受管域中，您必须指定 **--profile=PROFILE_NAME** 参数。

删除数据源后，服务器将需要重新加载。

12.5.2. 删除 XA 数据源

XA 数据源可以使用 **the xa-data-source remove management CLI** 命令删除。您还可以通过导航到 **Configuration → Subsystems → Datasources → Datasources → Datasources** 来使用管理控制台删除数据源 。

```
xa-data-source remove --name=XA_DATASOURCE_NAME
```



注意

在受管域中，您必须指定 **--profile=PROFILE_NAME** 参数。

在 **XA** 数据源被删除后，服务器将需要重新加载。

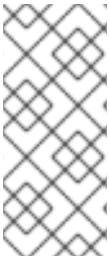
12.6. 测试数据源连接

您可以使用管理 **CLI** 或管理控制台测试数据源连接，以验证其设置是否正确。

使用管理 **CLI** 测试数据源连接

以下管理 **CLI** 命令可用于测试数据源的连接：

```
/subsystem=datasources/data-source=DATASOURCE_NAME:test-connection-in-pool
```



注意

在受管域中，您必须在此命令前加上 **/host=HOST_NAME/server=SERVER_NAME**。如果您要测试 **XA** 数据源，请将 **data-source=DATASOURCE_NAME** 替换为 **xa-data-source=XA_DATASOURCE_NAME**。

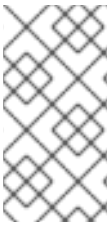
使用管理控制台测试数据源连接

在管理控制台中使用 **Add Datasource** 向导时，您有机会在创建数据源前测试连接。在向导的 **Test Connection** 屏幕上，单击 **Test Connection** 按钮。

添加数据源后，您可以通过导航到 **Configuration → Subsystems → Datasources → Datasources → Datasources**、选择数据源并从下拉菜单中选择 **Test Connection** 来测试连接。

12.7. 清空数据源连接

您可以使用以下管理 **CLI** 命令清空数据源连接。



注意

在受管域中，您必须在这些命令前使用 **/host=HOST_NAME/server=SERVER_NAME**。

- 清空池中的所有连接。

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-all-connection-in-pool
```

- 正常清空池中的所有连接。

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-gracefully-connection-in-pool
```

服务器将等待连接闲置，然后清空连接。

- 清空池中的所有空闲连接。

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-idle-connection-in-pool
```

- 清空池中的所有无效连接。

```
/subsystem=datasources/data-source=DATASOURCE_NAME:flush-invalid-connection-in-pool
```

服务器将清空其认为无效的所有连接，例如，由 **valid-connection-checker-class-name** 或 **check-valid-connection-sql** 验证机制清除。

您还可以使用管理控制台刷新连接。从 **Runtime** 选项卡中，选择服务器，选择 **Datasources**，选择数据源，然后使用下拉菜单选择相应的操作。

12.8. XA 数据源恢复

XA 数据源是可参与 **XA** 全局交易的数据源，由交易经理协调，并可在单个交易中跨越多个资源。如果其中一位参与者未能提交更改，则其他参与者中止交易并恢复交易发生前的状态。这是为了保持一致性，避免潜在的数据丢失或损坏。

XA 恢复是一个确保更新或回滚受交易影响的所有资源的过程，即使任何资源或交易参与者崩溃或不可用。**XA** 恢复发生，无需用户干预。

每个 **XA** 资源都需要关联一个恢复模块及其配置。恢复模块是执行恢复时执行的代码。**JBoss EAP** 自动注册 **JDBC XA** 资源的恢复模块。如果要实施自定义恢复代码，您可以使用 **XA** 数据源注册自定义模块。恢复模块必须扩展类 **com.arjuna.ats.jta.recovery.XAResourceRecovery**。

12.8.1. 配置 XA 恢复

对于大多数 **JDBC** 资源，恢复模块自动与资源关联。在这些情况下，您只需要配置允许恢复模块连接到资源以执行恢复的选项。

下表描述了特定于 **XA** 恢复的 **XA** 数据源参数。这些配置属性可以在数据源创建期间或之后设置。您可以使用管理控制台或管理 **CLI** 进行设置。有关配置 **XA** 数据源的详情，请参阅修改 **XA** 数据源。

表 **12.2. XA** 恢复的数据源参数

属性	描述
restore-username	用于连接资源以进行恢复的用户名。请注意，如果没有指定，将使用数据源安全设置。
restore-password	用于连接资源以进行恢复的密码。请注意，如果没有指定，将使用数据源安全设置。
recovery-security-domain	用于连接资源以进行恢复的安全域。
recovery-plugin-class-name	如果需要使用自定义恢复模块，将此属性设置为模块的完全限定类名称。模块应扩展类 **com.arjuna.ats.jta.recovery.XAResourceRecovery**。
recovery-plugin-properties	如果您使用需要设置属性的自定义恢复模块，请将此属性设置为属性的逗号分隔 ***KEY=VALUE*** 对的列表。

*禁用 **XA** 恢复*

*如果多个 **XA** 数据源连接到同一物理数据库，则通常只需要为其中一个配置 **XA** 恢复。*

*使用以下命令来禁用 **XA** 数据源的恢复：*

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-attribute(name=no-recovery,value=true)
```

12.8.2. 特定于厂商的 XA 恢复

特定厂商配置

*些数据库需要特定的配置，才能在由 **JBoss EAP** 事务管理器管理的 **XA** 交易中协作。有关详细信息和最新信息，请查看您的数据库供应商文档。*

MySQL

*不需要特殊配置。如需更多信息，请参阅 **MySQL** 文档。*

PostgreSQL 和 Postgres Plus Advanced Server

*要使 **PostgreSQL** 能够处理 **XA** 事务，请将配置参数 **max_prepared_transactions** 改为大于 **0** 的值，等于或大于 **max_connections**。*

Oracle

确保 **Oracle** 用户 **USER** 有权访问恢复所需的表。

```
GRANT SELECT ON sys.dba_pending_transactions TO USER;
GRANT SELECT ON sys.pending_trans$ TO USER;
GRANT SELECT ON sys.dba_2pc_pending TO USER;
GRANT EXECUTE ON sys.dbms_xa TO USER;
```

如果 **Oracle** 用户没有正确的权限，您可能会看到类似如下的错误：

```
WARN  [com.arjuna.ats.jta.logging.loggerI18N] [com.arjuna.ats.internal.jta.recovery.xarecovery1]
Local XARecoveryModule.xaRecovery  got XA exception javax.transaction.xa.XAException,
XAException.XAER_RMERR
```

Microsoft SQL Server

如需更多信息，请参阅 **Microsoft SQL Server** 文档，包括 **http://msdn.microsoft.com/en-us/library/aa342335.aspx**。

IBM DB2

不需要特殊配置。如需更多信息，请参阅 **IBM DB2** 文档。

Sybase

Sybase 希望在数据库中启用 **XA** 事务。如果没有正确的数据库配置，**XA** 事务将无法工作。**enable xact** 协调 参数启用或禁用自适应服务器事务协调服务。启用此参数后，自适应服务器服务器可确保对远程适配器服务器数据提交或回滚原始事务的更新。

要启用事务协调，请使用：

```
sp_configure 'enable xact coordination', 1
```

MariaDB

不需要特殊配置。如需更多信息，请参阅 **MariaDB** 文档。

已知问题

这些处理 **XA** 事务的已知问题与 **JBoss EAP 7** 支持的特定数据库和 **JDBC** 驱动程序版本有关。有关支持的数据库的最新信息，请参阅 **JBoss EAP** 支持的配置。

MySQL

MySQL 无法完全处理 **XA** 事务。如果客户端与 **MySQL** 断开连接，则此类交易的所有信息都将丢

失。如需更多信息，请参阅此 MySQL 错误。请注意，这个问题已在 MySQL 5.7 中解决。

PostgreSQL 和 Postgres Plus Advanced Server

当两阶段提交(2PC)的提交阶段发生网络故障时，JDBC 驱动程序会返回 XAER_RMERR XAException 错误代码。此错误向交易管理器表明一个无法恢复的灾难性事件，但事务保留在数据库端的 in doubt 状态，并在网络连接再次建立后易于更正。正确的返回代码应当是 XAER_RMFAIL 或 XAER_RETRY。错误代码不正确会导致事务保持在 JBoss EAP 侧的 Heuristic 状态，并在数据库中锁定需要人工干预。如需更多信息，请参阅此 PostgreSQL 错误。

如果使用单阶段提交优化时发生连接失败，JDBC 驱动程序将返回 XAER_RMERR，但应当返回 XAER_RMFAIL 错误代码。如果数据库在一阶提交期间提交数据并且连接在停机时中断，这可能会导致数据不一致，然后客户端被通知事务被回滚。

Postgres Plus JDBC 驱动程序为 Postgres Plus Server 中存在的所有准备事务返回 XID，因此无法确定 XID 所属的数据库。如果您在 JBoss EAP 中为同一数据库定义多个数据源，则内部事务恢复尝试可能会在错误的帐户下运行，这会导致恢复失败。

Oracle

当恢复管理器使用配置了一些用户凭据的数据源调用恢复时，JDBC 驱动程序会返回属于数据库实例上所有用户的 XID。JDBC 驱动程序抛出异常 ORA-24774：无法切换到指定的事务，因为它试图恢复属于其他用户的 XID。

这个问题的解决方法是向在恢复数据源配置中使用凭证的用户授予 FORCE ANY TRANSACTION 特权。有关配置权限的更多信息，请参阅
http://docs.oracle.com/database/121/ADMIN/ds_txnman.htm#ADMIN12259

Microsoft SQL Server

当两阶段提交(2PC)的提交阶段发生网络故障时，JDBC 驱动程序会返回 XAER_RMERR XAException 错误代码。此错误向交易管理器表明一个无法恢复的灾难性事件，但事务保留在数据库端的 in doubt 状态，并在网络连接再次建立后易于更正。正确的返回代码应当是 XAER_RMFAIL 或 XAER_RETRY。错误代码不正确会导致事务保持在 JBoss EAP 侧的 Heuristic 状态，并在数据库中锁定需要人工干预。如需更多信息，请参阅此 Microsoft SQL Server 问题报告。

如果使用单阶段提交优化时发生连接失败，JDBC 驱动程序将返回 XAER_RMERR，但应当返回 XAER_RMFAIL 错误代码。如果数据库在一阶提交期间提交数据并且连接在停机时中断，这可能会导致数据不一致，然后客户端被通知事务被回滚。

IBM DB2

如果在单阶段提交期间发生连接失败，JDBC 驱动程序将返回 XAER_RETRY，但应当返回 XAER_RMFAIL 错误代码。如果数据库在一阶提交期间提交数据并且连接在停机时中断，这可能会导

致数据不一致，然后客户端被通知事务被回滚。

Sybase

当两阶段提交(2PC)的提交阶段发生网络故障时，JDBC 驱动程序会返回 XAER_RMERR XAException 错误代码。此错误向交易管理器表明一个无法恢复的灾难性事件，但事务保留在数据库端的 in doubt 状态，并在网络连接再次建立后易于更正。正确的返回代码应当是 XAER_RMFAIL 或 XAER_RETRY。错误代码不正确会导致事务保持在 JBoss EAP 侧的 Heuristic 状态，并在数据库中锁定需要人工干预。

如果使用单阶段提交优化时发生连接失败，JDBC 驱动程序将返回 XAER_RMERR，但应当返回 XAER_RMFAIL 错误代码。如果数据库在一阶段提交期间提交数据并且连接在停机时中断，这可能会导致数据不一致，然后客户端被通知事务被回滚。

如果在 Sybase 事务分支处于就绪状态之前包含插入 Sybase 15.7 或 16 数据库的 XA 事务失败，则重复 XA 事务并插入具有相同主键的相同记录会失败，并出现 com.sybase.jdbc4.jdbc.SybSQLException: Attempt 以插入重复密钥行。这个异常将被抛出，直到原始的未完成 Sybase 事务分支回滚。

MariaDB

MariaDB 无法完全处理 XA 事务。如果客户端与 MariaDB 断开连接，则此类交易的所有信息都将丢失。

MariaDB Galera Cluster

MariaDB Galera 集群中不支持 XA 事务。

12.9. 数据库连接验证

数据库维护、网络问题或其他中断事件可能会导致 JBoss EAP 丢失与数据库的连接。为了从这些情形中恢复，您可以为数据源启用数据库连接验证。

要配置数据库连接验证，您可以指定验证时序方法、用于确定执行验证的方式的验证机制，以及用于定义异常处理方式的异常分类器。

1. 选择一种验证计时方法。

validate-on-match

当 validate-on-match 选项设为 true 时，每次使用下一步中指定的验证机制从连接池中签出时，都会验证数据库连接。

如果连接无效，则会向日志写入警告，并检索池中的下一个连接。此过程将继续，直到找到了有效的连接。如果您不希望循环池中的每一连接，您可以使用 **use-fast-fail** 选项。如果池中找不到有效的连接，则会创建一个新连接。如果连接创建失败，则会向请求的应用返回异常。

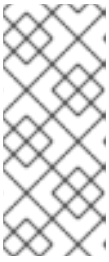
此设置可产生最快的恢复速度，但会给数据库造成最高负载。但是，如果对最低性能影响不是问题，则这是最安全的选择。

background-validation

当 **background-validation** 选项设为 **true** 时，会在使用前在后台线程中定期验证连接。验证的频率由 **background-validation-millis** 属性指定。**background-validation-millis** 的默认值为 **0**，表示它已被禁用。

在确定 **background-validation-millis** 属性的值时，请考虑以下几点：

- 这个值不应设置为与您 **idle-timeout-minutes** 设置相同的值。
- 数值越低，池的验证频率越高，无效连接越早从池中移除。
- 数值越低，会占用更多的数据库资源。数值越高，连接验证检查频率越少，使用的数据库资源更少，但死连接在较长时间内不会被检测到。



注意

这些选项是互斥的。如果 **validate-on-match** 设为 **true**，则 **background-validation** 必须设为 **false**。如果 **background-validation** 设为 **true**，则 **validate-on-match** 必须设置为 **false**。

2. 选择一种验证机制。

valid-connection-checker-class-name

使用 **valid-connection-checker-class-name** 是首选的验证机制。这指定了一个连接检查器类，用于验证正在使用的特定数据库的连接。**JBoss EAP** 提供以下连接检查器：

- *org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker*

check-valid-connection-sql

*使用 **check-valid-connection-sql** 时，您将提供用于验证连接的 **SQL** 语句。*

*以下是可以用来验证 **Oracle** 连接的 **SQL** 语句示例。*

```
select 1 from dual
```

以下是可以用来验证 **MySQL** 或 **PostgreSQL** 连接的示例 **SQL** 语句：

```
select 1
```

3. 设置异常分类器类名称。

当异常标记为致命时，即使连接参与交易，也会立即关闭连接。使用异常排序器类选项，在连接异常后正确地检测和清理。为您的数据源类型选择相应的 **JBoss EAP** 异常分类器。

- **org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.informix.InformixExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter**
- **org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter**

12.10. 数据源安全

数据源安全性指的是为数据源连接加密或模糊密码。这些密码可以以纯文本形式存储在配置文件中，但这代表了安全风险。

您可以通过多种方法实现数据源安全性。下面列出了每个示例。

使用安全域保护数据源

使用以下步骤通过安全域保护数据源：

1. 创建新的安全域。

```
/subsystem=security/security-domain=DsRealm:add(cache-type=default)
/subsystem=security/security-domain=DsRealm/authentication=classic:add(login-modules=
[{code=ConfiguredIdentity,flag=required,module-options={userName=sa,
principal=sa, password=sa}}])
```

定义了数据源的安全域。以下 **XML** 提取是调用 **CLI** 命令的结果。

```
<security-domain name="DsRealm" cache-type="default">
 <authentication>
  <login-module code="ConfiguredIdentity" flag="required">
   <module-option name="userName" value="sa"/>
   <module-option name="principal" value="sa"/>
   <module-option name="password" value="sa"/>
  </login-module>
 </authentication>
</security-domain>
```

2. 添加新数据源.

```
data-source add --name=securityDs
--jndi-name=java:jboss/datasources/securityDs
--connection-url=jdbc:h2:mem:test;DB_CLOSE_DELAY=-1 --driver-name=h2
--new-connection-sql="select current_user()"
```

3. 设置数据源上的安全域。

```
data-source --name=securityDs --security-domain=DsRealm
```

4. 重新加载服务器，让更改生效。

```
reload
```

注意

如果您使用具有多个数据源的安全域，请在安全域中禁用缓存。这可以通过将 **cache-type** 属性的值设置为 **none** 或完全删除属性来完成；但是，如果需要缓存，请为每个数据源使用单独的安全域。

以下 **XML** 提取显示了通过 **DsRealm** 保护的数据源。

```
<datasources>
  <datasource jndi-name="java:jboss/datasources/securityDs"
    pool-name="securityDs">
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <new-connection-sql>select current_user()</new-connection-sql>
      <security>
        <security-domain>DsRealm</security-domain>
      </security>
    </datasource>
</datasources>
```

有关使用安全域的更多信息，请参阅如何配置身份管理。

使用密码 **Vault** 保护数据源

使用以下步骤使用密码 **vault** 保护数据源：

1. 为 **ExampleDS** 数据源设置密码库。

```
data-source --name=ExampleDS
--
password=${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEtMWNlMDMyNDdmNmI2TElORV9CUkVBS3ZhdWx0}
```

2. 重新加载服务器以实施更改。

```
reload
```

以下 **XML** 安全元素添加到使用密码库保护的 **ExampleDS** 数据源中：

```
<security>
  <user-name>admin</user-name>
```

```
<password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEtMWNlMDMyNDdmNmI2TElORV9CUkVBS3ZhdWx0}</password>
</security>
```

有关使用密码库的更多信息，请参阅 **JBoss EAP How to Configure Server Security** 指南中的 **Password Vault** 部分。

使用凭证存储保护数据源

您还可以使用凭据存储来提供密码。**elytron** 子系统提供创建凭据存储的功能，以便在整个 **JBoss EAP** 中安全存放和使用您的密码。您可以在 **JBoss EAP How to Configure Server Security** 指南的 **Credential Store** 部分中找到关于创建和使用凭据存储的更多详细信息。

为示例**DS** 添加凭证存储参考

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=credential-reference,value={store=exampleCS, alias=example-ds-pw})
```

使用身份验证上下文保护数据源

您还可以使用 **Elytron** 身份验证上下文来提供用户名和密码。

使用以下步骤配置和使用数据源安全性的身份验证上下文。

1. 删除 密码 和 **user-name**。

```
/subsystem=datasources/data-source=ExampleDS:undefine-attribute(name=password)
/subsystem=datasources/data-source=ExampleDS:undefine-attribute(name=user-name)
```

2. 为数据源启用 **Elytron** 安全性。

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=elytron-enabled,value=true)

reload
```

3. 为您的凭证创建 身份验证配置。

   身份验证配置包含您希望在进行连接时使用的数据源的凭据。以下示例使用对凭据存储的引用，但也可以使用 **Elytron** 安全域。

```
/subsystem=elytron/authentication-configuration=exampleAuthConfig:add(authentication-name=sa,credential-reference={clear-text=sa})
```

4. 创建 身份验证上下文.

```
/subsystem=elytron/authentication-context=exampleAuthContext:add(match-rules=[{authentication-configuration=exampleAuthConfig}])
```

5. 更新数据源以使用身份验证上下文。

   以下示例更新了 **ExampleDS** 以使用身份验证上下文。

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=authentication-context,value=exampleAuthContext)

reload
```

注意

如果没有设置 **authentication-context** 属性，且 **elytron-enabled** 属性设为 **true**，则 **JBoss EAP** 将使用当前上下文进行身份验证。

使用 **Kerberos** 保护数据源

要使用 **kerberos** 身份验证保护数据源，需要以下配置：

- **Kerberos** 在数据库服务器上配置。
- **JBoss EAP** 主机服务器具有数据库服务器的 **keytab** 条目。

使用 **kerberos** 保护数据源：

1. 将 **JBoss EAP** 配置为使用 **kerberos.**

```
/system-property=java.security.krb5.conf:add(value="/path/to/krb5.conf")
/system-property=sun.security.krb5.debug:add(value="false")
/system-property=sun.security.spnego.debug:add(value="false")
```

若要进行调试，请将 **sun.security.krb5.debug** 和 **sun.security.spnego.debug** 的值更改为 **true**。在生产环境中，建议将值设为 **false**。

2. 配置安全性.

您可以使用旧安全性或 **Elytron** 安全来保护数据源。

- 将 **kerberos** 与旧安全性一起使用：

a. 配置 **infinispan** 缓存，以定期从缓存中删除过期的票据。

```
batch
/subsystem=infinispan/cache-container=security:add(default-cache=auth-cache)
/subsystem=infinispan/cache-container=security/local-cache=auth-cache:add()
/subsystem=infinispan/cache-container=security/local-cache=auth-
cache/expiration=EXPIRATION:add(lifespan=3540000,max-idle=3540000)
/subsystem=infinispan/cache-container=security/local-cache=auth-
cache/memory=object:add(size=1000)
run-batch
```

以下属性定义 **ticket** 过期：

- **Life span**：间隔，以毫秒为单位，从 **KDC** 请求新证书。将**lifespan** 属性 的值设置为比 **KDC** 定义的寿命小。
- **max-idle** ： **Interval**，以毫秒为单位，如果缓存未使用，则应从缓存中删除有效票据。
- **max-entries** ：要保留在缓存中的 **kerberos ticket** 的最大数量副本。该值对应于数据源中配置的连接数量。

b. 创建安全域。

```
batch
/subsystem=security/security-domain=KerberosDatabase:add(cache-type=infinispan)
/subsystem=security/security-domain=KerberosDatabase/authentication=classic:add
/subsystem=security/security-
domain=KerberosDatabase/authentication=classic/login-
module="KerberosDatabase-
Module":add(code="org.jboss.security.negotiation.KerberosLoginModule",module="org.
jboss.security.negotiation",flag=required, module-options={ "debug" => "false",
"storeKey" => "false", "useKeyTab" => "true", "keyTab" => "/path/to/eap.keytab",
"principal" => "PRINCIPAL@SERVER.COM", "doNotPrompt" => "true",
"refreshKrb5Config" => "true", "isInitiator" => "true", "addGSSCredential" => "true",
"credentialLifetime" => "-1"})
run-batch
```

- 将 **Microsoft JDBC** 驱动程序用于 **SQL** 服务器时，请在 **module-options** 中添加属性和值 **"wrapGSSCredential" SAS "true"**。
- 为进行调试，请将 **module-options** 中 **debug** 属性的值更改为 **true**。

- 在 **Elytron** 中使用 **kerberos:**

a. 在 **Elytron** 中设置 **kerberos** 工厂。

```
/subsystem=elytron/kerberos-security-factory=krbsf:add(debug=false,
principal=PRINCIPAL@SERVER.COM, path=/path/to/keytab, request-lifetime=-1,
obtain-kerberos-ticket=true, server=false)
```

- 要进行调试，请添加属性和值 **debug = true**。

有关支持的属性列表，请参阅 **How to Configure Server Security Guide** 中的 **Kerberos Security Factory Attributes** 部分。

b. 创建身份验证配置以使用 **kerberos** 工厂。

```
/subsystem=elytron/authentication-configuration=kerberos-conf:add(kerberos-
security-factory=krbsf)
```

c.

创建身份验证上下文.

```
/subsystem=elytron/authentication-context=ds-context:add(match-rules=
[{authentication-configuration=kerberos-conf}])
```

3.

*使用 **Kerberos** 保护数据源.*

- *如果您使用旧安全性：*

  a.

  将数据源配置为使用安全域。

  ```
  /subsystem=datasources/data-source=KerberosDS:add(connection-url="URL", min-
  pool-size=0, max-pool-size=10, jndi-name="java:jboss/datasource/KerberosDS",
  driver-name=<jdbc-driver>.jar, security-domain=KerberosDatabase, allow-multiple-
  users=false, pool-prefill=false, pool-use-strict-min=false, idle-timeout-minutes=2)
  ```

  b.

  配置特定于供应商的连接属性。

  ```
  /subsystem=datasources/data-source=KerberosDS/connection-properties=
  <connection-property-name>:add(value="(<kerberos-value>)")
  ```

  *示例：**Oracle** 数据库的连接属性.*

  ```
  /subsystem=datasources/data-source=KerberosDS/connection-
  properties=oracle.net.authentication_services:add(value="(KERBEROS5)")
  ```

- *如果使用 **Elytron:***

  a.

  配置数据源以使用身份验证上下文。

  ```
  /subsystem=datasources/data-source=KerberosDS:add(connection-url="URL", min-
  pool-size=0, max-pool-size=10, jndi-name="java:jboss/datasource/KerberosDS",
  driver-name=<jdbc-driver>.jar, elytron-enabled=true, authentication-context=ds-
  context, allow-multiple-users=false, pool-prefill=false, pool-use-strict-min=false, idle-
  timeout-minutes=2)
  ```

  b.

  配置特定于供应商的连接属性。

```
/subsystem=datasources/data-source=KerberosDS/connection-properties=
<connection-property-name>:add(value="(<kerberos-value>)")
```

*示例：**Oracle** 数据库的连接属性*

```
/subsystem=datasources/data-source=KerberosDS/connection-
properties=oracle.net.authentication_services:add(value="(KERBEROS5)")
```

*在使用 **kerberos** 身份验证时，建议对数据源使用以下属性和值：*

- ***pool-prefill=false***
- ***pool-use-strict-min=false***
- ***idle-timeout-minutes***

*如需支持的属性列表，请参阅 **Datasource Attributes**。*

12.11. DATASOURCE STATISTICS

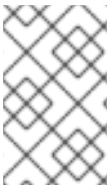
为数据源启用统计集合后，您可以查看数据源的运行时统计信息。

12.11.1. 启用数据源统计信息

*默认情况下不启用数据源统计。您可以使用管理 **CLI** 或管理控制台启用数据源统计信息收集。*

*使用管理 **CLI** 启用数据源统计信息*

*以下管理 **CLI** 命令启用 **ExampleDS** 数据源的统计信息收集：*



注意

*在受管域中，在此命令前加上 **/profile=PROFILE_NAME**。*

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-
enabled,value=true)
```

重新加载服务器以使更改生效。

使用管理控制台启用数据源统计信息

使用以下步骤，通过管理控制台为数据源启用统计信息收集。

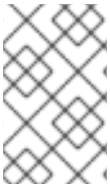
1. 导航到 **Configuration** → **Subsystems** → **Datasources & Drivers** → **Datasources**。
2. 选择数据源并点击 **View**。
3. 单击属性选项卡下的 编辑。
4. 将 **Statistics Enabled** 字段设置为 **ON**，再单击 **Save**。此时会出现一个弹出窗口，表示更改需要重新加载才能生效。
5. 重新加载服务器：
   - 对于单机服务器，请单击弹出窗口中的重新加载链接，以重新加载服务器。
   - 对于受管域，单击弹出窗口中的 **Topology** 链接。从 **Topology** 选项卡中，选择相应的服务器，再选择重新加载服务器的重新加载选项。

12.11.2. 查看数据源统计信息

您可以使用管理 **CLI** 或管理控制台查看数据源的运行时统计信息。

使用管理 **CLI** 查看数据源统计信息

以下管理 **CLI** 命令检索 **ExampleDS** 数据源的核心 池 统计信息：



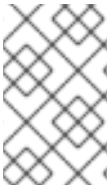
注意

在受管域中，在这些命令前加上 **/host=HOST_NAME/server=SERVER_NAME**。

```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => 1,
        "AvailableCount" => 20,
        "AverageBlockingTime" => 0L,
        "AverageCreationTime" => 122L,
        "AverageGetTime" => 128L,
        "AveragePoolTime" => 0L,
        "AverageUsageTime" => 0L,
        "BlockingFailureCount" => 0,
        "CreatedCount" => 1,
        "DestroyedCount" => 0,
        "IdleCount" => 1,
        ...
}
```

以下管理 **CLI** 命令检索 **ExampleDS** 数据源的 **JDBC** 统计数据：

```
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "PreparedStatementCacheAccessCount" => 0L,
        "PreparedStatementCacheAddCount" => 0L,
        "PreparedStatementCacheCurrentSize" => 0,
        "PreparedStatementCacheDeleteCount" => 0L,
        "PreparedStatementCacheHitCount" => 0L,
        "PreparedStatementCacheMissCount" => 0L,
        "statistics-enabled" => true
    }
}
```



注意

由于统计信息是运行时信息，因此务必指定 **include-runtime=true** 参数。

如需了解所有可用统计数据的详细列表，请参阅数据源统计信息。

使用管理控制台查看数据源统计信息

若要从管理控制台查看数据源统计信息，可从 **Runtime** 选项卡中导航到 **Datasources** 子系统，选择数据源，再单击 **View**。

如需了解所有可用统计数据的详细列表，请参阅数据源统计信息。

12.12. 数据源调优

有关监控和优化 数据源子 系统性能的提示，请参阅 性能调优指南 中的数据源和资源适配器 调优小节。

12.13. 容量策略

JBoss EAP 支持定义 **Jakarta Connectors** 部署的容量策略，包括数据源。容量策略定义池的物理连接的创建方式，称为容量递增和销毁，称为容量递减。默认策略设置为为每个请求创建一个连接以增加容量，并在为容量增加调度闲置超时时销毁所有连接。

要配置容量策略，您需要指定一个容量递增器类、容量递减器类或两者。

示例：定义容量策略

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-incrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer")

/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-decrementer-class,
value="org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer")
```

您还可以对指定容量递增器或递减器类配置属性。

示例： 为容量策略配置属性

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-incrementer-
properties.size, value=2)
```

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=capacity-decrementer-
properties.size, value=2)
```

MaxPoolSize Incrementer Policy

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.MaxPoolSizeIncrementer***

***MaxPoolSize incrementer** 策略将为每个请求填充池的最大大小。当您希望始终保持最大连接数时，此策略很有用。*

大小增加器策略

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.SizeIncrementer***

大小递增器策略将根据每个请求的指定连接数填充池。如果您想增加每个请求的连接数以增加连接数，从而预期下一个请求还需要连接，此策略很有用。

*表 **12.3.** 大小策略属性*

名称	描述
Size	应创建的连接数

注意

*这是默认的递增策略，大小值为 **1**。*

水印提高器策略

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkIncrementer***

***Watermark incrementer** 策略会将池填入每个请求的指定连接数。如果要让池中始终保持指定数量的连接，此策略很有用。*

*表 **12.4.** 水印策略属性*

名称	描述
spalmark	连接数的水位线

MinPoolSize Decrementer Policy

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.MinPoolSizeDecrementer***

***MinPoolSize** 修剪器策略会将池减小到每个请求的最小大小。如果要限制每个空闲超时请求后的连接数量，此策略很有用。池将以先出先出**(FIFO)**的方式运行。*

大小延迟策略

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.SizeDecrementer***

***Size** 减少程序策略会根据每个空闲超时请求的指定连接数来减少池。*

*表 **12.5.** 大小策略属性*

名称	描述
Size	应销毁的连接数

如果要减少每个空闲超时请求的连接数，以预计池使用量随着时间的推移会降低，此策略很有用。

*池将以先出先出**(FIFO)**的方式运行。*

***TimedOut Decrementer** 策略*

*类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutDecrementer***

***TimedOut** 减少程序策略将删除每个空闲超时请求已从池中超时的所有连接。该池将以第一个 **In Last Out(FILO)**方式运行。*

注意

此策略是默认的递减策略。

TimedOut/FIFO 发布者策略

类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.TimedOutFIFODecrementer**

TimedOutFIFO 减少**crementer** 策略将删除每个空闲超时请求已从池中超时的所有连接。池将以先出先出**(FIFO)**的方式运行。

水印 **Decrementer** 策略

类名称 ：**org.jboss.jca.core.connectionmanager.pool.capacity.WatermarkDecrementer**

Watermark 减少者策略会将池数减少到每个空闲超时请求的指定连接数。如果要让池中始终保持指定数量的连接，此策略很有用。池将以先出先出**(FIFO)**的方式运行。

表 **12.6.** 水印策略属性

名称	描述
spalmark	连接数的水位线

12.14. 加入跟踪

可以记录 **enlistment trace**，以帮助查找在 **XAResource** 实例获取期间发生的错误情况。启用插入跟踪后，**jca** 子系统会为每个池操作创建一个异常对象，以便在需要时生成准确的堆栈追踪；但也会产生性能开销。

自 **JBoss EAP 7.1** 起，默认情况下禁用了注册跟踪。您可以通过将 **enlistment -trace** 属性设置为 **true** 来使用管理 **CLI** 为数据源启用条目记录。

为非 **XA** 数据源启用加入跟踪。

```
data-source --name=DATASOURCE_NAME --enlistment-trace=true
```

启用 **XA** 数据源的加入跟踪。

```
xa-data-source --name=XA_DATASOURCE_NAME --enlistment-trace=true
```

警告

启用加入跟踪可能会对性能造成影响。

12.15. 数据源配置示例

12.15.1. MySQL Datasource 示例

以下是具有连接信息、基本安全性和验证选项的 **MySQL** 数据源配置示例。

示例：**MySQL Datasource** 配置

```
<datasources>
  <datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
    <connection-url>jdbc:mysql://localhost:3306/jbossdb</connection-url>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
      <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**MySQL JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
```

```
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MySQL JDBC** 驱动程序作为核心模块。*

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **MySQL JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-
name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MysqlXADataSource,
driver-class-name=com.mysql.cj.jdbc.Driver)
```

3. *添加 **MySQL** 数据源。*

```
data-source add --name=MySqlDS --jndi-name=java:jboss/MySqlDS --driver-name=mysql --
connection-url=jdbc:mysql://localhost:3306/jbossdb --user-name=admin --password=admin --
validate-on-match=true --background-validation=false --valid-connection-checker-class-
```

```
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

12.15.2. MySQL XA 数据源示例

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **MySQL XA** 数据源配置示例。

示例：**MySQL XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MySqlXADS" pool-name="MySqlXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mysqldb
    </xa-datasource-property>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
      <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**MySQL JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-8.0.12.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
```

```
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

管理 **CLI** 命令示例

此示例配置可通过以下管理 **CLI** 命令来实现：

1. 添加 **MySQL JDBC** 驱动程序作为核心模块。

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **MySQL JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.cj.jdbc.MysqlXADataSource,driver-class-name=com.mysql.cj.jdbc.Driver)
```

3. 添加 **MySQL XA** 数据源。

```
xa-data-source add --name=MySqlXADS --jndi-name=java:jboss/MySqlXADS --driver-name=mysql --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter --xa-datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mysqldb"}
```

12.15.3. PostgreSQL 数据源示例

以下是具有连接信息、基本安全性和验证选项的 **PostgreSQL** 数据源配置示例。

示例：**PostgreSQL** 数据源配置

```
<datasources>
  <datasource jndi-name="java:jboss/PostgresDS" pool-name="PostgresDS">
    <connection-url>jdbc:postgresql://localhost:5432/postgresdb</connection-url>
    <driver>postgresql</driver>
    <security>
        <user-name>admin</user-name>
        <password>admin</password>
    </security>
    <validation>
        <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
        <validate-on-match>true</validate-on-match>
        <background-validation>false</background-validation>
        <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="postgresql" module="com.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**PostgreSQL JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-42.x.y.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

在上例中，请确保将 **¼x.y** 替换为您的驱动程序版本号。

其他资源

- 有关 **JDBC** 驱动程序以及如何安装它们的信息，请参阅 **JDBC** 驱动程序。

- 有关已在 **JBoss EAP 7.3** 中测试的 **JDBC** 驱动程序的信息，请参阅 **JBoss EAP 7.3** 支持的配置。

管理 **CLI** 命令示例

您可以使用以下 **CLI** 命令将 **PostgreSQL JDBC** 驱动程序和 **PostgreSQL** 数据源添加到 **JDBC API** 中：

1. 添加 **PostgreSQL JDBC** 驱动程序作为核心模块。

```
module add --name=com.postgresql --resources=/path/to/postgresql-42.x.y.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

在上例中，请确保将¼**x.y** 替换为您的驱动程序版本号。

重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **PostgreSQL JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-name=postgresql,driver-module-
name=com.postgresql,driver-xa-datasource-class-
name=org.postgresql.xa.PGXADataSource)
```

3. 添加 **PostgreSQL** 数据源。

```
data-source add --name=PostgresDS --jndi-name=java:jboss/PostgresDS --driver-name=postgresql --connection-url=jdbc:postgresql://localhost:5432/postgresdb --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter
```

12.15.4. PostgreSQL XA Datasource 示例

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **PostgreSQL XA** 数据源配置示例。

示例：**PostgreSQL XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-name="PostgresXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="PortNumber">
      5432
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      postgresdb
    </xa-datasource-property>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="postgresql" module="com.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**PostgreSQL JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
```

```
<module xmlns="urn:jboss:module:1.1" name="com.postgresql">
  <resources>
    <resource-root path="postgresql-42.x.y.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

在上例中，请确保将 **x.y** 替换为您的驱动程序版本号。

其他资源

- 有关 **JDBC** 驱动程序以及如何安装它们的信息，请参阅 **JDBC** 驱动程序。
- 有关已在 **JBoss EAP 7.3** 中测试的 **JDBC** 驱动程序的信息，请参阅 **JBoss EAP 7.3** 支持的配置。

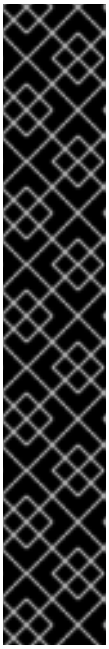
管理 **CLI** 命令示例

您可以使用以下 **CLI** 命令将 **PostgreSQL JDBC** 驱动程序和 **PostgreSQL** 数据源添加到 **JDBC API** 中：

1. 添加 **PostgreSQL JDBC** 驱动程序作为核心模块。

```
module add --name=com.postgresql --resources=/path/to/postgresql-42.x.y.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

在上例中，请确保将 **x.y** 替换为您的驱动程序版本号。



重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **PostgreSQL JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-name=postgresql,driver-module-name=com.postgresql,driver-xa-datasource-class-name=org.postgresql.xa.PGXADataSource)
```

3. 添加 **PostgreSQL XA** 数据源。

```
xa-data-source add --name=PostgresXADS --jndi-name=java:jboss/PostgresXADS --driver-name=postgresql --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter --xa-datasource-properties={"ServerName"=>"localhost","PortNumber"=>"5432","DatabaseName"=>"postgresdb"}
```

12.15.5. Oracle Datasource 示例

以下是具有连接信息、基本安全性和验证选项的 **Oracle** 数据源配置的示例。

示例：甲骨文数据源配置

```
<datasources>
  <datasource jndi-name="java:jboss/OracleDS" pool-name="OracleDS">
    <connection-url>jdbc:oracle:thin:@localhost:1521:XE</connection-url>
    <driver>oracle</driver>
    <security>
       <user-name>admin</user-name>
```

```
        <password>admin</password>
    </security>
    <validation>
        <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
        <validate-on-match>true</validate-on-match>
        <background-validation>false</background-validation>
        <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
        <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

*示例：**Oracle JDBC Driver module.xml** 文件*

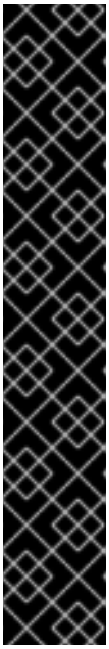
```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
    <resources>
        <resource-root path="ojdbc7.jar"/>
    </resources>
    <dependencies>
      <module name="javaee.api"/>
      <module name="sun.jdk"/>
      <module name="ibm.jdk"/>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **Oracle JDBC** 驱动程序作为核心模块。*

```
module add --name=com.oracle --resources=/path/to/ojdbc7.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```



重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

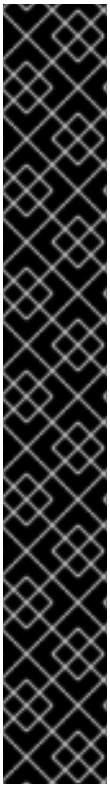
2. 注册 **Oracle JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=oracle:add(driver-name=oracle,driver-module-name=com.oracle,driver-xa-datasource-class-name=oracle.jdbc.xa.client.OracleXADataSource)
```

3. 添加 **Oracle** 数据源。

```
data-source add --name=OracleDS --jndi-name=java:jboss/OracleDS --driver-name=oracle --connection-url=jdbc:oracle:thin:@localhost:1521:XE --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter
```

12.15.6. Oracle XA 数据源示例



重要

要让 **XA** 恢复正常工作，必须应用以下设置才能访问 **Oracle XA** 数据源的用户：**value** 用户 定义为从 **JBoss EAP** 连接到 **Oracle** 的用户：

- **GRANT SELECT ON sys.dba_pending_transactions TO user;**
- **GRANT SELECT ON sys.pending_trans$ TO user;**
- **GRANT SELECT ON sys.dba_2pc_pending TO user;**
- **GRANT EXECUTE ON sys.dbms_xa TO user;**

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **Oracle XA** 数据源配置的示例。

示例：**Oracle XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/OracleXADS" pool-name="OracleXADS">
    <xa-datasource-property name="URL">
      jdbc:oracle:thin:@oracleHostName:1521:orcl
    </xa-datasource-property>
    <driver>oracle</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
      <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
```

```
    </driver>
  </drivers>
</datasources>
```

*示例：**Oracle JDBC Driver module.xml** 文件*

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc7.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **Oracle JDBC** 驱动程序作为核心模块。*

```
module add --name=com.oracle --resources=/path/to/ojdbc7.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **Oracle JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=oracle:add(driver-name=oracle,driver-module-name=com.oracle,driver-xa-datasource-class-name=oracle.jdbc.xa.client.OracleXADataSource)
```

3. *添加 **Oracle XA** 数据源.*

```
xa-data-source add --name=OracleXADS --jndi-name=java:jboss/OracleXADS --driver-name=oracle --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter --same-rm-override=false --xa-datasource-properties={"URL"=>"jdbc:oracle:thin:@oracleHostName:1521:orcl"}
```

12.15.7. Microsoft SQL Server Datasource 示例

*以下是包含连接信息、基本安全性和验证选项的 **Microsoft SQL Server** 数据源配置示例。*

*示例：**Microsoft SQL Server Datasource** 配置*

```
<datasources>
  <datasource jndi-name="java:jboss/MSSQLDS" pool-name="MSSQLDS">
    <connection-url>jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

*示例：**Microsoft SQL Server JDBC Driver module.xml File***

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
```

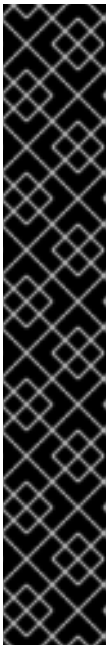
```
    <resource-root path="sqljdbc42.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="javax.xml.bind.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **Microsoft SQL Server JDBC** 驱动程序作为核心模块。*

```
module add --name=com.microsoft --resources=/path/to/sqljdbc42.jar --
dependencies=javax.api,javax.transaction.api,javax.xml.bind.api,javaee.api,sun.jdk,ibm.jdk
```



重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **Microsoft SQL Server JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-name=sqlserver,driver-module-
name=com.microsoft,driver-xa-datasource-class-
name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. *添加 **Microsoft SQL Server** 数据源。*

```
data-source add --name=MSSQLDS --jndi-name=java:jboss/MSSQLDS --driver-
name=sqlserver --connection-
url=jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase --user-name=admin --
password=admin --validate-on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter
```

12.15.8. Microsoft SQL Server XA Datasource 示例

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **Microsoft SQL Server XA** 数据源配置的示例。

示例：**Microsoft SQL Server XA Datasource** 配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MSSQLXADS" pool-name="MSSQLXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mssqldb
    </xa-datasource-property>
    <xa-datasource-property name="SelectMethod">
      cursor
    </xa-datasource-property>
    <driver>sqlserver</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**Microsoft SQL Server JDBC Driver module.xml File**

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc42.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="javax.xml.bind.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **Microsoft SQL Server JDBC** 驱动程序作为核心模块。*

```
module add --name=com.microsoft --resources=/path/to/sqljdbc42.jar --
dependencies=javax.api,javax.transaction.api,javax.xml.bind.api,javaee.api,sun.jdk,ibm.jdk
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **Microsoft SQL Server JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=sqlserver:add(driver-name=sqlserver,driver-module-
name=com.microsoft,driver-xa-datasource-class-
name=com.microsoft.sqlserver.jdbc.SQLServerXADataSource)
```

3. 添加 **Microsoft SQL Server XA** 数据源。

```
xa-data-source add --name=MSSQLXADS --jndi-name=java:jboss/MSSQLXADS --driver-name=sqlserver --user-name=admin --password=admin --validate-on-match=true --background-validation=false --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter --same-rm-override=false --xa-datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mssqldb","SelectMethod"=>"cursor"}
```

12.15.9. IBM DB2 数据源示例

以下是具有连接信息、基本安全性和验证选项的 **IBM DB2** 数据源配置的示例。

示例：**IBM DB2** 数据源配置

```
<datasources>
  <datasource jndi-name="java:jboss/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2://localhost:50000/ibmdb2db</connection-url>
    <driver>ibmdb2</driver>
    <pool>
      <min-pool-size>0</min-pool-size>
      <max-pool-size>50</max-pool-size>
    </pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
       <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**IBM DB2 JDBC Driver module.xml File**

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
```

```
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **IBM DB2 JDBC** 驱动程序作为核心模块。*

```
module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **IBM DB2 JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-name=ibmdb2,driver-module-
name=com.ibm,driver-xa-datasource-class-name=com.ibm.db2.jcc.DB2XADataSource)
```

3. *添加 **IBM DB2** 数据源。*

```
data-source add --name=DB2DS --jndi-name=java:jboss/DB2DS --driver-name=ibmdb2 --
connection-url=jdbc:db2://localhost:50000/ibmdb2db --user-name=admin --password=admin
```

```
--validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter --min-pool-size=0 --max-pool-size=50
```

12.15.10. IBM DB2 XA 数据源示例

这是使用 **XA** 数据源属性、基本安全性和验证选项的 **IBM DB2 XA** 数据源配置的示例。

示例：**IBM DB2 XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/DB2XADS" pool-name="DB2XADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      ibmdb2db
    </xa-datasource-property>
    <xa-datasource-property name="PortNumber">
      50000
    </xa-datasource-property>
    <xa-datasource-property name="DriverType">
      4
    </xa-datasource-property>
    <driver>ibmdb2</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <recovery>
      <recover-plugin class-name="org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin">
        <config-property name="EnableIsValid">
          false
        </config-property>
        <config-property name="IsValidOverride">
          false
        </config-property>
        <config-property name="EnableClose">
          false
        </config-property>
      </recover-plugin>
    </recovery>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"/>
```

```
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
      <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

*示例：**IBM DB2 JDBC Driver module.xml File***

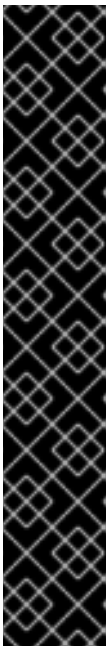
```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **IBM DB2 JDBC** 驱动程序作为核心模块。*

```
module add --name=com.ibm --resources=/path/to/db2jcc4.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```



重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **IBM DB2 JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=ibmdb2:add(driver-name=ibmdb2,driver-module-name=com.ibm,driver-xa-datasource-class-name=com.ibm.db2.jcc.DB2XADataSource)
```

3. 添加 **IBM DB2 XA** 数据源。

```
xa-data-source add --name=DB2XADS --jndi-name=java:jboss/DB2XADS --driver-name=ibmdb2 --user-name=admin --password=admin --validate-on-match=true --background-validation=false --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter --same-rm-override=false --recovery-plugin-class-name=org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin --recovery-plugin-properties={"EnableIsValid"=>"false","IsValidOverride"=>"false","EnableClose"=>"false"} --xa-datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"ibmdb2db","PortNumber"=>"50000","DriverType"=>"4"}
```

12.15.11. Sybase 数据源示例

以下是包含连接信息、基本安全性和验证选项的 **Sybase** 数据源配置示例。

示例：**Sybase Datasource Configuration**

```
<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?JCONNECT_VERSION=6</connection-url>
    <driver>sybase</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

-

示例：**Sybase JDBC Driver module.xml** 文件

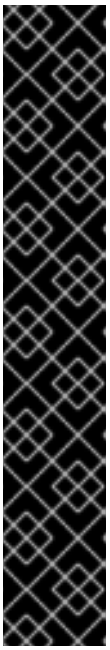
```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn4.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

管理 **CLI** 命令示例

此示例配置可通过以下管理 **CLI** 命令来实现：

1. 添加 **Sybase JDBC** 驱动程序作为核心模块。

```
module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```



重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **Sybase JDBC** 驱动程序。

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-name=sybase,driver-module-
name=com.sybase,driver-xa-datasource-class-
name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. 添加 **Sybase** 数据源。

```
data-source add --name=SybaseDB --jndi-name=java:jboss/SybaseDB --driver-name=sybase --connection-url=jdbc:sybase:Tds:localhost:5000/DATABASE?JCONNECT_VERSION=6 --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter
```

12.15.12. Sybase XA 数据源示例

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **Sybase XA** 数据源配置的示例。

示例：**Sybase XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mydatabase
    </xa-datasource-property>
    <xa-datasource-property name="PortNumber">
      4100
    </xa-datasource-property>
    <xa-datasource-property name="NetworkProtocol">
      Tds
    </xa-datasource-property>
    <driver>sybase</driver>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
```

```
    </driver>
  </drivers>
</datasources>
```

*示例：**Sybase JDBC Driver module.xml** 文件*

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn4.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **Sybase JDBC** 驱动程序作为核心模块。*

```
module add --name=com.sybase --resources=/path/to/jconn4.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **Sybase JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=sybase:add(driver-name=sybase,driver-module-name=com.sybase,driver-xa-datasource-class-name=com.sybase.jdbc4.jdbc.SybXADataSource)
```

3. 添加 **Sybase XA** 数据源。

```
xa-data-source add --name=SybaseXADS --jndi-name=java:jboss/SybaseXADS --driver-name=sybase --user-name=admin --password=admin --validate-on-match=true --background-validation=false --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter --same-rm-override=false --xa-datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mydatabase","PortNumber"=>"4100","NetworkProtocol"=>"Tds"}
```

12.15.13. MariaDB 数据源示例

以下是 **MariaDB** 数据源配置以及连接信息、基本安全性和验证选项的示例。

示例：**MariaDB** 数据源配置

```
<datasources>
  <datasource jndi-name="java:jboss/MariaDBDS" pool-name="MariaDBDS">
    <connection-url>jdbc:mariadb://localhost:3306/jbossdb</connection-url>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="mariadb" module="org.mariadb">
      <driver-class>org.mariadb.jdbc.Driver</driver-class>
      <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**MariaDB JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
```

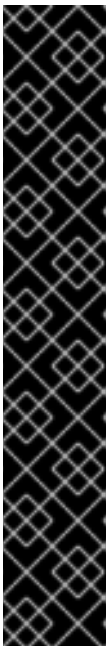
```
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MariaDB JDBC** 驱动程序作为核心模块。*

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.2.3.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```



重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **MariaDB JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-
name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,
driver-class-name=org.mariadb.jdbc.Driver)
```

3. *添加 **MariaDB** 数据源。*

```
data-source add --name=MariaDBDS --jndi-name=java:jboss/MariaDBDS --driver-
name=mariadb --connection-url=jdbc:mariadb://localhost:3306/jbossdb --user-name=admin -
-password=admin --validate-on-match=true --background-validation=false --valid-connection-
checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter
```

12.15.14. MariaDB XA 数据源示例

这是带有 **XA** 数据源属性、基本安全性和验证选项的 **MariaDB XA** 数据源配置的示例。

示例：**MariaDB XA** 数据源配置

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MariaDBXADS" pool-name="MariaDBXADS">
    <xa-datasource-property name="ServerName">
      localhost
    </xa-datasource-property>
    <xa-datasource-property name="DatabaseName">
      mariadbdb
    </xa-datasource-property>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mariadb" module="org.mariadb">
      <driver-class>org.mariadb.jdbc.Driver</driver-class>
      <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**MariaDB JDBC Driver module.xml** 文件

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.2.3.jar"/>
  </resources>
  <dependencies>
    <module name="javaee.api"/>
```

```
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MariaDB JDBC** 驱动程序作为核心模块。*

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.2.3.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **MariaDB JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-
name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,
driver-class-name=org.mariadb.jdbc.Driver)
```

3. *添加 **MariaDB XA** 数据源。*

```
xa-data-source add --name=MariaDBXADS --jndi-name=java:jboss/MariaDBXADS --driver-
name=mariadb --user-name=admin --password=admin --validate-on-match=true --
background-validation=false --valid-connection-checker-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --
```

```
exception-sorter-class-
name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter --xa-
datasource-properties={"ServerName"=>"localhost","DatabaseName"=>"mariadbdb"}
```

12.15.15. MariaDB Galera 集群数据源示例

这是 **MariaDB Galera** 集群数据源配置以及连接信息、基本安全性和验证选项的示例。

警告

MariaDB Galera 集群不支持 **XA** 事务。

示例：**Maria Galera Cluster Datasource** 配置

```
<datasources>
  <datasource jndi-name="java:jboss/MariaDBGaleraClusterDS" pool-
name="MariaDBGaleraClusterDS">
    <connection-url>jdbc:mariadb://192.168.1.1:3306,192.168.1.2:3306/jbossdb</connection-url>
    <driver>mariadb</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker"/>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"/>
    </validation>
  </datasource>
  <drivers>
    <driver name="mariadb" module="org.mariadb">
      <driver-class>org.mariadb.jdbc.Driver</driver-class>
      <xa-datasource-class>org.mariadb.jdbc.MySQLDataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

示例：**MariaDB JDBC Driver module.xml** 文件

```
<?xml version='1.0' encoding='UTF-8'?>
<module xmlns="urn:jboss:module:1.1" name="org.mariadb">
  <resources>
    <resource-root path="mariadb-java-client-1.5.4.jar"/>
```

```
  </resources>
  <dependencies>
    <module name="javaee.api"/>
    <module name="sun.jdk"/>
    <module name="ibm.jdk"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MariaDB JDBC** 驱动程序作为核心模块。*

```
module add --name=org.mariadb --resources=/path/to/mariadb-java-client-1.5.4.jar --dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **MariaDB JDBC** 驱动程序。*

```
/subsystem=datasources/jdbc-driver=mariadb:add(driver-name=mariadb,driver-module-name=org.mariadb,driver-xa-datasource-class-name=org.mariadb.jdbc.MySQLDataSource,driver-class-name=org.mariadb.jdbc.Driver)
```

3. *添加 **MariaDB Galera** 集群数据源.*

```
data-source add --name=MariaDBGaleraClusterDS --jndi-
```

name=java:jboss/MariaDBGaleraClusterDS --driver-name=mariadb --connection-url=jdbc:mariadb://192.168.1.1:3306,192.168.1.2:3306/jbossdb --user-name=admin --password=admin --validate-on-match=true --background-validation=false --valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker --exception-sorter-class-name=org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter

第 13 章 使用 AGROAL 进行数据源管理

13.1. 关于 AGROAL DATASOURCES 子系统

JBoss EAP 7.2 中引入了一个新的 数据源-agroal 子系统。这是一个由 Agroal 项目支持的高性能直接连接池。这可用作当前基于 JCA 的数据源 子系统的替代选择。

datasources-agroal 子系统默认为不可用，您必须 启用它 才能使用这一新的实施。

重要

Datasources-agroal 子系统仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

13.2. 启用 AGROAL DATASOURCES 子系统

默认的 JBoss EAP 配置中未启用 datasources-agroal 子系统。使用以下管理 CLI 命令启用它：

1. 添加 扩展名。

```
/extension=org.wildfly.extension.datasources-agroal:add
```

2. 添加 子系统。

```
/subsystem=datasources-agroal:add
```

3. 重新加载服务器以使更改生效。

```
reload
```

这会在服务器配置文件中添加下列 **XML**：

```
<server xmlns="urn:jboss:domain:8.0">
  <extensions>
    ...
    <extension module="org.wildfly.extension.datasources-agroal"/>
    ...
  </extensions>
  ...
  <subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0"/>
  ...
</server>
```

13.3. 将 JDBC 驱动程序安装为 AGROAL DATASOURCE 的核心模块

在 **JBoss EAP** 中定义 **Agroal** 数据源供应用使用之前，必须先安装相应的 **JDBC** 驱动程序。

若要将 **JDBC** 驱动程序作为 **Agroal** 数据源的核心模块安装，您必须首先 将 **JDBC** 驱动程序添加为核心模块， 然后在 **datasources-agroal** 子系统中 注册 **JDBC** 驱动程序。

13.3.1. 将 JDBC 驱动程序添加为核心模块

JDBC 驱动程序可以作为核心模块安装，使用管理 **CLI** 可以执行下列步骤：

1. 下载 **JDBC** 驱动程序。

   从您的数据库供应商下载适当的 **JDBC** 驱动程序。有关常见数据库的 **JDBC** 驱动程序的标准下载位置，请参阅 **JDBC** 驱动程序下载位置。

   如果 **JDBC** 驱动程序 **JAR** 文件包含在 **ZIP** 或 **TAR** 存档中，请确保提取存档。

2. 启动 **JBoss EAP** 服务器。

3. 启动管理 **CLI**。

   ```
   $ EAP_HOME/bin/jboss-cli.sh
   ```

4. 使用 模块 **add management CLI** 命令，添加新的核心模块。

```
module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --
dependencies=DEPENDENCIES
```

例如，以下命令添加 **MySQL JDBC** 驱动程序模块：

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

执行 模块 **--help** 获取关于使用此命令添加和删除模块的更多详细信息。

接下来，您必须将它注册为 **JDBC** 驱动程序，供应用数据源引用。

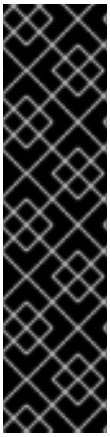
13.3.2. 为 Agroal Datasources 注册 JDBC 驱动程序

驱动程序作为核心模块安装之后，您必须使用以下管理 **CLI** 命令将它注册为 **JDBC** 驱动程序：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources-
agroal/driver=DRIVER_NAME:add(module=MODULE_NAME,class=CLASS_NAME)
```

CLASS_NAME 必须是完全限定类名称，用于为非 **XA** 数据源实施 **java.sql. Driver** 或 **javax.sql.DataSource** 或 **javax.sql.XADataSource**。

13.4. 配置非 XA 数据源



重要

Agroal 数据源的使用仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

13.4.1. 创建 Agroal Datasource

以下管理 **CLI** 命令创建一个 **Agroal** 数据源：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources-agroal/datasource=DATASOURCE_NAME:add(jndi-name=JNDI_NAME,connection-factory={driver=DRIVER_NAME,url=URL},connection-pool={max-size=MAX_POOL_SIZE})
```

有关所有可用 **Agroal** 数据源属性的完整列表，请参阅 **Agroal Datasource Attributes** 部分。

如需 **Agroal** 数据源配置示例，请参阅 **MySQL Agroal Datasource** 示例。

13.4.2. 删除 Agroal Datasource

以下管理 **CLI** 命令移除 **Agroal** 数据源：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources-agroal/datasource=DATASOURCE_NAME:remove
```

13.5. 配置 AGROAL XA 数据源

重要

Agroal 数据源的使用仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

13.5.1. 创建 Agroal XA 数据源

以下管理 **CLI** 命令创建一个 **Agroal XA** 数据源：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources-agroal/xa-datasource=XA_DATASOURCE_NAME:add(jndi-name=JNDI_NAME,connection-factory={driver=DRIVER_NAME,connection-properties={ServerName=HOST_NAME,PortNumber=PORT,DatabaseName=DATABASE_NAME}},connection-pool={max-size=MAX_POOL_SIZE})
```

有关所有可用 **Agroal** 数据源属性的完整列表，请参阅 **Agroal Datasource Attributes** 部分。

如需 **Agroal XA** 数据源配置示例，请参阅 **MySQL Agroal XA** 数据源示例。

13.5.2. 删除 Agroal XA 数据源

以下管理 **CLI** 命令删除 **Agroal XA** 数据源：在受管域中运行时，请确保在此命令前加上 **/profile=PROFILE_NAME**。

```
/subsystem=datasources-agroal/xa-datasource=DATASOURCE_NAME:remove
```

13.6. AGROAL DATASOURCE 配置示例

13.6.1. MySQL Agroal Datasource 示例

这是带有连接和基本安全设置的 **MySQL Agroal** 数据源配置的示例。

示例：**MySQL Agroal Datasource** 配置

```
<subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0">
```

```
    <datasource name="ExampleAgroalDS" jndi-name="java:jboss/datasources/ExampleAgroalDS">
      <connection-factory driver="mysql" url="jdbc:mysql://localhost:3306/jbossdb" username="admin"
password="admin"/>
      <connection-pool max-size="30"/>
    </datasource>
    <drivers>
      <driver name="mysql" module="com.mysql" class="com.mysql.cj.jdbc.Driver"/>
    </drivers>
</subsystem>
```

*示例：**MySQL JDBC Driver module.xml** 文件*

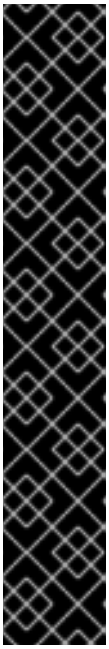
```
<?xml version='1.0' encoding='UTF-8'?>

<module xmlns="urn:jboss:module:1.1" name="com.mysql">

    <resources>
      <resource-root path="mysql-connector-java-8.0.12.jar"/>
    </resources>

    <dependencies>
      <module name="javaee.api"/>
      <module name="sun.jdk"/>
      <module name="ibm.jdk"/>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MySQL JDBC** 驱动程序作为核心模块。*

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```



重要

使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。

技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. 注册 **MySQL JDBC** 驱动程序。

```
/subsystem=datasources-
agroal/driver=mysql:add(module=com.mysql,class=com.mysql.cj.jdbc.Driver)
```

3. 添加 **MySQL** 数据源。

```
/subsystem=datasources-agroal/datasource=ExampleAgroalDS:add(jndi-
name=java:jboss/datasources/ExampleAgroalDS,connection-factory=
{driver=mysql,url=jdbc:mysql://localhost:3306/jbossdb,username=admin,password=admin},conn
ection-pool={max-size=30})
```

13.6.2. MySQL Agroal XA 数据源示例

这是带有 **XA** 数据源属性和基本安全设置的 **MySQL A** 数据源配置示例。

示例：**MySQL Agroal XA** 数据源配置

```
<subsystem xmlns="urn:jboss:domain:datasources-agroal:1.0">
    <xa-datasource name="ExampleAgroalXADS" jndi-
name="java:jboss/datasources/ExampleAgroalXADS">
        <connection-factory driver="mysqlXA" username="admin" password="admin">
            <connection-properties>
                <property name="ServerName" value="localhost"/>
                <property name="PortNumber" value="3306"/>
                <property name="DatabaseName" value="jbossdb"/>
            </connection-properties>
        </connection-factory>
        <connection-pool max-size="30"/>
```

```
    </xa-datasource>
    <drivers>
      <driver name="mysqlXA" module="com.mysql" class="com.mysql.cj.jdbc.MysqlXADataSource"/>
    </drivers>
  </subsystem>
```

*示例：**MySQL JDBC Driver module.xml** 文件*

```
<?xml version='1.0' encoding='UTF-8'?>

<module xmlns="urn:jboss:module:1.1" name="com.mysql">

    <resources>
      <resource-root path="mysql-connector-java-8.0.12.jar"/>
    </resources>

    <dependencies>
      <module name="javaee.api"/>
      <module name="sun.jdk"/>
      <module name="ibm.jdk"/>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

*管理 **CLI** 命令示例*

*此示例配置可通过以下管理 **CLI** 命令来实现：*

1. *添加 **MySQL JDBC** 驱动程序作为核心模块。*

```
module add --name=com.mysql --resources=/path/to/mysql-connector-java-8.0.12.jar --
dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
```

重要

*使用 模块 管理 **CLI** 命令添加和删除模块，仅作为技术预览提供。此命令不适合在受管域中使用，或在远程连接管理 **CLI** 时使用。在生产环境中，应当手动添加 和删除模块。*

*技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。*

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

2. *注册 **MySQL XA JDBC** 驱动程序。*

```
/subsystem=datasources-agroal/driver=mysqlXA:add(module=com.mysql,class=com.mysql.cj.jdbc.MysqlXADataSource)
```

3. *添加 **MySQL XA** 数据源。*

```
/subsystem=datasources-agroal/xa-datasource=ExampleAgroalXADS:add(jndi-name=java:jboss/datasources/ExampleAgroalXADS,connection-factory={driver=mysqlXA,connection-properties={ServerName=localhost,PortNumber=3306,DatabaseName=jbossdb},username=admin,password=admin},connection-pool={max-size=30})
```

第 14 章 配置事务子系统

通过 transactions 子系统，您可以配置事务管理器(TM)选项，如超时值、事务记录、统计信息收集，以及是否使用 JTS。JBoss EAP 使用 Narayana 框架提供事务服务。此框架利用基于标准的各种交易协议的支持，如 Jakarta Transactions、JTS 和 Web 服务事务。

如需更多信息，请参阅管理 JBoss EAP 上的事务。

第 15 章 ORB 配置

15.1. 关于通用对象请求代理架构(CORBA)

通用对象请求代理架构(CORBA)是一种标准，使应用程序和服务能够一起工作，即使它们以多种其他兼容语言、语言编写或在单独的平台上托管。CORBA 请求由服务器端组件代理，称为对象请求代理(ORB)。JBoss EAP 通过 Open JDKORB 组件提供 ORB 实例。

ORB 在内部用于 JTS 事务，也可供您自己的应用使用。

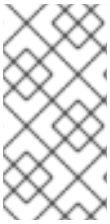
注意

对象交易服务(OTS)是一种跨平台服务，构成 CORBA 的一部分。OTS 规范由对象管理组维护。JTS 是构建事务管理器的规范，而 JTS 则是根据 OTS 规范设计的。

有关 CORBA 及其组件的详情，请参考通用对象请求代理架构。

15.2. 为 JTS 配置 ORB

在 JBoss EAP 的默认安装中，针对事务的对象请求代理(ORB)支持被禁用。您可以使用管理 CLI 或管理控制台，在 iiop-openjdk 子系统中配置 ORB 设置。



注意

在受管域中使用 full 或 full - ha 配置文件或 standalone- full.xml 或 standalone-full-ha.xml 配置文件时，可以使用 iiop- openjdk 子系统。

有关 iiop-openjdk 子系统的可用配置选项列表，请参阅 IIOP Subsystem Attributes。

使用管理 CLI 配置 ORB

您可以使用管理 CLI 配置 ORB 的各个方面。这是用于 JTS 的 ORB 的最小配置。

您可以使用 full 配置文件为受管域配置以下管理 CLI 命令：如有必要，请更改配置集，使其适合您需要配置的配置文件。如果您使用的是单机服务器，请省略命令的 /profile=full 部分。

启用安全拦截器

*通过将值设置为 **identity** 来启用 **security** 属性。*

```
/profile=full/subsystem=iiop-openjdk:write-attribute(name=security,value=identity)
```

*在 **IIOP** 子系统中启用事务*

*要为 **JTS** 启用 **ORB**，请将 **transactions** 属性 的值设置为 **full**，而不是默认的 **spec**。*

```
/profile=full/subsystem=iiop-openjdk:write-attribute(name=transactions, value=full)
```

*在事务子系统中启用 **JTS***

```
/profile=full/subsystem=transactions:write-attribute(name=jts,value=true)
```

注意

*对于 **JTS** 激活，服务器必须重新启动，因为重新加载是不够的。*

*使用管理控制台配置 **ORB***

1. *从管理控制台顶部选择 **Configuration** 选项卡。在受管域中，您必须选择要修改的适当配置文件。*
2. *选择 **Subsystems** → **IIOP(OpenJDK)** 并点击 **View**。*
3. *单击 **Edit**，并根据需要修改属性。*
4. *单击 **Save** 以保存更改。*

15.3. 配置 IIOP 以通过 ELYTRON 子系统使用 SSL/TLS

*您可以将 **iiop-openjdk** 子系统配置为使用 **SSL/TLS** 来保护客户端和服务器之间的通信。**elytron** 子系统以及旧版 安全 子系统为 **iiop-openjdk** 子系统以及 **JBoss EAP** 中的其他子系统配置 **SSL/TLS** 所需的组件。使用以下步骤将 **iiop-openjdk** 子系统配置为使用 **elytron** 子系统 **SSL/TLS**。*

1. *使用以下管理 **CLI** 命令，在 **iiop-openjdk** 子系统中显示当前传统的 **SSL/TLS** 配置：*

```
/subsystem=iiop-openjdk:read-attribute(name=security-domain)
{
    "outcome" => "success",
    "result" => "iiopSSLSecurityDomain"
}
```

iiop-openjdk 子系统必须使用传统 安全 子系统或 **elytron** 子系统 **SSL/TLS**。您不能同时使用这两者。以上命令显示 **iiop-openjdk** 子系统正在使用传统安全域来处理 **SSL/TLS**。在将 **iiop-openjdk** 子系统配置为使用 **SSL/TLS** 的 **elytron** 子系统之前，您需要删除此引用：

```
/subsystem=iiop-openjdk:undefine-attribute(name=security-realm)
```

如果未定义 **iiop-openjdk** 中的 **security- domain** 属性，您可以继续下一步。

2. 创建 **server-ssl-context**。

要将 **SSL/TLS** 与 **iiop-openjdk** 子系统结合使用，您需要定义 **server-ssl-context**。**JBoss EAP** 将 **SSL/TLS** 连接作为服务器时，使用 **server-ssl-context** 提供的配置。您可以使用 **Elytron Subsystem** 了解如何 配置服务器安全 指南中的 **Elytron Subsystem** 在为应用启用单向 **SSL/TLS** 中创建 **server- ssl-context**。

3. 创建 **client-ssl-context**。

要将 **SSL/TLS** 与 **iiop-openjdk** 子系统结合使用，您需要定义 **client-ssl-context**。**JBoss EAP** 在将 **SSL/TLS** 连接作为客户端时，使用 **client-ssl-context** 提供的配置。您可以在 如何配置服务器安全 指南 中的使用 **client-ssl-context** 中找到有关创建客户端**-ssl-context** 的更多详细信息。

4. 配置 **iiop-openjdk** 子系统，以使用 **client-ssl-context** 和 **server-ssl-context**。

示例：设置 **client-ssl-context** 和 **server-ssl-context**

```
batch

/subsystem=iiop-openjdk:write-attribute(name=client-ssl-context,value=iiopClientSSC)

/subsystem=iiop-openjdk:write-attribute(name=server-ssl-context,value=iiopServerSSC)
```

```
run-batch

reload
```

5. 从 **iiop-openjdk** 子系统配置与 和 的连接。

   您可以通过调整以下属性来指示在连接到 **iiop-openjdk** 子系统时是否需要 **SSL/TLS** 连接：

   - 要在 **iiop-openjdk** 子系统中启用对 **SSL** 的支持，可将 **support-ssl** 设置为 **true**。默认值为 **false**。
   - 要要求从 **iiop-openjdk** 子系统进行 **SSL/TLS** 连接，请将 **client-requires-ssl** 设置为 **true**。默认值为 **false**。
   - 要要求 **SSL/TLS** 连接到 **iiop-openjdk** 子系统，请将 **server-requires-ssl** 设置为 **true**。默认值为 **false**。请注意，此设置为 **true** 将阻止尝试连接到非 **SSL IIOP** 套接字。
   - 若要调整 **socket-binding**，请将 **ssl-socket-binding** 设置为所需的绑定：默认为 **iiop-ssl**。

   示例：将 **SSL/TLS** 连接设置为 **IIOP** 以及从 **IIOP** 设为 **Required**

```
/subsystem=iiop-openjdk:write-attribute(name=support-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=client-requires-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=server-requires-ssl,value=true)

/subsystem=iiop-openjdk:write-attribute(name=ssl-socket-binding,value=iiop-ssl)

reload
```

第 16 章 JAKARTA CONNECTORS MANAGEMENT

16.1. 关于 JAKARTA CONNECTORS

Jakarta Connectors 为 Jakarta EE 系统定义为外部异构企业信息系统(EIS)的标准架构。EIS 的示例包括企业资源规划(ERP)系统、大型机交易处理(TP)、数据库和消息传递系统。资源适配器是实施 Jakarta Connectors 架构的组件。

Jakarta Connectors 1.7 提供了管理以下功能：

- 连接
- 事务
- security
- life-cycle
- 工作实例
- 事务内流
- 消息内流

16.2. 关于资源适配器

资源适配器是可部署的 Jakarta EE 组件，提供使用 Jakarta Connectors 规范的 Jakarta EE 应用和企业信息系统(EIS)之间的通信。EIS 供应商通常提供资源适配器，以便其产品与 Jakarta EE 应用轻松集成。

企业信息系统可以是组织内部的任何其他软件系统。例如，企业资源规划(ERP)系统、数据库服务器、电子邮件服务器和专有消息传递系统。

资源适配器打包在资源适配器存档(RAR)文件中，可以部署到 JBoss EAP 中。RAR 文件也可以包含在企业存档(EAR)部署中。

资源适配器本身在 resource-adapters 子系统内定义，该子系统由 IronJacamar 项目提供。

16.3. 配置 JCA 子系统

jca 子系统控制 Jakarta Connectors 容器和资源适配器部署的常规设置。您可以使用管理控制台或管理 CLI 配置 jca 子系统。

要配置的主要 jca 子系统元素有：

- 归档验证
- bean 验证
- 工作经理
- 分布式工作经理
- bootstrap 上下文
- 缓存的连接管理器

从管理控制台配置 jca 子系统设置

jca 子系统可以从管理控制台进入 Configuration → Subsystems → JCA 并点击 View。然后，选择适当的标签页：

- 配置 - 包含缓存的连接管理器的设置、存档验证和 bean 验证。它们各自包含在各自的选项卡中。打开适当的选项卡并单击编辑链接，修改这些设置。
-

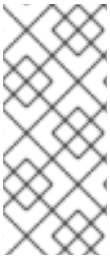
Bootstrap 上下文 - 包含配置的 **bootstrap** 上下文列表。可以添加、删除和配置新的 **bootstrap** 上下文对象。必须为每个 **bootstrap** 上下文分配一个工作管理器。

- **WorkManager -** 包含已配置的工作管理器的列表。可以在此处添加、移除新工作管理器及其线程池。每个工作管理器都有一个运行较短的线程池，以及可选的长时间运行线程池。

单击所选工作管理器上的线程池属性，可以配置线程池属性。

从管理 **CLI** 配置 **jca** 子系统设置

jca 子系统可以从管理 **CLI** 从 **/subsystem=jca** 地址进行配置。在受管域中，您必须在命令前加上 **/profile=PROFILE_NAME**。



注意

以下部分中表中的属性名称列在管理模型中时，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-jca_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。

归档验证

这决定了是否将对部署单元执行存档验证。下表描述了您可以为归档验证设置的属性：

表 **16.1.** 归档验证属性

属性	默认值	描述
enabled	true	指定是否启用归档验证。
fail-on-error	true	指定归档验证错误报告是否失败。
fail-on-warn	false	指定存档验证警告报告是否失败。

如果存档没有正确实施 **Jakarta Connectors** 规格并启用了存档验证，则在描述问题的部署期间将显示错误消息。例如：

```
Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public int hashCode()" method.
Code: com.mycompany.myproject.ResourceAdapterImpl

Severity: ERROR
```

> *Section: 19.4.2*
> *Description: A ResourceAdapter must implement a "public boolean equals(Object)" method.*
> *Code: com.mycompany.myproject.ResourceAdapterImpl*

如果未指定存档验证，则被视为存在存档，并且 已启用 的属性默认为 ***true***。

Bean *验证*

此设置确定是否对部署单元执行 ***JSR-303*** *中定义的* ***bean*** *验证。 下表介绍了您可以为* ***bean*** *验证设置的属性：用于* ***bean*** *验证的* ***Jakarta*** *等效于* ***Jakarta Bean Validation***。

表 ***16.2. Bean*** *验证属性*

属性	默认值	描述
enabled	true	指定是否启用了 bean 验证。

如果未指定 ***bean*** *验证， 它将被视为存在， 并且 已启用 的属性默认为* ***true***。

工作管理器

有两种工作管理器：

默认工作管理器

默认工作管理器及其线程池。

自定义工作管理器

自定义工作管理器定义及其线程池。

下表描述了您可以为工作管理器设置的属性：

表 ***16.3.*** *工作管理器属性*

属性	描述
name	指定工作管理器的名称。
elytron-enabled	此属性为 **workmanager** 启用 Elytron 安全性。

工作管理器还包含以下子元素：

*表 **16.4.** 工作管理器子元素*

子元素	描述
short-running-threads	标准工作实例的线程池.每个工作管理器都有一个运行较短的线程池。
long-running-threads	Jakarta Connectors 1.7 线程池，用于设置 **LONG_RUNNING 提示** 的实例。每个工作管理器都可以有一个可选的长时间运行线程池。

下表描述了您可以为工作管理器线程池设置的属性。

*表 **16.5.** 线程池属性*

属性	描述
allow-core-timeout	确定核心线程是否可以超时的布尔设置。默认值为 **false**。
core-threads	内核线程池大小。这必须等于或小于最大线程池大小。
handoff-executor	在任务无法接受的情况下，将任务委派给 的执行者。如果未指定，则无法接受的任务将被静默丢弃。
keepalive-time	指定池线程在工作后应保留的时间。
max-threads	最大线程池大小。
name	指定线程池的名称。
queue-length	最大队列长度。
thread-factory	指线程工厂.

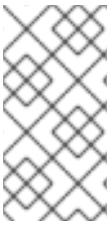
分布式工作管理器

分布式工作管理器是一个工作管理器实例，可以在另一个工作管理器实例上重新排期工作。

*以下示例管理 **CLI** 命令配置分布式工作管理器：请注意，您必须使用提供高可用性功能的配置，如 **standalone-ha.xml** 或 **standalone-full-ha.xml** 配置文件（用于单机服务器）。*

示例：配置分布式工作管理器

```
batch
/subsystem=jca/distributed-workmanager=myDistWorkMgr:add(name=myDistWorkMgr)
/subsystem=jca/distributed-workmanager=myDistWorkMgr/short-running-
threads=myDistWorkMgr:add(queue-length=10,max-threads=10)
/subsystem=jca/bootstrap-
context=myCustomContext:add(name=myCustomContext,workmanager=myDistWorkMgr)
run-batch
```



注意

***short-running-threads** 元素的名称必须与 **distributed-workmanager** 元素的名称相同。*

下表描述了您可以为分布式工作管理器配置的属性：

*表 **16.6.** 分布式工作管理器属性*

属性	描述
elytron-enabled	为工作管理器启用 Elytron 安全性。
name	分布式工作管理器的名称。
policy	该策略决定何时重新分发工作实例。允许的值有： • **NEVER** - 始终将工作实例分发到另一节点。 • **ALWAYS** - 将 work 实例分发到另一节点。 • **WATERMARK** - 根据当前节点可用的可用工作线程数量，将工作实例分发到另一节点。
policy-options	策略的键/值对选项列表。如果您使用 **WATERMARK** 策略，您可以使用 **水mark** 策略选项指定应该分发的可用线程数量。例如： `/subsystem=jca/distributed-workmanager=myDistWorkMgr:write-attribute(name=policy-options,value={watermark=3})`

属性	描述
selector	选择器决定网络中哪些节点重新分发工作实例。允许的值有： • **FIRST_AVAILABLE** - 选择列表中的第一个可用节点。 • **PING_TIME** - 选择 ping 时间最低的节点。 • **MAX_FREE_THREADS** - 选择可用 worker 线程数最多的节点。
selector-options	选择器的键/值对选项列表。

分布式工作管理器还包含以下子元素：

*表 **16.7.** 分布式工作管理器子元素*

子元素	描述
long-running-threads	设置 **LONG_RUNNING** 提示的工作实例的线程池。每个分布式工作管理器都可以有长时间运行的线程池。
short-running-threads	标准工作实例的线程池。每个分布式工作管理器都必须有一个运行较短的线程池。

***bootstrap** 上下文*

*这用于定义自定义 **bootstrap** 上下文。下表描述了您可以为 **bootstrap** 上下文设置的属性。*

*表 **16.8. Bootstrap** 上下文属性*

属性	描述
name	指定 bootstrap 上下文的名称。
WorkManager	指定用于此上下文的工作管理器的名称。

缓存的连接管理器

*这可用于调试连接并支持事务中的连接的 **lazy enlist**，跟踪这些连接是否被应用正确使用和发布。下表描述了您可以为缓存的连接管理器设置的属性：*

*表 **16.9.** 缓存的连接管理器属性*

属性	默认值	描述
debug	false	输出失败时的警告以明确关闭连接。
错误	false	在失败时抛出异常以明确关闭连接.
ignore-unknown-connections	false	指定不会缓存未知连接。
install	false	启用或禁用缓存的连接管理器 valve 和拦截器。

16.4. 配置资源适配器

16.4.1. 部署资源适配器

*资源适配器可以像其他部署一样使用管理 **CLI** 或管理控制台进行部署。在运行单机服务器时，您还可以将存档复制到部署目录，供部署扫描器提取。*

*使用管理 **CLI** 部署资源适配器*

要将资源适配器部署到单机服务器，请输入以下命令。

```
deploy /path/to/resource-adapter.rar
```

*要将资源适配器部署到受管域中的所有服务器组，请输入以下命令管理 **CLI**：*

```
deploy /path/to/resource-adapter.rar --all-server-groups
```

使用管理控制台部署资源适配器

1. *登录管理控制台，再单击 **Deployments** 选项卡。*
2. *单击添加**(+)**按钮。在受管域中，您将首先需要选择内容存储库。*
3. *选择 **Upload Deployment** 选项。*
4. *浏览资源适配器存档并单击 **Next**。*

5. 验证上传，然后单击"完成"。

6. 在受管域中，将部署部署到适当的服务器组并启用部署。

使用 **Deployment Scanner** 部署资源适配器

若要手动将资源适配器部署到单机服务器，请将资源适配器存档复制到服务器部署目录，如 **EAP_HOME/standalone/deployments/**。这将由部署扫描器提取和部署。

注意

此选项不适用于受管域。您必须使用管理控制台或管理 **CLI** 将资源适配器部署到服务器组。

16.4.2. 配置资源适配器

您可以使用管理接口配置资源适配器。以下示例演示了如何使用管理 **CLI** 配置资源适配器。有关支持的属性和其他重要信息，请参阅您的资源适配器厂商文档。

添加资源适配器配置

添加资源适配器配置。

```
/subsystem=resource-adapters/resource-adapter=eis.rar:add(archive=eis.rar, transaction-support=XATransaction)
```

配置资源适配器设置

根据需要配置以下任何设置：

- 配置配置属性.

  添加 服务器配置 属性。

  ```
  /subsystem=resource-adapters/resource-adapter=eis.rar/config-properties=server:add(value=localhost)
  ```

  添加 端口 配置属性。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/config-properties=port:add(value=9000)
```

- 配置 **admin-object**。

  添加 **admin** 对象。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName:add(class-name=com.acme.eis.ra.EISAdminObjectImpl, jndi-name=java:/eis/AcmeAdminObject)
```

  配置 **admin** 对象配置属性。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName/config-properties=threshold:add(value=10)
```

- 配置 连接定义**.**

  为受管连接工厂添加连接定义。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName:add(class-name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-name=java:/eis/AcmeConnectionFactory)
```

  配置受管连接工厂配置属性。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName/config-properties=name:add(value=Acme Inc)
```

  配置是否记录加入跟踪。您可以通过将 **enlistment -trace** 属性设置为 **true** 来启用条目记录。

```
/subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName:write-attribute(name=enlistment-trace,value=true)
```

警告

启用采用跟踪功能可以简化事务加载期间跟踪错误，但对性能有影响。

如需资源适配器的所有可用配置选项，请参阅资源适配器属性。

激活资源适配器

激活资源适配器。

```
/subsystem=resource-adapters/resource-adapter=eis.rar:activate
```

注意

您还可以为资源适配器定义容量策略。如需了解更多详细信息，请参阅 **Capacity Policies** 部分。

16.4.3. 配置资源适配器以使用 Elytron 子系统

在 **IronJacamar** 中的服务器和资源适配器之间有两种通信。

其中一个是服务器何时打开资源适配器连接。如规格中所定义，这可以通过容器管理的登录进行保护，这需要在打开连接时将 **JAAS** 主题及主体和凭证传播到资源适配器。此登录可委派给 **Elytron**。

IronJacamar 支持安全性内流。这种机制使得资源适配器能够在向工作管理器提交工作时，以及在将消息传送到驻留在同一 **JBoss EAP** 实例中的端点时建立安全信息。

容器管理的登录

若要通过 **Elytron** 实现容器管理的登录，需要将 **elytron-enabled** 属性设为 **true**。这将导致与 **Elytron** 保护的资源适配器的所有连接。

```
/subsystem=resource-adapters/resource-adapter=RAR_NAME/connection-
definitions=FACTORY_NAME:write-attribute(name=elytron-enabled,value=true)
```

若要配置 **elytron-enabled** 属性，可以使用管理 **CLI** 将 **resource -adapters** 子系统中的 **elytron-enabled** 属性设置为 **true**。默认情况下，此属性设为 **false**。

authentication-context 属性定义要用于执行登录的 **Elytron** 身份验证上下文的名称。

Elytron authentication-context 属性可以包含一个或多个 身份验证配置 元素，后者包含您要使用的凭据。

如果未设置 **authentication-context** 属性，**JBoss EAP** 将使用当前的 **authentication-context**，这是供开启 连接的调用者代码使用的 **authentication- context**。

示例：创建 身份验证配置

```
/subsystem=elytron/authentication-configuration=exampleAuthConfig:add(authentication-name=sa,credential-reference={clear-text=sa})
```

示例：使用上述配置创建 **authentication-context**

```
/subsystem=elytron/authentication-context=exampleAuthContext:add(match-rules=[{authentication-configuration=exampleAuthConfig}])
```

安全流

资源管理器也可以在提交将由工作管理器执行的工作时引入安全凭据。安全进流允许工作在执行前对自身进行身份验证。如果身份验证成功，则提交的工作将在生成的身份验证上下文下执行。如果失败，则工作执行将被拒绝。

要启用 **Elytron** 安全内流，请在配置资源适配器工作管理器时设置 **wm-elytron-security-domain** 属性。**Elytron** 将根据指定的域执行身份验证。

注意

*当资源适配器工作管理器配置为使用 **Elytron** 安全域 **wm-elytron-security-domain** 时，引用的工作管理器应将 **elytron-enabled** 属性设为 **true**。*

```
/subsystem=jca/workmanager=customWM:add(name=customWM, elytron-enabled=true)
```

注意

*如果使用 **wm-security-domain** 属性而不是 **wm-elytron -security-domain** 属性，则安全流将由传统 安全 子系统来执行。*

*在下面 **jca** 子系统的示例配置中，我们可以看到名为 **ra-with-elytron-security-domain** 的资源适配器的配置。此资源适配器将工作管理器安全性配置为使用 **Elytron** 安全域的 **wm-realm**。*

```
<subsystem xmlns="urn:jboss:domain:jca:5.0">
    <archive-validation enabled="true" fail-on-error="true" fail-on-warn="false"/>
    <bean-validation enabled="true"/>
    <default-workmanager>
        <short-running-threads>
            <core-threads count="50"/>
            <queue-length count="50"/>
            <max-threads count="50"/>
            <keepalive-time time="10" unit="seconds"/>
        </short-running-threads>
        <long-running-threads>
            <core-threads count="50"/>
            <queue-length count="50"/>
            <max-threads count="50"/>
            <keepalive-time time="10" unit="seconds"/>
        </long-running-threads>
    </default-workmanager>
    <workmanager name="customWM">
        <elytron-enabled>true</elytron-enabled>
        <short-running-threads>
            <core-threads count="20"/>
            <queue-length count="20"/>
            <max-threads count="20"/>
        </short-running-threads>
    </workmanager>
    <bootstrap-contexts>
        <bootstrap-context name="customContext" workmanager="customWM"/>
    </bootstrap-contexts>
    <cached-connection-manager/>
</subsystem>
```

然后，使用 **resource-adapter** 子系统中的 **boostrap** 上下文引用工作管理器。

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:5.0">
   <resource-adapters>
      <resource-adapter id="ra-with-elytron-security-domain">
         <archive>
            ra-with-elytron-security-domain.rar
         </archive>
         <bootstrap-context>customContext</bootstrap-context>
         <transaction-support>NoTransaction</transaction-support>
         <workmanager>
            <security>
               <elytron-security-domain>wm-realm</elytron-security-domain>
               <default-principal>wm-default-principal</default-principal>
               <default-groups>
                  <group>
                     wm-default-group
                  </group>
               </default-groups>
            </security>
         </workmanager>
      </resource-adapter>
   </resource-adapters>
</subsystem>
```

示例：安全域的配置

```
/subsystem=elytron/properties-realm=wm-properties-realm:add(users-properties={path=/security-dir/users.properties, plain-text=true}, groups-properties={path=/security-dir/groups.properties})

/subsystem=elytron/simple-role-decoder=wm-role-decoder:add(attribute=groups)

/subsystem=elytron/constant-permission-mapper=wm-permission-mapper:add(permissions=[{class-name="org.wildfly.security.auth.permission.LoginPermission"}])

/subsystem=elytron/security-domain=wm-realm:add(default-realm=wm-properties-realm, permission-mapper=wm-permission-mapper, realms=[{role-decoder=wm-role-decoder, realm=wm-properties-realm}])
```

Work 类负责提供指定域下 **Elytron** 身份验证的凭据。为此，它必须实施 **javax.resource.spi.work.WorkContextProvider**。

```
public interface WorkContextProvider {
   /**
```

```
     * Gets an instance of <code>WorkContexts</code> that needs to be used
     * by the <code>WorkManager</code> to set up the execution context while
     * executing a <code>Work</code> instance.
     *
     * @return an <code>List</code> of <code>WorkContext</code> instances.
     */
    List<WorkContext> getWorkContexts();
}
```

此界面允许 工作 类使用 **WorkContext** 来配置要在其中执行任务的上下文的某些方面。其中一个方面是安全进化。为此， **List<WorkContext> getWorkContexts** 方法必须提供 **javax.resource.spi.work.SecurityContext**。此上下文将使用 **JSR-196** 规范 中定义的 **javax.security.auth.callback.Callback** 对象来提供凭据。用于容器服务供应商接口身份验证的 **Jakarta** 等同于 **Jakarta** 身份验证。

示例：使用上下文创建回调

```
public class ExampleWork implements Work, WorkContextProvider {

   private final String username;
   private final String role;

   public MyWork(TestBean bean, String username, String role) {
      this.principals = null;
      this.roles = null;
      this.bean = bean;
      this.username = username;
      this.role = role;
   }

   public List<WorkContext> getWorkContexts() {
      List<WorkContext> l = new ArrayList<>(1);
      l.add(new MySecurityContext(username, role));
      return l;
   }

   public void run() {
      ...
   }

   public void release() {
      ...
   }

   public class ExampleSecurityContext extends SecurityContext {

      public void setupSecurityContext(CallbackHandler handler, Subject executionSubject,
Subject serviceSubject) {
         try {
            List<javax.security.auth.callback.Callback> cbs = new ArrayList<>();
            cbs.add(new CallerPrincipalCallback(executionSubject, new
```

```
SimplePrincipal(username)));
                cbs.add(new GroupPrincipalCallback(executionSubject, new String[]{role}));
                handler.handle(cbs.toArray(new javax.security.auth.callback.Callback[cbs.size()]));
            } catch (Throwable t) {
                throw new RuntimeException(t);
            }
        }
    }
```

在上例中，**Examp leWork** 实施 **WorkContextProvider** 接口来提供 **ExampleSecurityContext**。该上下文将创建必要的回调，以提供 **Elytron** 在工作执行时验证的安全信息。

16.4.4. 部署和配置 IBM MQ 资源适配器

您可以在配置 **JBoss EAP** 的消息传递中找到 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/configuring_messaging/#deploy_the_ibm_mq_resource_adapter** 部署 **IBM MQ** 资源适配器的说明。

16.4.5. 部署和配置通用 JMS 资源适配器

您可以在配置 **JBoss EAP** 的消息传递中查看配置通用 **JMS** 资源适配器的说明。

16.5. 配置受管连接池

JBoss EAP 提供三种 **ManagedConnectionPool** 接口实施：

org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool

这是 **JBoss EAP 7** 中的默认连接池，提供最佳的现成性能。

org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool

这是之前 **JBoss EAP** 版本中的默认连接池。

org.jboss.jca.core.connectionmanager.pool.mcp.LeakDumperManagedConnectionPool

此连接池仅用于调试目的，并将在关机或池清空时报告任何泄漏。

您可以使用以下管理 **CLI** 命令，为数据源设置受管连接池实施。

```
/subsystem=datasources/data-source=DATA_SOURCE:write-
attribute(name=mcp,value=MCP_CLASS)
```

您可以使用以下管理 **CLI** 命令，为资源适配器设置受管连接池实施。

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-
definitions=CONNECTION_DEFINITION:write-attribute(name=mcp,value=MCP_CLASS)
```

您可以使用以下管理 **CLI** 命令，为消息传递服务器设置受管连接池实施。

```
/subsystem=messaging-activemq/server=SERVER/pooled-connection-
factory=CONNECTION_FACTORY:write-attribute(name=managed-connection-
pool,value=MCP_CLASS)
```

16.6. 查看连接统计信息

您可以从 **/deployment=NAME.rar** 子树读取已定义连接的统计信息。这样，您就可以访问任何 **RAR** 的统计信息，即使它没有在 **standalone.xml** 或 **domain.xml** 配置中定义。

```
/deployment=NAME.rar/subsystem=resource-adapters/statistics=statistics/connection-
definitions=java\:/testMe:read-resource(include-runtime=true)
```

注意

务必指定 **include-runtime=true** 参数，因为所有统计信息都是仅运行时信息。

如需有关可用统计数据的信息，请参阅资源适配器统计信息。

16.7. 清空资源适配器连接

您可以使用以下管理 **CLI** 命令清空资源适配器连接。

注意

在受管域中，您必须在这些命令前使用 **/host=HOST_NAME/server=SERVER_NAME**。

- 清空池中的所有连接。

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-all-connection-in-pool
```

- 正常清空池中的所有连接。

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-gracefully-connection-in-pool
```

服务器将等待连接闲置，然后清空连接。

- 清空池中的所有空闲连接。

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-idle-connection-in-pool
```

- 清空池中的所有无效连接。

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER/connection-definitions=CONNECTION_DEFINITION:flush-invalid-connection-in-pool
```

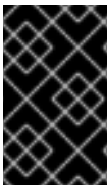
服务器将清空其确定无效的所有连接。

16.8. 调整资源适配器子系统

有关监控和优化 资源适配器子 系统性能的提示，请参阅 性能调优指南 中的数据源和资源适配器 调整小节。

第 17 章 配置 WEB 服务器(UNDERTOW)

17.1. UNDERTOW 子系统概述



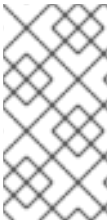
重要

在 **JBoss EAP 7** 中，**undertow** 子系统取代了 **JBoss EAP 6** 中的 **Web** 子系统。

undertow 子系统可用于配置 **Web** 服务器和 **servlet** 容器设置。它实施 **Jakarta Servlet 4.0** 规范和 **websocket**。它还支持 **HTTP** 升级，并在 **servlet** 部署中使用高性能非阻塞处理程序。**undertow** 子系统也能够充当支持 **mod_cluster** 的高性能反向代理。

在 **undertow** 子系统内，需要配置五个主要组件：

- 缓冲区缓存
- **server**
- **Servlet** 容器
- 处理程序
- 过滤器



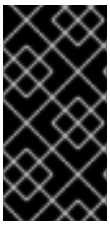
注意

尽管 **JBoss EAP** 确实提供了更新每个组件的配置，但默认配置适用于大多数用例，并且提供合理的性能设置。

默认 **Undertow** 子系统配置

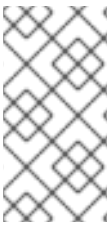
```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-host="default-host" default-servlet-container="default" default-security-domain="other">
    <buffer-cache name="default"/>
    <server name="default-server">
```

```
        <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true"/>
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content"/>
            <http-invoker security-realm="ApplicationRealm"/>
        </host>
    </server>
    <servlet-container name="default">
        <jsp-config/>
        <websockets/>
    </servlet-container>
    <handlers>
        <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
    </handlers>
</subsystem>
```



重要

***undertow** 子系统也依赖于 **io** 子系统来提供 **XNIO** 工作线程和缓冲区池。**The io** 子系统单独配置，提供默认配置，这在大部分情形中都能提供最佳性能。*



注意

*与 **JBoss EAP 6** 中的 **Web** 子系统相比，**JBoss EAP 7** 中的 **undertow** 子系统具有不同的 **HTTP** 方法默认行为。*

*将 **Elytron** 与 **Undertow** 子系统搭配使用*

*部署 **Web** 应用时，将识别该应用所需的安全域名称。这可以来自部署内部，或者部署没有安全域，则将假定 **undertow** 子系统中定义的 **default-security-domain**。默认情况下，假定安全域映射到旧安全子系统中定义的 **PicketBox**。不过，可以将 **application-security-domain** 资源添加到 **undertow** 子系统中，该子系统从应用所需的安全域名称映射到适当的 **Elytron** 配置。*

示例：添加映射.

```
/subsystem=undertow/application-security-domain=ApplicationDomain:add(security-
domain=ApplicationDomain)
```

如果结果为：

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" ... default-security-domain="other">
...
    <application-security-domains>
        <application-security-domain name="ApplicationDomain" security-domain="ApplicationDomain"/>
    </application-security-domains>
...
</subsystem>
```

注意

- 如果此时部署部署，应重新加载应用服务器，使应用安全域映射生效。
- 在当前的 **Web service-Elytron** 集成中，为保护 **Web** 服务端点而指定的安全域名称，**Elytron** 安全域名必须相同。

这种简单形式适合在部署中使用 **Servlet** 规范中定义的标准 **HTTP** 机制，如 **BASIC**、**CLIENT_CERT**、**DIGEST**、**FORM**。此处将对 **ApplicationDomain** 安全域进行身份验证。此表单也适用于应用不使用任何身份验证机制，而是使用编程身份验证，或者试图获取与部署关联的 **SecurityDomain** 并直接使用。

示例：映射的高级表单：

```
/subsystem=undertow/application-security-domain=MyAppSecurity:add(http-authentication-
factory=application-http-authentication)
```

如果结果为：

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" ... default-security-domain="other">
...
    <application-security-domains>
        <application-security-domain name="MyAppSecurity" http-authentication-factory="application-
http-authentication"/>
    </application-security-domains>
...
</subsystem>
```

在这种配置形式中，引用 **http-authentication-factory**，而不是引用安全域。这是用于获取身份验证机制实例且与安全域关联的工厂。

在使用自定义 **HTTP** 身份验证机制时，或者必须针对主体转换器、凭据工厂和机制域等机制定义其他配置时，您应该引用 **http-authentication-factory** 属性。在使用 **Servlet** 规范中描述的四种机制之外，最好使用 **http-authentication-factory** 属性引用 **http-authentication-factory** 属性。

使用高级形式的映射时，可以使用另一个配置选项 **override-deployment-config**。引用的 **http-authentication-factory** 可以返回一整套身份验证机制。默认情况下，这些将被过滤为仅与应用请求的机制匹配。如果此选项设为 **true**，则工厂提供的机制将覆盖应用请求的机制。

application-security-domain 资源还有一个额外的选项 **enable-jacc**。如果设置为 **true**，则 对于与此映射匹配的任何部署，将启用 **JACC**。

Runtime Information

如果使用 **application-security-domain** 映射，可以重复检查部署是否按预期与其匹配。如果资源使用 **include-runtime=true** 读取，与映射关联的部署也将显示为：

```
/subsystem=undertow/application-security-domain=MyAppSecurity:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "enable-jacc" => false,
        "http-authentication-factory" => undefined,
        "override-deployment-config" => false,
        "referencing-deployments" => ["simple-webapp.war"],
        "security-domain" => "ApplicationDomain",
        "setting" => undefined
    }
}
```

在此输出中，**reference -deployments** 属性显示已使用该映射部署了 **simple-webapp.war** 部署。

17.2. 配置缓冲器缓存

缓冲区缓存用于缓存静态资源。**JBoss EAP** 支持根据部署配置和引用多个缓存，允许不同的部署使用不同的缓存大小。缓冲区在区域中分配，是固定大小。将缓冲区大小乘以每个区域的缓冲区数量，再乘以

区域的最大数量，便可计算出所用的空间总量。缓冲区缓存的默认大小为 **10MB**。

JBoss EAP 默认提供一个缓存：

默认 **Undertow** 子系统配置

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-host="default-host" default-servlet-container="default" default-security-domain="other">
    <buffer-cache name="default"/>
    ...
</subsystem>
```

更新现有缓冲区缓存

更新现有缓冲缓存：

```
/subsystem=undertow/buffer-cache=default/:write-attribute(name=buffer-size,value=2048)
```

```
reload
```

创建新缓冲缓存

创建新缓冲缓存：

```
/subsystem=undertow/buffer-cache=new-buffer:add
```

删除缓冲器缓存

删除缓冲缓存：

```
/subsystem=undertow/buffer-cache=new-buffer:remove
```

```
reload
```

有关可用于配置缓冲区缓存的属性的完整列表，请参阅 **Undertow Subsystem Attributes** 部分。

17.3. 配置缓冲池

Undertow 字节缓冲区池用于分配池 **NIO Buffer** 实例。所有监听器都有一个字节缓冲区池，您可以为每个监听器使用不同的缓冲区池和工作程序。字节缓冲区池可以在不同的服务器实例之间共享。

这些缓冲区用于 **IO** 操作，缓冲区的大小对应用性能有极大影响。对于大多数服务器而言，理想的大小通常是 **16k**。

更新现有缓冲池

更新现有的字节缓冲池：

```
/subsystem=undertow/byte-buffer-pool=myByteBufferPool:write-attribute(name=buffer-size,value=1024)
```

```
reload
```

创建字节缓冲区池

创建新的字节缓冲池：

```
/subsystem=undertow/byte-buffer-pool=newByteBufferPool:add
```

删除 **Byte Buffer Pool**

删除字节缓冲池：

```
/subsystem=undertow/byte-buffer-pool=newByteBufferPool:remove
```

```
reload
```

有关可用于配置字节缓冲区池的属性的完整列表，请参阅 **Byte Buffer Pool Attributes** 引用。

17.4. 配置服务器

服务器代表 **Undertow** 的实例，由多个元素组成：

- 主机
- **http-listener**
- **https-listener**

- ***ajp-listener***

主机元素提供虚拟主机配置，而三个侦听器则将该类型的连接提供给 **Undertow** 实例。

服务器的默认行为是在服务器启动时对请求进行队列。您可以使用主机上的 **queue-requests-on-start** 属性来更改默认行为。如果此属性设为 **true** （默认值），则服务器启动时到达的请求将被保留，直到服务器就绪为止。如果此属性设为 **false**，则在服务器完全启动前到达的请求将通过默认响应代码被拒绝。

无论属性值是什么，在服务器完全启动前请求处理不会启动。

您可以使用管理控制台配置 **queue-requests-on-start** 属性，方法是导航到 **Configuration → Subsystems → Web(Undertow) → Server**，选择服务器并点击 **View**，然后选择 **Hosts** 选项卡。对于受管域，您必须指定要配置的配置文件。

注意

可以配置多个服务器，从而完全隔离部署和服务器。这在某些情况下很有用，比如多租户环境。

JBoss EAP 默认提供一个服务器：

默认 **Undertow** 子系统配置

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-host="default-host" default-servlet-container="default" default-security-domain="other">
    <buffer-cache name="default"/>
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content"/>
            <http-invoker security-realm="ApplicationRealm"/>
        </host>
    </server>
    ...
</subsystem>
```

以下示例演示了如何使用管理 **CLI** 配置服务器。您还可以通过导航到 **Configuration → Subsystems → Web(Undertow) → Server**，使用管理控制台配置服务器。

更新现有服务器

更新现有服务器：

```
/subsystem=undertow/server=default-server:write-attribute(name=default-host,value=default-host)
```

```
reload
```

创建新服务器

创建新服务器：

```
/subsystem=undertow/server=new-server:add
```

```
reload
```

删除服务器

删除服务器：

```
/subsystem=undertow/server=new-server:remove
```

```
reload
```

有关可用于配置服务器的属性的完整列表，请参阅 ***Undertow Subsystem Attributes*** *部分。*

17.4.1. 访问日志记录

您可以在您定义的每个主机上配置访问日志。

有两个可用的访问日志记录选项：标准访问日志和控制台访问日志。

请注意，访问日志记录所需的额外处理可能会影响系统性能。

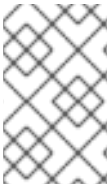
17.4.1.1. 标准访问日志

标准访问日志将日志条目写入到日志文件。

默认情况下，日志文件存储在 **standalone/log/access_log.log** 目录中。

要启用标准访问日志，请将 **access-log** 设置添加到您要捕获访问日志数据的主机。以下 **CLI** 命令演示了默认 **JBoss EAP** 服务器中默认主机上的配置：

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:add
```



注意

您必须在启用标准访问日志记录后重新加载服务器。

默认情况下，访问日志记录包括以下数据：

- 远程主机名
- 远程逻辑用户名**(always -)**
- 经过身份验证的远程用户
- 请求的日期和时间，格式为通用日志格式
- 请求的第一行
- 响应的 **HTTP** 状态代码
- 发送的字节数，**HTTP** 标头除外

这组数据被定义为常见模式。也提供另一种组合模式。除了常见模式中记录的数据外，组合模式还包括来自传入标头的引用者和用户代理。

*您可以使用 **pattern** 属性更改记录的数据。以下 **CLI** 命令演示了更新 **pattern** 属性以使用组合模式：*

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:write-attribute(name=pattern,value="combined"
```

注意

*您必须在更新 **pattern** 属性后重新加载服务器。*

*表 **17.1.** 可用模式*

pattern	描述
%a	远程 IP 地址
%A	本地 IP 地址
%b	发送字节，除 HTTP 标头或 **-** 如果没有发送字节
%B	发送的字节数，HTTP 标头除外
%h	远程主机名
%H	请求协议
%l	远程逻辑用户名 from **identd** （始终返回 **-** ；为 Apache 访问日志兼容性而包含）
%m	请求方法
%p	本地端口
%q	查询字符串（除 **?** 字符外）
%r	请求的第一行
%s	响应的 HTTP 状态代码
%t	日期和时间，采用通用日志格式格式
%u	经过身份验证的远程用户
%U	请求的 URL 路径

pattern	描述
%v	本地服务器名称
%D	处理请求所需的时间，以毫秒为单位
%T	处理请求的时间（以秒为单位）
%I	当前 Request 线程名称（稍后与堆栈追踪进行比较）
common	**%h %l %u %t "%r" %s %b**
combined	**%h %l %u %t "%r" %s %b "%{i,Referer}" "%{i,User-Agent}"**

*您还可以从 **Cookie**、传入的标头和响应标头或会话中写入信息。该语法采用 **Apache** 语法建模：*

- *用于传入标头的 **%{I,xx}***
- *用于传出响应标头的 **%{O,xxx}***
- *特定 **Cookie %{C,xxx}***
- ***%{R,xxx} where xxx 是 ServletRequest**中的属性*
- ***%{s,xxx} where xxx 是 HttpSession**中的属性*

此日志提供了其他配置选项。如需更多信息，请参阅附录中的"访问日志属性"。

17.4.1.2. 控制台访问日志

*控制台访问日志将数据写入 **stdout**，结构为 **JSON** 数据。*

每个访问记录都是一行数据。您可以捕获此数据以供日志聚合系统处理。

*要配置控制台访问日志，请向要捕获访问日志数据的主机添加 **console-access-log** 设置。以下 **CLI** 命令演示了默认 **JBoss EAP** 服务器中默认主机上的配置：*

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:add
```

默认情况下，控制台访问日志记录包括以下数据：

*表 **17.2.** 默认控制台访问日志数据*

日志数据字段名称	描述
eventSource	请求中事件的源
hostName	处理请求的 JBoss EAP 主机
bytesSent	JBoss EAP 服务器响应请求而发送的字节数
dateTime	JBoss EAP 服务器处理请求的日期和时间
remoteHost	请求源自的机器的 IP 地址
remoteUser	与远程请求关联的用户名
requestLine	已提交请求
responseCode	JBoss EAP 服务器返回的 HTTP 响应代码

*日志输出中始终包含默认属性。您可以使用 **properties** 属性 来更改默认日志数据的标签，在某些情况下，也可更改数据配置。您还可以使用 **properties** 属性 向输出中添加额外的日志数据。*

*表 **17.3.** 可用的控制台访问日志数据*

日志数据字段名称	描述	格式
authentication-type	用于验证与请求关联的用户的身份验证类型。Default label: authenticationType 使用 key 选项更改此属性的标签。	authentication-type{} authentication-type= {key="authType"}
bytes-sent	请求返回的字节数，不包括 HTTP 标头。默认标签：bytesSent 使用 key 选项更改此属性的标签。	bytes-sent={} bytes-sent= {key="sent-bytes"}

日志数据字段名称	描述	格式
date-time	接收和处理请求的日期和时间。默认标签：dateTime 使用 key 选项更改此属性的标签。使用 date-format 定义用于格式化日期记录的模式。模式必须是 Java SimpleDateFormatter 模式。使用 time-zone 选项指定要格式化 date-format 选项时用于格式化日期和/或时间数据的时区。此值必须是有效的 java.util.TimeZone。	date-time={key="<keyname>", date-format="<date-time format>"} date-time={key="@timestamp", date-format="yyyy-MM-dd'T'H:mm:ssSSS"}
host-and-port	请求查询的主机和端口。默认标签：hostAndPort 使用 key 选项更改此属性的标签。	host-and-port{} host-and-port={key="port-host"}
local-ip	本地连接的 IP 地址。使用 key 选项更改此属性的标签。默认标签：localIp 使用 key 选项更改此属性的标签。	local-ip{} local-ip{key=”localIP”}
local-port	本地连接的端口。默认标签：localPort 使用 key 选项更改此属性的标签。	local-port{} local-port{key=”LocalPort”}
local-server-name	处理请求的本地服务器的名称。默认标签：localServerName 使用 key 选项更改此属性的标签。	local-server-name {} local-server-name {key=LocalServerName}
path-parameter	请求中包含的一个或多个路径或 URI 参数。name 属性是一个用逗号分开的名称列表，用于解析交换值。使用 key-prefix 属性使键唯一。如果指定了 key-prefix，则前缀前面是输出中每个路径参数的名称的前面。	path-parameter{names={store,section}} path-parameter{names={store,section}, key-prefix="my-"}
predicate	predicate 上下文的名称。name 属性是一个用逗号分开的名称列表，用于解析交换值。使用 key-prefix 属性使键唯一。如果指定了 key-prefix，则前缀前面是输出中每个路径参数的名称的前面。	predicate{names={store,section}} predicate{names={store,section}, key-prefix="my-"}

日志数据字段名称	描述	格式
query-parameter	请求中包含的一个或多个查询参数。name 属性是一个用逗号分开的名称列表，用于解析交换值。使用 key-prefix 属性使键唯一。如果指定了 key-prefix，则前缀前面是输出中每个路径参数的名称的前面。	query-parameter{names={store,section}} query-parameter{names={store,section}, key-prefix="my-"}
query-string	请求的查询字符串。默认标签：queryString 使用键选项更改此属性的标签。使用 include-question-mark 属性指定查询字符串是否应包含问标记。默认情况下不包含问号。	query-string{} query-string{key="QueryString", include-question-mark="true"}
relative-path	请求的相对路径。默认标签：relativePath 使用 key 选项更改此属性的标签。	relative-path{} relative-path{key="RelativePath"}
remote-host	远程主机名。默认标签：remoteHost 使用 key 选项更改此属性的标签。	remote-host{} remote-host{key="RemoteHost"}
remote-ip	远程 IP 地址。默认标签：remoteIp 使用键选项更改此属性的标签。使用模糊的属性，模糊处理输出日志记录中的 IP 地址。默认值为 false。	remote-ip{} remote-ip{key="RemoteIP", obfuscated="true"}
remote-user	经过身份验证的远程用户.默认标签：remoteUser 使用密钥选项更改此属性的标签。	remote-user{} remote-user{key="RemoteUser"}
request-header	请求标头的名称。结构化数据的键是标头的名称；值是已命名标头的值。name 属性是一个用逗号分开的名称列表，用于解析交换值。使用 key-prefix 属性使键唯一。如果指定了 key-prefix，则该前缀前面是日志输出中请求标头名称的前面。	request-header{names={store,section}} request-header{names={store,section}, key-prefix="my-"}
request-line	请求行。默认标签： requestLine 使用 key 选项更改此属性的标签。	request-line{} request-line{key="Request-Line"}
request-method	请求方法。默认 label: requestMethod 使用 key 选项更改此属性的标签。	request-method{} request-method{key="RequestMethod"}

日志数据字段名称	描述	格式
request-path	请求的相对路径。默认 label: requestPath 使用 key 选项更改此属性的标签。	request-path{} request-path{key="RequestPath"}
request-protocol	请求的协议。默认标签：requestProtocol 使用 key 选项更改此属性的标签。	request-protocol{} request-protocol{key="RequestProtocol"}
request-scheme	请求的 URI 架构。默认 label: requestScheme 使用 key 选项更改此属性的标签。	request-scheme{} request-scheme{key="RequestScheme"}
request-url	原始请求 URI。如果客户端指定，包括主机名、协议等。默认 label: requestUrl 使用 key 选项更改此属性的标签。	request-url{} request-url{key="RequestURL"}
resolved-path	解析路径.Default Label: resolvedPath 使用 key 选项更改此属性的标签。	resolved-path{} resolved-path{key="ResolvedPath"}
response-code	响应代码。默认 label: responseCode 使用 key 选项更改此属性的标签。	response-code{} response-code{key="ResponseCode"}
response-header	响应标头的名称。结构化数据的键是标头的名称；值是已命名标头的值。name 属性是一个用逗号分开的名称列表，用于解析交换值。使用 key-prefix 属性使键唯一。如果指定了 key-prefix，则该前缀前面是日志输出中请求标头名称的前面。	response-header{names={store,section}} response-header{names={store,section}, key-prefix="my-"}
response-reason-phrase	响应代码的文本原因。默认标签：backendReasonPhrase 使用 key 选项更改此属性的标签。	response-reason-phrase{} response-reason-phrase{key="ResponseReasonPhrase"}
response-time	用于处理请求的时间。Default label: responseTime 使用 key 选项更改此属性的标签。默认的时间单元是 MILLISECONDS。可用时间单元包括：* NANOSECONDS * MICROSECONDS * MILLISECONDS * SECONDS	response-time{} response-time{key="ResponseTime", time-unit=SECONDS}

日志数据字段名称	描述	格式
secure-exchange	指明交换是否安全。默认标签：secureExchange 使用 key 选项更改此属性的标签。	secure-exchange{} secure-exchange{key="SecureExchange"}
ssl-cipher	请求的 SSL 密码。默认标签：sslCipher 使用 key 选项更改此属性的标签。	ssl-cipher{} ssl-cipher{key="SSLCipher"}
ssl-client-cert	请求的 SSL 客户端证书。默认标签：sslClientCert 使用 key 选项更改此属性的标签。	ssl-client-cert{} ssl-client-cert{key="SSLClientCert"}
ssl-session-id	请求的 SSL 会话 ID。默认标签：sslSessionId 使用 key 选项更改此属性的标签。	SSL-session-id{} stored-response
存储的对请求的响应。默认 label：storageResponse 使用 key 选项更改此属性的标签。	stored-response{} stored-response{key="StoredResponse"}	thread-name
当前线程的线程名称。默认 label: threadName 使用 key 选项更改此属性的标签。	thread-name{} thread-name{key="ThreadName"}	transport-protocol

*您可以使用 **metadata** 属性配置其他任意数据，以包含在访问日志记录中。**metadata** 属性的值是 一组 **key:value** 对，用于定义要在访问日志记录中包含的数据。对中的值可以是管理模型表达式。服务器启动或重新加载后，即可解决管理模型表达式。键值对用逗号分开。*

*以下 **CLI** 命令演示了复杂的控制台日志配置示例，包括附加日志数据、自定义日志数据和其他元数据：*

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:add(metadata={"@version"="1", "qualifiedHostName"=${jboss.qualified.host.name:unknown}}, attributes={bytes-sent={}, date-time={key="@timestamp", date-format="yyyy-MM-dd'T'HH:mm:ssSSS"}, remote-host={}, request-line={}, response-header={key-prefix="responseHeader", names=["Content-Type"]}, response-code={}, remote-user={}})
```

*生成的访问日志记录类似于以下额外 **JSON** 数据（注：以下示例输出被格式化为可读性；实际记录中，所有数据将以一行形式输出）：*

```
{
```

```
    "eventSource":"web-access",
    "hostName":"default-host",
    "@version":"1",
    "qualifiedHostName":"localhost.localdomain",
    "bytesSent":1504,
    "@timestamp":"2019-05-02T11:57:37123",
    "remoteHost":"127.0.0.1",
    "remoteUser":null,
    "requestLine":"GET / HTTP/2.0",
    "responseCode":200,
    "responseHeaderContent-Type":"text/html"
}
```

以下命令演示了在激活控制台访问日志后对日志数据的更新：

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:write-attribute(name=attributes,value={bytes-sent={}, date-time={key="@timestamp", date-format="yyyy-MM-dd'T'HH:mm:ssSSS"}, remote-host={}, request-line={}, response-header={key-prefix="responseHeader", names=["Content-Type"]}, response-code={}, remote-user={}})
```

以下命令演示了在激活控制台访问日志后自定义元数据的更新：

```
/subsystem=undertow/server=default-server/host=default-host/setting=console-access-log:write-attribute(name=metadata,value={"@version"="1",
"qualifiedHostName"=${jboss.qualified.host.name:unknown}}
```

17.5. 配置 SERVLET 容器

servlet 容器提供所有 **servlet**、**JSP** 和 **websocket** 相关配置，包括会话相关的配置。虽然大多数服务器只需要一个 **servlet** 容器，但可以通过添加额外的 **servlet -container** 元素来配置多个 **servlet** 容器。拥有多个 **servlet** 容器可以实现以下行为：允许将多个部署部署到不同虚拟主机上的同一上下文路径。

注意

servlet 容器 中提供的大部分配置可以被使用 **web.xml** 文件部署的应用单独覆盖。

JBoss EAP 默认提供一个 **servlet** 容器：

默认 **Undertow** 子系统配置

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0">
  <buffer-cache name="default"/>
  <server name="default-server">
```

```
    ...
  </server>
  <servlet-container name="default">
    <jsp-config/>
    <websockets/>
  </servlet-container>
 ...
 </subsystem>
```

以下示例演示了如何使用管理 **CLI** 配置 **servlet** 容器。您还可以通过导航到 **Configuration → Subsystems → Web(Undertow) → Servlet Container**，使用管理控制台配置 **servlet** 容器。

更新现有 **Servlet** 容器

更新现有的 **servlet** 容器：

```
/subsystem=undertow/servlet-container=default:write-attribute(name=ignore-flush,value=true)
```

```
reload
```

创建新 **Servlet** 容器

创建新的 **servlet** 容器：

```
/subsystem=undertow/servlet-container=new-servlet-container:add
```

```
reload
```

删除 **Servlet** 容器

删除 **servlet** 容器：

```
/subsystem=undertow/servlet-container=new-servlet-container:remove
```

```
reload
```

有关可用于配置 **servlet** 容器的属性的完整列表，请参阅 **Undertow Subsystem Attributes** 部分。

17.6. 配置 SERVLET 扩展

借助 **Servlet** 扩展，您可以 **hook** 到 **servlet** 部署过程并修改 **servlet** 部署的各个方面。如果您需要为部署添加额外的身份验证机制或使用原生 **Undertow** 处理程序作为 **servlet** 部署的一部分，这时非常有用。

要创建自定义 **servlet** 扩展，需要实施 **the io.undertow.servlet.ServletExtension** 接口，然后将您的实施类的名称添加到部署中的 **META-INF/services/io.undertow.servlet.ServletExtension** 文件中。您还需要包含 **ServletExtension** 实施编译的类文件。当 **Undertow** 部署 **servlet** 时，它将加载来自 部署 类加载器的所有服务，然后调用它们的 **handleDeployment** 方法。

将包含完整且可更改的部署描述的 **Undertow DeploymentInfo** 结构传递到此方法。您可以修改此结构来更改部署的任何方面。

DeploymentInfo 结构的结构与嵌入式 **API** 使用的相同，因此实际上 **ServletExtension** 具有 与您在嵌入式模式中使用 **Undertow** 时所具有的相同灵活性。

17.7. 配置处理程序

JBoss EAP 允许配置两种类型的处理程序：

- 文件处理程序
- **reverse-proxy** 处理程序

文件处理程序提供静态文件。每一文件处理程序必须关联到虚拟主机中的一个位置。**reverse-proxy** 处理程序允许 **JBoss EAP** 充当高性能反向代理。

JBoss EAP 默认提供一个文件处理程序：

默认 **Undertow** 子系统配置

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-host="default-host" default-servlet-container="default" default-security-domain="other">
    <buffer-cache name="default"/>
    <server name="default-server">
        ...
    </server>
    <servlet-container name="default">
        ...
    </servlet-container>
    <handlers>
        <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
    </handlers>
</subsystem>
```

*将 **WebDAV** 用于静态资源*

*早期版本的 **JBoss EAP** 允许通过 **Web davServlet** 将 **Web DAV** 与 **Web** 子系统搭配使用，以托管静态资源，并且启用额外的 **HTTP** 方法来访问和操作这些文件。在 **JBoss EAP 7** 中，**undertow** 子系统提供了使用文件处理程序提供静态文件的机制，但 **undertow** 子系统不支持 **WebDAV**。如果要将 **WebDAV** 与 **JBoss EAP 7** 搭配使用，您可以编写自定义 **WebDAVservlet**。*

更新现有文件处理程序

更新现有文件处理程序：

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-attribute(name=case-sensitive,value=true)
```

```
reload
```

创建新文件处理程序

要创建新文件处理程序，请执行以下操作：

```
/subsystem=undertow/configuration=handler/file=new-file-handler:add(path="${jboss.home.dir}/welcome-content")
```

警告

*如果您将文件处理程序 的路径 直接设置为文件而不是目录，则引用该文件处理程序的任何 位置 元素都必须以正斜杠(/)结尾。否则，服务器将返回 **404 - 未找到** 响应。*

删除文件处理程序

删除文件处理程序

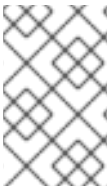
```
/subsystem=undertow/configuration=handler/file=new-file-handler:remove
```

```
reload
```

*有关可用于配置处理程序的属性的完整列表，请参阅 **Undertow Subsystem Attributes** 部分。*

17.8. 配置过滤器

过滤器允许修改请求的某些方面，并且可以使用 **predicates** 来控制过滤器的执行时间。过滤器的一些常见用例包括设置标头或执行 **GZIP** 压缩。



注意

过滤器在功能上等同于 **JBoss EAP 6** 中使用的全局值。

可以定义以下类型的过滤器：

- **custom-filter**
- **error-page**
- **expression-filter**
- **gzip**
- **mod-cluster**
- **request-limit**
- **response-header**
- 重写

以下示例演示了如何使用管理 **CLI** 配置过滤器。您还可以通过导航到 **Configuration → Subsystems → Web(Undertow) → Filters** 来配置使用管理控制台的过滤器。

更新现有过滤器

更新现有过滤器：

```
/subsystem=undertow/configuration=filter/response-header=myHeader:write-attribute(name=header-value,value="JBoss-EAP")
```

```
reload
```

创建新过滤器

要创建新过滤器，请执行以下操作：

```
/subsystem=undertow/configuration=filter/response-header=new-response-header:add(header-name=new-response-header,header-value="My Value")
```

删除过滤器

删除过滤器：

```
/subsystem=undertow/configuration=filter/response-header=new-response-header:remove
```

```
reload
```

有关可用于配置过滤器的属性的完整列表，请参阅 **Undertow Subsystem Attributes** 部分。

17.8.1. 配置 buffer-request 处理程序

客户端或浏览器的请求由两个部分组成：标题和正文。在典型的情形中，标头和正文发送至 **JBoss EAP**，两者之间没有任何延迟。但是，如果首先发送标题，在几秒钟后发送正文，则会延迟发送完整的请求。此方案在 **JBoss EAP** 中创建线程，以显示为 等待 执行完整的请求。

可以使用 **buffer-request** 处理程序纠正发送标头和请求正文时造成的延迟。**buffer-request** 处理程序会在将其分配到 **worker** 线程前尝试使用来自非阻塞 **IO** 线程的请求。如果没有添加 **buffer-request** 处理程序，则直接为 **worker** 线程分配线程。但是，当添加 **buffer-request** 处理程序时，处理程序会尝试读取它在分配至 **worker** 线程前可以使用 **IO** 线程以非阻塞方式缓冲的数据量。

您可以使用以下管理 **CLI** 命令配置 **buffer-request** 处理程序：

```
/subsystem=undertow/configuration=filter/expression-filter=buf:add(expression="buffer-request(buffers=1)")

/subsystem=undertow/server=default-server/host=default-host/filter-ref=buf:add
```

可以处理的缓冲区请求的大小有限制。此限制由缓冲区大小和缓冲区总数的组合决定，如下方的公式所示。

Total_size = num_buffers * buffer_size

在以上因素中：

- **Total_size** 是在请求分配给 **worker** 线程前缓冲的数据大小。
- **num_buffers** 是缓冲区的数量。缓冲区数量由处理程序上的 **buffers** 参数设置。
- **buffer_size** 是每个缓冲区的大小。缓冲区大小在 **io** 子系统中设置，默认情况下为 **16KB**。

警告

避免配置非常大的缓冲区请求，否则您可能耗尽内存。

17.9. 配置默认欢迎 WEB 应用程序

JBoss EAP 包含一个默认的 **Welcome** 应用，默认显示在端口 **8080** 的根上下文中。

Undertow 中预配置了默认服务器，可提供欢迎内容。

默认 **Undertow** 子系统配置

```
<subsystem xmlns="urn:jboss:domain:undertow:10.0" default-server="default-server" default-virtual-host="default-host" default-servlet-container="default" default-security-domain="other">
    ...
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>
        <host name="default-host" alias="localhost">
```

```
            <location name="/" handler="welcome-content"/>
            <http-invoker security-realm="ApplicationRealm"/>
        </host>
    </server>
    ...
    <handlers>
        <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
    </handlers>
</subsystem>
```

默认服务器 **default-server** 配置了默认主机 **default-host**。默认主机被配置为使用 **<location>** 元素和 **welcome-content** 文件处理程序来处理对服务器的 **root** 的请求。**welcome-content** 处理程序提供 **path** 属性中指定的位置的内容。

此默认 **Welcome** 应用可替换为您自己的 **Web** 应用。这可以通过以下两种方式之一进行配置：

- 更改 **welcome-content** 文件处理程序
- 更改 **default-web-module**

您还可以禁用欢迎内容。

更改 welcome-content 文件处理程序

1. 修改现有的 **welcome-content** 文件处理程序路径，以指向新部署。

```
/subsystem=undertow/configuration=handler/file=welcome-content:write-
attribute(name=path,value="/path/to/content")
```

注意

或者，您也可以创建供服务器的 **root** 使用的其他文件处理程序。

```
/subsystem=undertow/configuration=handler/file=NEW_FILE_HANDLER:add(
path="/path/to/content")
/subsystem=undertow/server=default-server/host=default-
host/location=\/:write-attribute(name=handler,value=NEW_FILE_HANDLER)
```

2. 重新加载服务器以使更改生效。

```
reload
```

更改 **default-web-module**

1. 将部署的 **Web** 应用映射到服务器的根目录**.**

```
/subsystem=undertow/server=default-server/host=default-host:write-attribute(name=default-web-module,value=hello.war)
```

2. 重新加载服务器以使更改生效。

```
reload
```

禁用默认欢迎 **Web** 应用程序

1. 删除 **default-host** 的位置 条目 / 来禁用 **welcome** 应用。

```
/subsystem=undertow/server=default-server/host=default-host/location=\/:remove
```

2. 重新加载服务器以使更改生效。

```
reload
```

17.10. 配置 HTTPS

有关为 **Web** 应用程序配置 **HTTPS** 的详情，请参考如何配置服务器安全性为应用程序配置单向和双向 **SSL/TLS**。

有关配置用于 **JBoss EAP** 管理接口的 **HTTPS** 的详情，请参考 如何保护管理接口（如何 配置服务器安全性）。

17.11. 配置 HTTP 会话超时

HTTP 会话超时定义声明 **HTTP** 会话无效所需的时间。例如，用户访问部署到 **JBoss EAP** 的应用，该应用将创建 **HTTP** 会话。如果该用户随后尝试在 **HTTP** 会话超时后再次访问该应用，则原始 **HTTP** 会话

将无效，并且将强制用户创建新的 **HTTP** 会话。这可能导致丢失未经保护的数据，或者用户需要重新进行身份验证。

HTTP 会话超时在应用的 **web.xml** 文件中配置，但可以在 **JBoss EAP** 中指定默认的 **HTTP** 会话超时。服务器的超时值将应用于所有已部署的应用，但应用的 **web.xml** 将覆盖服务器的值。

server 值在 **default-session-timeout** 属性中指定，该属性可在 **undertow** 子系统的 **servlet-container** 部分中找到。**default-session-timeout** 的值以分钟为单位指定，默认值为 **30**。

配置默认会话超时

配置 **default-session-timeout** ：

```
/subsystem=undertow/servlet-container=default:write-attribute(name=default-session-timeout,
value=60)
```

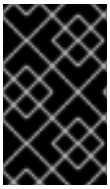
```
reload
```

17.12. 配置仅 HTTP 会话管理 COOKIES

会话管理 **Cookie** 可通过 **HTTP API** 和非 **HTTP API**（如 **JavaScript**）访问。**JBoss EAP** 提供了将 **HttpOnly** 标头作为 **Set-Cookie** 响应标题的一部分发送至客户端的功能，通常是浏览器。在支持的浏览器中，启用此标头会告知浏览器，它应该阻止通过非 **HTTP API** 访问会话管理 **Cookie**。将会话管理 **Cookie** 限制为仅 **HTTP API** 有助于缓解会话 **Cookie** 失窃的风险。若要启用此行为，**http-only** 属性应设为 **true**。

重要

使用 **HttpOnly** 标头本身不会防止跨站点脚本攻击，而是仅通知浏览器。浏览器还必须支持 **HttpOnly** 才能影响此行为。



重要

使用 **http-only** 属性仅将限制应用到会话管理 **Cookie**，而非其他浏览器 **Cookie**。

http-only 属性在 **undertow** 子系统的两个位置中设置：

- 在 **servlet** 容器中作为会话 **Cookie** 设置
- 在服务器的主机部分中作为单点登录属性

为 **Servlet** 容器会话 **Cookie** 配置 仅限主机

为 **servlet** 容器会话 **Cookie** 配置 只主机 属性：

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
```

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:write-attribute(name=http-only,value=true)
```

```
reload
```

为 主机单点 登录配置只主机

为 主机单点登录 配置只主机属性：

```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:add
```

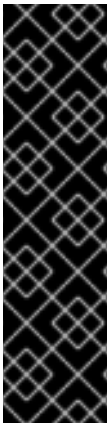
```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:write-attribute(name=http-only,value=true)
```

```
reload
```

17.13. 配置 HTTP/2

Undertow 允许使用 **HTTP/2** 标准，这可以通过压缩标头和在同一 **TCP** 连接上多路复用来缩短延迟。它还使服务器能够在请求资源之前将资源推送至客户端，从而加快页面负载。

请注意，**HTTP/2** 仅适用于同时支持 **HTTP/2** 标准的客户端和浏览器。



重要

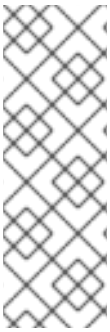
大多数现代浏览器通过安全 **TLS** 连接（称为 **h2** ）强制 **HTTP/2**，可能不支持通过普通 **HTTP**（称为 **h2c** ）的 **HTTP/2**。换而言之，仍可以将 **JBoss EAP** 配置为使用 **HTTP/2** 和 **h2c**，无需使用 **HTTPS**，仅使用 **HTTP** 升级时使用普通 **HTTP**。在这种情况下，您只需在 **HTTP** 侦听器 **Undertow** 中启用 **HTTP/2**：

```
/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=enable-http2,value=true)
```

要将 **Undertow** 配置为使用 **HTTP/2**，请在 **Undertow** 中启用 **HTTPS** 侦听器以使用 **HTTP/2**，方法是将 **enable-http2** 属性 设置为 **true** ：

```
/subsystem=undertow/server=default-server/https-listener=https:write-attribute(name=enable-http2,value=true)
```

有关 **HTTPS** 侦听器并将 **Undertow** 配置为将 **HTTPS** 用于 **Web** 应用程序的更多信息，请参阅如何配置服务器安全性为应用配置单向和双向 **SSL/TLS**。



注意

为了使用带有 **elytron** 子系统的 **HTTP/2**，您需要确保 **Undertow** 的 **https -listener** 中的 **configuressl-context** 配置为可修改。这可以通过将相应的 **server-ssl-context** 的 **wrap** 属性设置为 **false** 来实现。默认情况下，**wrap** 属性设为 **false**。**Undertow** 需要此功能，才能对 **ALPN** 对 **ssl-context** 进行修改。如果提供的 **ssl-context** 不可写，则无法使用 **ALPN**，连接将返回 **HTTP/1.1**。

使用 **HTTP/2** 时支持 **ALPN**

在安全 **TLS** 连接中使用 **HTTP/2** 时，需要支持 **ALPN TLS** 协议扩展的 **TLS** 堆栈。获取这个堆栈会根据安装的 **JDK** 的不同而有所不同。

- 使用 **Java 8** 时，**ALPN** 实施直接引入到 **JBoss EAP** 中，其依赖性依赖于 **Java** 内部。因此，**ALPN** 实施仅适用于 **Oracle** 和 **OpenJDK**。它不适用于 **IBM Java**。红帽强烈建议您利用 **JBoss EAP** 中 **OpenSSL** 供应商的 **ALPN TLS** 协议扩展支持，以及实施 **ALPN** 功能的 **OpenSSL** 库。使用 **OpenSSL** 供应商提供的 **ALPN TLS** 协议扩展支持应该可以提高性能。

- 自 **Java 9** 起，**JDK** 以原生方式支持 **ALPN**；但是，使用来自 **OpenSSL** 提供商的 **ALPN TLS** 协议扩展支持还应提高使用 **Java 9** 或更高版本的性能。

安装 OpenSSL 的说明（获得 ALPN TLS 协议扩展支持），请参见 JBoss 核心服务中的安装 OpenSSL。Red Hat Enterprise Linux 8 支持标准系统 OpenSSL，且不需要额外的 JBoss Core Services OpenSSL。

安装了 OpenSSL 后，请按照配置 JBoss EAP 使用 OpenSSL 中的说明操作。

验证 HTTP/2 是否正在使用

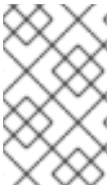
若要验证 Undertow 是否在使用 HTTP/2，您需要检查来自 Undertow 的标头。使用 https（如 https://localhost:8443 ）导航到您的 JBoss EAP 实例，再使用浏览器的开发人员工具来检查标题。使用 HTTP/2 时，一些浏览器（如 Google Chrome）将显示 HTTP/2 伪标头，如 :path、:authority、:method 和：scheme。其他浏览器（如 Firefox 和 Safari）会将标头的状态或版本报告为 HTTP/2.0。

17.14. 配置 REQUESTDUMPING HANDLER

RequestDumping 处理程序, io.undertow.server.handlers.RequestDumpingHandler 记录 JBoss EAP 中 Undertow 处理的请求和对应响应对象的详细信息。

重要

虽然此处理程序对调试很有用，但也可能会记录敏感信息。在启用此处理程序时，请牢记这一点。



注意

RequestDumping 处理程序 取代了 JBoss EAP 6 中的 RequestDumperValve。

您可以在服务器级别上 直接在 JBoss EAP 或单个应用中配置 RequestDumping 处理程序。

17.14.1. 在服务器上配置 RequestDumping Handler

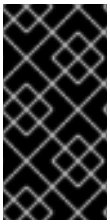
RequestDumping 处理程序应配置为表达式过滤器。要将 RequestDumping 处理程序配置为表达式过滤器，您需要执行以下操作：

使用 RequestDumping Handle r 创建一个新的 Expression Filter

```
/subsystem=undertow/configuration=filter/expression-
filter=requestDumperExpression:add(expression="dump-request")
```

在 **Undertow Web** 服务器中启用 **Expression Filter**

```
/subsystem=undertow/server=default-server/host=default-host/filter-
ref=requestDumperExpression:add
```



重要

在启用 **RequestDumping** 处理程序 作为表达式过滤器时，**Undertow Web** 服务器处理的所有请求和对应的响应都会记录下来。

为特定 **URL** 配置 **RequestDumping Handler**

除了记录所有请求外，您还可以使用表达式过滤器来仅记录特定 **URL** 的请求和对应的响应。这可以使用您的表达式中的 **predicate** 来实现，如 **path** 、**path-prefix** 或 **path-suffix**。例如，如果您想要记录所有请求及对 **/myApplication/test** 的对应响应，您可以在创建表达式过滤器时使用表达式 **"path(/myApplication/test)-> dump -request"** 而不是 表达式**"dump-request "**。这只会将路径与 **/myApplication/test** 完全匹配的请求定向到 **RequestDumping** 处理程序。

17.14.2. 在应用程序中配置 RequestDumping Handler

除了在服务器上配置 **RequestDumping** 处理程序 外，您还可以在单个应用程序中进行配置。这会将处理程序的范围限制为仅该特定应用。应在 **WEB-INF/undertow-handlers.conf** 中配置 **RequestDumping** 处理程序。

要在 **WEB-INF/undertow-handlers.conf** 中配置 **RequestDumping** 处理程序，将这个应用程序的所有请求和对应的响应记录到 **WEB-INF/undertow-handlers.conf** 中：

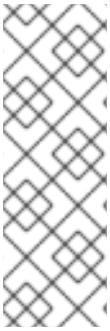
示例： **WEB-INF/undertow-handlers.conf**

```
dump-request
```

要将 **WEB-INF/undertow-handlers.conf** 中的 **RequestDumping** 处理程序 配置为仅记录此应用中特定 **URL** 的请求和对应的响应，您可以在表达式中使用 **predicate**，如 **path** 、**prefix** 或 **path-suffix**。例如，要记录应用程序中 测试 的所有请求和对应的响应，可以使用以下带有 **path predicate** 的表达式：

示例： **WEB-INF/undertow-handlers.conf**

```
path(/test) -> dump-request
```



注意

*在使用应用 **WEB-INF/undertow -handlers.conf** 中定义的表达式（如 **path**、**path-prefix** 或 **path-suffix**） 中定义的 **predicates** 时，所使用的值相对于应用程序的上下文根目录。例如，如果应用程序的上下文根目录是 **myApplication**，且表达式 路径**(/test)-> dump-request** 在 **WEB-INF/undertow-handlers.conf** 中配置，则它只会记录请求以及对 **/myApplication/test** 的对应响应。*

17.15. 配置 COOKIE 安全性

*您可以使用 **secure-cookie** 处理程序来增强通过服务器和客户端之间的连接创建的 **Cookie** 的安全性。在这种情况下，如果将 **Cookie** 设置的连接标记为安全，**cookie** 将将其 **secure** 属性设置为 **true**。*

*您可以通过配置监听程序或使用 **HTTPS** 来保护连接。您可以通过在 **undertow** 子系统中定义 **expression-filter**， 来配置 **secure-cookie** 处理程序。如需更多信息，请参阅配置过滤器。*

*使用 **secure-cookie** 处理程序时，通过安全连接设置的 **Cookie** 将隐式设置为安全，永远不会通过不安全连接发送。*

17.16. 调整 UNDERTOW 子系统

*有关优化 **undertow** 子系统性能的提示，请参阅 性能调优指南中的 **Undertow Subsystem Tuning** 部分。*

第 18 章 配置远程

18.1. 关于删除子系统

远程 子系统 允许您为本地和远程服务配置入站和出站连接，以及这些连接的设置。

JBoss 远程连接包括下列可配置的元素：端点、连接器和一系列本地和远程连接 **URI**。除非大多数人将自定义连接器用于自己的应用，否则他们根本不需要配置远程 子系统。充当远程客户端（如 **EJB**）的应用需要单独的配置来连接特定的连接器。

默认删除子系统配置

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
    <endpoint/>
    <http-connector name="http-remoting-connector" connector-ref="default" security-
realm="ApplicationRealm"/>
</subsystem>
```

如需了解可用于 远程子系统的属性 的完整列表，请参阅 删除子系统属性。

移除端点

远程端点使用由 **io** 子系统声明和配置的 **XNIO** 工作程序。

如需有关如何配置远程端点的详情，请参阅配置 **Endpoint**。

连接器

连接器是主远程配置元素。允许多个连接器。每个连接器由一个带有多个子元素的 **<connector>** 元素和少量其他属性组成。默认连接器供多个 **JBoss EAP** 子系统使用。自定义连接器元素和属性的具体设置取决于您的应用程序。请联系红帽全球支持服务以了解更多信息。

有关如何配置连接器的详情，请参阅配置连接器。

出站连接

您可以指定三种不同类型的出站连接：

- 由 **URI** 指定出站连接

- *本地出站连接，它连接到本地资源，如套接字*
- *远程出站连接，它连接到远程资源并使用安全域进行身份验证*

其他配置

*远程操作取决于在远程 子系统 之外配置的多个元素，如网络接口和 **IO** 工作程序。*

如需更多信息，请参阅附加远程配置。

18.2. 配置端点

重要

*在 **JBoss EAP 6** 中，工作程序线程池已直接在 远程子系统 中配置。在 **JBoss EAP 7** 中，远程 端点 配置引用了来自 **io** 子系统的 **worker**。*

***JBoss EAP** 默认提供以下 端点 配置：*

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
  <endpoint/>
  ...
</subsystem>
```

更新现有端点配置

```
/subsystem=remoting/configuration=endpoint:write-attribute(name=authentication-retries,value=2)
```

```
reload
```

创建新端点配置

```
/subsystem=remoting/configuration=endpoint:add
```

删除端点配置

```
/subsystem=remoting/configuration=endpoint:remove
```

```
reload
```

如需可用于端点配置的属性的完整列表，请参阅 **Endpoint Attributes**。

18.3. 配置连接器

连接器是与远程相关的主要配置元素，包含多个用于额外配置的子元素。

更新现有连接器配置

```
/subsystem=remoting/connector=new-connector:write-attribute(name=socket-binding,value=my-socket-binding)
```

```
reload
```

创建新连接器

```
/subsystem=remoting/connector=new-connector:add(socket-binding=my-socket-binding)
```

删除连接器

```
/subsystem=remoting/connector=new-connector:remove
```

```
reload
```

有关可用于配置连接器的属性的完整列表，请参阅 **Remoting Subsystem Attributes** 部分。

18.4. 配置 HTTP 连接器

HTTP 连接器为基于 **HTTP** 升级的远程连接器提供配置。**JBoss EAP** 默认提供以下 **http-connector** 配置：

```
<subsystem xmlns="urn:jboss:domain:remoting:4.0">
    ...
    <http-connector name="http-remoting-connector" connector-ref="default" security-realm="ApplicationRealm"/>
</subsystem>
```

默认情况下，此 **HTTP** 连接器连接到名为 **default** 的 **HTTP** 侦听器，该侦听器在 **undertow** 子系统中配置。如需更多信息，请参阅配置 **Web** 服务器**(Undertow)**。

更新现有的 **HTTP** 连接器配置

```
/subsystem=remoting/http-connector=new-connector:write-attribute(name=connector-ref,value=new-connector-ref)
```

```
reload
```

*创建新 **HTTP** 连接器*

```
/subsystem=remoting/http-connector=new-connector:add(connector-ref=default)
```

*删除 **HTTP** 连接器*

```
/subsystem=remoting/http-connector=new-connector:remove
```

*有关可用于配置 **HTTP** 连接器的属性的完整列表，请参阅 **Connector Attributes**。*

18.5. 配置出站连接

*出站连接是一种通用的远程远程连接，由 **URI** 完全指定的出站连接。*

更新现有出站连接

```
/subsystem=remoting/outbound-connection=new-outbound-connection:write-attribute(name=uri,value=http://example.com)
```

创建新的出站连接

```
/subsystem=remoting/outbound-connection=new-outbound-connection:add(uri=http://example.com)
```

删除出站连接

```
/subsystem=remoting/outbound-connection=new-outbound-connection:remove
```

有关可用于配置出站连接的属性的完整列表，请参阅出站连接属性。

18.6. 配置远程出站连接

*远程出站连接由协议、出站套接字绑定、用户名和安全域指定。协议可以是 远程、**http -remoting** 或 **https-remoting**。*

更新现有的远程出站连接

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:write-attribute(name=outbound-socket-binding-ref,value=outbound-socket-binding)
```

创建新的远程出站连接

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

删除远程出站连接

```
/subsystem=remoting/remote-outbound-connection=new-remote-outbound-connection:remove
```

如需配置远程出站出站连接属性的完整属性列表，请参阅远程出站连接属性。

18.7. 配置本地出站连接

本地出站连接是与本地协议的远程连接，仅由出站套接字绑定 指定。

更新现有本地出站连接

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:write-attribute(name=outbound-socket-binding-ref,value=outbound-socket-binding)
```

创建新的本地出站连接

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:add(outbound-socket-binding-ref=outbound-socket-binding)
```

删除本地出站连接

```
/subsystem=remoting/local-outbound-connection=new-local-outbound-connection:remove
```

有关可用于配置本地出站连接的属性的完整列表，请参阅本地出站连接。

18.8. 额外的远程配置

在远程 子系统之外配置多个远程 元素。

IO worker

使用以下命令设置用于远程的 **IO worker**：

```
/subsystem=remoting/configuration=endpoint:write-attribute(name=worker,
value=WORKER_NAME)
```

如需有关如何配置 **IO worker** 的详细信息，请参阅配置工作程序。

网络接口

远程 子系统 使用的网络接口是 公共 接口。此接口也供其他几个子系统使用，因此在修改时请谨慎操作。

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

在受管域中，每个主机的 **host.xml** 文件中定义 公共 接口。

套接字绑定

远程 子系统 使用的默认套接字绑定绑定到端口 **8080**。

如需有关套接字绑定和套接字绑定组的更多信息，请参阅 **Socket Bindings**。

远程连接器参考 EJB

ejb3 子系统包含对远程方法调用的远程连接器的引用。以下是默认配置：

```
<remote connector-ref="remoting-connector" thread-pool-name="default"/>
```

安全传输配置

如果客户端请求，远程传输使用 **STARTTLS** 使用安全连接，如 **HTTPS**、安全 **Servlet**。同一套接字绑定或网络端口用于受保护且不受保护的连接，因此不需要额外的服务器端配置。根据需求，客户端请求安全或不安全的传输。使用远程（如 **EJB**、**ORB** 和 **JMS** 提供程序）的 **JBoss EAP** 组件默认请求受保护的接口。

警告

STARTTLS 的工作原理是在客户端请求安全连接时激活安全连接，否则默认为不安全的连接。它本身会受到中间人攻击的影响，攻击者拦截客户端的请求并修改它以请求不安全的连接。如果客户端没有收到安全连接，则必须正确编写客户端失败，除非不安全的连接是适当的回退。

第 19 章 配置 IO 子系统

19.1. IO 子系统概述

The io 子系统定义供 Undertow 和远程等其他子系统使用的 XNIO 工作程序 和 缓冲区池。这些 worker 和缓冲池在 io 子系统的以下组件中定义：

默认 IO 子系统配置

```
<subsystem xmlns="urn:jboss:domain:io:3.0">
    <worker name="default"/>
    <buffer-pool name="default"/>
</subsystem>
```

19.2. 配置工作程序

Worker 是 XNIO 工作程序实例。XNIO 工作程序实例是 Java NIO API 的抽象层，提供 IO 和 worker 线程管理等功能，以及 SSL 支持。默认情况下，JBoss EAP 提供一个名为 default 的单个工作程序，但可以定义更多工作程序。

更新现有工作程序

更新现有的 worker：

```
/subsystem=io/worker=default:write-attribute(name=io-threads,value=10)
```

```
reload
```

创建新工作程序

创建新 worker：

```
/subsystem=io/worker=newWorker:add
```

删除工作程序

删除 worker：

```
/subsystem=io/worker=newWorker:remove
```

```
reload
```

有关可用于配置 **worker** 的属性的完整列表，请参阅 **IO** 子系统属性部分。

19.3. 配置缓冲池

注意

IO 缓冲池已弃用，但它们仍设置为当前版本中的默认设置。缓冲区池是池式 **NIO** 缓冲区实例。更改缓冲区大小对应用程序性能有显著影响。对于大多数服务器而言，理想的缓冲区大小通常是 **16k**。有关配置 **Undertow** 字节缓冲区池的更多信息，请参阅 **JBoss EAP** 配置指南中的配置缓冲区池 章节。

更新现有缓冲池

更新现有缓冲池：

```
/subsystem=io/buffer-pool=default:write-attribute(name=direct-buffers,value=true)
reload
```

创建缓冲池

创建新缓冲池：

```
/subsystem=io/buffer-pool=newBuffer:add
```

删除缓冲池

删除缓冲池：

```
/subsystem=io/buffer-pool=newBuffer:remove
reload
```

有关可用于配置缓冲区池的属性的完整列表，请参阅 **IO** 子系统属性部分。

19.4. 调整 IO 子系统

有关监控和优化 **io** 子系统性能的提示，请参阅 性能调优指南 中的 **IO** 子系统 调优部分。

第 20 章 配置 WEB 服务

JBoss EAP 提供使用管理控制台或管理 CLI 通过 webservices 子系统配置已部署的 Web 服务行为的功能。您可以配置发布的端点地址和处理程序链。您还可以为 Web 服务启用运行时统计信息集合。

如需更多信息，请参阅为 JBoss EAP 开发 Web 服务应用程序配置 Web 服务子系统 。

第 21 章 JAKARTA SERVER FACES 配置

21.1. MULTIIPLE JAKARTA SERVER FACES 实施 JAKARTA SERVER FACES

通过使用 **jsf** 子系统，您可以在相同的 **JBoss EAP** 服务器实例上安装多个 **Jakarta** 服务器 **Faces** 实施。您可以安装一个采用 **Jakarta** 服务器 **Faces** 规范 **2.3** 或更高版本的 **Sun Mojarra** 或 **Apache MyFaces** 版本。

21.1.1. 安装 Jakarta 服务器 Faces 实施

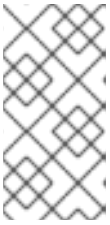
以下流程描述了如何手动安装新的 **Jakarta Server Faces** 实施，并使其成为默认实施。

1. 添加 **Jakarta Server Faces** 实施 **JAR** 文件。
2. 添加 **Jakarta Server Faces API JAR** 文件。
3. 添加 **Jakarta** 服务器 **Faces** 注入 **JAR** 文件。
4. 如果要安装 **MyFaces**，请添加 **commons-digester JAR** 文件。
5. 设置默认的 **Jakarta Server Faces** 实施。

添加 **Jakarta** 服务器 **Faces** 实施 **JAR** 文件

1. 在 **EAP_HOME/modules/** 目录中为 **Jakarta Server Faces** 实施创建适当的目录结构：

```
$ cd EAP_HOME/modules/
$ mkdir -p com/sun/jsf-impl/IMPL_NAME-VERSION
```



注意

例如，将 **IMPL_NAME-VERSION** 替换为支持 **Jakarta Server Faces** 规格 **2.3** 或更高版本的 **Mojarra** 版本。

2. 将 **Jakarta** 服务器 **Faces** 实施 **JAR** 文件复制到 **IMPL_NAME-VERSION/** 子目录中。

3. 在 **IMPL_NAME-VERSION/** 子目录中，创建一个与这个 **Mojarra** 模板或 **MyFaces** 模板 类似的 **module.xml** 文件。如果使用模板，请务必对记下的可替换变量使用适当的值。

添加 **Jakarta Server Faces API JAR** 文件

1. 在 **EAP_HOME/modules/** 目录中为 **Jakarta Server Faces** 实施创建适当的目录结构：

```
$ cd EAP_HOME/modules/
$ mkdir -p javax/faces/api/IMPL_NAME-VERSION
```

2. 将 **Jakarta Server Faces API JAR** 文件复制到 **IMPL_NAME-VERSION/** 子目录。

3. 在 **IMPL_NAME-VERSION/** 子目录中，创建一个与这个 **Mojarra** 模板或 **MyFaces** 模板 类似的 **module.xml** 文件。如果使用模板，请务必对记下的可替换变量使用适当的值。

添加 **Jakarta Server Faces Injection JAR** 文件

1. 在 **EAP_HOME/modules/** 目录中为 **Jakarta Server Faces** 实施创建适当的目录结构：

```
$ cd EAP_HOME/modules/
$ mkdir -p org/jboss/as/jsf-injection/IMPL_NAME-VERSION
```

2. 按照补丁和升级指南中所述，下载 **JBoss EAP** 实例的最新累积修补程序。接下来，完成以下步骤之一：

   - 如果您尚未将补丁更新应用到服务器，然后将 **wildfly-jsf-injection** 和 **weld- core-jsf JAR** 文件从 **EAP_HOME/modules/system/layers/base/org/jboss/as/jsf-injection/main/** 复制到 **IMPL_NAME-VERSION/** 子目录。

   - 如果您已将补丁更新应用到服务器，然后将 **wildfly-jsf-injection** 和 **weld -core-jsf JAR** 文件从最新的补丁更新目录复制到 **IMPL_NAME-VERSION/** 子目录。例如，**EAP_HOME/modules/system/layers/base/.overlays/layer-base-jboss-eap-7.3.z.CP/org/jboss/as/jsf-injection**，其中 **z** 是最新版本的版本号。

3. 在 **IMPL_NAME-VERSION/** 子目录中，创建一个与这个 **Mojarra** 模板或 **MyFaces** 模板 类似的 **module.xml** 文件。如果使用模板，请务必对记下的可替换变量使用适当的值。

为 **MyFaces** 添加 **commons-digester JAR** 文件

1. 在 **EAP_HOME/modules/** 目录中为 **commons-digester JAR** 创建适当的目录结构：

```
$ cd EAP_HOME/modules/
$ mkdir -p org/apache/commons/digester/main
```

2. 下载 **commons-digester JAR** 文件，并将它复制到 **main/** 子目录。

3. 在 **main/** 子目录中，创建与此 模板 类似的 **module.xml** 文件。如果使用模板，请务必对记下的可替换变量使用适当的值。

设置默认 **Jakarta Server Faces** 实施

1. 运行以下管理 **CLI** 命令，将新的 **Jakarta Server Faces** 实施设置为默认实施：

```
/subsystem=jsf:write-attribute(name=default-jsf-impl-slot,value=IMPL_NAME-VERSION)
```

2. 重新启动 **JBoss EAP** 服务器以使更改生效。

21.1.2. 多Jakarta 服务器 Faces 实施支持

JBoss EAP 7.3 随附一个 **Jakarta Server Faces** 实施，即 **Jakarta Server Faces 2.3** 实施，基于 **Mojarra**。

多**Jakarta** 服务器 **Faces** 允许在同一 **JBoss EAP** 服务器上安装多个 **Jakarta** 服务器 **Faces** 实施和版本。目标是允许使用 **Jakarta Server Faces**、**MyFaces** 或 **Mojarra** 以及 **Java EE JSF 2.1** 及更高版本中的任何版本的实施，以及 **Jakarta Server Faces 2.3** 及更高版本。多**Jakarta** 服务器 **Faces** 提供了与容器完全集成的一种实施，可实现更有效的注解处理、生命周期处理和其他集成优势。

21.1.2.1. Multi-Jakarta Server Faces 实施

多**Jakarta** 服务器 **Faces** 的工作方式是，对于每个 **Jakarta Server Faces** 版本，在 **com.sun.jsf-impl**、**javax. faces.api** 和 **org.jboss.as.jsf-injection** 下创建一个新的插槽。当 **jsf** 子系统启动时，它将扫描模块路径，以查找所有已安装的 **Jakarta Server Faces** 实施。当 **jsf** 子系统部署包含指定上下文参数的 **Web** 应用时，它会将插槽的模块添加到部署中。

例如，若要指明 **Jakarta Server Faces** 应用应当使用 **MyFaces 2.2.12**（假设服务器上已安装了 **MyFaces 2.2.12**，则必须添加以下上下文参数：

```
<context-param>
  <param-name>org.jboss.jbossfaces.JSF_CONFIG_NAME</param-name>
  <param-value>myfaces-2.2.12</param-value>
</context-param>
```

21.1.2.2. 更改默认的 Jakarta 服务器 Faces 实施

multi-Jakarta 服务器 **Faces** 功能在 **jsf** 子系统 中包含 **default-jsf-impl-slot** 属性。此属性允许您更改默认的 **Jakarta Server Faces** 实现，如下流程所述：

1. 使用 **write-attribute** 命令将 **default-jsf-impl-slot** 属性的值设置为活跃的 **Jakarta Server Faces** 实施之一：

```
/subsystem=jsf:write-attribute(name=default-jsf-impl-slot,value=JSF_IMPLEMENTATION)
```

2. 重新启动 **JBoss EAP** 服务器以使更改生效。

```
reload
```

要查看安装了哪些 **Jakarta Server Faces** 实施，您可以执行 **list-active-jsf-impls** 操作。

```
/subsystem=jsf:list-active-jsf-impls
{
    "outcome" => "success",
    "result" => [
        "myfaces-2.1.12",
        "mojarra-2.2.0-m05",
        "main"
    ]
}
```

21.2. 禁止 DOCTYPES

您可以使用以下管理 **CLI** 命令在 **Jakarta Server Faces** 部署中禁止 **DOCTYPE** 声明：

```
/subsystem=jsf:write-attribute(name=disallow-doctype-decl,value=true)
reload
```

*您可以通过在部署的 **web.xml** 文件中添加 **com.sun.faces.disallowDoctypeDecl** 上下文参数，为特定的 **Jakarta Server Faces** 部署 覆盖此设置：*

```
<context-param>
  <param-name>com.sun.faces.disallowDoctypeDecl</param-name>
  <param-value>false</param-value>
</context-param>
```

第 22 章 配置批处理应用程序

JBoss EAP 7 引入了对 JSR-352 中定义的 Java 批处理应用的支持。批处理应用的 Jakarta 等效于 Jakarta Batch。您可以配置环境以运行批处理应用，并且利用 batch-jberet 子系统管理批处理作业。

如需关于开发批处理应用的信息，请参阅 JBoss EAP 开发指南中的 Jakarta Batch 应用开发。

22.1. 配置批处理作业

您可以使用 batch-jberet 子系统配置批处理作业的设置，该子系统基于 JBeret 实施。

默认的 batch-jberet 子系统配置定义内存中作业存储库和默认线程池设置。

```
<subsystem xmlns="urn:jboss:domain:batch-jberet:2.0">
    <default-job-repository name="in-memory"/>
    <default-thread-pool name="batch"/>
    <job-repository name="in-memory">
        <in-memory/>
    </job-repository>
    <thread-pool name="batch">
        <max-threads count="10"/>
        <keepalive-time time="30" unit="seconds"/>
    </thread-pool>
</subsystem>
```

默认情况下，服务器恢复后将重新启动服务器暂停期间停止的任何批处理作业。您可以将 restart-jobs-on-resume 属性设置为 false，以将作业保留为 STOPPED 状态。

```
/subsystem=batch-jberet:write-attribute(name=restart-jobs-on-resume,value=false)
```

您还可以配置批处理作业存储库和 线程池的设置。

22.1.1. 配置批处理作业存储库

本节介绍如何使用管理 CLI 配置内存中和 JDBC 作业存储库来存储批处理作业信息。您还可以通过使用管理控制台配置作业存储库，方法是导航到 Configuration → Subsystems → Batch(JBeret)，点 View，然后从左侧菜单中选择 In Memory 或 JDBC。

添加内存中作业存储库

您可以添加将批处理作业信息存储在内存中的作业存储库。

```
/subsystem=batch-jberet/in-memory-job-repository=REPOSITORY_NAME:add
```

添加 **JDBC** 作业存储库

您可以添加将批处理作业信息存储在数据库中的作业存储库。您必须指定用于连接数据库的数据源的名称。

```
/subsystem=batch-jberet/jdbc-job-repository=REPOSITORY_NAME:add(data-
source=DATASOURCE)
```

设置默认作业存储库

您可以将内存中或 **JDBC** 作业存储库设置为批处理应用的默认作业存储库。

```
/subsystem=batch-jberet:write-attribute(name=default-job-repository,value=REPOSITORY_NAME)
```

这将需要重新加载服务器。

```
reload
```

22.1.2. 配置批处理线程池

本节介绍如何使用管理 **CLI** 配置线程池和线程工厂，以用于批处理作业。您还可以使用管理控制台配置线程池和线程工厂，方法是导航到 **Configuration → Subsystems → Batch(JBeret)**，点 **View**，然后从左侧菜单中选择 **Thread Factory** 或 **Thread Pool**。

配置线程池

在添加线程池时，您必须指定 **max-threads**，其应始终大于 **3**，因为保留了两个线程，以确保分区作业可以按预期执行。

1. 添加线程池。

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:add(max-threads=10)
```

2. 如果需要，设置 **keepalive-time** 值。

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-
attribute(name=keepalive-time,value={time=60,unit=SECONDS})
```

使用线程事实

1. 添加线程工厂。

```
/subsystem=batch-jberet/thread-factory=THREAD_FACTORY_NAME:add
```

2. 为线程工厂配置所需的属性。

- **Group-name -** 为此线程工厂创建的线程组的名称。
- 优先级 **-** 创建的线程的线程优先级。
- **thread-name-pattern -** 用于为线程创建名称的模板。可以使用以下模式：
  - **%% -** 百分比符号
  - **%t -** 每个事实线程序列号
  - **%g -** 全局线程序列号
  - **% F -** 工厂序列号
  - **%I -** 线程 **ID**

3. 将线程工厂分配到线程池。

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:write-attribute(name=thread-factory,value=THREAD_FACTORY_NAME)
```

这将需要重新加载服务器。

```
reload
```

设置默认线程池

您可以将不同的线程池设置为默认线程池。

```
/subsystem=batch-jberet:write-attribute(name=default-thread-pool,value=THREAD_POOL_NAME)
```

这将需要重新加载服务器。

```
reload
```

查看线程池统计信息

您可以使用 **read-resource** 管理 **CLI** 操作查看批处理线程池的运行时信息。您必须使用 **include-runtime=true** 参数来查看此运行时信息。

```
/subsystem=batch-jberet/thread-pool=THREAD_POOL_NAME:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "active-count" => 0,
        "completed-task-count" => 0L,
        "current-thread-count" => 0,
        "keepalive-time" => undefined,
        "largest-thread-count" => 0,
        "max-threads" => 15,
        "name" => "THREAD_POOL_NAME",
        "queue-size" => 0,
        "rejected-count" => 0,
        "task-count" => 0L,
        "thread-factory" => "THREAD_FACTORY_NAME"
    }
}
```

您还可以从 **Runtime** 选项卡导航到 **Batch** 子系统，从而使用管理控制台查看批处理线程池的运行时信息。

22.2. 管理批处理作业

通过 用于部署的 **batch-jberet** 子系统资源，您可以启动、停止、重新启动和查看批处理作业的执行详细信息。批处理作业可以通过管理 **CLI** 或管理控制台进行管理。

通过管理 **CLI** 管理批处理作业
重启批处理作业

您可以通过提供执行 **ID** 和重启批处理作业时要使用的任何属性，重启 **STOPPED** 或 **FAILED** 状态的作业。

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:restart-job(execution-
id=EXECUTION_ID,properties={PROPERTY=VALUE})
```

执行 **ID** 必须是作业实例的最新执行。

启动批处理作业

您可以通过提供作业 **XML** 文件以及启动批处理作业时要使用的任何属性（可选）来启动批处理作业。

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:start-job(job-xml-
name=JOB_XML_NAME,properties={PROPERTY=VALUE})
```

停止批处理作业

您可以通过提供执行 **ID** 来停止正在运行的批处理作业。

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:stop-job(execution-
id=EXECUTION_ID)
```

查看批处理作业执行详情

您可以查看批处理作业执行的详细信息。您必须在 **read -resource** 操作中使用 **include- runtime=true** 参数来查看此运行时信息。

```
/deployment=DEPLOYMENT_NAME/subsystem=batch-jberet:read-resource(recursive=true,include-
runtime=true)
{
    "outcome" => "success",
    "result" => {"job" => {"import-file" => {
        "instance-count" => 2,
        "running-executions" => 0,
        "execution" => {
            "2" => {
                "batch-status" => "COMPLETED",
                "create-time" => "2016-04-11T22:03:12.708-0400",
                "end-time" => "2016-04-11T22:03:12.718-0400",
                "exit-status" => "COMPLETED",
                "instance-id" => 58L,
                "last-updated-time" => "2016-04-11T22:03:12.719-0400",
                "start-time" => "2016-04-11T22:03:12.708-0400"
            },
            "1" => {
                "batch-status" => "FAILED",
                "create-time" => "2016-04-11T21:57:17.567-0400",
                "end-time" => "2016-04-11T21:57:17.596-0400",
```

```
            "exit-status" => "Error : org.hibernate.exception.ConstraintViolationException: could not
execute statement",
            "instance-id" => 15L,
            "last-updated-time" => "2016-04-11T21:57:17.597-0400",
            "start-time" => "2016-04-11T21:57:17.567-0400"
        }
    }
  }}}
}
```

从管理控制台管理批处理作业

若要从管理控制台管理批处理作业，可导航到 **Runtime** 选项卡，选择服务器，选择 **Batch(JBeret)**，然后从列表中选择作业。

重启批处理作业

选择执行并单击重启来重启 **STOPPED** 作业。

启动批处理作业

通过选择作业并从下拉菜单中选择 **Start**，启动批处理作业的新执行。

停止批处理作业

选择执行并单击 **Stop**，停止正在运行的批处理作业。

查看批处理作业执行详情

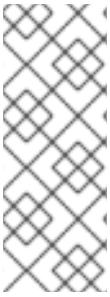
表中列出的每个执行都显示作业执行详细信息。

22.3. 为批处理作业配置安全性

您可以配置 **batch-jberet** 子系统，以使用 **Elytron** 安全域运行批处理作业。这允许使用相同的安全身份安全地暂停和恢复批处理作业。例如，创建了一个安全的 **RESTful** 端点，以使用 **batch-jberet** 子系统启动批处理作业。如果 **RESTful** 端点和 **batch-jberet** 子系统都使用相同的安全域进行保护，或者 **batch-jberet** 安全域信任 **RESTful** 端点的安全域，则以这种方式启动的批处理作业可以通过相同的安全身份安全地暂停和恢复。

使用以下管理 **CLI** 命令，更新 **security-domain** 属性，以配置批处理作业的安全性。

```
/subsystem=batch-jberet:write-attribute(name=security-domain, value=ExampleDomain)

reload
```



注意

批处理作业需要 ***org.wildfly.extension.batch.jberet.deployment.BatchPermission*** *权限。它提供与* ***javax.batch.operations.JobOperator*** *一致的* ***start***、 ***stop***、 ***restart***、 ***rewal*** *和* ***read*** *权限。* ***default-permission-mapper*** *映射 程序提供* ***org.wildfly.extension.batch.jberet.deployment.BatchPermission*** *权限。*

第 23 章 配置命名子系统

23.1. 关于命名子系统

naming 子系统为 JBoss EAP 提供 JNDI 实施。您可以配置此子系统来绑定全局 JNDI 命名空间中的条目。您还可以将其配置为激活或停用远程 JNDI 接口。

以下是 命名 子系统 XML 配置示例示例，其中包含指定的所有元素和属性：

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <simple name="java:global/simple-integer-binding" value="100" type="int" />
        <simple name="java:global/jboss.org/docs/url" value="https://docs.jboss.org"
type="java.net.URL" />
        <object-factory name="java:global/foo/bar/factory" module="org.foo.bar"
class="org.foo.bar.ObjectFactory" />
        <external-context name="java:global/federation/ldap/example"
class="javax.naming.directory.InitialDirContext" cache="true">
            <environment>
                <property name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory" />
                <property name="java.naming.provider.url" value="ldap://ldap.example.com:389" />
                <property name="java.naming.security.authentication" value="simple" />
                <property name="java.naming.security.principal" value="uid=admin,ou=system" />
                <property name="java.naming.security.credentials" value="secret" />
            </environment>
        </external-context>
        <lookup name="java:global/new-alias-name" lookup="java:global/original-name" />
    </bindings>
    <remote-naming/>
</subsystem>
```

23.2. 配置全局绑定

命名 子系统允许您将条目绑定到 java:global、java:jboss 或 java 全局 JNDI 命名空间；不过，建议您使用标准的可移植 java:global 命名空间。

全局绑定在 naming 子系统的 <bindings> 元素中配置。支持四种绑定类型：

- 简单绑定
- 对象工厂绑定

- 外部上下文绑定
- 绑定查找别名

配置简单绑定

简单的 **XML** 配置元素绑定了原语或 **java.net.URL** 条目。

- **name** 属性是必需的，它指定条目的目标 **JNDI** 名称。
- **value** 属性是强制的，定义条目的值。
- 可选的 **type** 属性（默认为 **java.lang.String** ）指定条目值的类型。除了 **java.lang.String** 外，您还可以指定原语类型及其对应的对象打包程序类，如 **int** 或 **java.lang.Integer** 和 **java.net.URL**。

以下是管理 **CLI** 命令的示例，它可创建一个简单的绑定：

```
/subsystem=naming/binding=java\:global\/simple-integer-binding:add(binding-type=simple, type=int,
value=100)
```

生成 **XML** 配置

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <simple name="java:global/simple-integer-binding" value="100" type="int"/>
    </bindings>
    <remote-naming/>
</subsystem>
```

使用以下命令来删除绑定：

```
/subsystem=naming/binding=java\:global\/simple-integer-binding:remove
```

绑定对象事实

object-factory XML 配置元素绑定了 javax.naming.spi.ObjectFactory 条目。

- name 属性是必需的，它指定条目的目标 JNDI 名称。
- class 属性是强制的，定义对象工厂的 Java 类型。
- module 属性是必须的，用于指定可从中加载对象工厂 Java 类的 JBoss 模块 ID。
- 可选 的环境 子元素可用于为对象工厂提供自定义环境。

以下是管理 CLI 命令的示例，它创建一个对象工厂绑定：

```
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:add(binding-type=object-factory,
module=org.foo.bar, class=org.foo.bar.ObjectFactory, environment=[p1=v1, p2=v2])
```

生成 XML 配置

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
    <bindings>
        <object-factory name="java:global/foo/bar/factory" module="org.foo.bar"
class="org.foo.bar.ObjectFactory">
            <environment>
                <property name="p1" value="v1" />
                <property name="p2" value="v2" />
            </environment>
        </object-factory>
    </bindings>
</subsystem>
```

使用以下命令来删除绑定：

```
/subsystem=naming/binding=java\:global\/foo\/bar\/factory:remove
```

绑定外部上下文

外部 **JNDI** 上下文的联合（如 **LDAP** 上下文）可使用 外部上下文 **XML** 配置元素来实现。

- **name** 属性是必需的， 它指定条目的目标 **JNDI** 名称。
- **class** 属性是强制的， 指示用于创建联合上下文的 **Java** 初始命名上下文类型。请注意， 此类类型必须具有带有单个环境映射参数的构造器。
- 可选的 **module** 属性指定 **JBoss** 模块 **ID**， 可在其中加载外部 **JNDI** 上下文所需的任何类。
- 可选的 **cache** 属性（默认为 **false** ）指明是否应缓存外部上下文实例。
- 可选 的环境 子元素可用于提供查找外部上下文所需的自定义环境。

以下是用于创建外部上下文绑定的管理 **CLI** 命令示例：

```
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:add(binding-type=external-
context, cache=true, class=javax.naming.directory.InitialDirContext, module=org.jboss.as.naming,
environment=[java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory,
java.naming.provider.url="ldap://ldap.example.com:389", java.naming.security.authentication=simple,
java.naming.security.principal="uid=admin,ou=system", java.naming.security.credentials=secret])
```

生成 **XML** 配置

```
<subsystem xmlns="urn:jboss:domain:naming:2.0">
  <bindings>
    <external-context name="java:global/federation/ldap/example" module="org.jboss.as.naming"
class="javax.naming.directory.InitialDirContext" cache="true">
      <environment>
        <property name="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>
        <property name="java.naming.provider.url" value="ldap://ldap.example.com:389"/>
        <property name="java.naming.security.authentication" value="simple"/>
        <property name="java.naming.security.principal" value="uid=admin,ou=system"/>
        <property name="java.naming.security.credentials" value="secret"/>
      </environment>
    </external-context>
  </bindings>
</subsystem>
```

使用以下命令来删除绑定：

```
/subsystem=naming/binding=java\:global\/federation\/ldap\/example:remove
```

注意

资源查找未正确实施 **lookup(Name)** 方法的 **JNDI** 提供程序，可能会导致 **"javax.naming.InvalidNameException: onlyly support CompoundName name"** 错误。

您可以通过指定外部上下文环境改为使用 **lookup(String)** 方法来解决这个问题；但是，这可能会导致性能下降。

```
<property name="org.jboss.as.naming.lookup.by.string" value="true"/>
```

绑定查找别名

lookup 元素允许您将现有条目绑定到其他名称或别名中。

- **name** 属性是必需的，它指定条目的目标 **JNDI** 名称。
- **lookup** 属性是必需的，指明源 **JNDI** 名称。

以下是将现有条目绑定到别名的管理 **CLI** 命令示例：

```
/subsystem=naming/binding=java\:global\/new-alias-name:add(binding-type=lookup,
lookup=java\:global\/original-name)
```

生成 **XML** 配置

```
<lookup name="java:global/new-alias-name" lookup="java:global/original-name" />
```

使用以下命令来删除绑定：

```
/subsystem=naming/binding=java\:global\/c:remove
```

23.3. 动态更改 JNDI 绑定

JBoss EAP 7.1 引入了无需强制重新加载或重新启动服务器即可动态更改 **JNDI** 绑定的功能。如果因为版本更新、测试要求或应用功能更新而动态重新配置网络服务端点，此功能很有用。

若要更新 **JNDI** 绑定，请使用 **rebind** 操作。**rebind** 操作使用与 **add** 操作相同的参数。除了 外部上下文绑定类型外，此命令适用于所有 绑定类型。外部上下文绑定需要额外的依赖项，这会影响 **Modular Service Container(MSC)**状态，因此在重新启动服务的情况下无法重启它们。

以下命令动态更改配置简单绑定示例中定义的 **JNDI** 绑定。

```
/subsystem=naming/binding=java\:global\/simple-integer-binding:rebind(binding-type=simple,
type=int, value=200)
```

有关如何在 命名 子系统中配置全局绑定的更多信息，请参阅 配置全局绑定。

23.4. 配置远程 JNDI 接口

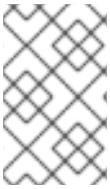
远程 **JNDI** 接口允许客户端在远程 **JBoss EAP** 实例中查找条目。可以将 **naming** 子系统配置为停用或激活此接口，该接口默认为激活。远程 **JNDI** 接口使用 **<remote-naming>** 元素进行配置。

使用以下管理 **CLI** 命令，激活或重新激活远程 **JNDI** 接口：

```
/subsystem=naming/service=remote-naming:add
```

使用以下管理 **CLI** 命令停用远程 **JNDI** 接口：

```
/subsystem=naming/service=remote-naming:remove
```



注意

*只有 **java:jboss/exported** 上下文中的条目才能通过远程 **JNDI** 访问。*

第 24 章 配置高可用性

24.1. 高可用性简介

JBoss EAP 提供下列高可用性服务来确保所部署的 Jakarta EE 应用可用：

负载平衡

这使得服务能够通过将工作负载分散到多个服务器来处理大量请求。即使出现大量请求，客户端也可以从服务及时响应。

故障切换

这使得客户端即使在硬件或网络故障时也能不间断地访问服务。如果服务失败，则另一群集成员接管客户端的请求，以便其能够继续处理。

群集是一个包括所有这些功能的术语。可以将群集成员配置为共享工作负载，称为负载平衡，并在另一群集成员出现故障时（称为故障转移）选择客户端处理。

注意

务必牢记，所选的 JBoss EAP 操作模式（单机服务器或 受管域 ）与您希望如何管理服务器相关。高可用性服务可以在 JBoss EAP 中配置，无论其工作模式如何。

JBoss EAP 支持使用各种组件在多个不同级别上实现高可用性。运行时的部分和可高度可用的应用程序组件有：

- 应用服务器的实例
- Web 应用程序与内部 JBoss Web 服务器、Apache HTTP 服务器、Microsoft IIS 或 Oracle iPlanet Web 服务器结合使用时
- 有状态和无状态会话 Enterprise JavaBeans(EJB)
- 单点登录(SSO)机制

- HTTP 会话
- JMS 服务和消息驱动的 Bean(MDB)
- 单例 MSC 服务
- 单例部署

群集由 jgroups、in finispan 和 modcluster 子系统提供给 JBoss EAP。ha 和 full-ha 配置文件启用了这些系统。在 JBoss EAP 中，这些服务按需启动和关闭，但只有在服务器上部署了配置为可分布式的应用时，它们才会启动。

有关如何将应用标记为可分布式，请参见 JBoss EAP 开发指南。

24.2. 使用 JGROUPS 进行群集通信

24.2.1. 关于 JGroups

JGroups 是用于可靠消息传递的工具包，可用于创建节点可以互相发送消息的群集。

jgroups 子系统提供对 JBoss EAP 中高可用性服务的组通信支持。它允许您配置指定的频道和协议堆栈，以及查看频道的运行时统计信息。当使用提供高可用性功能的配置（如受管域中的 ha 或 full-ha 配置文件）或 standalone-ha.xml 或 standalone- full-ha. xml 配置文件时，可以使用 jgroups 子系统。

JBoss EAP 预配置了两个 JGroups 堆栈：

udp

群集中的节点使用用户数据报协议(UDP)多播互相通信。这是默认的堆栈。

tcp

群集中的节点使用传输控制协议(TCP)相互通信。

注意

TCP 的开销更大，通常被视为比 UDP 慢，因为它处理错误检查、数据包排序和拥塞控制本身。JGroups 为 UDP 处理这些功能，而 TCP 则保证其自身。在不可靠或高拥塞网络上使用 JGroups 时，或者多播不可用时，TCP 是一种不错的选择。

您可以使用预配置的堆栈或自行定义以满足系统特定要求。有关可用协议及其属性的更多信息，请参见以下部分。

- *JGroups Subsystem Attributes*
- *JGroups Protocols*

24.2.2. 将默认 JGroups 频道切换为使用 TCP

默认情况下，集群节点使用为 ee JGroups 频道配置的 udp 协议堆栈进行通信。

```
<channels default="ee">
  <channel name="ee" stack="udp"/>
</channels>
<stacks>
  <stack name="udp">
    <transport type="UDP" socket-binding="jgroups-udp"/>
    <protocol type="PING"/>
    ...
  </stack>
  <stack name="tcp">
    <transport type="TCP" socket-binding="jgroups-tcp"/>
    <protocol type="MPING" socket-binding="jgroups-mping"/>
    ...
  </stack>
</stacks>
```

使用多播配置 TCP

某些网络仅允许使用 TCP。使用以下管理 CLI 命令，切换 ee 频道以使用预配置的 tcp 堆栈：

```
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcp)
```

此默认 tcp 堆栈使用 MPING 协议，它使用 IP 多播来发现初始群集成员资格。

在没有多播的情况下配置 TCP

当安全策略不首选或允许多播时，您可以将默认协议堆栈更改为使用 **TCP**。要在没有多播的情况下配置基于 **TCP** 的集群，请执行以下步骤：

1. 运行以下命令，将 **ee** 频道切换为使用 **JGroups** 子系统中预先配置的 **tcp** 堆栈：

```
<channel name="ee" stack="tcp" cluster="ejb"/>
```

2. 设置集群节点的名称：

- 在独立配置模式中，执行以下步骤之一：
  - 运行以下命令：

```
<server xmlns="urn:jboss:domain:8.0" name="node_1">
```

  - 在启动实例时，为系统属性 **jboss.node.name** 指定唯一名称。

- 在域模式中，集群服务器在 **server** 标签的 **host-*.xml** 文件中列出。默认配置指定以下服务器名称，您可以根据需要进行编辑。

```
<servers>
   <server name="server-one" group="main-server-group"/>
   <server name="server-two" group="other-server-group">
      <socket-bindings port-offset="150"/>
   </server>
</servers>
```

3. 选择以下协议之一来发现其他群集成员：

- **TCPGOSSIP** ：此协议使用外部 **Gosip** 路由器服务来发现群集成员。这需要配置和管理其他进程，但允许单个 **EAP** 实例列出其他群集成员。如果群集成员频繁变化，此协议很有用。如需了解更多详细信息，请参阅 **TCPPING**。

- **TCPPING** ：此协议定义了一个静态群集成员列表，并且要求每个节点列出所有潜在的群集成员。当群集成员地址未知且不会经常更改时，首选此协议。如需了解更多详细信息，

请参阅 **TCPGOSSIP**。

24.2.3. 配置 TCPPING

此流程创建一个新的 **JGroups** 堆栈，它使用 **TCPPING** 协议来定义静态集群成员资格列表。系统提供了基础脚本，该脚本可创建 **tcpping** 堆栈并将默认 频道 设置为使用此新堆栈。此脚本中的管理 **CLI** 命令必须为您的环境自定义，并将作为批处理处理。

1. 将以下脚本复制到文本编辑器中，并将它保存到本地文件系统。

```
# Define the socket bindings
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=jgroups-host-a:add(host=HOST_A,port=7600)
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=jgroups-host-b:add(host=HOST_B,port=7600)
batch
# Add the tcpping stack
/subsystem=jgroups/stack=tcpping:add
/subsystem=jgroups/stack=tcpping/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpping/protocol=TCPPING:add(socket-bindings=[jgroups-host-a,jgroups-host-b])
/subsystem=jgroups/stack=tcpping/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpping/protocol=FD_SOCK:add
/subsystem=jgroups/stack=tcpping/protocol=FD_ALL:add
/subsystem=jgroups/stack=tcpping/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpping/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpping/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpping/protocol=MFC:add
/subsystem=jgroups/stack=tcpping/protocol=FRAG2:add
# Set tcpping as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcpping)
run-batch
reload
```

请注意，定义的协议的顺序非常重要。您还可以通过传递 **add-index** 值到 **add** 命令，将协议插入到特定的索引中。索引基于零，因此以下管理 **CLI** 命令添加 **UNICAST3** 协议作为第七个协议：

```
/subsystem=jgroups/stack=tcpping/protocol=UNICAST3:add(add-index=6)
```

2. 为您的环境修改 脚本。

- 如果您在受管域中运行，则必须通过 **/profile= PROFILE_NAME**在

/subsystem=jgroups 命令之前指定要更新的配置集。

- 根据您的环境调整以下属性：
  - 套接字绑定： 以逗号分隔的主机和端口组合列表，这些组合被视为众所周知的，并可用于查找初始成员身份。有关定义套接字绑定的更多信息，请参阅配置套接字绑定。
  - initial_hosts ：以逗号分隔的主机和端口组合列表，使用语法 HOST[PORT]，这些 [PORT] 被视为广为人知且可用于查找初始成员资格，如 host1[1000],host2[2000]。
  - port_range ：此属性用于将 initial_hosts 端口范围扩展为指定的值。例如，如果您将 initial_hosts 设置为 host1[1000],host2[2000]，并且 port_range 设为 1，则 initial_hosts 设置将扩展到 host1[1000],host1[1001],host2[2000],host2[2001]。此属性仅适用于 initial_hosts 属性。

3. 通过将 脚本文件传递到管理 CLI 来运行 脚本。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

TCPPING 堆栈现已可用， TCP 用于网络通信。

24.2.3.1. 在独立模式中配置 TCPPING

此流程可帮助您在独立模式下为集群应用程序配置 TCP 堆栈和节点。

流程

1. 在 JGroups 子系统中将默认堆栈从 udp 改为 tcp ：

```
<channel name="ee" stack="tcp" cluster="ejb"/>
```

2. 将 TCP 堆栈配置为使用 TCPPING 协议代替默认的 MPING 协议。在以下代码中， initial_hosts 属性与群集内所有节点的列表相关联，而 7600 表示默认的 jgroups-tcp 端口可以根据您的配置和环境而变化。

```
<stack name="tcp">
  <transport type="TCP" socket-binding="jgroups-tcp"/>
  <protocol type="TCPPING">
    <property name="initial_hosts">192.168.1.5[7600],192.168.1.9[7600]</property>
    <property name="port_range">0</property>
  </protocol>
  <protocol type="MERGE3"/>
  <protocol type="FD_SOCK" socket-binding="jgroups-tcp-fd"/>
  <protocol type="FD_ALL"/>
  <protocol type="VERIFY_SUSPECT"/>
  <protocol type="pbcast.NAKACK2"/>
  <protocol type="UNICAST3"/>
  <protocol type="pbcast.STABLE"/>
  <protocol type="pbcast.GMS"/>
  <protocol type="MFC"/>
  <protocol type="FRAG3"/>
</stack>
```

注意

***initial_hosts** 中设置的端口号 **7600** 必须与 **jgroups-tcp** 套接字绑定定义中定义的端口号相同。如果将 **port-offset** 功能用于 **socket-binding**，则需要在 **initial_hosts** 中指定偏移后相同的值。*

3. *设置专用接口的 **IP** 地址，供 **JGroups** 组件使用。**IP** 地址应与 **initial_hosts** 中指定的 **IP** 地址之一相关联：*

```
<interface name="private">
  <inet-address value="${jboss.bind.address.private:192.168.1.5}"/>
</interface>
```

4. *重复上述步骤来配置集群中的其他节点。配置节点时，启动每个节点并部署集群应用。*

验证

- *您可以检查日志以验证节点是否已启动并在运行：*

```
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-2,ee,node_1)
ISPN000094: Received new cluster view for channel server: [node_1|1] (2) [node_1, node_2]
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-2,ee,node_1)
ISPN000094: Received new cluster view for channel web: [node_1|1] (2) [node_1, node_2]
```

24.2.3.2. 在域模式中配置 TCPPING

此流程可帮助您在域模式中为集群应用程序配置 **TCP** 堆栈和节点。

流程

1. 如果同一配置集用于多个集群，请将系统属性值设置为 **initial_hosts** ：

```
<protocol type="TCPPING">
  <property name="initial_hosts">${jboss.cluster.tcp.initial_hosts}</property>

Set the system property at the `server-group` level:
<server-groups>
  <server-group name="a-server-group" profile="ha">
  <socket-binding-group ref="ha-sockets"/>
  <system-properties>
    <property name="jboss.cluster.tcp.initial_hosts"
value="192.168.1.5[7600],192.168.1.9[7600]" />
  </system-properties>
```

2. 在主机控制器的 **XML** 配置中设置专用接口的 **IP** 地址。专用接口的 **IP** 地址应当与 **initial_hosts** 中列出的 **IP** 地址之一相关联。

```
<interfaces>
    ....
  <interface name="private">
    <inet-address value="${jboss.bind.address.private:192.168.1.5}"/>
  </interface>
</interfaces>
```

验证

- 您可以检查日志以验证节点是否已启动并在运行：

```
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-2,ee,node_1)
ISPN000094: Received new cluster view for channel server: [node_1|1] (2) [node_1, node_2]
INFO  [org.infinispan.remoting.transport.jgroups.JGroupsTransport] (thread-2,ee,node_1)
ISPN000094: Received new cluster view for channel web: [node_1|1] (2) [node_1, node_2]
```

24.2.4. 配置 TCPGOSSIP

此流程创建一个新的 **JGroups** 堆栈，它使用 **TCPGOSSIP** 协议来使用外部 **Gosip** 路由器发现群集的成员。我们提供了一个基础脚本，该脚本可创建 **tcpgosip** 堆栈，并将 **default ee** 通道设置为使用此新堆栈。此脚本中的管理 **CLI** 命令必须为您的环境自定义，并将作为批处理处理。

1. 将以下脚本复制到文本编辑器中，并将它保存到本地文件系统。

```
# Define the socket bindings
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=jgroups-host-a:add(host=HOST_A,port=13001)
batch
# Add the tcpgossip stack
/subsystem=jgroups/stack=tcpgossip:add
/subsystem=jgroups/stack=tcpgossip/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=tcpgossip/protocol=TCPGOSSIP:add(socket-bindings=[jgroups-host-a])
/subsystem=jgroups/stack=tcpgossip/protocol=MERGE3:add
/subsystem=jgroups/stack=tcpgossip/protocol=FD_SOCK:add
/subsystem=jgroups/stack=tcpgossip/protocol=FD_ALL:add
/subsystem=jgroups/stack=tcpgossip/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=tcpgossip/protocol=UNICAST3:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=tcpgossip/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=tcpgossip/protocol=MFC:add
/subsystem=jgroups/stack=tcpgossip/protocol=FRAG2:add
# Set tcpgossip as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=tcpgossip)
run-batch
reload
```

请注意，定义的协议的顺序非常重要。您还可以通过传递 **add-index** 值到 **add** 命令，将协议插入到特定的索引中。索引基于零，因此以下管理 **CLI** 命令添加 **UNICAST3** 协议作为第七个协议：

```
/subsystem=jgroups/stack=tcpgossip/protocol=UNICAST3:add(add-index=6)
```

2. 为您的环境修改 脚本。

- 如果您在受管域中运行，则必须通过 **/profile= PROFILE_NAME**在 **/subsystem=jgroups** 命令之前指定要更新的配置集。

- 根据您的环境调整以下属性：

  - 套接字绑定：以逗号分隔的主机和端口组合列表，这些组合被视为众所周知的，并可用于查找初始成员身份。有关定义套接字绑定的更多信息，请参阅配置套接字绑定。

  - **initial_hosts** ：以逗号分隔的主机和端口组合列表，使用语法 **HOST[PORT],**

这些 **[PORT]** 被视为广为人知且可用于查找初始成员资格，如 **host1[1000],host2[2000]**。

- **port_range** ：此属性用于将 **initial_hosts** 端口范围扩展为指定的值。例如，如果您将 **initial_hosts** 设置为 **host1[1000],host2[2000]**，并且 **port_range** 设为 **1**，则 **initial_hosts** 设置将扩展到 **host1[1000],host1[1001],host2[2000],host2[2001]**。此属性仅适用于 **initial_hosts** 属性。

- **reconnect_interval** ：断开连接的 **stub** 尝试重新连接到 **gossip** 路由器的间隔（以毫秒为单位）。

- **sock_conn_timeout** ：套接字创建的最大时间。默认值为 **1000** 毫秒。

- **sock_read_timeout** ：阻止读取的最大时间（以毫秒为单位）。值 **0** 将无限期阻止。

3. 通过将 脚本文件传递到管理 **CLI** 来运行 脚本。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

TCPGOSSIP 堆栈现已可用，**TCP** 用于网络通信。此堆栈配置为与 **gossip** 路由器搭配使用，以便 **JGroups** 群集成员能够查找其他群集成员。

24.2.5. 配置 JDBC_PING

您可以使用 **JDBC_PING** 协议管理和发现群集中的成员资格。

JDBC_PING 使用数据源中指定的数据库来列出群集的成员。

1. 创建一个数据源，以连接到您要用于管理集群成员资格的数据库。

2. 将以下脚本复制到文本编辑器中，并将它保存到本地文件系统。

```
batch
# Add the JDBC_PING stack
```

```
/subsystem=jgroups/stack=JDBC_PING:add
/subsystem=jgroups/stack=JDBC_PING/transport=TCP:add(socket-binding=jgroups-tcp)
/subsystem=jgroups/stack=JDBC_PING/protocol=JDBC_PING:add(data-
source=ExampleDS)
/subsystem=jgroups/stack=JDBC_PING/protocol=MERGE3:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FD_SOCK:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FD_ALL:add
/subsystem=jgroups/stack=JDBC_PING/protocol=VERIFY_SUSPECT:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.NAKACK2:add
/subsystem=jgroups/stack=JDBC_PING/protocol=UNICAST3:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.STABLE:add
/subsystem=jgroups/stack=JDBC_PING/protocol=pbcast.GMS:add
/subsystem=jgroups/stack=JDBC_PING/protocol=MFC:add
/subsystem=jgroups/stack=JDBC_PING/protocol=FRAG2:add
# Set JDBC_PING as the stack for the ee channel
/subsystem=jgroups/channel=ee:write-attribute(name=stack,value=JDBC_PING)
run-batch
reload
```

请注意，定义的协议的顺序非常重要。您还可以通过传递 **add-index** 值到 **add** 命令，将协议插入到特定的索引中。索引基于零，因此以下管理 **CLI** 命令添加 **UNICAST3** 协议作为第七个协议：

```
/subsystem=jgroups/stack=JDBC_PING/protocol=UNICAST3:add(add-index=6)
```

3. 为您的环境修改 脚本。

   - 如果您在受管域中运行，则必须通过 **/profile= PROFILE_NAME**在 **/subsystem=jgroups** 命令之前指定要更新的配置集。
   - 将 **'ExampleDS'** 替换为在第 **1** 步中定义的数据源的名称。

4. 通过将 脚本文件传递到管理 **CLI** 来运行 脚本。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --file=/path/to/SCRIPT_NAME
```

JDBC_PING 堆栈现已可用，**TCP** 用于网络通信。

相关信息

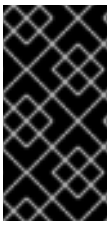
JDBC_PING: **https://developer.jboss.org/wiki/JDBCPING**

创建数据源：关于 JBoss EAP 数据源

24.2.6. 将 JGroups 绑定到网络接口

默认情况下，JGroups 仅绑定到 专用 网络接口，后者指向默认配置中的 localhost。出于安全考虑，JGroups 不会绑定到 JBoss EAP 启动期间由 -b 参数定义的网络接口，因为集群流量不应在公共网络接口上公开。

有关如何配置网络接口的详情，请查看本指南中的网络配置章节。



重要

出于安全考虑，JGroups 仅应绑定到非公共网络接口。出于性能原因，我们还建议 JGroups 流量的网络接口应该是专用虚拟局域网(VLAN)的一部分。

24.2.7. 保护集群

要安全运行集群，需要解决几个问题：

- 防止未经授权的节点加入集群。这可以通过需要身份验证来解决。
- 防止非成员与群集成员通信.这可以通过加密消息来解决。

24.2.7.1. 配置身份验证

JGroups 身份验证由 AUTH 协议执行。目的是确保只有经过身份验证的节点才能加入群集。

在适用的服务器配置文件中，使用适当的属性设置添加 AUTH 协议。AUTH 协议应该在 pbcast.GMS 协议之前配置。

以下示例演示了如何将 AUTH 与不同类型的授权令牌搭配使用。

带有简单令牌的AUTH

```
...
```

```
<protocol type="pbcast.STABLE"/>
<auth-protocol type="AUTH">
  <plain-token>
    <shared-secret-reference clear-text="my_password"/>
  </plain-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...
```

*带有 **Digest Algorithm** 令牌的 **AUTH***

*此格式可与任何摘要算法一起使用，如 **MD5** 或 **SHA-2**。**JBoss EAP 7.3** 的默认摘要算法是 **SHA-256**，**JVM** 支持最强大的摘要算法。很多 **JVM** 还将实施 **SHA-512**。*

```
...
<protocol type="pbcast.STABLE"/>
<auth-protocol type="AUTH">
  <digest-token algorithm="SHA-512">
    <shared-secret-reference clear-text="my_password"/>
  </digest-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...
```

*带有 **X509** 令牌的 **AUTH***

*本例在 **elytron** 子系统中创建新的密钥存储，并在 **JGroups AUTH** 配置中引用它。*

1. *创建密钥存储：*

```
$ keytool -genkeypair -alias jgroups_key -keypass my_password -storepass my_password -storetype jks -keystore jgroups.keystore -keyalg RSA
```

2. *使用管理 **CLI** 将密钥存储添加到 **elytron** 子系统：*

```
/subsystem=elytron/key-store=jgroups-token-store:add(type=jks,path=/path/to/jgroups.keystore,credential-reference={clear-text=my_password}, required=true)
```

3. *在 **JGroups** 堆栈定义中，将 **AUTH** 配置为使用密钥存储：*

```
...
<protocol type="pbcast.STABLE"/>
<auth-protocol type="AUTH">
  <cipher-token algorithm="RSA" key-alias="jgroups_key" key-store="jgroups-token-store">
    <shared-secret-reference clear-text="my_password"/>
    <key-credential-reference clear-text="my_password"/>
```

```
  </cipher-token>
</auth-protocol>
<protocol type="pbcast.GMS"/>
...
```

24.2.7.2. 配置加密

为加密消息，**JGroups** 使用由群集成员共享的机密密钥。发送方使用共享机密密钥加密消息，接收方使用相同的机密密钥解密消息。通过使用 **SYM_ENCRYPT** 协议配置的 对称加密，节点使用共享密钥存储来检索密钥。使用 **ASYM_ENCRYPT** 协议配置的非 对称加密 时，节点在使用 **AUTH** 进行身份验证后，从集群的协调人员检索机密密钥。

使用对称加密

要使用 **SYM_ENCRYPT**，您必须设置一个密钥存储，该存储将在每个节点的 **JGroups** 配置中引用。

1. 创建密钥存储.

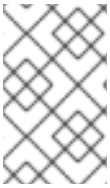
   在以下命令中，将 **VERSION** 替换为适当的 **JGroups JAR** 版本，并将 **PASSWORD** 替换为密钥存储密码。

   ```
   $ java -cp EAP_HOME/modules/system/layers/base/org/jgroups/main/jgroups-VERSION.jar
   org.jgroups.demos.KeyStoreGenerator --alg AES --size 128 --storeName
   defaultStore.keystore --storepass PASSWORD --alias mykey
   ```

   这将生成一个 **defaultStore.keystore** 文件，该文件将在 **JGroups** 配置中引用。

2. 生成密钥存储后，将使用以下两种方法之一在 **SYM_PROTOCOL** 中定义它。

   - 在配置中直接指定密钥存储。
   - 使用 **Elytron** 子系统引用密钥存储。



注意

使用 **SYM_ENCRYPT** 时配置 **AUTH** 是可选的。

通过直接引用密钥存储使用对称加密

1. 在 **jgroups** 子系统中配置 **SYM_ENCRYPT** 协议。

在适用的服务器配置文件中，使用适当的属性设置添加 **SYM_ENCRYPT** 协议。此协议将引用之前创建的密钥存储。**SYM_ENCRYPT** 协议应该在 **pbcast.NAKACK2** 协议之前立即配置。

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="SYM_ENCRYPT">
        <property name="provider">SunJCE</property>
        <property name="sym_algorithm">AES</property>
        <property name="encrypt_entire_message">true</property>
        <property name="keystore_name">/path/to/defaultStore.keystore</property>
        <property name="store_password">PASSWORD</property>
        <property name="alias">mykey</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

*通过 **Elytron** 使用对称加密*

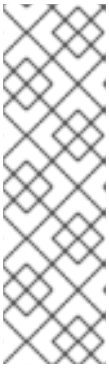
1. 利用管理 **CLI**，在 **elytron** 子系统中创建密钥存储，该子系统引用 *使用对称加密* 在 中创建的 **defaultStore.keystore**。

```
/subsystem=elytron/key-store=jgroups-
keystore:add(path=/path/to/defaultStore.keystore,credential-reference={clear-
text=PASSWORD},type=JCEKS)
```

2. 使用适当的属性设置，在 **jgroups** 子系统中添加 **SYM_ENCRYPT** 协议。**SYM_ENCRYPT** 协议应该在 **pbcast.NAKACK2** 协议之前配置，如以下配置所示。

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
```

```
        <transport type="UDP" socket-binding="jgroups-udp"/>
        <protocol type="PING"/>
        <protocol type="MERGE3"/>
        <protocol type="FD_SOCK"/>
        <protocol type="FD_ALL"/>
        <protocol type="VERIFY_SUSPECT"/>
        <encrypt-protocol type="SYM_ENCRYPT" key-alias="mykey" key-store="jgroups-
keystore">
          <key-credential-reference clear-text="PASSWORD"/>
          <property name="provider">SunJCE</property>
          <property name="encrypt_entire_message">true</property>
        </encrypt-protocol>
        <protocol type="pbcast.NAKACK2"/>
        <protocol type="UNICAST3"/>
        <protocol type="pbcast.STABLE"/>
        <protocol type="pbcast.GMS"/>
        <protocol type="UFC"/>
        <protocol type="MFC"/>
        <protocol type="FRAG2"/>
      </stack>
    </stacks>
</subsystem>
```



注意

上例使用明文密码；但是，也可以定义凭据存储来定义配置文件外的密码。有关配置此存储的更多信息，请参阅 如何配置服务器安全指南中 ***https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/how_to_configure_server_security/#credential_store*** *的凭证存储部分。*

使用对称加密

要使用 ***ASYM_ENCRYPT****，必须定义* ***AUTH*** *协议。有关在* ***jgroups*** *子系统中配置* ***AUTH*** *协议的说明，请参阅配置 身份验证 部分。*

ASYM_ENCRYPT *使用以下两种方法之一进行配置：*

- *生成机密密钥并在配置中直接引用该密钥。*
- *创建密钥存储并使用* ***Elytron*** *子系统引用它。*

通过生成 ***secret*** *密钥使用对称加密*

1. 在 **jgroups** 子系统中配置 **ASYM_ENCRYPT** 协议。

   在适用的服务器配置文件中，使用适当的属性设置添加 **ASYM_ENCRYPT** 协议。**ASYM_ENCRYPT** 协议应该在 **pbcast.NAKACK2** 协议之前配置，如以下配置所示。

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <protocol type="ASYM_ENCRYPT">
        <property name="encrypt_entire_message">true</property>
        <property name="sym_keylength">128</property>
        <property name="sym_algorithm">AES/ECB/PKCS5Padding</property>
        <property name="asym_keylength">512</property>
        <property name="asym_algorithm">RSA</property>
      </protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <!-- Configure AUTH protocol here -->
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```

通过 **Elytron** 使用非对称加密

1. 创建密钥存储以包含密钥对。以下命令使用 **mykey** 条目创建一个密钥存储：

```
$ keytool -genkeypair -alias mykey -keyalg RSA -keysize 1024 -keystore
defaultKeystore.keystore -dname "CN=localhost" -keypass secret -storepass secret
```

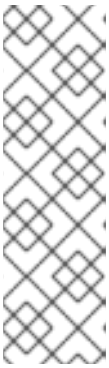
2. 使用管理 **CLI**，在 **elytron** 子系统中创建密钥存储，以引用 **defaultStore.keystore**。

```
/subsystem=elytron/key-store=jgroups-
keystore:add(path=/path/to/defaultStore.keystore,credential-reference={clear-
text=PASSWORD},type=JCEKS)
```

3. 使用适当的属性设置，在 **jgroups** 子系统中添加 **ASYM_ENCRYPT** 协

议。**ASYM_ENCRYPT** 协议应该在 **pbcast.NAKACK2** 协议之前配置，如以下配置所示。

```
<subsystem xmlns="urn:jboss:domain:jgroups:6.0">
  <stacks>
    <stack name="udp">
      <transport type="UDP" socket-binding="jgroups-udp"/>
      <protocol type="PING"/>
      <protocol type="MERGE3"/>
      <protocol type="FD_SOCK"/>
      <protocol type="FD_ALL"/>
      <protocol type="VERIFY_SUSPECT"/>
      <encrypt-protocol type="ASYM_ENCRYPT" key-alias="mykey" key-store="jgroups-
keystore">
        <key-credential-reference clear-text="secret" />
        <property name="encrypt_entire_message">true</property>
      </encrypt-protocol>
      <protocol type="pbcast.NAKACK2"/>
      <protocol type="UNICAST3"/>
      <protocol type="pbcast.STABLE"/>
      <!-- Configure AUTH protocol here -->
      <protocol type="pbcast.GMS"/>
      <protocol type="UFC"/>
      <protocol type="MFC"/>
      <protocol type="FRAG2"/>
    </stack>
  </stacks>
</subsystem>
```



注意

上例使用明文密码；但是，也可以定义凭据存储来定义配置文件外的密码。有关配置此存储的更多信息，请参阅 如何配置服务器安全指南中 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/how_to_configure_server_security/#credential_store** 的凭证存储部分。

24.2.8. 配置 JGroups 线程池

jgroups 子系统包含 **default**、**internal**、 **oob** 和 **timer** 线程池。可以为任何 **JGroups** 堆栈配置这些池，并且不影响本地节点上配置的 **bean** 或其他池。**JGroup** 线程池用于支持集群通信。

下表列出了您可以为每个线程池配置的属性以及每个线程的默认值。

线程池名称	描述	keepalive-time	max-threads	Min-threads	queue-length
default	此池用于处理没有标记为带外的传入信息。	60000L	300	20	100
internal	此池用于处理 EAP 维护所需的内部进程。	60000L	4	2	100
OOB	此池用于处理传入的带外信息。	60000L	300	20	0
计时器	此池用于处理时间绑定调度程序消息。	5000L	4	2	500

*利用下列语法，通过管理 **CLI** 配置 **JGroups** 线程池：*

```
/subsystem=jgroups/stack=STACK_TYPE/transport=TRANSPORT_TYPE/thread-pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME, value=ATTRIBUTE_VALUE)
```

*以下是管理 **CLI** 命令的示例，可将 **the udp** 堆栈的默认线程池中的 **max-threads** 值设置为 **500** ：*

```
/subsystem=jgroups/stack=udp/transport=UDP/thread-pool=default:write-attribute(name="max-threads", value="500")
```

*24.2.9. 配置 **JGroups** 发送和接收缓冲区*

解决缓冲区大小警告

*默认情况下，**JGroups** 配置有特定的发送和接收缓冲区值；但是，您的操作系统可能会限制可用的缓冲区大小，**JBoss EAP** 可能无法使用其配置的缓冲区值。在这种情况下，您将在 **JBoss EAP** 日志中看到类似如下的警告：*

```
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the send buffer of socket DatagramSocket was set to 640KB, but the OS only allocated 212.99KB.
This might lead to performance problems. Please set your max send buffer in the OS correctly (e.g. net.core.wmem_max on Linux)
WARNING [org.jgroups.protocols.UDP] (ServerService Thread Pool -- 68)
JGRP000015: the receive buffer of socket DatagramSocket was set to 20MB, but the OS only
```

> *allocated 212.99KB.*
> *This might lead to performance problems. Please set your max receive buffer in the OS correctly (e.g. net.core.rmem_max on Linux)*

要解决这个问题，请查阅您的操作系统文档来了解如何增加缓冲区大小。对于 ***Red Hat Enterprise Linux*** *系统，以* ***root*** *用户身份编辑* ***/etc/sysctl.conf*****，** *为在系统重启后保留的缓冲区大小配置最大值。例如：*

```
# Allow a 25MB UDP receive buffer for JGroups
net.core.rmem_max = 26214400
# Allow a 1MB UDP send buffer for JGroups
net.core.wmem_max = 1048576
```

修改 ***/etc/sysctl.conf*** *后，运行* ***sysctl -p*** *以使更改生效。*

配置 ***JGroups*** *缓冲器大小*

您可以通过在 ***UDP*** *和* ***TCP JGroups*** *堆栈中设置以下传输属性来配置* ***JBoss EAP*** *使用的* ***JGroups*** *缓冲区大小：*

UDP Stack

- ***ucast_recv_buf_size***
- ***ucast_send_buf_size***
- ***mcast_recv_buf_size***
- ***mcast_send_buf_size***

TCP Stack

- ***recv_buf_size***
- ***send_buf_size***

可以使用管理控制台或管理 **CLI** 来配置 **JGroups** 缓冲区大小。

使用以下语法，通过管理 **CLI** 设置 **JGroups** 缓冲区大小属性：

```
/subsystem=jgroups/stack=STACK_NAME/transport=TRANSPORT/property=PROPERTY_NAME:add(value=BUFFER_SIZE)
```

以下是一个管理 **CLI** 命令示例，可将 **tcp** 堆栈上的 **recv_buf_size** 属性设置为 **20000000** ：

```
/subsystem=jgroups/stack=tcp/transport=TRANSPORT/property=recv_buf_size:add(value=20000000)
```

也可以使用管理控制台，从 **Configuration** 选项卡导航到 **JGroups** 子系统，单击 **View**，选择 **Stack** 选项卡，选择适当的堆栈，再单击传输，编辑 **Properties** 字段，以此配置 **JGroups** 缓冲区大小。

24.2.10. 调整 JGroups 子系统

有关监控 **jgroups** 子系统性能并优化性能的提示，请参阅 性能调优指南中的 **JGroups Subsystem Tuning** 部分。

24.2.11. JGroups Troubleshooting

24.2.11.1. 节点不构建集群

确保您的计算机已针对 **IP** 多播正确设置。**JBoss EAP** 随附了两个测试程序，它们可用于测试 **IP** 多播： **McastReceiverTest** 和 **McastSenderTest**。

在终端中，启动 **McastReceiverTest**。

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastReceiverTest -mcast_addr 230.11.11.11 -port 5555
```

然后，在另一个终端窗口中启动 **McastSenderTest**。

```
$ java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastSenderTest -mcast_addr 230.11.11.11 -port 5555
```

如果要绑定到特定网络接口卡(NIC)，请使用 **-bind_addr YOUR_BIND_ADDRESS**，其中 **YOUR_BIND_ADDRESS** 是您要绑定的 **NIC** 的 **IP** 地址。在发送方和接收方中使用此参数。

在 **McastSenderTest** 终端窗口中键入时，您应当会在 **McastReceiverTest** 窗口中看到输出。如果没有，请尝试以下步骤：

- 通过向 **sender** 命令添加 **-ttl VALUE** 来提高多播数据包的生存时间。此测试程序使用的默认值为 **32**，**VALUE** 不得大于 **255**。
- 如果计算机有多个接口，请验证您是否正在使用正确的接口。
- 联系系统管理员以确保多播能够处理您选择的接口。

且您知道多播在集群中的每一台机器上正常工作，您可以重复上述测试来测试网络，将发送者放在一台机器上，接收方放在另一台机器上。

24.2.11.2. 缺少 Heartbeats 失败检测的原因

有时，群集成员因故障检测(FD)而怀疑，因为某些时间段内未收到心跳确认，**T** 是由 超时 和 **max_tries** 定义的。

例如，节点 **A**、**B**、**C** 和 **D** 的集群，其中 **A ping B**、**B ping C**、**C ping D** 和 **D ping A**，**C** 可能会因为以下原因而怀疑：

- **B** 或 **C** 以 **100% CPU** 运行，超过 **T** 秒。因此，即使 **C** 发送心跳确认到 **B**，**B** 可能无法处理它，因为它处于 **100%** 的 **CPU** 使用率。
- **B** 或 **C** 是垃圾收集方式，导致的情况与上方相同。
- 以上两个案例的组合。
- 网络丢失数据包。网络中有很多流量时通常会出现这种情况，交换机开始丢弃数据包，通常首先广播，然后是 **IP** 多播，最后是 **TCP** 数据包。

- B 或 C 正在处理回调。例如，如果 C 在其通道上收到处理用 T + 1 秒的远程方法调用，此时 C 将不会处理任何其他消息，包括心跳。因此，B 将不会收到心跳确认，并将怀疑 C。

24.3. INFINISPAN

24.3.1. 关于 Infinispan

Infinispan 是 Java 数据网格平台，提供 JSR-107兼容缓存接口，用于管理缓存的数据。Jakarta 同等管理缓存数据是 Jakarta Persistence 2.2。

有关 Infinispan 功能和配置选项的更多信息，请参阅 Infinispan 文档。

infinispan 子系统提供对 JBoss EAP 的缓存支持。它允许您配置和查看指定缓存容器和缓存的运行时指标。

在使用提供高可用性功能的配置时，如受管域中的 ha 或 full-ha 配置文件，或者 standalone-ha.xml 或 standalone-full-ha.xml 配置文件，in finispan 子系统提供缓存、状态复制和状态分布支持。在非高可用性配置中，in finispan 子系统提供本地缓存支持。

重要

Infinispan 在 JBoss EAP 中以私有模块的形式交付，以提供 JBoss EAP 的缓存功能。Infinispan 不支持由应用直接使用。

24.3.2. 缓存容器

缓存容器是子系统使用的缓存存储库。每个缓存容器定义要使用的默认缓存。

JBoss EAP 7 定义以下默认 Infinispan 缓存容器：

- 用于单例缓存 的服务器
- Web 会话集群

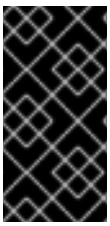
- *用于 有 状态会话 **Bean** 集群的 **EJB***
- *用于实体 缓存的 **Hibernate***

*示例：默认 **Infinispan** 配置*

```
<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
  <cache-container name="server" aliases="singleton cluster" default-cache="default"
module="org.wildfly.clustering.server">
    <transport lock-timeout="60000"/>
    <replicated-cache name="default">
      <transaction mode="BATCH"/>
    </replicated-cache>
  </cache-container>
  <cache-container name="web" default-cache="dist" module="org.wildfly.clustering.web.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="ejb" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
    <transport lock-timeout="60000"/>
    <distributed-cache name="dist">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <file-store/>
    </distributed-cache>
  </cache-container>
  <cache-container name="hibernate" default-cache="local-query" module="org.hibernate.infinispan">
    <transport lock-timeout="60000"/>
    <local-cache name="local-query">
      <object-memory size="1000"/>
      <expiration max-idle="100000"/>
    </local-cache>
    <invalidation-cache name="entity">
      <transaction mode="NON_XA"/>
      <object-memory size="1000"/>
      <expiration max-idle="100000"/>
    </invalidation-cache>
    <replicated-cache name="timestamps" mode="ASYNC"/>
  </cache-container>
</subsystem>
```

注意每个缓存容器中定义的默认缓存。例如，**Web** 缓存容器将 **dist** 分布式缓存定义为默认值。因此，集群 **Web** 会话时将使用 **dist** 缓存。

如需有关更改默认缓存并添加额外缓存的信息，请参阅配置缓存容器。



重要

您可以添加额外的缓存和缓存容器，例如用于 **HTTP** 会话、有状态会话 **Bean** 或单例服务或部署。不支持直接由用户应用使用这些缓存。

24.3.2.1. 配置缓存容器

可以使用管理控制台或管理 **CLI** 配置缓存容器和缓存属性。

警告

您应该避免更改缓存或缓存容器名称，因为配置中的其他组件可能会引用它们。

使用管理控制台配置缓存

从管理控制台中的 **Configuration** 选项卡导航到 **Infinispan** 子系统后，您可以配置缓存和缓存容器。在受管域中，确保选择要配置的适当配置文件。

- 添加缓存容器。

  单击 **Cache Container** 标题旁边的 **Add(+)**按钮，选择 **Add Cache Container**，然后输入新缓存容器的设置。

- 更新缓存容器设置。

  选择适当的缓存容器，再单击 **View**。根据需要配置缓存容器设置。

- 更新缓存容器传输设置。

  选择适当的缓存容器，再单击 **View**。选择传输选项卡，并根据需要配置缓存容器传输设置。

- 配置缓存.

  选择适当的缓存容器，再单击 **View**。例如，从相应的缓存选项卡中，您可以添加、更新和删除缓存。

使用管理 **CLI** 配置缓存

您可以使用管理 **CLI** 配置缓存和缓存容器。在受管域中，您必须通过 **/profile=PROFILE_NAME** 在前面指定要更新的配置集。

- 添加缓存容器。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:add
```

- 添加复制的缓存.

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-cache=CACHE:add(mode=MODE)
```

- 设置缓存容器的默认缓存。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-attribute(name=default-cache,value=CACHE)
```

- 为复制的缓存配置批处理。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/replicated-cache=CACHE/component=transaction:write-attribute(name=mode,value=BATCH)
```

以下示例演示了如何将 并发 分布式缓存添加到 **Web** 缓存容器中。此缓存配置可以减轻默认缓存的锁定限制，允许多个并发请求同时访问同一 **Web** 会话。它允许无锁定读和获得专用锁定的频率更高，但时间较长。

使用以下管理 **CLI** 命令，将 并发 分布式缓存添加到 **web** 缓存容器中，并使其成为默认缓存：

```
batch
/subsystem=infinispan/cache-container=web/distributed-cache=concurrent:add
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=concurrent)
/subsystem=infinispan/cache-container=web/distributed-cache=concurrent/store=file:add
run-batch
```

这会导致以下服务器配置：

```
<cache-container name="web" default-cache="concurrent"
module="org.wildfly.clustering.web.infinispan">
 ...
 <distributed-cache name="concurrent">
    <file-store/>
 </distributed-cache>
</cache-container>
```

更改默认 **EJB** 缓存容器

您可以在 **ejb3** 子系统中使用缓存容器，如下所述：

- 若要支持 **EJB** 会话 **Bean** 的传递，您可以使用 **infinispan** 子系统中定义的 **ejb** 缓存容器来存储会话。
- 对于连接到服务器上集群部署的远程 **EJB** 客户端，您必须向这些客户端提供集群拓扑信息，以便在它们与节点交互失败时能够切换到集群中的其他节点。

如果要更改或重命名名为 **ejb** 的默认缓存容器（支持传递和调配拓扑信息），您必须将 **cache-container** 属性添加到 **passivation-stores** 元素中，并将 **cluster** 属性添加到 远程 元素中，如下例所示：如果您只是添加新的缓存供您自己的使用，则不需要进行这些更改。

```
<subsystem xmlns="urn:jboss:domain:ejb3:5.0">
   <passivation-stores>
      <passivation-store name="infinispan" cache-container="ejb-cltest" max-size="10000"/>
   </passivation-stores>

   <remote cluster="ejb-cltest" connector-ref="http-remoting-connector" thread-pool-name="default"/>
</subsystem>

<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
   ...
```

```
    <cache-container name="ejb-cltest" aliases="sfsb" default-cache="dist"
module="org.wildfly.clustering.ejb.infinispan">
</subsystem>
```

Hibernate 缓存容器中的驱除功能

hibernate 缓存容器的驱除功能会从内存中移除缓存条目。此功能有助于减少子系统的内存负载。

size 属性设置在缓存条目驱除开始前存储的最大缓存条目数。

示例：驱除功能

```
<cache-container name="hibernate" default-cache="local-query" module="org.hibernate.infinispan">
  <transport lock-timeout="60000"/>
  <local-cache name="local-query">
    <object-memory size="1000"/>
    <expiration max-idle="100000"/>
```

请注意，驱除只在内存中发生。缓存存储包含被驱除的缓存条目，以防止永久丢失信息。有关驱除功能的更多信息，请参阅 **Infinispan** 用户指南中的驱除和数据容器部分。

24.3.3. 集群模式

可以使用 **Infinispan** 以两种不同的方式在 **JBoss EAP** 中配置集群。应用程序的最佳方法将取决于您的要求。每种模式的可用性、一致性、可靠性和可扩展性之间有一个权衡。在选择群集模式之前，您必须确定网络最重要的功能，并平衡这些要求。

缓存模式

复制

复制模式会自动检测并添加新实例。对这些实例所做的更改将复制到群集上的所有节点。复制模式通常最适合在小型集群中工作，因为需要通过网络复制大量信息。**Infinispan** 可以配置为使用 **UDP** 多播，从而将网络流量拥塞减少到某种程度。

distribution

分发模式允许 **Infinispan** 线性扩展群集。分发模式使用一致的哈希算法来确定新节点的放置位置。要保留的信息的副本数或所有者是可配置的。在保留的副本数、数据的持久性和性能之间有一个

权衡。所保留的副本越多，对性能的影响越高，但在服务器故障中丢失数据的可能性就越少。哈希算法也致力于通过查找不多播或存储元数据的条目来减少网络流量。

当集群大小超过 **6-8** 节点时，您应该考虑将分发模式视为缓存策略。使用分发模式时，数据仅分发到集群中的一个节点子集，而不是所有节点。

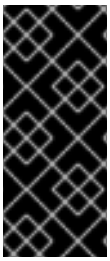
分散

分散模式类似于分发模式，因为它使用一致的哈希算法来确定所有权。但是，所有权仅限于两个成员，或原始器或接收给定会话请求的节点，始终假定所有权用于协调锁定和缓存条目更新。以分散模式使用的缓存写入算法确保写入操作仅产生单个 **RPC** 调用，其中具有两个所有者的分发缓存通常可以使用两个 **RPC** 调用。这对分布式 **Web** 会话非常有用，因为负载平衡器故障转移往往会将流量定向到非主要所有者，甚至备份节点。这可以在集群拓扑更改后减少争用并提高性能。

分散模式不支持事务或 **L1** 缓存。但是，它确实支持错误的读取，允许发起给定条目的缓存写入的节点继续在某些持续时间内为该条目执行读取服务，即使它不是根据一致的哈希的所有者。其效果类似于 **L1** 缓存，尽管冲突读取的配置属性和 **L1** 缓存的配置属性不同。

同步和异步复制

复制可以采用同步或异步模式执行，所选的模式则取决于您的要求和应用。



重要

从 **JBoss EAP 7.1** 开始，您必须始终使用同步**(SYNC)**缓存模式。**SYNC** 也是默认的缓存模式。使用异步**(ASYNC)**模式无效，从而导致锁定争用。如需有关会话复制和适当缓存模式的更多信息，请参阅如何为 **EAP** 配置和调优会话复制。

同步复制

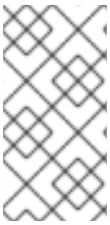
通过同步复制，复制过程在处理用户请求的同一线程中运行。会话复制在完成响应后开始，线程只有在复制完成后才会发布。同步复制对网络流量有影响，因为它需要来自集群中的每个节点的响应。但是，其优点是确保对集群中的所有节点都进行了所有修改。

异步复制

通过异步复制，**Infinispan** 使用线程池在后台进行复制。发送方不等待来自集群中其他节点的回复。但是，在上一个复制完成之前，同一会话的缓存读取将阻止，以便不读取陈旧数据。复制会根据时间或队列大小触发。失败的复制尝试写入日志，而不是实时通知。

24.3.3.1. 配置缓存模式

您可以使用管理 **CLI** 更改默认缓存。



注意

本节显示与配置 **Web** 会话缓存相关的说明，它默认为分发模式。可以轻松调整步骤和管理 **CLI** 命令，以应用到其他缓存容器。

进入 **Replication Cache** 模式

Web 会话缓存的默认 **JBoss EAP 7** 配置不包括 复制 缓存。必须先添加此缓存。

注意

以下管理 **CLI** 命令用于单机服务器。在受管域中运行时，您必须通过 **/profile =infinispan** 命令并使用 **/profile= PROFILE_NAME** 指定要更新的配置集。

1. 添加 复制 缓存并将其设置为默认缓存。

```
batch
/subsystem=infinispan/cache-container=web/replicated-cache=repl:add(mode=ASYNC)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=transaction:add(mode=BATCH)
/subsystem=infinispan/cache-container=web/replicated-
cache=repl/component=locking:add(isolation=REPEATABLE_READ)
/subsystem=infinispan/cache-container=web/replicated-cache=repl/store=file:add
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=repl)
run-batch
```

2. 重新加载服务器：

```
reload
```

进入发布缓存模式

Web 会话缓存的默认 **JBoss EAP 7** 配置已包含 离散分发 缓存。

注意

*以下管理 **CLI** 命令用于单机服务器。在受管域中运行时，您必须通过 **/profile =infinispan** 命令并使用 **/profile= PROFILE_NAME** 指定要更新的配置集。*

1. *将默认缓存更改为 **dist** 分发缓存。*

```
/subsystem=infinispan/cache-container=web:write-attribute(name=default-cache,value=dist)
```

2. *设置分发缓存的所有者数量。以下命令设置 **5** 所有者：默认值为 **2**。*

```
/subsystem=infinispan/cache-container=web/distributed-cache=dist:write-
attribute(name=owners,value=5)
```

3. *重新加载服务器：*

```
reload
```

*进入 **Scattered Cache** 模式*

***Web** 会话缓存的默认 **JBoss EAP** 配置不包括 分布式缓存。以下示例显示了管理 **CLI** 命令，以添加分散的缓存并将其设置为默认缓存。*

注意

*以下管理 **CLI** 命令适用于使用 **HA** 配置文件的单机服务器：在受管域中运行时，您必须通过 **/profile =infinispan** 命令并使用 **/profile= PROFILE_NAME** 指定要更新的配置集。*

1. *创建分散的缓存，读取寿命等于默认的 **web** 会话超时值 **30** 分钟。*

```
/subsystem=infinispan/cache-container=web/scattered-cache=scattered:add(bias-
lifespan=1800000)
```

2. *将 分散 设置为默认缓存。*

```
/subsystem=infinispan/cache-container=web:write-attribute(name=default-
cache,value=scattered)
```

这将产生以下服务器配置：

```
<cache-container name="web" default-cache="scattered"
module="org.wildfly.clustering.web.infinispan">
    ...
    <scattered-cache name="scattered" bias-lifespan="1800000"/>
    ...
</cache-container>
```

24.3.3.2. 缓存策略性能

使用 **SYNC** 缓存策略时，复制的成本很容易在响应时间进行测量和直接可见，因为请求在复制完成后才会完成。

尽管 **ASYNC** 缓存策略的响应时间应该比 **SYNC** 缓存策略要低，但这只在适当的条件下才会发生。**ASYNC** 缓存策略更难测量，但当请求之间的持续时间足够长时，它可以提供更好的性能，让缓存操作能够完成。这是因为在响应时间没有立即看到复制成本。

如果对同一会话的请求速度过快，则上一个请求的复制成本将转移到后续请求的前面，因为它必须等待上一个请求中的复制完成。对于在收到响应后立即发送后续请求的快速触发请求，**ASY NC** 缓存策略将比 **SYNC** 缓存策略更差。因此，同一会话请求之间有一个阈值，其中 **SYNC** 缓存策略实际上比 **ASYNC** 缓存策略更佳。在现实世界使用情况中，对同一会话的请求通常不会快速连续收到。相反，请求之间通常有几秒钟或以上的时间段。在这种情况下，**ASY NC** 缓存策略是明智的默认策略，提供最快的响应时间。

24.3.4. 状态传输

状态传输既是基本数据网格，也是群集缓存功能。如果不进行状态传输，数据将会丢失，因为节点会添加到集群或从集群中删除。

State transfer 会根据缓存成员资格的变化调整缓存的内部状态。当节点加入或离开集群时、两个或多个群集分区合并或这些事件组合后，会自动进行这一更改。新启动缓存的初始状态传输成本最为昂贵，因为新缓存必须根据缓存模式接收最大状态量，如下所述。

timeout 属性可用于控制新启动的缓存等待多久才能接收其状态。如果 **timeout** 属性是一个正数，则缓存将等待接收其所有初始状态，然后才能提供给服务请求。如果状态传输未在指定时间内完成，则默认值为 **240000** 毫秒，缓存将抛出错误并取消启动。如果 超时 设为 **0**，则缓存将立即可用，并在后台操作期间收到初始状态。在初始状态传输完成之前，任何对缓存尚未接收的缓存条目请求都需要从远程节点获取。

可通过以下命令将 **timeout** 属性设为 **0** ：

```
/subsystem=infinispan/cache-container=server/CACHE_TYPE=CACHE/component=state-transfer:write-attribute(name=timeout,value=0)
```

状态传输行为由缓存模式决定。

- 在复制模式中，新节点加入缓存会从现有节点接收整个缓存状态。当节点离开集群时，没有状态转移。
- 在分发模式中，新节点仅从现有节点接收状态的一部分，而现有节点则移除了其中一些状态，以便在缓存中保留每个密钥 的所有者 副本，具体通过一致的散列来确定。当节点离开集群时，分发缓存需要创建该节点上存储的密钥的额外副本，以便每个密钥的所有者继续存在。
- 在无效模式中，初始状态传输与复制模式类似，唯一的区别在于无法保证节点具有相同的状态。当节点离开集群时，没有状态转移。

状态传输默认同时传输内存中和持久状态，但可以在配置中禁用这两者。禁用状态传输时，必须配置 **ClusterLoader**，否则节点将成为密钥的所有者或备份所有者，而无需将数据加载到其缓存中。另外，如果以分布模式禁用状态传输，则密钥偶尔会比缓存中 的所有者 副本少。

24.3.5. 配置 Infinispan 线程池

infinispan 子系统包含 **async-operations**、过期、侦听器、持久性、**remote-command**、**state-transfer** 和 传输 线程池。可以为任何 **Infinispan** 缓存容器配置这些池。

下表列出了您可以为 **infinispan** 子系统中的每个线程池配置的属性和每个线程的默认值：

线程池名称	keepalive-time	max-threads	Min-threads	queue-length
async-operations	60000L	25	25	1000
过期	60000L	1	N/A	N/A
监听程序	60000L	1	1	100000
持久性	60000L	4	1	0

线程池名称	keepalive-time	max-threads	Min-threads	queue-length
remote-command	60000L	200	1	0
state-transfer	60000L	60	1	0
transport	60000L	25	25	100000

使用以下语法，通过管理 **CLI** 配置 **Infinispan** 线程池：

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER_NAME/thread-pool=THREAD_POOL_NAME:write-attribute(name=ATTRIBUTE_NAME, value=ATTRIBUTE_VALUE)
```

以下是管理 **CLI** 命令的示例，可将 服务器 缓存容器的 **persistence** 线程池中的 **max-threads** 值设置为 **10** ：

```
/subsystem=infinispan/cache-container=server/thread-pool=persistence:write-attribute(name="max-threads", value="10")
```

24.3.6. Infinispan Statistics

可以启用有关 **Infinispan** 缓存和缓存容器的运行时统计信息，用于监控目的。出于性能原因，默认情况下不启用统计数据集合。

可以为每个缓存容器、缓存或两者启用统计信息集合。每个缓存的 **statistics** 选项将覆盖缓存容器的选项。为缓存容器启用或禁用统计集合将导致该容器中的所有缓存都继承该设置，除非它们明确指定自己的设置。

24.3.6.1. 启用 Infinispan Statistics

警告

启用 **Infinispan** 统计数据可能会对 **infinispan** 子系统的性能造成负面影响。只有在需要时才应启用统计数据。

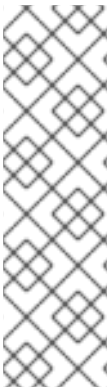
您可以使用管理控制台或管理 **CLI** 来启用或禁用 **Infinispan** 统计数据的集合。从管理控制台，从 **Configuration** 选项卡导航到 **Infinispan** 子系统，选择相应的缓存或缓存容器，然后编辑 **Statistics Enabled** 属性。使用下列命令，通过管理 **CLI** 启用统计信息：

为缓存容器启用统计信息集合。需要重新加载服务器。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-attribute(name=statistics-enabled,value=true)
```

为缓存启用统计信息集合。需要重新加载服务器。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-attribute(name=statistics-enabled,value=true)
```



注意

您可以使用以下命令取消定义缓存的启用了 **statistics-enabled** 属性，以便它继承缓存容器的 **statistics-enabled** 属性的设置。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-attribute(name=statistics-enabled)
```

24.3.7. Infinispan 分区处理

Infinispan 集群构建自存储数据的多个节点。为防止在多个节点失败时数据丢失，**Infinispan** 会在多个节点上复制相同的数据。这种级别的数据冗余使用 **owners** 属性来配置。只要少于配置的节点数同时崩溃，**Infinispan** 将具有可用数据的副本。

但是，当集群中太多节点消失时，可能会发生一些灾难性情况：

脑裂

这会将集群分成两个或者多个分区（独立运行）的子集群。在这些情况下，多个客户端从不同分区进行读写时看到同一缓存条目的不同版本，在许多应用程序中存在问题。

注意

有办法减少脑裂发生的可能性，如冗余网络或 IP 绑定 ；但是，这只缩短了问题的发生时间。

多个节点按顺序崩溃

如果多个节点（特别是所有者数量）崩溃，且 **Infinispan** 没有时间正确地在崩溃间重新平衡其状态，则结果为部分数据丢失。

目标是避免因为脑裂或多个节点快速崩溃而导致数据返回给用户的情况。

24.3.7.1. split Brain

在脑裂情况下，每个网络分区都会安装自己的 **JGroups** 视图，从其他分区中删除节点。我们无法直接确定集群是否已分割为两个或者多个分区，因为这些分区相互不知道。相反，我们假设群集会在 **JGroups** 群集中的一个或多个节点消失而未发送显式离开消息时进行拆分。

禁用分区处理后，每个这样的分区将继续作为独立集群运行。每个分区可能只看到数据的某一部分，每个分区可能会在缓存中写入冲突的更新。

启用分区处理时，如果检测到分割，每个分区不会立即启动重新平衡，而是首先检查它是否应该进入降级模式：

- 如果至少有一个部分丢失了其所有所有者，这意味着自上一次重新平衡结束以来，至少指定的所有者数量已经保留，则分区将进入降级模式。
- 如果分区没有在最新的稳定拓扑中包含简单大多数节点 （**floor(numNodes/2)+ 1**），则分区也会进入降级模式。
- 否则，分区会保持正常运行，并启动重新平衡。

每次重新平衡操作结束时都会更新稳定拓扑，协调者决定不需要再进行重新平衡。这些规则可确保，最多一个分区保持可用模式，其他分区进入降级模式。

当分区处于降级模式时，它只允许访问完全拥有的密钥：

- 此分区的节点上具有所有副本的条目的请求（读取和写入）将被满足。
- 对部分或完全归已消失的节点拥有的条目的请求将通过 **AvailabilityException** 拒绝。

这可保证分区无法为同一密钥写入不同的值（缓存是一致的），并且一个分区无法读取已在其他分区中更新的密钥（无过时数据）。

注意

两个分区可以启动隔离，只要它们不合并，就可以读取和写入不一致的数据。在未来，我们可能允许自定义可用性策略（例如，检查某个节点是否属于群集，或检查是否可以访问外部计算机）来应对这种情况。

24.3.7.2. 配置分区处理

目前分区处理被默认禁用。使用以下管理 **CLI** 命令启用分区处理：

```
/subsystem=infinispan/cache-container=web/distributed-cache=dist/component=partition-handling:write-attribute(name=enabled, value=true)
```

24.3.8. 配置远程缓存容器

您必须为受管域中的每一服务器组配置远程缓存。您可以使用启用了 **statistics** 的属性为给定 **remote-cache-container** 和关联的运行时缓存 启用 指标集合。

24.3.8.1. 创建远程缓存容器

受管域中的每一服务器组需要一个唯一的远程缓存。缓存可属于同一数据网格。因此，用户必须通过为服务器组定义套接字绑定并将套接字绑定与远程缓存容器关联，为每个服务器组配置远程缓存。

流程

1. 定义一个 套接字绑定，根据需要在集群中的每个远程红帽数据网格实例重复 命令。

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=SOCKET_BINDING:add(host=HOSTNAME,port=PORT)
```

2. 定义一个 **remote-cache-container**，以引用新创建的套接字绑定。

```
batch
/subsystem=infinispan/remote-cache-container=CACHE_CONTAINER:add(default-remote-cluster=data-grid-cluster)
/subsystem=infinispan/remote-cache-container=CACHE_CONTAINER/remote-cluster=data-grid-cluster:add(socket-bindings=[SOCKET_BINDING,SOCKET_BINDING_2,...])
run-batch
```

24.3.8.2. 为远程缓存容器启用统计信息

启用 **statistics-enabled** 属性可为给定的 **remote-cache-container** 和关联的运行时缓存启用指标集合。

- 对于名为**"foo"**的 **remote-cache-container**，请使用以下操作启用统计信息：

```
/subsystem=infinispan/remote-cache-container=foo:write-attribute(name=statistics-enabled, value=true)
```

- 对于 **remote-cache-container "foo"**，在运行时可以看到以下指标：

```
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=connections)
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=active-connections)
/subsystem=infinispan/remote-cache-container=foo:read-attribute(name=idle-connections)
```

- 对于这些指标的描述，请对 **remote-cache-container** 执行 **read-resource- description** 操作：

```
/subsystem=infinispan/remote-cache-container=foo:read-resource-description
```

- 以下指标特定于所选部署使用的远程缓存：

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:read-resource(include-runtime=true, recursive=true)
{
    "average-read-time" : 1,
    "average-remove-time" : 0,
    "average-write-time" : 2,
```

```
    "hits" : 9,
    "misses" : 0,
    "near-cache-hits" : 7,
    "near-cache-invalidations" : 8,
    "near-cache-misses" : 9,
    "near-cache-size" : 1,
    "removes" : 0,
    "time-since-reset" : 82344,
    "writes" : 8
}
```

- 如需这些指标的描述，请对远程缓存执行 **read-resource-description** 操作：

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:read-resource-description
```

- 其中一些指标是计算的值（如 **average-***），另一些则被截断，如命中和未命中。通过以下操作可以重置 **tallied** 指标：

```
/subsystem=infinispan/remote-cache-container=foo/remote-cache=bar.war:reset-statistics()
```

24.3.9. 将 HTTP 会话外部化到红帽数据网格

注意

您需要红帽数据网格订阅才能使用此功能。

红帽数据网格可用作 **JBoss EAP** 中应用特定数据的外部缓存容器，如 **HTTP** 会话。这允许扩展独立于应用的数据层，并使可能驻留于不同域中的不同 **JBoss EAP** 集群能够访问同一红帽数据网格集群的数据。此外，其他应用程序还可与红帽数据网格提供的缓存相连接。

下例演示了如何使 **HTTP** 会话外部化。它适用于 **JBoss EAP** 和受管域的单机实例。

1. 创建 **remote-cache-container**。如需更多信息，请参阅配置远程缓存容器。

2. 配置 **HotRod** 存储.**HotRod** 存储将专用远程缓存用于 **JBoss EAP** 服务器创建的每个缓存。通常，**JBoss EAP** 服务器上使用一个无效缓存，如下 **CLI** 脚本中所示。

注意

*远程缓存需要在红帽数据网格服务器上手动配置。建议对这些缓存进行配置，是带有保守锁定的事务分发模式缓存。缓存名称必须与 **test.war** 等部署文件名对应。*

*配置了远程缓存容器后，可以配置热门 存储 来取代任何现有的存储。以下 **CLI** 脚本演示了与无效缓存结合使用会话的典型用例：*

```
batch
/subsystem=infinispan/cache-container=web/invalidation-cache=CACHE_NAME:add()
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/store=hotrod:add(remote-cache-
container=CACHE_CONTAINER,fetch-
state=false,purge=false,passivation=false,shared=true)
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/component=transaction:add(mode=BATCH)
/subsystem=infinispan/cache-container=web/invalidation-
cache=CACHE_NAME/component=locking:add(isolation=REPEATABLE_READ)
/subsystem=infinispan/cache-container=web:write-attribute(name=default-
cache,value=CACHE_NAME)
run-batch
```

该脚本配置一个新的无效缓存。然后，会话数据被保留在缓存中以获得性能并写入存储以获得弹性。

*可以使用 **@Resource** 注释将 **HotRod** 客户端直接注入到 **Jakarta EE** 应用中。在下例中，@ **Resource** 注释在 **hotrod-client.properties** 文件中查找类路径中的配置属性。*

```
@Resource(lookup = "java:jboss/infinispan/remote-container/web-sessions")
private org.infinispan.client.hotrod.RemoteCacheContainer client;
```

*示例： **hotrod-client.properties** 文件*

```
infinispan.client.hotrod.transport_factory =
org.infinispan.client.hotrod.impl.transport.tcp.TcpTransportFactory
infinispan.client.hotrod.server_list = 127.0.0.1:11222
infinispan.client.hotrod.marshaller =
org.infinispan.commons.marshall.jboss.GenericJBossMarshaller
infinispan.client.hotrod.async_executor_factory =
org.infinispan.client.hotrod.impl.async.DefaultAsyncExecutorFactory
infinispan.client.hotrod.default_executor_factory.pool_size = 1
infinispan.client.hotrod.default_executor_factory.queue_size = 10000
infinispan.client.hotrod.hash_function_impl.1 =
```

```
org.infinispan.client.hotrod.impl.consistenthash.ConsistentHashV1
infinispan.client.hotrod.tcp_no_delay = true
infinispan.client.hotrod.ping_on_startup = true
infinispan.client.hotrod.request_balancing_strategy =
org.infinispan.client.hotrod.impl.transport.tcp.RoundRobinBalancingStrategy
infinispan.client.hotrod.key_size_estimate = 64
infinispan.client.hotrod.value_size_estimate = 512
infinispan.client.hotrod.force_return_values = false

## below is connection pooling config

maxActive=-1
maxTotal = -1
maxIdle = -1
whenExhaustedAction = 1
timeBetweenEvictionRunsMillis=120000
minEvictableIdleTimeMillis=300000
testWhileIdle = true
minIdle = 1
```

保护远程缓存容器

*可以使用 **SSL** 保护与远程红帽数据网格实例的通信。这可以通过在 **JBoss EAP** 实例上配置 **remote-cache-container** 并在红帽数据网格实例上调整 **hotrod** 连接器以使用活动的安全域来完成。*

1. *在 **JBoss EAP** 中创建 客户端**-ssl-context**。有关创建 客户端**-ssl-context** 的更多信息，包括生成其他 **elytron** 组件，请参阅 如何为 **JBoss EAP** 配置服务器 安全性中的 客户端**-ssl-context**。*

```
/subsystem=elytron/client-ssl-context=CLIENT_SSL_CONTEXT:add(key-
manager=KEY_MANAGER,trust-manager=TRUST_MANAGER)
```

2. *配置远程缓存容器，以使用客户端 **SSL** 上下文。*

```
/subsystem=infinispan/remote-cache-
container=CACHE_CONTAINER/component=security:write-attribute(name=ssl-
context,value=CLIENT_SSL_CONTEXT)
```

3. *保护远程红帽数据网格实例，并为每个实例重复此操作。*

   a. *将 **client-ssl-context** 中使用的密钥存储复制到远程红帽数据网格实例。*

b.

配置 **ApplicationRealm** 以使用此密钥存储。

```
/core-service=management/security-realm=ApplicationRealm/server-identity=ssl:add(keystore-path="KEYSTORE_NAME",keystore-relative-to="jboss.server.config.dir",keystore-password="KEYSTORE_PASSWORD")
```

c.

调整热连接器以指向此安全域。

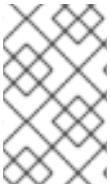
```
/subsystem=datagrid-infinispan-endpoint/hotrod-connector=hotrod-connector/encryption=ENCRYPTION:add(require-ssl-client-auth=false,security-realm="ApplicationRealm")
```

d.

重新载入远程红帽数据网格实例。

```
reload
```

24.3.10. 使用远程存储外部化到红帽数据网格的 HTTP 会话



注意

您需要红帽数据网格订阅才能使用此功能。

此处的说明代表一种使会话外部化的旧方式。**JBoss EAP 7.2** 引入了基于与 **elytron** 子系统集成的 **HotRod** 协议的自定义优化缓存存储。建议您使用新的 热门存储，如红帽数据网格外部化 **HTTP** 会话 中所述。

注意

必须为每个 **distributable** 应用创建一个全新的缓存。它可以在现有的缓存容器中创建，如 **web**。

使 **HTTP** 会话外部化：

1.

通过添加网络信息到 **socket-binding-group**，定义远程红帽数据网格服务器的位置。

示例：添加远程套接字绑定

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-rhdg-server1:add(host=RHDGHostName1, port=11222)

/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-rhdg-server2:add(host=RHDGHostName2, port=11222)
```

*生成 **XML***

```
<socket-binding-group name="standard-sockets" ... >
  ...
  <outbound-socket-binding name="remote-rhdg-server1">
    <remote-destination host="RHDGHostName1" port="11222"/>
  </outbound-socket-binding>
  <outbound-socket-binding name="remote-rhdg-server2">
    <remote-destination host="RHDGHostName2" port="11222"/>
  </outbound-socket-binding>
</socket-binding-group>
```

注意

您需要为每个红帽数据网格服务器配置远程套接字绑定。

2. *确保在 **JBoss EAP** 的 **infinispan** 子系统中定义了远程缓存容器；在位于 **remote-store** 元素的 **cache** 属性的示例中，定义远程红帽数据网格服务器上的缓存名称。*

*如果您在受管域中运行，请在这些命令之前使用 **/profile=PROFILE_NAME**。*

示例：添加远程缓存容器

```
/subsystem=infinispan/cache-container=web/invalidation-cache=rhdg:add(mode=SYNC)

/subsystem=infinispan/cache-container=web/invalidation-cache=rhdg/component=locking:write-attribute(name=isolation,value=REPEATABLE_READ)
```

```
/subsystem=infinispan/cache-container=web/invalidation-
cache=rhdg/component=transaction:write-attribute(name=mode,value=BATCH)

/subsystem=infinispan/cache-container=web/invalidation-
cache=rhdg/store=remote:add(remote-servers=["remote-rhdg-server1","remote-rhdg-
server2"], cache=default, socket-timeout=60000, passivation=false, purge=false,
shared=true)
```

*生成 **XML***

```
<subsystem xmlns="urn:jboss:domain:infinispan:7.0">
  ...
  <cache-container name="web" default-cache="dist"
module="org.wildfly.clustering.web.infinispan" statistics-enabled="true">
    <transport lock-timeout="60000"/>
    <invalidation-cache name="rhdg" mode="SYNC">
      <locking isolation="REPEATABLE_READ"/>
      <transaction mode="BATCH"/>
      <remote-store cache="default" socket-timeout="60000" remote-servers="remote-rhdg-
server1 remote-rhdg-server2" passivation="false" purge="false" shared="true"/>
    </invalidation-cache>
    ...
  </cache-container>
</subsystem>
```

3. *将缓存信息添加到应用 **jboss-web.xml** 文件中。在以下示例中，**web** 是 缓存容器的名称，**rh dg** 是此容器中相应缓存的名称。*

*示例：**jboss-web.xml** 文件*

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web xmlns="http://www.jboss.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-web_10_0.xsd"
           version="10.0">
   <replication-config>
      <replication-granularity>SESSION</replication-granularity>
```

```
        <cache-name>web.rhdg</cache-name>
    </replication-config>
</jboss-web>
```

24.4. 将 JBOSS EAP 配置为前端负载平衡器

您可以配置 JBoss EAP 和 undertow 子系统，充当前端负载平衡器，以代理到后端 JBoss EAP 服务器的请求。由于 Undertow 使用异步 IO，负责连接的 IO 线程是唯一与请求相关的线程。同样的线程也用于连接后端服务器。

您可以使用以下协议：

- 通过纯文本(http)的HTTP，支持 HTTP/1 和 HTTP/2(h2c)
- 通过安全连接(https)的 HTTP，支持 HTTP/1 和 HTTP/2(h2)
- AJP(ajp)

您可以定义静态负载平衡器并在配置中指定后端主机，或者使用 mod_cluster 前端动态更新主机。

有关将 Undertow 配置为使用 HTTP/2 的说明，请参阅配置 HTTP/2。

24.4.1. 使用 mod_cluster 将 Undertow 配置为负载平衡器

您可以使用内置的 mod_cluster 前端负载平衡器来负载平衡其他 JBoss EAP 实例。

此流程假定您在受管域中运行，并且已配置了以下配置：

- 将充当负载平衡器的 JBoss EAP 服务器。
  - 此服务器使用 load-balancer 配置文件，它绑定到 load-balancer-sockets 套接字绑定

组。

注意

load-balancer 配置集已预先配置了套接字绑定 **mod-cluster** **Undertow** 过滤器，并在默认主机中引用过滤器以将此服务器用作前端负载平衡器。

- 两台 **JBoss EAP** 服务器，用作后端服务器。
  - 这些服务器在集群中运行，并使用 **ha** 配置文件，它绑定到 **ha-sockets** 套接字绑定组。
- 将负载平衡部署到后端服务器的 **distributable** 应用。

配置 mod_cluster Front-end Load Balancer

下列步骤对受管域中的服务器进行负载平衡，但可以调整它们以应用到一组单机服务器。务必更新管理 **CLI** 命令值以适合您的环境。

1. 设置 **mod_cluster** 公告安全密钥。

   添加广播安全密钥可让负载平衡器和服务器在发现期间进行身份验证。

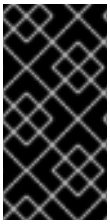
   使用以下管理 **CLI** 命令，设置 **mod_cluster** 公告安全密钥：

   ```
   /profile=ha/subsystem=modcluster/proxy=default:write-attribute(name=advertise-security-key,
   value=mypassword)
   ```

2. 更新 **mod_cluster** 负载平衡器的安全密钥。

   使用以下管理 **CLI** 命令，为 **mod_cluster** 过滤器设置安全密钥：

   ```
   /profile=load-balancer/subsystem=undertow/configuration=filter/mod-cluster=load-
   balancer:write-attribute(name=security-key,value=mypassword)
   ```



重要

*建议由 **mod_cluster** 使用的管理和公告套接字绑定仅公开给内部网络，而非公共 **IP** 地址。*

*负载平衡器 **JBoss EAP** 服务器现在可以对两个后端 **JBoss EAP** 服务器进行负载平衡。*

*多个 **mod_cluster** 配置*

***mod_cluster** 子系统支持多个命名代理配置，允许使用反向代理在非默认 下注册服务器。此外，这允许单个应用服务器节点注册到不同的代理服务器组。*

*以下示例将 **ajp-listener**、**server** 和主机添加到 **undertow** 服务器。它还添加新的 **mod_cluster** 配置，以使用广播机制注册主机。*

```
/socket-binding-group=standard-sockets/socket-binding=ajp-other:add(port=8010)
/subsystem=undertow/server=other-server:add
/subsystem=undertow/server=other-server/ajp-listener=ajp-other:add(socket-binding=ajp-other)
/subsystem=undertow/server=other-server/host=other-host:add(default-web-module=root-other.war)
/subsystem=undertow/server=other-server/host=other-host
/location=other:add(handler=welcome-content)
/subsystem=undertow/server=other-server/host=other-host:write-attribute(name=alias,value=
[localhost]))

/socket-binding-group=standard-sockets/socket-binding=modcluster-other:add(multicast-
address=224.0.1.106,multicast-port=23364)
/subsystem=modcluster/proxy=other:add(advertise-socket=modcluster-other,balancer=other-
balancer,connector=ajp-other)

reload
```

24.4.2. 在负载均衡器中启用等级的会话关联

*您必须在负载平衡器中启用等级会话关联，才能在 **distribut-web** 子系统中具有多个排序的路由。有关 **distributable-web** 子系统和不同关联性选项的更多信息，请参阅 **JBoss EAP** 开发指南中 适用于分布式 **Web** 会话配置的 **distribut-web** 子系统。*

*分隔节点路由的默认分隔符为： 如果要使用其他值，您可以配置 **affinity** 资源的 分隔符 属性。*

流程

1. *为负载均衡器启用排序的会话关联：*

```
/subsystem=undertow/configuration=filter/mod-cluster=load-balancer/affinity=ranked:add
```

2. 可选：配置 关联性 资源的 分隔符 属性：

```
/subsystem=undertow/configuration=filter/mod-cluster=load-balancer/affinity=ranked:write-attribute(name=delimiter,value=':')
```

24.4.3. 将 Undertow 配置为静态负载平衡器

若要使用 **Undertow** 配置静态负载平衡器，您需要在 **undertow** 子系统中配置代理处理程序。要在 **Undertow** 中配置代理处理程序，您需要对将充当静态负载平衡器的 **JBoss EAP** 实例进行以下操作：

1. 添加反向代理处理程序
2. 定义每个远程主机的出站套接字绑定
3. 将每个远程主机添加到反向代理处理程序
4. 添加反向代理位置

下例演示了如何将 **JBoss EAP** 实例配置为静态负载平衡器。**JBoss EAP** 实例位于 **lb.example.com**，并在另外两个服务器之间负载平衡：**server 1.example.com** 和 **server2.example.com**。负载均衡器将反向代理到位置 **/app**，并将使用 **AJP** 协议。

1. 添加反向代理处理程序：

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-handler:add
```

2. 为每个远程主机定义出站套接字绑定：

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-host1/:add(host=server1.example.com, port=8009)

/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=remote-host2/:add(host=server2.example.com, port=8009)
```

3. 将每个远程主机添加到反向代理处理程序：

```
/subsystem=undertow/configuration=handler/reverse-proxy=my-handler/host=host1:add(outbound-socket-binding=remote-host1, scheme=ajp, instance-id=myroute1, path=/test)

/subsystem=undertow/configuration=handler/reverse-proxy=my-handler/host=host2:add(outbound-socket-binding=remote-host2, scheme=ajp, instance-id=myroute2, path=/test)
```

4. 添加反向代理位置：

```
/subsystem=undertow/server=default-server/host=default-host/location=\/test:add(handler=my-handler)
```

访问 **lb.example.com:8080/app** 时，您现在会看到从 **server1.example.com** 和 **server2.example.com** 进行代理的内容。

24.5. 使用外部 WEB 服务器作为代理服务器

JBoss EAP 可以使用支持的 **HTTP**、**HTTPS** 或 **AJP** 协议接受来自外部 **Web** 服务器的请求，具体取决于外部 **Web** 服务器配置。

如需有关每个 **web** 服务器支持的 **HTTP** 连接器的详细信息，请参阅 **HTTP Connectors** 概述。且决定使用哪个 **Web** 服务器和 **HTTP** 连接器，请参阅有关配置连接器的相应部分：

- 请参见 **Apache HTTP** 服务器的 **mod_cluster ???**、**mod_jk** 或 **mod_proxy** 部分。
- 请参阅 **Microsoft IIS** 的 **ISAPI** 连接器部分。
- 请参阅 **Oracle iPlanet Web** 服务器的 **NSAPI** 连接器部分。

有关 **HTTP** 连接器受支持配置的最新信息，请参阅 **JBoss EAP** 支持的配置。

您还需要确保将 **JBoss EAP** 配置为接受来自外部 **Web** 服务器的请求。

24.5.1. HTTP 连接器概述

JBoss EAP 能够使用内置于外部 Web 服务器的负载平衡和群集机制，如 Apache HTTP 服务器、Microsoft IIS 和 Oracle iPlanet 以及通过 Undertow。JBoss EAP 使用连接器与 Web 服务器通信。这些连接器在 JBoss EAP 的 undertow 子系统内配置。

Web 服务器包含软件模块，这些模块控制 HTTP 请求路由到 JBoss EAP 节点的方式。每个模块的运行方式和配置方式都有所不同。模块配置为在多个 JBoss EAP 节点之间平衡工作负载，从而在出现故障时或同时将工作负载迁移到备用服务器。

JBoss EAP 支持多种不同的连接器：您选择的某个服务器取决于正在使用的 Web 服务器以及您需要的功能。下表中列出了与 JBoss EAP 兼容的各种 HTTP 连接器所支持配置和功能的比较。

注意

如需将 JBoss EAP 7 用作多平台负载平衡器，请参阅使用 mod_cluster 将 Undertow 配置为负载平衡器。

有关 HTTP 连接器受支持配置的最新信息，请参阅 JBoss EAP 支持的配置。

表 24.1. HTTP 连接器支持的配置

连接器	Web 服务器	支持的操作系统	支持的协议
mod_cluster	红帽 JBoss 核心服务 Apache HTTP 服务器, 红帽 JBoss Web 服务器 Apache HTTP 服务器, JBoss EAP(Undertow)	红帽企业 Linux、Microsoft Windows Server、Oracle Solaris	HTTP、HTTPS、AJP、WebSocket
mod_jk	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	红帽企业 Linux、Microsoft Windows Server、Oracle Solaris	AJP
mod_proxy	Red Hat JBoss Core Services Apache HTTP Server, Red Hat JBoss Web Server Apache HTTP Server	红帽企业 Linux、Microsoft Windows Server、Oracle Solaris	HTTP、HTTPS、AJP
ISAPI 连接器	Microsoft IIS	Microsoft Windows Server	AJP
NSAPI 连接器	Oracle iPlanet Web 服务器	Oracle Solaris	AJP

表 24.2. HTTP 连接器功能

连接器	支持粘滞会话	适应部署状态
mod_cluster	是	可以。检测应用的部署和取消部署，并根据应用是否在该服务器上部署，动态决定是否将客户端请求定向到服务器。
mod_jk	是	否。只要容器可用，无论应用状态如何，直接向容器发出客户端请求。
mod_proxy	是	否。只要容器可用，无论应用状态如何，直接向容器发出客户端请求。
ISAPI 连接器	是	否。只要容器可用，无论应用状态如何，直接向容器发出客户端请求。
NSAPI 连接器	是	否。只要容器可用，无论应用状态如何，直接向容器发出客户端请求。

24.5.2. Apache HTTP 服务器

现在，可以使用 **Red Hat JBoss Core Services** 单独下载 **Apache HTTP Server** 捆绑包。这简化了安装和配置，并实现了更加一致的更新体验。

24.5.2.1. 安装 Apache HTTP 服务器

有关安装 **Apache HTTP** 服务器的详情请参考 **JBoss** 核心服务**Apache HTTP** 服务器安装指南。

24.5.3. 接受来自外部 Web 服务器的请求

只要配置了正确的协议处理程序，如 **AJP**、**HTTP** 或 **HTTPS**，**JBoss EAP** 不要求任何特殊配置开始接受来自代理服务器的请求。

如果代理服务器使用 **mod_jk**、**mod_proxy**、**ISAPI** 或 **NSAPI**，它将请求发送到 **JBoss EAP** 和 **JBoss EAP** 只需提供响应。使用 **mod_cluster**，您还必须配置网络，以允许 **JBoss EAP** 向其当前负载、应用程序生命周期事件和健康状态等信息发送信息，以帮助它确定路由请求的位置。有关配置 **mod_cluster** 代理服务器的详情，请参考 **mod_cluster HTTP** 连接器。

更新 **JBoss EAP** 配置

在以下步骤中，将示例中的协议和端口替换为您需要配置的协议和端口。

1. 配置 **Undertow** 的 **instance-id** 属性。

外部 **Web** 服务器使用 **instance-id** 在其连接器配置中标识 **JBoss EAP** 实例。使用以下管理 **CLI** 命令，在 **Undertow** 中设置 **instance-id** 属性：

```
/subsystem=undertow:write-attribute(name=instance-id,value=node1)
```

在上例中，外部 **Web** 服务器将当前的 **JBoss EAP** 实例识别为 **node1**。

2. 将所需的监听程序添加到 **Undertow**：

为了使外部 **Web** 服务器能够连接 **JBoss EAP**，**Undertow** 需要侦听器。每一协议需要自己的侦听器，它们绑定到套接字绑定。

注意

根据您所需的协议和端口配置，此步骤可能并不是必需的。在所有默认 **JBoss EAP** 配置中配置 **HTTP** 侦听器，如果您使用 **ha** 或 **full-ha** 配置文件，则会配置 **AJP** 侦听器。

您可以通过读取默认服务器配置来检查是否已配置了所需的监听程序：

```
/subsystem=undertow/server=default-server:read-resource
```

若要向 **Undertow** 添加侦听器，它必须具有套接字绑定：套接字绑定添加到您的服务器或服务器组使用的套接字绑定组中。以下管理 **CLI** 命令向 标准套接字绑定添加了 **ajp** 套接字 绑定（绑定到端口 **8009**）

```
/socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
```

以下管理 **CLI** 命令使用 **ajp** 套接字绑定向 **Undertow** 添加 **ajp** 侦听器：

```
/subsystem=undertow/server=default-server/ajp-listener=ajp:add(socket-binding=ajp)
```

24.6. MOD_CLUSTER HTTP 连接器

mod_cluster 连接器是基于 Apache HTTP Server 的负载平衡器。它使用通信通道将来自 Apache HTTP 服务器的请求转发到一组应用服务器节点。

mod_cluster 连接器与其他连接器相比具有多个优势。

- mod_cluster 管理协议(MCMP)是 JBoss EAP 服务器和启用了 mod_cluster 模块的 Apache HTTP 服务器之间的额外连接。它供 JBoss EAP 服务器用于通过一组自定义 HTTP 方法将服务器端负载均衡因素和生命周期事件传回到 Apache HTTP 服务器。
- 使用 mod_cluster 的 Apache HTTP Server 的动态配置允许 JBoss EAP 服务器加入负载平衡协议，而无需手动配置。
- JBoss EAP 执行负载平衡因子计算，而不依赖于具有 mod_cluster 的 Apache HTTP 服务器。这使得负载平衡指标比其他连接器更准确。
- mod_cluster 连接器提供精细的应用程序生命周期控制。每个 JBoss EAP 服务器将 Web 应用程序上下文生命周期事件转发到 Apache HTTP 服务器，通知它启动或停止对给定上下文的路由请求。这可以防止最终用户因为不可用资源而看到 HTTP 错误。
- 可以使用 AJP、HTTP 或 HTTPS 传输。

有关 modcluster 子系统特定配置选项的更多详细信息，请参阅 ModCluster Subsystem Attributes。

24.6.1. 在 Apache HTTP 服务器中配置 mod_cluster

安装 JBoss Core Services Apache HTTP 服务器或使用 JBoss Web 服务器时已包含 mod_cluster 模块，默认情况下即可加载。

注意

自 3.1.0 版开始，Apache HTTP 服务器不再通过 JBoss Web 服务器分发。

请参阅以下步骤来配置 mod_cluster 模块以适应您的环境。

注意

红帽客户还可以使用红帽客户门户网站中的负载平衡器配置工具为 **mod_cluster** 和其他连接器快速生成最佳配置模板。请注意，您必须登录才能访问此工具。

配置 **mod_cluster**

Apache HTTP 服务器已包含 **mod_cluster** 配置文件 **mod_cluster.conf**，它加载 **mod_cluster** 模块并提供基本配置。可以配置此文件中的 **IP** 地址、端口和其他设置（如下所示）。

```
# mod_proxy_balancer should be disabled when mod_cluster is used
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_cluster

<IfModule manager_module>
  Listen 6666
  <VirtualHost *:6666>
    <Directory />
      Require ip 127.0.0.1
    </Directory>
    ServerAdvertise on
    EnableMCPMReceive
    <Location /mod_cluster_manager>
      SetHandler mod_cluster-manager
      Require ip 127.0.0.1
    </Location>
  </VirtualHost>
</IfModule>
```

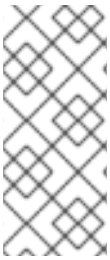
Apache HTTP 服务器配置为负载平衡器，可以用于 **JBoss EAP** 上运行的 **modcluster** 子系统。您必须配置 **mod_cluster** 工作程序节点，使 **JBoss EAP** 知道 **mod_cluster**。

如果要禁用 **mod_cluster** 的广播并配置静态代理列表，请参阅 **mod_cluster** 的 **Disable Advertisement**。有关 **Apache HTTP** 服务器中可用 **mod_cluster** 配置选项的更多信息，请参阅 **Apache HTTP** 服务器 **mod_cluster** 指令

有关配置 **mod_cluster** 的详情，请参阅 **JBoss Web** 服务器 **HTTP** 连接器和负载平衡指南中的配置使用 **Apache HTTP** 服务器的负载平衡和 **mod_cluster** 部分。

24.6.2. 为 mod_cluster 禁用广播

默认情况下，**mod cluster** 子系统的负载平衡器使用多播 **UDP** 来公告其对后台工作线程的可用性。您可以使用以下步骤禁用广播并使用代理列表。



注意

以下流程中的管理 **CLI** 命令假定您在受管域中使用 **full-ha** 配置文件。如果您使用除 **full-ha** 以外的配置文件，请在 命令中使用相应的配置集名称。如果您正在运行单机服务器，请完全删除 **/profile=full-ha**。

1. 修改 **Apache HTTP** 服务器配置。

编辑 **httpd.conf Apache HTTP** 服务器配置文件。使用 **EnableMCPMReceive** 指令，对侦听 **MCPM** 请求的虚拟主机进行以下更新：

a. 添加 指令，以禁用服务器公告。

将 **ServerAdvertise** 指令设置为 **Off**，以禁用服务器公告。

```
ServerAdvertise Off
```

b. 禁用广播频率。

如果您的配置指定了 **AdvertiseFrequency** 参数，请使用 **#** 字符进行注释。

```
# AdvertiseFrequency 5
```

c. 启用接收 **MCPM** 消息的功能。

确保 **EnableMCPMReceive** 指令存在，以允许 **Web** 服务器从工作程序节点接收 **MCPM** 消息。

```
EnableMCPMReceive
```

2. 在 **JBoss EAP modcluster** 子系统中禁用广播。

使用以下管理 **CLI** 命令禁用广播：

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-
attribute(name=advertise,value=false)
```

重要

务必继续下一步，以提供代理列表。如果代理列表为空，则不会禁用广播。

3.

在 **JBoss EAP modcluster** 子系统中提供代理列表。

必须提供代理列表，因为如果禁用了广播，**mod cluster** 子系统将无法自动发现代理。

首先，在适当的套接字绑定组中定义出站套接字绑定。

```
/socket-binding-group=full-ha-sockets/remote-destination-outbound-socket-
binding=proxy1:add(host=10.33.144.3,port=6666)
/socket-binding-group=full-ha-sockets/remote-destination-outbound-socket-
binding=proxy2:add(host=10.33.144.1,port=6666)
```

接下来，将代理添加到 **mod_cluster** 配置中。

```
/profile=full-ha/subsystem=modcluster/proxy=default:list-add(name=proxies,value=proxy1)
/profile=full-ha/subsystem=modcluster/proxy=default:list-add(name=proxies,value=proxy2)
```

Apache HTTP 服务器平衡器不再将其存在性公告给 **worker** 节点，并且不再使用 **UDP** 多播。

24.6.3. 配置 mod_cluster Worker 节点

mod_cluster 工作程序节点由 **JBoss EAP** 服务器组成。此服务器可以是单机服务器，也可以是受管域中服务器组的一部分。单独的进程在 **JBoss EAP** 内运行，后者管理群集的所有工作程序节点。这称为 **master**。

受管域中的工作程序节点在服务器组间共享相同的配置。作为单机服务器运行的 **worker** 节点是单独配置的。否则，配置步骤是相同的。

- 单机服务器必须使用 **standalone-ha** 或 **standalone- full-ha** 配置文件启动。

- 受管域中的服务器组必须使用 **ha** 或 **full-ha** 配置文件，以及 **ha-sockets** 或 **full-ha-sockets** 套接字绑定组。**JBoss EAP** 附带一个支持群集的服务器组，名为 **other-server-group**，它满足这些要求。

配置工作程序节点

此流程中的管理 **CLI** 命令假定您使用带 **full-ha** 配置文件的受管域。如果您正在运行单机服务器，请删除命令的 **/profile=full-ha** 部分。

1. 配置网络接口。

默认情况下，网络接口默认为 **127.0.0.1**。托管单机服务器或服务器组中一个或多个服务器的每个物理主机都需要将其接口配置为使用其公共 **IP** 地址，由其他服务器看到。

根据您的环境，使用以下管理 **CLI** 命令为管理 、 公共 和不安全 接口 修改外部 **IP** 地址。务必将 命令中的 **EXTERNAL_IP_ADDRESS** 替换为主机的实际外部 **IP** 地址。

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:EXTERNAL_IP_ADDRESS}")
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address.public:EXTERNAL_IP_ADDRESS}")
/interface=unsecure:write-attribute(name=inet-
address,value="${jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}")
```

重新加载服务器：

```
reload
```

2. 配置主机名.

为参与受管域的每个主机设置唯一的主机名。此名称必须在从卷之间唯一，并将供从系统标识到集群，因此请记录您使用的名称。

a. 使用适当的 **host.xml** 配置文件启动 **JBoss EAP** 从主机。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

b.

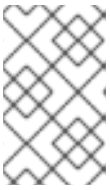
使用以下管理 **CLI** 命令，设置唯一的主机名：此示例使用 **slave1** 作为新主机名。

```
/host=EXISTING_HOST_NAME:write-attribute(name=name,value=slave1)
```

有关配置主机名的更多信息，请参阅配置主机的名称。

3.

配置每一主机以连接域控制器。



注意

此步骤不适用于单机服务器。

对于需要加入受管域的新配置主机，您必须删除本地元素，再添加指向域控制器的 **remote element host** 属性。

a.

使用适当的 **host.xml** 配置文件启动 **JBoss EAP** 从主机。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml
```

b.

使用下列管理 **CLI** 命令，配置域控制器设置：

```
/host=SLAVE_HOST_NAME:write-remote-domain-
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.master.por
t:9990},security-realm="ManagementRealm")
```

这会修改 **host-slave.xml** 文件中的 **XML**，如下所示：

```
<domain-controller>
    <remote host="DOMAIN_CONTROLLER_IP_ADDRESS"
port="${jboss.domain.master.port:9990}" security-realm="ManagementRealm"/>
</domain-controller>
```

如需更多信息，请参阅主机控制器配置。

4.

为每个从主机配置身份验证。

每个从属服务器都需要在域控制器的 **managementRealm** 中创建用户名和密码。在域控制器或单机 **master** 上，为每个主机运行 **EAP_HOME/bin/add-user.sh** 命令。为每个主机添加一个管理用户，用户名与从设备的主机名匹配。

确保对最后一个问题 回答是 **"**这个新用户将用于一个 **AS** 进程连接到另一个 **AS** 进程吗？**"**，以便为您提供一个 **secret** 值。

示例： **add-user.sh Script** 输出 **(trimmed)**

```
$ EAP_HOME/bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Username : slave1
Password : changeme
Re-enter Password : changeme
What groups do you want this user to belong to? (Please enter a comma separated list, or
leave blank for none)[  ]:
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for
server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition <secret
value="SECRET_VALUE" />
```

复制此输出中提供的 **Base64** 编码 **secret** 值 **SECRET_VALUE**，可在下一步中使用。

如需更多信息，请参阅 **JBoss EAP** 如何配置服务器安全指南中的将 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/how_to_configure_server_security/#add_user_master_domain_controller** 用户添加到主域控制器一节。

5. 修改从主机的安全域，以使用新的身份验证。

您可以通过在服务器配置中设置 **secret** 值、从凭证存储或 **vault** 中获取密码，或者将密码作为系统属性来指定密码。

- 使用 **Management CLI**，在服务器配置文件中指定 **Base64** 编码的密码值。

使用以下管理 **CLI** 命令指定机密值：务必将 **SECRET_VALUE** 替换为上一步中 **add-user** 输出返回的机密值。

```
/host=SLAVE_HOST_NAME/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="SECRET_VALUE")
```

您将需要重新加载服务器。**host** 参数不适用于单机服务器。

```
reload --host=HOST_NAME
```

如需更多信息，请参阅《**JBoss EAP** 如何配置服务器安全指南》中的"将从属控制器配置为使用凭据 "一节。

- 将主机配置为从凭据存储获取密码。

如果您在凭证存储中存储了 **secret** 值，您可以使用以下命令将服务器 **secret** 设置为来自凭证存储的值：

```
/host=SLAVE_HOST_NAME/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(credential-reference={store=STORE_NAME,alias=ALIAS}
```

您将需要重新加载服务器。**host** 参数不适用于单机服务器。

```
reload --host=HOST_NAME
```

如需更多信息，请参阅 **JBoss EAP** 如何配置服务器安全指南 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/how_to_configure_server_security/#credential_store** 的凭据存储部分。

- 将主机配置为从密码库获取密码。

a. 使用 **EAP_HOME/bin/vault.sh** 脚本生成屏蔽的密码。它将生成一个字符串，格式为 **VAULT::secret::password::VAULT_SECRET_VALUE**，例如：

```
VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNlLWE4ODMtZjQ1NWNmNDU4ZDc1TElORV9CUkVBS3ZhdWx0.
```

注意

在密码库中创建密码时，必须以纯文本形式指定，而不是 **Base64** 编码。

b. 使用以下管理 **CLI** 命令指定机密值：务必将 **VAULT_SECRET_VALUE** 替换为上一步中生成的掩码密码。

```
/host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="${VAULT::secret::password::VAULT_SECRET_VALUE}")
```

您将需要重新加载服务器。**host** 参数不适用于单机服务器。

```
reload --host=HOST_NAME
```

如需更多信息，请参阅 **JBoss EAP** 如何配置服务器安全指南的 **Password Vault** 部分。

- 指定密码作为系统属性。

以下示例使用 **server.identity.password** 作为密码的系统属性名称。

a. 在服务器配置文件中指定密码的系统属性。

使用以下 **managemente CLI** 命令，将机密身份配置为使用系统属性：

```
/host=SLAVE_HOST_NAME/core-service=management/security-
realm=ManagementRealm/server-
identity=secret:add(value="${server.identity.password}")
```

您将需要重新加载服务器。**host** 参数不适用于单机服务器。

```
reload --host=master
```

b.

在启动服务器时设置系统属性的密码。

您可以通过将 **server.identity.password** 系统属性作为命令行参数或在属性文件中传递来设置 **server.identity.password** 系统属性。

i.

作为纯文本命令行参数传递.

启动服务器并传递 **server.identity.password** 属性。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Dserver.identity.password=changeme
```

警告

密码必须以纯文本形式输入，并且对于发出 **ps -ef** 命令的任何人可见。

ii.

在属性文件中设置 属性。

创建属性文件，并将键/值对添加到属性文件中，例如：

```
server.identity.password=changeme
```

警告

密码为纯文本，对于有权访问此属性文件的任何人可见。

使用命令行参数启动服务器。

```
$ EAP_HOME/bin/domain.sh --host-config=host-slave.xml --
properties=PATH_TO_PROPERTIES_FILE
```

6. *重新启动服务器。*

现在，从系统将使用其主机名作为用户名，并将加密字符串用作密码，向主设备进行身份验证。

*您的单机服务器或受管域的服务器组中的服务器现在配置为 **mod_cluster worker** 节点。如果部署集群应用，其会话将复制到所有集群节点进行故障转移，并且可以接受来自外部 **Web** 服务器或负载平衡器的请求。默认情况下，集群的每个节点都使用自动发现功能发现其他节点。*

24.6.4. 配置 mod_cluster fail_on_status Parameter

***fail_on_status** 参数列出了这些 **HTTP** 状态代码，当被集群中的 **worker** 节点返回时，会将该节点标记为失败。然后，负载均衡器将将来的请求发送到集群中的另一个 **worker** 节点。失败的 **worker** 节点将保持在 **not OK** 状态，直到它会向负载均衡器发送 **STATUS** 消息。*

***fail_on_status** 参数必须在负载平衡器的 **httpd** 配置文件中配置。可以通过逗号分隔列表来指定 **fail_on_status** 的多个 **HTTP** 状态代码。以下示例为 **fail_on_status** 指定 **HTTP** 状态代码 **203** 和 **204**。*

*示例：配置 **fail_on_status***

```
ProxyPass / balancer://MyBalancer stickysession=JSESSIONID|jsessionid nofailover=on
failonstatus=203,204
ProxyPassReverse / balancer://MyBalancer
ProxyPreserveHost on
```

24.6.5. 在集群间迁移流量

使用 **JBoss EAP** 创建新群集后，您可以在升级过程中将来自上一集群的流量迁移到新的群集。在此任务中，您将看到可用于在最短的中断或停机时间下迁移此流量的策略。

- 新群集设置。我们将将此群集命名为 **ClusterNEW**。
- 旧集群设置正在变得冗余。我们将将此群集命名为 **ClusterOLD**。

集群升级过程 - 负载平衡组

1. 使用先决条件中描述的步骤设置新集群。
2. 在 **ClusterNEW** 和 **ClusterOLD** 中，确保将配置选项 **sticky-session** 设置为 **true** 的默认设置。启用此选项意味着向任何群集中的群集节点发出的所有新请求将继续转至相应的群集节点。

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-attribute(name=sticky-
session,value=true)
```

3. 将 **load-balancing-group** 设置为 **ClusterOLD**，假设 **ClusterOLD** 中的所有集群节点都是 **ClusterOLD** 负载平衡组的成员。

```
/profile=full-ha/subsystem=modcluster/proxy=default:write-attribute(name=load-balancing-
group,value=ClusterOLD)
```

4. 使用 **Configure a mod_cluster Worker Node** 部分中描述的进程，将 **ClusterNEW** 中的节点单独添加到 **mod_cluster** 配置中。另外，使用上述步骤，将其负载平衡组设置为 **ClusterNEW**。

此时，您可以在 **mod_cluster-manager** 控制台中看到类似于下简示例的输出：

```
        mod_cluster/<version>

LBGroup ClusterOLD: [Enable Nodes]  [Disable Nodes]  [Stop Nodes]
   Node node-1-jvmroute (ajp://node1.oldcluster.example:8009):
      [Enable Contexts]  [Disable Contexts]  [Stop Contexts]
```

```
        Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
        Virtual Host 1:
          Contexts:
            /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

      Node node-2-jvmroute (ajp://node2.oldcluster.example:8009):
        [Enable Contexts]  [Disable Contexts]  [Stop Contexts]
        Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
        Virtual Host 1:
          Contexts:
            /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]


   LBGroup ClusterNEW: [Enable Nodes]  [Disable Nodes]  [Stop Nodes]
      Node node-3-jvmroute (ajp://node3.newcluster.example:8009):
        [Enable Contexts]  [Disable Contexts]  [Stop Contexts]
        Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
        Virtual Host 1:
          Contexts:
            /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

      Node node-4-jvmroute (ajp://node4.newcluster.example:8009):
        [Enable Contexts]  [Disable Contexts]  [Stop Contexts]
        Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
        Virtual Host 1:
          Contexts:
            /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]
```

5.

*任何旧的活跃会话都在 **ClusterOLD** 组内，任何新会话都会在 **ClusterOLD** 或 **CLusterNEW** 组中创建。接下来，我们要禁用整个 **ClusterOLD** 组，以便其集群节点可以被删除，而不会给当前活跃的客户端会话造成任何错误。*

*在 **mod_cluster-manager web** 控制台中，单击 **LBGroup ClusterOLD** 的 **Disable Nodes** 链接。*

*从这一刻起，只有属于已建立的会话的请求才会路由到 **ClusterOLD** 负载平衡组的成员。任何新客户端的会话都仅在 **ClusterNEW** 组中创建。只要在 **ClusterOLD** 组中没有活动的会话，我们可以安全地删除其成员。*

注意

*使用 **Stop Nodes** 会命令负载平衡器停止将任何请求路由到此域。这将强制切换到另一个负载平衡组，如果 **ClusterNEW** 和 **Cluster OLD** 之间没有会话复制，则会导致客户端出现会话数据丢失。*

默认负载均衡组

*如果当前的 **ClusterOLD** 设置不包含任何负载均衡组设置（从 **mod_cluster-manager** 控制台上的 **LBGroup:** 中找到），仍然可以使用禁用 **ClusterOLD** 节点的优势。在本例中，单击每个 **ClusterOLD** 节点的 **Disable Contexts**。这些节点的上下文将被禁用，如果没有活跃的会话，它们就会准备好被删除。只有在启用了上下文的节点上才会创建新客户端会话，本例中为 **ClusterNEW** 成员。*

*使用管理 **CLI***

*除了使用 **mod_cluster-manager web** 控制台外，您还可以使用 **JBoss EAP** 管理 **CLI** 来停止或禁用特定的上下文。*

停止上下文

```
/host=master/server=server-one/subsystem=modcluster:stop-context(context=/my-deployed-application-context, virtualhost=default-host, waittime=0)
```

*停止将 **waittime** 设置为 **0**（无超时）的上下文可指示平衡器立即停止向其路由任何请求，这会强制故障转移到另一个可用的上下文。*

*如果您使用 **waittime** 参数设置了超时值，则不会在此上下文上创建新的会话，但现有会话将继续定向到此节点，直到它们完成或者指定的超时已过。**waittime** 参数默认为 **10** 秒。*

禁用上下文

```
/host=master/server=server-one/subsystem=modcluster:disable-context(context=/my-deployed-application-context, virtualhost=default-host)
```

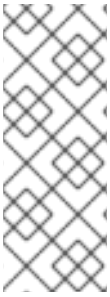
禁用上下文会告知平衡器，不应在此上下文上创建新的会话。

24.7. APACHE MOD_JK HTTP CONNECTOR

Apache mod_jk 是一个 HTTP 连接器，为出于兼容性需要它的客户提供。

JBoss EAP 可以接受来自 Apache HTTP 代理服务器的工作负载。代理服务器接受来自 Web 前端的客户端请求，并将工作传递到参与的 JBoss EAP 服务器。如果启用了粘性会话，则相同的客户端请求始终发送到同一 JBoss EAP 服务器，除非该服务器不可用。

mod_jk 通过 AJP 1.3 协议进行通信。其他协议可与 mod_cluster 或 mod_proxy 一起使用。如需更多信息，请参阅 HTTP 连接器概述。



注意

mod_cluster 是比 mod_jk 更高级的负载平衡器，是推荐的 HTTP 连接器。mod_cluster 提供 mod_jk 的所有功能，以及其他功能。与 JBoss EAP mod_cluster HTTP 连接器不同，Apache mod_jk HTTP 连接器不知道服务器或服务器组上的部署状态，也无法适应其相应地发送工作的位置。

如需更多信息，请参阅 Apache mod_jk 文档。

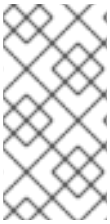
24.7.1. 在 Apache HTTP 服务器中配置 mod_jk

安装 JBoss Core Services Apache HTTP 服务器或使用 JBoss Web 服务器时，已包含 mod_jk 模块 mod_jk.so ；但是，默认情况下不加载它。

注意

自 3.1.0 版开始，Apache HTTP 服务器不再通过 JBoss Web 服务器分发。

使用以下步骤在 Apache HTTP 服务器中加载和配置 mod_jk：请注意，这些步骤假定您已导航到 Apache HTTP 服务器的 httpd/ 目录，这些服务器将根据您的平台而有所不同。如需更多信息，请参阅 JBoss Core ServicesApache HTTP Server 安装指南中的平台安装说明。



注意

红帽客户还可以使用红帽客户门户上的负载平衡器配置工具，为 mod_jk 和其他连接器快速生成最佳配置模板。请注意，您必须登录才能访问此工具。

1.

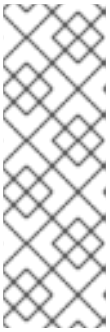
*配置 **mod_jk** 模块。*

注意

*在**conf. d/mod_jk.conf.sample** 中提供了一个示例 **mod_jk** 配置文件。您可以使用这个示例而不是自行创建文件，方法是删除 **.sample** 扩展名并根据需要修改其内容。*

*创建名为 **conf.d/mod_jk.conf** 的新文件。将以下配置添加到 文件中，确保将内容修改为您需要的套件。*

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf.d/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Require ip 127.0.0.1
</Location>
```



注意

JkMount** 指令指定 **Apache HTTP** 服务器必须转发到 **mod_jk** 模块的 **URL**。根据指令的配置，**mod_jk** 会将收到的 **URL** 发送到正确的工作程序。要直接提供静态内容并只使用 **Java** 应用程序的负载均衡器，**URL** 路径必须是 **/application/。要将 **mod_jk** 用作负载均衡器，请使用值 **/*** 将所有 **URL** 转发到 **mod_jk**。*

*除了常规的 **mod_jk** 配置外，此文件还指定加载 **mod_jk.so** 模块，并定义在何处查找 **workers.properties** 文件。*

2.

*配置 **mod_jk** 工作程序节点。*

注意

***at conf.d/workers.properties.sample** 提供了 **worker** 配置文件示例。您可以使用这个示例而不是自行创建文件，方法是删除 **.sample** 扩展名并根据需要修改其内容。*

*创建名为 **conf.d/workers.properties** 的新文件。将以下配置添加到 文件中，确保将内容修改为您需要的套件。*

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
```

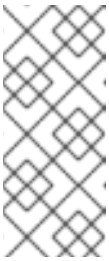
```
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

有关 **mod_jk workers.properties** 文件和其他高级配置选项的语法详情，请参阅 **mod_jk Worker Properties**。

3. （可选）指定 **JKMountFile** 指令。

除了 **mod-jk.conf** 中的 **JKMount** 指令外，您还可以指定一个文件，其中包含要转发到 **mod_jk** 的多个 **URL** 模式。

a. 创建 **uriworkermap.properties** 文件。



注意

at conf.d/uriworkermap.properties.sample 提供了 **URI worker** 映射配置文件示例。您可以使用这个示例而不是自行创建文件，方法是删除 **.sample** 扩展名并根据需要修改其内容。

创建名为 **conf.d/uriworkermap.properties** 的新文件。为要匹配的每个 **URL** 模式添加一行，例如：

```
# Simple worker configuration file
/*=loadbalancer
```

b. 更新配置，以指向 **uriworkermap.properties** 文件。

将以下内容附加到 **conf.d/mod_jk.conf** ：

```
# Use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf.d/uriworkermap.properties
```

有关配置 **mod_jk** 的详情，请参阅 **JBoss Web** 服务器 **HTTP** 连接器和负载平衡指南中的将 **Apache HTTP** 服务器配置为载入 **mod_jk** 部分。

24.7.2. 将 JBoss EAP 配置为与 mod_jk 通信

JBoss EAP undertow 子系统需要指定侦听器，以便接受来自 的请求并将回复发回给外部 Web 服务器。由于 mod_jk 使用 AJP 协议，因此必须配置 AJP 侦听器。

如果您使用的是默认高可用性配置 ha 或 full-ha，则已经配置了 AJP 侦听器。

具体步骤请查看 从外部 Web 服务器接受请求。

24.8. APACHE MOD_PROXY HTTP 连接器

Apache mod_proxy 是一个 HTTP 连接器，支持通过 AJP、HTTP 和 HTTPS 协议进行连接。mod_proxy 可以在负载均衡或非负载均衡的配置中配置，并且支持粘性会话的概念。

mod_proxy 模块要求 JBoss EAP 在 undertow 子系统中配置 HTTP、HTTPS 或 AJP 侦听器，具体取决于您计划使用的协议。

注意

mod_cluster 是比 mod_proxy 更为高级的负载平衡器，是推荐的 HTTP 连接器。mod_cluster 提供 mod_proxy 的所有功能，以及其他功能。与 JBoss EAP mod_cluster HTTP 连接器不同，Apache mod_proxy HTTP 连接器不知道服务器或服务器组上的部署状态，也无法适应其相应地发送工作的位置。

如需更多信息，请参阅 Apache mod_proxy 文档。

24.8.1. 在 Apache HTTP 服务器中配置 mod_proxy

安装 JBoss Core Services Apache HTTP 服务器或使用 JBoss Web 服务器时已包含 mod_proxy 模块，并且默认加载。

注意

自 3.1.0 版开始，Apache HTTP 服务器不再通过 JBoss Web 服务器分发。

请参阅以下适当的部分来配置基本的负载平衡或非 负载平衡代理。这些步骤假定您已导航到 Apache HTTP 服务器的 httpd/ 目录，这些服务器将根据您的平台而有所不同。如需更多信息，请参阅 JBoss Core ServicesApache HTTP Server 安装指南中的平台安装说明。这些步骤还假定 JBoss EAP undertow 子系统中已配置了所需的 HTTP 侦听器。

注意

红帽客户还可以使用红帽客户门户网站中的负载平衡器配置工具为 mod_proxy 和其他连接器快速生成最佳配置模板。请注意，您必须登录才能访问此工具。

添加非负载平衡代理

将以下配置添加到您的 conf/httpd.conf 文件中，直接在您拥有的任何其他 <VirtualHost> 指令下。将值替换为适合您的设置的值。

```
<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On

# The IP and port of JBoss
# These represent the default values, if your httpd is on the same host
# as your JBoss managed domain or server

ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
</VirtualHost>
```

添加负载平衡代理

注意

*默认 **Apache HTTP** 服务器配置禁用 **mod_proxy_balancer.so** 模块，因为它与 **mod_cluster** 不兼容。要完成此任务，您需要加载此模块并禁用 **mod_cluster** 模块。*

*若要将 **mod_proxy** 用作负载平衡器并将工作发送到多个 **JBoss EAP** 实例，请将以下配置添加到您的 **conf/httpd.conf** 文件中：**IP** 地址示例为虚构。使用适合您的环境的值替换它们。*

```
<Proxy balancer://mycluster>

Order deny,allow
Allow from all

# Add each JBoss Enterprise Application Server by IP address and port.
# If the route values are unique like this, one node will not fail over to the other.
BalancerMember http://192.168.1.1:8080 route=node1
BalancerMember http://192.168.1.2:8180 route=node2
</Proxy>

<VirtualHost *:80>
 # Your domain name
 ServerName YOUR_DOMAIN_NAME

 ProxyPreserveHost On
 ProxyPass / balancer://mycluster/

 # The location of the HTML files, and access control information DocumentRoot /var/www
 <Directory /var/www>
  Options -Indexes
  Order allow,deny
  Allow from all
 </Directory>

</VirtualHost>
```

*以上示例都使用 **HTTP** 协议进行通信。如果载入适当的 **mod_proxy** 模块，您可以使用 **AJP** 或 **HTTPS** 协议。如需了解更多详细信息，请参阅 **Apache mod_proxy** 文档。*

启用粘滞会话

*粘性会话意味着，如果某个客户端请求最初发送到特定的 **JBoss EAP** 工作程序，则以后的所有请求都将发送到同一工作程序，除非变得不可用。这几乎都是推荐的行为。*

*要为 **mod_proxy** 启用粘性会话，请将 **stickysession** 参数添加到 **ProxyPass** 语句中。*

```
ProxyPass / balancer://mycluster stickysession=JSESSIONID
```

您可以为 **ProxyPass** 语句指定其他参数，如 **lbmethod** 和 **nofailover**。有关可用参数的更多信息，请参阅 **Apache mod_proxy** 文档。

24.8.2. 将 JBoss EAP 配置为与 mod_proxy 通信

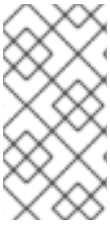
JBoss EAP undertow 子系统需要指定侦听器，以便接受来自 的请求并将回复发回给外部 **Web** 服务器。根据您要使用的协议，您可能需要配置侦听器。

JBoss EAP 默认配置中配置了 **HTTP** 侦听器。如果您使用的是默认高可用性配置 **ha** 或 **full-ha**，还会预配置 **AJP** 侦听器。

具体步骤请查看 从外部 **Web** 服务器接受请求。

24.9. MICROSOFT ISAPI CONNECTOR

Internet 服务器 **API(ISAPI)**是一组 **API**，用于为 **Web** 服务器（如 **Microsoft** 的 **Internet Information Services(IIS)**）编写 **OLE** 服务器扩展和过滤器。**isapi_redirect.dll** 是 **mod_jk** 的扩展，调整为 **IIS**。**isapi_redirect.dll** 允许您将 **JBoss EAP** 实例配置为将 **IIS** 用作负载平衡器的 **worker** 节点。



注意

如需有关 **Windows Server** 和 **IIS** 支持的配置的信息，请参阅 **JBoss EAP** 支持的配置。

24.9.1. 将 Microsoft IIS 配置为使用 ISAPI 连接器

从红帽客户门户网站下载 **ISAPI** 连接器：

1. 打开浏览器并登录红帽客户门户 **JBoss** 软件下载页面。
2. 在 **Product** 下拉菜单中选择 **Web Connectors**。
3. 从 **Version** 下拉菜单中选择最新的 **JBoss Core Services** 版本。

4. 在列表中找到 **Red Hat JBoss Core Services ISAPI Connector**，然后单击 **Download** 链接。

5. 提取存档，并将 **sbin** 目录的内容复制到服务器的一个位置。以下说明假定内容已复制到 **C:\connectors**。

使用 **IIS Manager(IIS 7)**配置 **IIS** 重定向器：

1. 点 **Start** → **Run** 并输入 **inetmgr** 来打开 **IIS** 管理器。

2. 在左侧的树视图窗格中，展开 **IIS 7**。

3. 双击 **ISAPI** 和 **CGI** 注册以在新窗口中将其打开。

4. 在 **Actions** 窗格中，单击 **Add**。此时将打开 **Add ISAPI** 或 **CGI Restriction** 窗口。

5. 指定以下值：

   - **ISAPI** 或 **CGI** 路径 ：**C:\connectors\isapi_redirect.dll**
   - 描述 ：**jboss**
   - 允许执行扩展路径 ：选中复选框。

6. 单击确定以关闭 添加 **ISAPI** 或 **CGI Restriction** 窗口。

7. 定义 **JBoss** 原生虚拟目录

   - 右键单击 **Default Web Site**，再单击 **Add Virtual Directory**。此时将打开 **Add Virtual Directory** 窗口。

- *指定以下值来添加虚拟目录：*
  - *别名 ：**jboss***
  - *物理路径 ：**C:\connectors***
- *单击 **OK** 以保存这些值并关闭 **Add Virtual Directory** 窗口。*

8. *定义 **JBoss** 原生 **ISAPI** 重定向过滤器*
   - *在树视图窗格中，展开 **Sites** → **Default Web Site**。*
   - *双击 **ISAPI** 过滤器。此时会显示 **ISAPI** 过滤器功能视图。*
   - *在 **Actions** 窗格中，单击 **Add**。此时将显示 **Add ISAPI Filter** 窗口。*
   - *在 **Add ISAPI Filter** 窗口中指定以下值：*
     - *过滤器名称 ：**jboss***
     - *可执行 ：**C:\connectors\isapi_redirect.dll***
   - *单击 **OK** 以保存这些值并关闭 **Add ISAPI Filters** 窗口。*
9. *启用 **ISAPI-dll** 处理程序*
   - *双击树视图窗格中的 **IIS 7** 项目。**IIS 7 Home Features View** 将打开。*

- 双击 **Handler** 映射。**Handler** 映射功能视图显示。

- 在 **Group by combo** 框中，选择 **State**。**Handler Mappings** 显示在 **Enabled** 和 **Disabled Groups** 中。

- 查找 **ISAPI-dll.**如果它位于 **Disabled** 组中，请右键单击它，然后选择 **Edit Feature Permissions**。

- 启用以下权限：

  - 读

  - 脚本

  - 执行

- 单击 **OK** 以保存这些值，然后关闭 **Edit Feature Permissions** 窗口。

Microsoft IIS 现已配置为使用 **ISAPI** 连接器。

24.9.2. 配置 ISAPI 连接器以将客户端请求发送到 JBoss EAP

此任务配置一组 **JBoss EAP** 服务器，以接受来自 **ISAPI** 连接器的请求。它不包括用于负载平衡或高可用性故障转移的配置。

此配置在 **IIS** 服务器上完成，并且假定您已将 **JBoss EAP** 配置为接受来自外部 **Web** 服务器的请求。您还需要对 **IIS** 服务器的完整管理员访问权限，并将 **IIS** 配置为使用 **ISAPI** 连接器。

创建属性文件和设置重定向

1. 创建用于存储日志、属性文件和锁定文件的目录。

   此流程的其余部分假定您为此使用了目录 **C:\connectors**。如果您使用不同的目录，请相应

地修改说明。

2. 创建 **isapi_redirect.properties** 文件。

创建名为 **C:\connectors\isapi_redirect.properties** 的新文件。将以下内容复制到 文件。

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

如果您不想使用 **rewrite.properties** 文件，请在行首加上 **#** 字符来注释掉最后一行。

3. 创建 **uriworkermap.properties** 文件

uriworkermap.properties 文件包含已部署应用程序 **URL** 之间的映射，以及哪个 **worker** 处理向它们的请求。下例中的 文件显示了 文件的语法。将您的 **uriworkermap.properties** 文件放在 **C:\connectors** 中。

```
# images and css files for path /status are provided by worker01
/status=worker01
/images/*=worker01
/css/*=worker01

# Path /web-console is provided by worker02
# IIS (customized) error page is used for http errors with number greater or equal to 400
# css files are provided by worker01
/web-console/*=worker02;use_server_errors=400
/web-console/css/*=worker01

# Example of exclusion from mapping, logo.gif won't be displayed
# !/web-console/images/logo.gif=*
```

```
# Requests to /app-01 or /app-01/something will be routed to worker01
/app-01|/*=worker01

# Requests to /app-02 or /app-02/something will be routed to worker02
/app-02|/*=worker02
```

4. 创建 **workers.properties** 文件。

Worker .properties 文件包含 **worker** 标签和服务器实例之间的映射定义。此文件遵循用于 **Apache mod_jk** 工作程序属性配置的同一文件的语法。

以下是 **workers.properties** 文件的示例：工作程序名称（**worker01** 和 **worker02**）必须与 **JBoss EAP undertow** 子系统中配置的 **instance-id** 相匹配。

将此文件放入 **C:\connectors** 目录中。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers

# First JBoss EAP server definition, port 8009 is standard port for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second JBoss EAP server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. 创建 **rewrite.properties** 文件。

rewrite.properties 文件包含为特定应用重写规则的简单 **URL**。重写路径使用名称值对来指定，如下例所示。将此文件放入 **C:\connectors** 目录中。

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
```

6. 使用 **net stop** 和 **net start** 命令重新启动 **IIS** 服务器。

```
C:\> net stop was /Y
C:\> net start w3svc
```

IIS 服务器配置为根据特定应用将客户端请求发送到您配置的特定 **JBoss EAP** 服务器。

24.9.3. 配置 ISAPI Connector to Balance Client Requests Across 多 JBoss EAP 服务器

此配置在您指定的 **JBoss EAP** 服务器之间平衡客户端请求。此配置在 **IIS** 服务器上完成，并且假定您已将 **JBoss EAP** 配置为接受来自外部 **Web** 服务器的请求。您还需要对 **IIS** 服务器的完整管理员访问权限，并将 **IIS** 配置为使用 **ISAPI** 连接器。

在客户端请求多个服务器之间取得平衡

1. 创建用于存储日志、属性文件和锁定文件的目录。

   此流程的其余部分假定您为此使用了目录 **C:\connectors**。如果您使用不同的目录，请相应地修改说明。

2. 创建 **isapi_redirect.properties** 文件。

   创建名为 **C:\connectors\isapi_redirect.properties** 的新文件。将以下内容复制到 文件。

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

   如果您不想使用 **rewrite.properties** 文件，请在行首加上 **#** 字符来注释掉最后一行。

3. 创建 **uriworkermap.properties** 文件。

uriworkermap.properties 文件包含已部署应用程序 **URL** 之间的映射，以及哪个 **worker** 处理向它们的请求。以下示例文件显示了 文件的语法，以及负载均衡配置。通配符**(*)**字符将各种 **URL** 子目录的所有请求发送到名为 **router** 的负载均衡器。下一步中介绍了负载平衡器的配置。

将您的 **uriworkermap.properties** 文件放在 **C:\connectors** 中。

```
# images, css files, path /status and /web-console will be
# provided by nodes defined in the load-balancer called "router"
/css/*=router
/images/*=router
/status=router
/web-console|/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
# !/web-console/images/logo.gif=*

# Requests to /app-01 and /app-02 will be routed to nodes defined
# in the load-balancer called "router"
/app-01|/*=router
/app-02|/*=router

# mapping for management console, nodes in cluster can be enabled or disabled here
/jkmanager|/*=status
```

4. 创建 **workers.properties** 文件。

Worker .properties 文件包含 **worker** 标签和服务器实例之间的映射定义。此文件遵循用于 **Apache mod_jk** 工作程序属性配置的同一文件的语法。

以下是 **workers.properties** 文件的示例 ：负载均衡器在文件的末尾配置，组成 **worker worker01** 和 **worker02**。这些工作程序名称必须与 **JBoss EAP undertow** 子系统中配置的 **instance-id** 相匹配。

将此文件放入 **C:\connectors** 目录中。

```
# The advanced router LB worker
worker.list=router,status

# First EAP server definition, port 8009 is standard port for AJP in EAP
#
# lbfactor defines how much the worker will be used.
```

```
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A – all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker for jkmanager
worker.status.type=status
```

5. 创建 **rewrite.properties** 文件。

rewrite.properties 文件包含为特定应用重写规则的简单 **URL**。重写路径使用名称值对来指定，如下例所示。将此文件放入 **C:\connectors** 目录中。

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
Restart the IIS server.

Restart your IIS server by using the net stop and net start commands.
C:\> net stop was /Y
C:\> net start w3svc
```

IIS 服务器配置为将客户端请求发送到 **workers.properties** 文件中引用的 **JBoss EAP** 服务器，以 **1:3** 比例在服务器之间分散负载。这一比例派生自分配给每台服务器的负载平衡因子(**lbfactor**)。

24.10. ORACLE NSAPI 连接器

Netscape Server API(NSAPI)是由 Oracle iPlanet Web 服务器（以前称为 Netscape Web Server）提供的 API，用于实施服务器的扩展。这些扩展称为服务器插件。NSAPI 连接器使用 in nsapi_redirector.so，它是 mod_jk 的扩展，调整为 Oracle iPlanet Web 服务器。NSAPI 连接器使您能够将 JBoss EAP 实例配置为 worker 节点，并将 Oracle iPlanet Web 服务器配置为负载平衡器。

注意

有关 Solaris 和 Oracle iPlanet Web 服务器支持的配置的信息，请参阅 JBoss EAP 支持的配置。

24.10.1. 将 Oracle iPlanet Web 服务器配置为使用 NSAPI 连接器

先决条件

- JBoss EAP 在每台服务器上安装和配置，这些服务器上将充当工作程序。

从红帽客户门户网站下载 NSAPI 连接器：

1. 打开浏览器并登录红帽客户门户 JBoss 软件下载页面。
2. 在 Product 下拉菜单中选择 Web Connectors。
3. 从 Version 下拉菜单中选择最新的 JBoss Core Services 版本。
4. 在列表中找到 Red Hat JBoss Core Services NSAPI Connector，确保您为您的系统选择正确的平台和架构，然后单击 Download 链接。
5. 将位于 lib/ 或 lib 64/ 目录中的 then sapi_redirector.so 文件提取到 IPLANET_CONFIG/lib/ 或 IPLANET_CONFIG/lib64/ 目录中。

设置 NSAPI 连接器：

注意

*在这些说明中，**IPLANET_CONFIG** 是指 **Oracle iPlanet** 配置目录，通常是 **/opt/oracle/webserver7/config/**。如果您的 **Oracle iPlanet** 配置目录不同，请相应地修改说明。*

1. *禁用 **servlet** 映射。*

   *打开 **IPLANET_CONFIG/default.web.xml** 文件，找到标题 **Built In Server Mappings** 部分。通过用 **XML** 注释字符（**<!--** 和 **-->**）来禁用以下三个**servlet** 的映射。*

   - ***default***
   - ***invocationr***
   - ***jsp***

   以下示例配置显示了禁用的映射。

```
<!-- ============== Built In Servlet Mappings =============== -->
<!-- The servlet mappings for the built in servlets defined above. -->
<!-- The mapping for the default servlet -->
<!--servlet-mapping>
 <servlet-name>default</servlet-name>
 <url-pattern>/</url-pattern>
</servlet-mapping-->
<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
 <servlet-name>invoker</servlet-name>
 <url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->
<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
 <servlet-name>jsp</servlet-name>
 <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

   保存并退出文件。

2. 配置 **iPlanet Web** 服务器以加载 **NSAPI** 连接器模块。

将以下行添加到 **IPLANET_CONFIG/magnus.conf** 文件的末尾，修改文件路径以适应您的配置。这些行定义 **then sapi_redirector.so** 模块的位置，以及 **workers.properties** 文件，其中列出了 **worker** 及其属性。

```
Init fn="load-modules" funcs="jk_init,jk_service" shlib="/lib/nsapi_redirector.so" shlib_flags="(global|now)"

Init fn="jk_init" worker_file="IPLANET_CONFIG/connectors/workers.properties" log_level="info" log_file="IPLANET_CONFIG/connectors/nsapi.log" shm_file="IPLANET_CONFIG/connectors/tmp/jk_shm"
```

以上配置适用于 **32** 位架构。如果使用 **64** 位 **Solaris**，请将字符串 **lib/nsapi_redirector.so** 更改为 **lib64/nsapi_redirector.so**。

保存并退出文件。

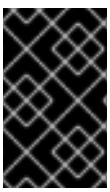
3. 配置 **NSAPI** 连接器。

您可以为基本配置配置 **NSAPI** 连接器，且无负载平衡或负载平衡配置。选择以下选项之一，之后您的配置将完成。

- 配置 **NSAPI** 连接器以将客户端请求发送到 **JBoss EAP**
- 配置 **NSAPI Connector to Balance Client Requests Across** 多 **JBoss EAP** 服务器

24.10.2. 配置 NSAPI 连接器以将客户端请求发送到 JBoss EAP

此任务配置 **NSAPI** 连接器，以将客户端请求重定向到 **JBoss EAP** 服务器，而无需负载平衡或故障转移。重定向以每个部署为基础，因此按 **URL** 进行。



重要

在继续此任务前，您必须已配置了 **NSAPI** 连接器。

设置基本 **HTTP** 连接器

1.

定义重定向到 **JBoss EAP** 服务器的 **URL** 路径。

注意

在 **IPLANET_CONFIG/obj.conf** 中，行首不允许有空格，除非该行是上一行的延续。

编辑 **IPLANET_CONFIG/obj.conf** 文件。找到以 **<Object name="default">** 开头的部分，并添加要匹配的每个 **URL** 模式，其格式为以下示例文件所示。**string jknsapi** 指的是 **HTTP** 连接器，将在下一步中定义。示例演示了使用通配符进行模式匹配。

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
</Object>
```

2.

定义提供每个路径的 **worker**。

继续编辑 **IPLANET_CONFIG/obj.conf** 文件。在刚刚完成编辑的部分的关闭标签后直接添加以下内容：**& lt;/Object>**。

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="worker01" path="/status"
Service fn="jk_service" worker="worker02" path="/nc(/*)"
Service fn="jk_service" worker="worker01"
</Object>
```

上例将请求重定向到名为 **worker 01** 的 **worker** 路径 **/status**，并将 **/nc/** 下的所有 **URL** 路径重定向到名为 **worker 02** 的 **worker**。第三行指出分配给**jknsapi**对象且前面行不匹配的所有 **URL** 都服务于 **worker01**。

保存并退出文件。

3.

定义 **worker** 及其属性。

在 **IPLANET_CONFIG/connectors/** 目录中创建一个名为 **workers.properties** 的文件。将以下内容粘贴到 文件中，并进行修改以适合您的环境。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

Worker .properties 文件的语法与 **Apache mod_jk** 相同。

保存并退出文件。

4. 重启 **iPlanet Web** 服务器

发出以下命令以重新启动 **iPlanet Web** 服务器：

```
IPLANET_CONFIG/../bin/stopserv
IPLANET_CONFIG/../bin/startserv
```

iPlanet Web 服务器现在将客户端请求发送到您已配置在 **JBoss EAP** 上部署的 **URL**。

24.10.3. 配置 NSAPI Connector to Balance Client Requests Across 多 JBoss EAP 服务器

此任务配置 **NSAPI** 连接器，以将客户端请求发送到负载平衡配置中的 **JBoss EAP** 服务器。

重要

在继续此任务前，您必须已配置了 **NSAPI** 连接器。

配置用于负载平衡的连接器

1.

定义重定向到 **JBoss EAP** 服务器的 **URL** 路径。

注意

在 **IPLANET_CONFIG/obj.conf** 中，行首不允许有空格，除非该行是上一行的延续。

编辑 **IPLANET_CONFIG/obj.conf** 文件。找到以 **<Object name="default">** 开头的部分，并添加要匹配的每个 **URL** 模式，其格式为以下示例文件所示。**string jknsapi** 指的是要在下一步中定义的 **HTTP** 连接器。示例演示了使用通配符进行模式匹配。

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jkmanager/*" name="jknsapi"
</Object>
```

2.

定义提供每个路径的 **worker**。

继续编辑 **IPLANET_CONFIG/obj.conf** 文件。直接在上一步中修改的部分结束标签后 **</Object>**，添加以下新部分，并根据您的需要进行修改：

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="status" path="/jkmanager(/*)"
Service fn="jk_service" worker="router"
</Object>
```

This jksnapi 对象定义 **worker** 节点，用于提供映射到 默认 对象中 **name="jksnapi"** 映射的每个路径。与 **/jkmanager/*** 匹配的 **URL** 之外的所有内容都重定向到名为 **router** 的 **worker**。

3.

定义 **worker** 及其属性。

在 **IPLANET_CONFIG/connector/** 中创建一个名为 **workers.properties** 的文件。将以下内容粘贴到 文件中，并进行修改以适合您的环境。

```
# The advanced router LB worker
# A list of each worker
worker.list=router,status

# First JBoss EAP server
# (worker node) definition.
# Port 8009 is the standard port for AJP
#

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second JBoss EAP server
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the load-balancer called "router"
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status
```

Worker .properties 文件的语法与 **Apache mod_jk** 相同。

保存并退出文件。

4. 重新启动 **iPlanet Web** 服务器 **7.0**。

```
IPLANET_CONFIG/../bin/stopserv
IPLANET_CONFIG/../bin/startserv
```

iPlanet Web 服务器将在负载平衡配置中将您配置的 **URL** 模式重定向到 **JBoss EAP** 服务器。

第 25 章 ECLIPSE MICROPROFILE

25.1. 使用 ECLIPSE MICROPROFILE 配置管理配置

重要

Eclipse Microprofile Config 仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

25.1.1. 关于 MicroProfile Config SmallRye 子系统

配置数据可以动态变化，应用需要能够在不重新启动服务器的情况下访问最新的配置信息。**Eclipse MicroProfile** 配置提供了配置数据的可移植外部化。这使得应用和微服务可以配置为在多个环境中运行，无需修改或重新打包。**Eclipse MicroProfile** 配置功能在 **JBoss EAP** 中使用 **SmallRye Config** 组件实施，由 **microprofile-config-smallrye** 子系统提供。此子系统包含在默认的 **JBoss EAP 7.3** 配置中。

25.1.2. 受 MicroProfile Config SmallRye 子系统支持的 ConfigSources

MicroProfile 配置属性可以是不同格式的，可以来自不同位置，由 **ConfigSources** 提供，它们是 **org.eclipse.microprofile.config.spi.ConfigSource** 接口的实施。

MicroProfile 配置规范提供下列默认 **ConfigSource** 实施，用于检索配置值：

- 使用 **System.getProperties**（）.
- 使用 **System.getenv**（）.
- 来自类路径上的所有 **META-INF/microprofile-config.properties** 文件。

The microprofile-config-smallrye 子系统支持以下额外类型的 **ConfigSource** 资源，用于检索配置值：

- （从属性中）。
- 以及目录中的一个文件。
- （来自 **ConfigSource** 类）。
- （来自 **ConfigSourceProvider** 类）。

注意

虽然以下部分使用管理 **CLI** 配置 **microprofile-config-smallrye** 子系统，但您也可以通过导航到 **Configuration → Subsystems → MicroProfile Config → 并点击 View** 来使用管理控制台完成这些任务。

如需可用于配置 **MicroProfile Config SmallRye** 子系统的属性列表，请参阅 **MicroProfile Config SmallRye** 子系统。

从属性获取 **ConfigSource** 配置

您可以通过添加 **config-source** 属性并将属性指定为列表，直接存储 **ConfigSource** 的属性 以供访问。

以下管理 **CLI** 命令创建一个 **ConfigSource**，其中包含 **property1** 和 **properties 2** 属性 的配置数据。

```
/subsystem=microprofile-config-smallrye/config-source=props:add(properties=
{property1=value1,property2=value2})
```

此命令将为 **microprofile-config-smallrye** 子系统生成以下 **XML** 配置：

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
    <config-source name="props">
        <property name="property1" value="value1"/>
        <property name="property2" value="value2"/>
    </config-source>
</subsystem>
```

从目录获取 ConfigSource 配置

您可以通过添加 config-source 属性并指定包含文件的目录来创建 ConfigSource 以从目录中的文件读取属性。dir 目录中每个文件的名称是属性的名称，文件内容则是 属性的值，

以下管理 CLI 命令创建一个 ConfigSource，它将从 /etc/config/numbers-app/ 目录中的文件读取配置属性。

```
/subsystem=microprofile-config-smallrye/config-source=file-props:add(dir={path=/etc/config/numbers-app})
```

此命令将为 microprofile-config-smallrye 子系统生成以下 XML 配置：

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
    <config-source name="file-props">
        <dir path="/etc/config/numbers-app"/>
    </config-source>
</subsystem>
```

此结构与 OpenShift ConfigMap 使用的结构对应。The dir 值对应于 OpenShift 或 Kubernetes 中的 ConfigMap 定义中的 mountPath。

JBoss EAP 中部署的任何应用能够以编程方式访问 目录中存储的属性。

假设目录 /etc/config/numbers-app/ 包含以下两个文件：

- num.size ：此文件包含值 5。
- num.max ：此文件包含值 100。

在下面的代码示例中，num.size 返回 5 and num.max 返回 100。

```
@Inject
@ConfigProperty(name = "num.size")
int numSize;
```

```
@Inject
@ConfigProperty(name = "num.max")
int numMax;
```

从 **ConfigSource** 类获取 **ConfigSource** 配置

您可以创建和配置自定义 **org.eclipse.microprofile.config.spi.ConfigSource** 实施类，以提供配置值的来源。

以下管理 **CLI** 命令为名为 **org.example.MyConfig Source** 的实施类创建一个 **Config Source**，它由名为 **org.example** 的 **JBoss** 模块提供。

```
/subsystem=microprofile-config-smallrye/config-source=my-config-source:add(class=
{name=org.example.MyConfigSource, module=org.example})
```

此命令将为 **microprofile-config-smallrye** 子系统生成以下 **XML** 配置：

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
    <config-source name="my-config-source">
        <class name="org.example.MyConfigSource" module="org.example"/>
    </config-source>
</subsystem>
```

此 **ConfigSource** 类提供的属性可用于任何 **JBoss EAP** 部署。

有关如何将全局模块添加到 **JBoss EAP** 服务器的详情，请参考定义全局模块。

从 **ConfigSourceProvider** 类获取 **ConfigSource** 配置

您可以创建和配置自定义 **org.eclipse.microprofile.config.spi.ConfigSourceProvider** 实施类，以注册多个 **ConfigSource** 实例的实施。

以下管理 **CLI** 命令为名为 **org.example.MyConfigSourceProvider** 的实施类创建一个 **config-source-provider**，该类由名为 **org.example** 的 **JBoss** 模块提供。

```
/subsystem=microprofile-config-smallrye/config-source-provider=my-config-source-
provider:add(class={name=org.example.MyConfigSourceProvider, module=org.example})
```

此命令将为 **microprofile-config-smallrye** 子系统生成以下 **XML** 配置：

```
<subsystem xmlns="urn:wildfly:microprofile-config-smallrye:1.0">
```

```
    <config-source-provider name="my-config-source-provider">
        <class name="org.example.MyConfigSourceProvider" module="org.example"/>
    </config-source-provider>
</subsystem>
```

ConfigSourceProvider 实施提供的属性可用于任何 **JBoss EAP** 部署。

有关如何将全局模块添加到 **JBoss EAP** 服务器的详情，请参考定义全局模块。

25.1.3. 部署访问 ConfigSources 的应用程序

在 **Java** 代码中访问 **MicroProfile** 配置的应用必须启用 **CDI**，方法是在部署存档 中包含 **META-INF/beans.xml** 或 **WEB-INF/beans.xml** 文件，或者包含 **CDI bean** 注释。

有关 **CDI** 的更多信息，请参阅 **JBoss EAP** 开发指南中的上下文和依赖注入(**CDI**)。
https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#contexts_and_dependency_injection

25.2. 使用 MICROPROFILE OPENTRACING SMALLRYE 子系统跟踪请求

重要

Eclipse Microprofile OpenTracing 仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

25.2.1. 关于 Eclipse MicroProfile OpenTracing

跟踪服务边界请求的能力非常重要，特别是在请求可以在其生命周期内穿过多个服务的微服务环境中。**Eclipse Microprofile OpenTracing** 规范定义了用于访问 **Jakarta RESTful Web Services** 应用中 **OpenTracing** 合规 **Tracer** 对象的行为和 **API**。行为指定如何为传入和传出请求自动创建 **OpenTracing Spans**。**API** 定义如何显式禁用或启用给定端点的追踪。

25.2.2. 关于 MicroProfile OpenTracing SmallRye 子系统

Eclipse Microprofile OpenTracing 功能在 JBoss EAP 中使用 SmallRye OpenTracing 组件实施，并由 microprofile-opentracing-smallrye 子系统提供。此子系统实施 Microprofile 1.3.0，其中包括对对 JAX-RS 端点和 CDI Bean 的请求进行追踪的支持，并包含在默认的 JBoss EAP 7.3 配置中。

The microprofile-opentracing-smallrye 子系统附带了 Jaeger Client 作为默认 tracer，以及 Jakarta EE 应用程序中常用组件的一组检测库，如 Jakarta RESTful Web 服务和上下文以及依赖注入。部署到 JBoss EAP 服务器的每个单独的 WAR 都会自动拥有自己的 跟踪器 实例。EAR 中的每个 WAR 都被视为单独的 WAR，每个 WAR 都有自己的 Tracer 实例。默认情况下，用于 Jaeger Client 的服务名称派生自部署名称，通常是 WAR 文件名。

25.2.3. 配置 MicroProfile OpenTracing SmallRye 子系统

The microprofile-opentracing-smallrye 子系统包含在默认的 JBoss EAP 7.3 配置中。由于启用了 OpenTracing，可能会有内存或性能成本，因此您可能需要禁用此子系统。

使用下列管理 CLI 命令，从服务器配置中删除 子系统，以全局禁用服务器实例的 MicroProfile OpenTracing 功能：

1. 移除子系统：

```
/subsystem=microprofile-opentracing-smallrye:remove()
```

2. 重新加载服务器以使更改生效。

```
reload
```

使用下列管理 CLI 命令，通过将 子系统添加到服务器配置，为服务器实例全局启用 MicroProfile OpenTracing 功能：

1. 添加 子系统。

```
/subsystem=microprofile-opentracing-smallrye:add()
```

2. 重新加载服务器以使更改生效。

```
reload
```

没有可用于 **microprofile-opentracing-smallrye** 子系统的其他配置选项。相反，您可以通过设置系统属性或环境变量来配置 **Jaeger Client**。如需有关如何配置 **Jaeger** 客户端的信息，请参阅 **Jaeger** 文档。如需有效系统属性列表，请参阅 **Jaeger** 文档中的通过环境配置。

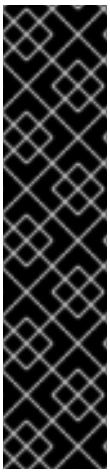
重要

因为这个功能是作为技术预览提供的，所以当前的配置选项，特别是那些使用系统属性和环境变量配置 **Jaeger** 客户端追踪器的配置选项，可能会在以后的发行版本中以不兼容的方式改变。

另请注意，在默认情况下，**Jakarta** 的 **Jaeger** 客户端有一个可探测性的抽样策略，它被设置为 **0.001**，这意味着只会抽样大约一千个 **trace**。若要对每个请求进行示例，请将系统属性 **JAEGER_SAMPLER_TYPE** 设置为 **const**，将 **JAEGER_SAMPLER_PARAM** 设置为 **1**。

有关如何覆盖默认追踪器以及如何追踪 **CDI Bean** 的信息，请参阅《开发指南 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#using_microprofile_opentracing_smallrye_tracer**》中使用 **Eclipse MicroProfile OpenTracing to Trace Requests**。

25.3. 使用 ECLIPSE MICROPROFILE 健康监控服务器健康状况



重要

Eclipse MicroProfile 健康仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（**SLA**）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

25.3.1. MicroProfile Health SmallRye Subsystem

JBoss EAP 包含 **SmallRye Health** 组件，它使用 **microprofile-health-smallrye** 子系统提供 **Eclipse MicroProfile** 健康功能。此子系统用于确定 **JBoss EAP** 实例是否按预期响应，并且默认启用。

重要

默认情况下，**MicroProfile Health SmallRye** 子系统仅评估服务器是否在运行。要包含自定义健康检查，请参阅开发指南中的实施 **https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#microprofile_health_smallrye_custom_check** 自定义健康检查部分。

MicroProfile 健康仅在将 **JBoss EAP** 作为单机服务器运行时才可用。

25.3.2. 使用管理 CLI 示例进行健康检查

可以使用管理 **CLI** 检查健康检查。

- **:check** 属性演示了获取服务器的当前运行状况（存活度和 就绪度探测 ）。

```
/subsystem=microprofile-health-smallrye:check
{
    "outcome" => "success",
    "result" => {
        "outcome" => "UP",
        "checks" => []
    }
}
```

- **check -live** 属性仅获取 存活度 探测的健康数据。

```
/subsystem=microprofile-health-smallrye:check-live
{
    "outcome" => "success",
    "result" => {
        "outcome" => "UP",
        "checks" => []
    }
}
```

- **check-ready** 属性仅获取 就绪度探测 的健康数据。

```
/subsystem=microprofile-health-smallrye:check-ready
{
    "outcome" => "success",
```

```
    "result" => {
        "outcome" => "UP",
        "checks" => []
    }
}
```

25.3.3. 使用管理控制台检查健康检查

管理控制台包含一个检查运行时操作，可将健康检查和全局结果显示为布尔值。

从管理控制台获取服务器的当前健康状况：

1. 导航到 **Runtime** 选项卡，再选择 服务器。

2. 在 **Monitor** 列中，点击 **MicroProfile Health → View**。

25.3.4. 使用 HTTP 端点检查健康检查

健康检查自动部署到 **JBoss EAP** 上的 健康 上下文中。

That means that in addition to being able to examine it using the management CLI, you can also obtain the current health using the HTTP endpoint. The default address for the `/health` endpoint, accessible from the management interfaces, is `http://127.0.0.1:9990/health`.

1. 要使用 **HTTP** 端点来获取服务器当前的健康状况，可使用以下 **URL**：

```
http://HOST:9990/health
```

访问此上下文时，以 **JSON** 格式显示健康检查，指示服务器是否正常运行。此端点检查 存活度和 就绪度探测 的健康状况。

1. **/health/live** 端点检查 存活度 探测的当前健康状况。

```
http://HOST:9990/health/live
```

1. **/health/ready** 端点检查 就绪 探测的当前健康状况。

```
http://HOST:9990/health/ready
```

25.3.5. 探测没有定义时的全局状态

:empty-readiness-checks-status 和 **:empty-liveness-checks-status** 是管理属性，可在未定义就绪度 或 存活度 探测时指定全局状态。这些属性允许应用报告**"DOWN"**，直至其探测验证应用程序是否已就绪/处于活动状态。默认情况下，这些属性将报告 **'UP'**。

1. **:empty-readiness-checks-status** 属性指定 就绪 探测的全局状态（如果没有定义 就绪度探测 ）。

```
/subsystem=microprofile-health-smallrye:read-attribute(name=empty-readiness-checks-status)
{
    "outcome" => "success",
    "result" => expression "${env.MP_HEALTH_EMPTY_READINESS_CHECKS_STATUS:UP}"
}
```

1. **:empty-liveness-checks-status'attribute** 指定如果没有定义 存活度 探测的全局状态。

```
/subsystem=microprofile-health-smallrye:read-attribute(name=empty-liveness-checks-status)
{
    "outcome" => "success",
    "result" => expression "${env.MP_HEALTH_EMPTY_LIVENESS_CHECKS_STATUS:UP}"
}
```

检查 就绪 度和存活度 探测的 **/health HTTP** 端点和 **:check** 操作也会考虑这些属性。

25.3.6. 为 HTTP 端点启用身份验证

健康 上下文可以被配置为要求用户被授权访问上下文。

在这个端点中启用身份验证：

1. 在 **microprofile -health-smallrye** 子系统上，将 **security- enabled** 属性设为 **true**。

```
/subsystem=microprofile-health-smallrye:write-attribute(name=security-enabled,value=true)
```

2. 重新加载服务器以使更改生效。

```
reload
```

随后尝试访问 健康 端点会导致身份验证提示符。

25.4. ECLIPSE MICROPROFILE 指标

重要

Eclipse MicroProfile 指标仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

25.4.1. 关于 MicroProfile 指标子系统

JBoss EAP 包含 SmallRye 指标 组件，它利用 microprofile-metrics-smallrye 子系统提供 Eclipse MicroProfile 指标 功能。此子系统用于为 JBoss EAP 实例提供监控数据，并且默认启用。

重要

The microprofile-metrics-smallrye 子系统仅在单机配置中启用。

25.4.2. 使用 HTTP 端点检查指标

指标在 JBoss EAP 管理界面上自动可用，包含以下上下文可用：

- /metrics/ - 包含 MicroProfile 3.0 规范中指定的指标。
- /metrics/vendor - 包含特定于供应商的指标，如内存池。

- */metrics/application - 包含来自使用 MicroProfile 指标 API 的部署和子系统的指标。*

JBoss EAP 子系统指标以 Prometheus 格式公开。

指标名称基于子系统和属性名称。例如，undertow 子系统为应用部署中的每个 servlet 公开指标属性 request-count。此指标的名称为 jboss_undertow_request_count。前缀 jboss 有助于识别指标来源。

要查看 EAP 公开的指标列表，可在 CLI 中执行以下命令：

```
$ curl -v http://localhost:9990/metrics | grep -i type
```

示例：获取 JAX-RS 应用的请求数

以下示例演示了如何查找向 JBoss EAP 上部署的 Web 服务发出的请求数。该示例使用 helloworld-rs quickstart 作为 Web 服务。下载快速入门自：jboss-eap-quickstarts.

在 helloworld -rs 快速启动中检查 request-count 指标：

1. *为 undertow 子系统启用统计信息：*
   - *在启用了统计信息的情况下启动单机服务器：*

     ```
     $ ./standalone.sh -Dwildfly.statistics-enabled=true
     ```
   - *对于已经运行的服务器，启用 undertow 子系统的统计信息：*

     ```
     /subsystem=undertow/:write-attribute(name=statistics-enabled,value=true)
     ```
2. *部署 helloworld-rs 快速入门：*
   - *在快速启动的根目录中，使用 Maven 部署 Web 应用程序：*

     ```
     $ mvn clean install wildfly:deploy
     ```

3. 使用 **curl** 命令在 **CLI** 中查询 **http** 端点，并为 **request_count** 过滤：

```
$ curl -v http://localhost:9990/metrics |  grep request_count
```

输出：

```
jboss_undertow_request_count_total{server="default-server",http_listener="default",} 0.0
```

返回的属性值为 **0.0**。

4. 在 **Web** 浏览器中访问位于 **http://localhost:8080/helloworld-rs/** 的快速入门，再单击任何链接。

5. 通过 **CLI** 再次查询 **http** 端点：

```
$ curl -v http://localhost:9990/metrics |  grep request_count
```

输出：

```
jboss_undertow_request_count_total{server="default-server",http_listener="default",} 1.0
```

该值更新至 **1.0**。

您可以通过重复最后两个步骤来验证请求数是否正确更新。

25.4.3. 使用管理控制台配置 MicroProfile 指标

在管理控制台中，您可以执行以下配置：

- 启用或禁用公开指标。
- 编辑公开的指标的前缀。

- 启用或禁用安全性。
- 将非必需字段重置为初始或默认值。

使用管理控制台配置 **MicroProfile** 指标：

- 访问管理控制台，再导航到 **Configuration → Subsystems → MicroProfile Metrics**，然后点击 **View** 打开 **MicroProfile Metrics** 页面。

25.4.4. 为 HTTP 端点启用身份验证

可以将 指标 上下文和所有子上下文配置为要求用户有权访问该上下文。以下流程在这个端点中启用身份验证。

1. 在 **microprofile -metrics-smallrye** 子系统上，将 **security- enabled** 属性设为 **true**。

```
/subsystem=microprofile-metrics-smallrye:write-attribute(name=security-enabled,value=true)
```

2. 重新加载服务器以使更改生效。

```
reload
```

之后任何访问 指标 端点的尝试都将产生身份验证提示。

附录 A. 参考资料

A.1. 服务器运行时参数

应用服务器启动脚本在运行时接受参数和交换机。这允许服务器在 **standalone.xml**、**domain.xml**和 **host.xml** 配置文件中定义的配置的替代配置下启动。

其他配置可能包括启动服务器，并设置替代套接字绑定或辅助配置。

可以通过在启动时传递帮助开关 **-h** 或 **--help** 来访问可用的参数列表。

表 **A.1.** 运行时切换和参数

参数或切换	操作模式	描述
--admin-only	standalone	将服务器的运行类型设置为 **ADMIN_ONLY**。这将导致它打开管理接口并接受管理请求，但不能启动其他运行时服务或接受最终用户请求。请注意，建议您改为使用 **--start-mode=admin-only**。
--admin-only	域	将主机控制器的运行类型设置为 **ADMIN_ONLY**，导致它打开管理接口并接受管理请求，但不启动服务器；如果此主机控制器是域的主设备，则接受从属主机控制器的传入连接。
-b=<value>, -b <value>	独立，域	设置系统属性 **jboss.bind.address**，用于为公共接口配置绑定地址。如果未指定值，则默认为 **127.0.0.1**。有关为其他接口设置绑定地址，请参阅 **-b<interface>=<value>** 条目。
-b<interface>=<value>	独立，域	将系统属性 **jboss.bind.address.<interface>** 设置为给定值。例如，**-bmanagement=*IP_ADDRESS***
--backup	域	即使此主机不是域控制器，也保留永久域配置的副本。
-c=<config>, -c <config>	standalone	要使用的服务器配置文件的名称。默认值为 **standalone.xml**。
-c=<config>, -c <config>	域	要使用的服务器配置文件的名称。默认值为 **domain.xml**。
--cached-dc	域	如果主机不是域控制器，并且无法在引导时联系域控制器，则使用域配置的本地缓存副本引导。

参数或切换	操作模式	描述
--debug [<port>]	standalone	使用可选参数激活调试模式以指定端口。只有启动脚本支持时才起作用。
-D<name>[=<value>]	独立，域	设置系统属性.
--domain-config=<config>	域	要使用的服务器配置文件的名称。默认值为 **domain.xml**。
--git-repo	standalone	用于管理和持久服务器配置数据的 Git 存储库的位置。如果要在本地存储， 或者远程存储库的 URL，这可能是本地的。
--git-branch	standalone	Git 存储库中要使用的分支或标签名称。*此参数应命名现有的分支或标签名称，因为如果不存在，则不会创建该分支或标签名称*。如果使用标签名称，请将存储库置于分离的 HEAD 状态，这意味着以后的提交不会附加到任何分支。标签名称为只读，通常在多个节点之间复制配置时使用。
--git-auth	standalone	指向 Elytron 配置文件的 URL，该文件包含连接远程 Git 存储库时要使用的凭据。如果您的远程 Git 存储库需要身份验证，则需要此参数。虽然 Git 支持 SSH 身份验证，但 Elytron 不支持；因此，目前只能指定要通过 HTTP 或 HTTPS 进行身份验证的凭据，而不是通过 SSH 进行身份验证。此参数不与 **本地存储库** 一起使用。
-h, --help	独立，域	显示帮助消息并退出。
--host-config=<config>	域	要使用的主机配置文件的名称。默认值为 **host.xml**。
--interprocess-hc-address=<address>	域	主机控制器应侦听进程控制器通信的地址。
--interprocess-hc-port=<port>	域	主机控制器应侦听进程控制器通信的端口。
--master-address=<address>	域	将系统属性 **jboss.domain.master.address** 设置为给定的值。在默认的从属主机控制器配置中，这用于配置主控主机控制器的地址。
--master-port=<port>	域	将系统属性 **jboss.domain.master.port** 设置为给定的值。在默认的从属主机控制器配置中，这用于配置供主主机控制器用于原生管理通信的端口。
--read-only-server-config=<config>	standalone	要使用的服务器配置文件的名称。这与 **--server-config** 和 **-c** 不同，原始文件不会被覆盖。

参数或切换	操作模式	描述
--read-only-domain-config=<config>	域	要使用的域配置文件的名称。这与 **--domain-config** 和 **-c** 不同，因为初始文件不会被覆盖。
--read-only-host-config=<config>	域	要使用的主机配置文件的名称。这与 **--host-config** 的不同之处在于，初始文件永远不会被覆盖。
-P=<url>, -P <url>, --properties=<url>	独立，域	从指定的 URL 加载系统属性。
--pc-address=<address>	域	进程控制器侦听来自其控制的进程的通信的地址。
--pc-port=<port>	域	进程控制器在其上侦听其控制进程的通信的端口。
-S<name>[=<value>]	standalone	设置安全属性。
-secmgr	独立，域	运行安装有安全管理器的服务器。
--server-config=<config>	standalone	要使用的服务器配置文件的名称。默认值为 **standalone.xml**。
--start-mode=<mode>	standalone	设置服务器的启动模式。这个选项不能与 **--admin-only** 一起使用。有效值为： • **Normal** ：服务器将正常启动。 • 仅管理 **：** 服务器将仅打开管理接口并接受管理请求，而不启动其他运行时服务或接受最终用户请求。 • **暂停** ：服务器将以暂停模式启动，并且在恢复之前不会服务请求。
-u=<value>, -u <value>	独立，域	设置系统属性 **jboss.default.multicast.address**，用于在配置文件中的 socket-binding 元素中配置多播地址。如果没有指定值，则默认为 **230.0.0.4**。
-v, -V, --version	独立，域	显示应用服务器版本并退出。

警告

JBoss EAP 附带的配置文件已设置为处理交换机的行为，例如 -b 和 -u。如果您将配置文件更改为不再使用交换机控制的系统属性，那么将其添加到启动命令将无效。

A.2. RPM 服务配置文件

与 ZIP 或安装程序安装相比，JBoss EAP 的 RPM 安装包括两个额外的配置文件：这些文件供服务初始化脚本用于指定 JBoss EAP 启动环境。这些服务配置文件的位置与 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 及更新的版本不同。

重要

对于 Red Hat Enterprise Linux 7 及更新的版本，使用 systemd 加载 RPM 服务配置文件，因此不会扩展变量表达式。

表 A.2. Red Hat Enterprise Linux 6 的 RPM 配置文件

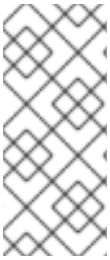
File	描述
/etc/sysconfig/eap7-standalone	特定于在红帽企业 Linux 6 上独立 JBoss EAP 服务器的设置.
/etc/sysconfig/eap7-domain	特定于 JBoss EAP 的设置，作为受管域在红帽企业 Linux 6 上运行.

表 A.3. Red Hat Enterprise Linux 7 及更新的版本的 RPM 配置文件

File	描述
/etc/opt/rh/eap7/wildfly/eap7-standalone.conf	特定于在红帽企业 Linux 7 和更高版本上独立 JBoss EAP 服务器的设置.
/etc/opt/rh/eap7/wildfly/eap7-domain.conf	特定于 JBoss EAP 的设置，作为受管域在红帽企业 Linux 7 及更高版本上运行.

A.3. RPM 服务配置属性

*下表显示了 **JBoss EAP RPM** 服务的可用配置属性及其默认值的列表。*



注意

*如果在 **RPM** 服务配置文件中具有相同的名称（如 **/etc/sysconfig/eap7-standalone** ），并且在 **JBoss EAP** 启动配置文件中，如 **EAP_HOME/bin/standalone.conf**，则优先的值就是 **JBoss EAP** 启动配置文件中的值。个这样的属性是 **JAVA_HOME**。*

*表 **A.4. RPM** 服务配置属性*

属性	描述
JAVA_HOME	安装 Java 运行时环境的目录。 默认值： **/usr/lib/jvm/jre**
JAVAPTH	安装 Java 可执行文件的路径。 默认值： **$JAVA_HOME/bin**
WILDFLY_STARTUP_WAIT	初始化脚本将等待的秒数，直到在收到 start 或 restart 命令后确认服务器启动成功为止。此属性仅适用于 Red Hat Enterprise Linux 6。 默认值： **60**
WILDFLY_SHUTDOWN_WAIT	在接收停止或重新启动命令之前，初始化脚本将等待服务器关闭的秒数。此属性仅适用于 Red Hat Enterprise Linux 6。 默认值为： **20**
WILDFLY_CONSOLE_LOG	将重定向到 CONSOLE 日志处理程序的文件。 默认值：单机服务器的 **/var/opt/rh/eap7/log/wildfly/standalone/console.log**，或者受管域的 **/var/opt/rh/eap7/log/wildfly/domain/console.log**。
WILDFLY_SH	用于启动至 JBoss EAP 服务器的 脚本。 默认值：单机服务器的 **/opt/rh/eap7/root/usr/share/wildfly/bin/standalone.sh** 或 **/opt/rh/eap7/root/usr/share/wildfly/bin/domain.sh**。
WILDFLY_SERVER_CONFIG	要使用的服务器配置文件。 此属性没有默认值。启动时可以定义 **standalone. xml 或 domain**.xml。
WILDFLY_HOST_CONFIG	对于受管域，此属性允许用户指定主机配置文件，如 **host.xml**。它没有设置为默认值。

属性	描述
WILDFLY_MODULEPATH	JBoss EAP 模块目录的路径。 默认值： **/opt/rh/eap7/root/usr/share/wildfly/modules**
WILDFLY_BIND	设置 **jboss.bind.address** 系统属性，用于配置公共接口的绑定地址。如果没有指定值，则默认为 **0.0.0.0**。
WILDFLY_OPTS	启动时要包含 的其他参数.例如： -Dorg.wildfly.openssl.path=*PATH_TO_OPENSSL_LIBS*

A.4. JBOSS EAP 子系统概述

*下表简要描述了 **JBoss EAP** 子系统。*

*表 **A.5. JBoss EAP** 子系统*

JBoss EAP 子系统	描述
batch-jberet	配置运行批处理应用的环境， 并管理批处理作业。
bean-validation	配置 bean 验证以验证 Java 对象数据。
core-management	注册监听服务器生命周期事件的监听程序并跟踪 配置更改。
datasources	创建和管理数据源，以及 JDBC 数据库驱动程序。
deployment-scanner	配置部署扫描器，以监控要部署的应用的特定位置。
EE	在 Jakarta EE 平台中配置通用功能，如定义全局模块 、启用基于描述符的属性替换和配置默认绑定。
ejb3	配置企业 JavaBeans(EJB)，包括会话和消息驱动型 Bean. **ejb3** 子系统的更多信息，请参阅为 JBoss EAP *开发 EJB 应用*。
elytron	配置服务器和应用安全性. 有关 **elytron** 子系统的更多信息，请参阅 *JBoss* EAP 的安全架构。
iiop-openjdk	为 JTS 事务和其他 ORB 服务（包括安全性）配置通用对象请求代理架构 (CORBA)服务。在 JBoss EAP 6 中，此功能包含在**jacorb** 子系统中。
Infinispan	为 JBoss EAP 高可用性服务配置缓存功能。

JBoss EAP 子系统	描述
io	定义供其他子系统使用的 worker 和缓冲区池。
jaxrs	启用 JAX-RS 应用的部署和功能。
jca	配置 JCA 容器和资源适配器部署的常规设置。雅加达等效者是 Jakarta Connectors。
jdr	启用诊断数据的收集，以帮助进行故障排除。JBoss EAP 订阅者可在请求支持时向红帽提供此信息。
jgroups	为群集中的服务器互相通信配置协议堆栈和通信机制。
jmx	配置远程 Java 管理扩展(JMX)访问。
jpa	管理 Jakarta Persistence 2.2 容器管理要求，并允许您部署持久的单元定义、注解和描述符。 **jpa** 子系统的更多信息，请参阅 JBoss EAP *开发指南*。
jSF	管理 JavaServer Faces(JSF)实施。Jakarta 等效于 Jakarta Server Faces。
jsr77	提供根据 Jakarta 管理规范定义的 Jakarta EE 管理功能。
logging	通过日志类别和日志 处理程序系统配置系统和应用级日志记录。
mail	配置邮件服务器属性和 自定义邮件传输，以创建邮件服务，允许部署到 JBoss EAP 的应用使用该服务发送邮件。
messaging-activemq	配置集成消息传递提供商 Artemis 的 JMS 目的地、连接工厂和其他设置。在 JBoss EAP 6 中，消息传递功能包含在 **messaging** 子系统中。 有关 **messaging-activemq** 子系统的更多信息，*请参阅配置 JBoss EAP 的消息传递*。
microprofile-config-smallrye	使用 MicroProfile Config SmallRye 提供配置数据的可移植外部化，允许应用在无需重新启动服务器的情况下访问最新的配置属性。  **重要** Eclipse Microprofile Config 仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。 如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

JBoss EAP 子系统	描述
microprofile-health-smallrye	使用 SmallRye Health 来监控服务器健康状况。如需有关 **microprofile-health-smallrye** 子系统的信息，请参阅使用 Eclipse MicroProfile 健康的监控服务器 健康状况以及如何进行配置。*有关创建自定义健康检查的信息，请参阅《开发指南》*中的使用 Eclipse MicroProfile 健康来监控服务器健康状况。  **重要** Eclipse Microprofile Health 仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。 如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。
microprofile-opentracing-smallrye	使用 SmallRye OpenTracing 跟踪服务边界的请求。如需了解 有关 **microprofile-opentracing-smallrye 子系统的信息，请参阅 MicroProfile OpenTracing** SmallRye 子系统追踪请求。有关如何自定义 CDI Bean 和 JAX-RS *端点追踪的信息，请参阅《开发指南 https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/development_guide/#using_microprofile_opentracing_smallrye_tracer 》中使用 Eclipse MicroProfile OpenTracing to Trace 请求*。  **重要** Eclipse Microprofile OpenTracing 仅作为技术预览提供。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。 如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。
modcluster	配置服务器端 mod_cluster 工作程序节点。
命名	将条目绑定到全局 JNDI 命名空间，并配置远程 JNDI 接口。
picketlink-federation	配置 PicketLink 基于 SAML 的单点登录(SSO)。 有关 picket **link-federation** 子系统的更多信息，请参阅 *如何为 JBoss EAP 设置 SSO 和 SAML v2*。
picketlink-identity-management	配置 PicketLink 身份管理服务。不支持此子系统。

JBoss EAP 子系统	描述
pojo	支持部署包含 JBoss Microcontainer 服务的应用，如之前版本的 JBoss EAP 所支持。
远程	为本地和远程服务的入站和出站连接配置设置。
discovery	发现子系统目前仅用于内部子系统；它是私有 API，不可用于公共用途。
request-controller	配置设置以正常暂停和关闭服务器.
resource-adapters	使用 Jakarta Connectors 规范配置和维护资源适配器，以在 Jakarta EE 应用和企业信息系统(EIS)之间进行通信。
rTS	REST-AT 不受支持的实施.
sar	支持部署包含 MBean 服务的 SAR 存档，如之前版本的 JBoss EAP 所支持。
security	用于配置应用安全设置的传统方法. **有关安全** 子系统的更多信息，请参阅 *JBoss* EAP 的安全架构。
security-manager	配置 Java 安全管理器要使用的 Java 安全策略。 如需有关 **security-manager** 子系统的更多信息，请参阅 *如何为 JBoss EAP 配置服务器* 安全性。
singleton	定义单例策略，以配置单例部署的行为或创建单例 MSC 服务。 有关 **singleton** 子系统的更多信息，请参阅 JBoss EAP *开发指南*。
事务	配置事务管理器(TM)选项，如超时值、事务记录以及是否使用 Java 事务服务(JTS)。 有关 **事务** 子系统的更多信息，请参阅为 JBoss EAP *管理 JBoss EAP 事务*。
Undertow	配置 JBoss EAP 的 Web 服务器和 servlet 容器设置。在 JBoss EAP 6 中，此功能包含在 **web** 子系统中。
Web 服务	配置公布的端点地址和端点处理程序链，以及 Web 服务提供商的主机名、端口和 WSDL 地址。 有关 **Webservices** 子系统的更多信息，请参阅为 JBoss EAP *开发 Web 服务应用*。
weld	为 JBoss EAP 配置上下文和依赖注入(CDI)功能.
XTS	配置用于协调事务中的 Web 服务的设置。

A.5. 附加用户实用程序参数

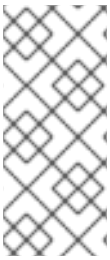
*下表描述了适用于 **add-user.sh** 或 **add-user.bat** 脚本的参数，它是将新用户添加到用于开箱即用身份验证的属性文件中的实用程序。*

*表 **A.6.** 附加用户命令参数*

命令行参数	描述
-a	在应用域中创建用户。如果省略，则默认设置是在管理域中创建用户。
-dc <value>	将包含属性文件的域配置目录。如果省略，默认目录为 ***EAP_HOME*/domain/configuration/**。
-SC <value>	将包含属性文件的替代单机服务器配置目录。如果省略，默认目录为 ***EAP_HOME*/standalone/configuration/**。
-up, --user-properties <value>	备用用户属性文件的名称。它可以是绝对路径，也可以是与 **-sc 或 - dc** 参数一起使用的文件名，该参数指定替代配置目录。
-g, --group <value>	要分配给此用户的组的逗号分隔列表。
-GP, --group-properties <value>	备用组属性文件的名称。它可以是绝对路径，也可以是与 **-sc 或 - dc** 参数一起使用的文件名，该参数指定替代配置目录。
-p, --password <value>	用户的密码。
-u, --user <value>	用户名称。用户名只能以任何数字和顺序包含以下字符： • 字母数字字符（a-z、A-Z、0-9） • 短划线(-)、句点(.)、逗号(@) • 反斜杠(\) • Equals (=)
-r, --realm <value>	用于保护管理接口的域名称。如果省略，默认值为 **ManagementRealm**。
-s, --silent	运行对控制台不带输出的 **add-user** 脚本。
-e, --enable	启用 用户。
-d, --disable	禁用 用户。
-cw, --confirm-warning	以交互模式自动确认警告.

命令行参数	描述
-h, --help	显示 **add-user** 脚本的使用信息。
-DS, --display-secret	以非交互模式打印机密值。

A.6. 管理审计日志记录属性



注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.7.** 管理审计日志记录：日志记录器属性*

属性	描述
enabled	是否启用审计日志记录。
log-boot	是否应在服务器引导中记录操作。
log-read-only	无论是不修改配置的操作还是任何运行时服务，都应记录下来。

*表 **A.8.** 管理审计日志记录：日志格式条件属性*

属性	描述
compact	如果为 **true**，它将在一个行上格式化 JSON。可能仍然包含新行的值；因此，如果将转义 **行或转义 控制字符设置为 true，请在一行上** 包含整个记录。
date-format	**java.text.SimpleDateFormat** 所了解的日期格式。如果 **include-date** 设为 **false，则会忽略此项**。
date-separator	日期和格式化日志消息其余部分之间的分隔符。如果 **include-date** 设为 **false，则会忽略此项**。
escape-control-characters	如果为 **true**，它将转义所有控制字符，ASCII 条目的十进制值大于 **32**，以八进制表示 ASCII 代码。例如，新行变为 **#012**。如果为 **true**，这将覆盖 **转义新行=false**。
escape-new-line	如果为 **true**，它将转义所有带有八进制 ASCII 代码的新行：**#012**。

属性	描述
include-date	是否在格式化的日志记录中包含日期。

*表 **A.9.** 管理审计日志记录：文件处理程序属性*

属性	描述
disabled-due-to-failure	是否因为日志记录失败而禁用了此处理程序（只读）。
failure-count	处理程序初始化后的日志失败数量（只读）。
formatter	用于格式化日志消息的 JSON 格式器。
max-failure-count	禁用此处理程序前的最大日志记录失败次数。
路径	审计日志文件的路径。
relative-to	之前命名的路径的名称，或者系统提供的一个标准路径的名称。如果提供了 **relative-to**，则 **path** 属性的值将被视为此属性指定的路径的相对值。
rotate-at-startup	是否应在服务器启动时轮转旧日志文件。

*表 **A.10.** 管理审计日志记录：**Syslog** 处理程序属性*

属性	描述
app-name	要添加到 syslog 记录的应用程序名称，如 *RFC-5424* 第 6.2.5 节中所定义。如果未指定，它将默认为产品名称。
disabled-due-to-failure	是否因为日志记录失败而禁用了此处理程序（只读）。
功能	*按照 RFC-5424 以及 RFC-3164 第 4.1.1 节所述，用于 syslog 记录的功能。*
failure-count	处理程序初始化后的日志失败数量（只读）。
formatter	用于格式化日志消息的 JSON 格式器。
max-failure-count	禁用此处理程序前的最大日志记录失败次数。
max-length	允许日志消息（包括标头）的最大长度，以字节为单位。如果未定义，如果 **syslog-format** 为 **RFC3164**，则默认为 **1024** 字节，如果 **syslog-format** 为 **RFC5424**，则默认为 **2048** 字节。

属性	描述
协议	用于 syslog 处理程序的协议。必须是其中一个，并且只能是一个 a **udp**、**tcp** 或 **tls**。
syslog-format	syslog 格式： **RFC5424** 或 **RFC3164**。
truncate	如果以字节为单位的长度大于 **max-length** 属性的值，是否应截断消息（包括标头）。如果设置为 **false**，则消息将被分割并使用相同的标头值发送。

注意

*系统日志服务器的实施各不相同，因此并非所有设置都适用于所有 **syslog** 服务器。已使用 **rsyslog syslog** 实施进行了测试。*

此表仅列出高级别属性。每一属性具有配置参数，另一些则具有子配置参数。

A.7. 接口属性

注意

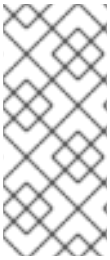
*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.11.** 接口属性和值*

接口元素	描述
任意	表明接口选择条件的部分内容应当是它至少满足一组嵌套条件，但不一定全部满足。
any-address	表示使用此接口的套接字应绑定到通配符地址的空元素。将使用 IPv6 通配符地址(**::**)，除非 **java.net.preferIPv4Stack** 系统属性设为 true，否则将使用 IPv4 通配符地址(**0.0.0.0**)。如果套接字绑定到双栈计算机上的 IPv6 任意本地地址，则它可以接受 IPv6 和 IPv4 通信；如果套接字绑定到 IPv4（IPv4 映射）本地地址，则它只能接受 IPv4 流量。
inet-address	IPv6 或 IPv4 中的 IP 地址增量十进制表示法，或者解析为 IP 地址的主机名。

接口元素	描述
link-local-address	表示接口的部分选择条件的空元素应当是与其关联的地址是否为本地链接。
环回	表明接口的部分选择条件的空元素应该是它是否为环回接口。
loopback-address	可能不会在计算机的回环接口上实际配置的环回地址。与 inet-address 类型不同，即使找不到具有与其关联的 IP 地址的 NIC，也会使用给定值。
多播	表明接口选择条件的部分空白元素应该是它是否支持多播。
name	接口的名称。
nic	网络接口的名称（如 eth0、eth1、lo）。
nic-match	个正则表达式，可以匹配计算机上可用的网络接口名称来查找可接受的接口。
not	表明接口选择条件的部分元素应当是它不符合任何嵌套条件集。
point-to-point	表明接口的部分选择条件的空元素应该是它是否为点对点接口。
public-address	表示接口的部分选择条件的空元素应当是它是否具有可公开路由的地址。
site-local-address	表示接口的部分选择条件为与它关联的地址是否本地的空元素。
subnet-match	以 *斜杠表示法* 编写的网络 IP 地址和地址网络前缀中的位数，例如 **192.168.0.0/16**。
up	表示接口选择条件的部分为空的元素应当是当前是否启动。
虚拟	表示接口的部分选择条件的空元素应当是它是否为虚拟接口。

A.8. 套接字绑定属性



注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

下表显示了可以为三种套接字绑定中的每种类型配置的属性：

- ***socket-binding***
- ***remote-destination-outbound-socket-binding***
- ***local-destination-outbound-socket-binding***

*表 **A.12.** 入站套接字绑定**(socket-binding)Attributes***

属性	描述
client-mappings	指定此套接字绑定的客户端映射。连接到此套接字的客户端应使用与所需的出站接口匹配的映射中指定的目标地址。这允许使用网络地址转换或绑定多个网络接口的高级网络拓扑。每个映射应当按照声明的顺序进行评估，第一个成功匹配项用于确定目的地。
Fix-port	即使将数值偏移应用到套接字组中的其他套接字，端口值是否应保持固定。
Interface	套接字应绑定到的接口名称，或者对于多播套接字，则为它应侦听的接口。这应该是声明的接口之一。如果未定义，则将使用括起套接字绑定组中 **default-interface** 属性的值。
multicast-address	套接字应接收多播流量的多播地址。如果未指定，套接字将不会配置为接收多播。
多播端口	套接字应接收多播流量的端口。如果配置了 **多播地址，** 则必须配置多播地址。
name	套接字的名称。需要访问套接字配置信息的服务将使用此名称找到它。此属性是必需的。
port	套接字应绑定到的端口号。请注意，如果服务器应用 port-offset 以递增或减少所有端口值，则可以覆盖此值。

*表 **A.13.** 远程出站套接字绑定**(remote-destination-outbound-socket-binding)Attributes***

属性	描述
fixed-source-port	即使将数字偏移应用到套接字组中的其他出站套接字，端口值是否应保持固定。
主机	此出站套接字将连接到的远程目标的主机名或 IP 地址。
port	出站套接字应连接到的远程目的地的端口号。

属性	描述
source-interface	用于出站套接字的源地址的接口名称。
source-port	用作出站套接字源端口的端口号。

表 A.14. 本地出站套接字绑定(local-destination-outbound-socket-binding)Attributes

属性	描述
fixed-source-port	即使将数字偏移应用到套接字组中的其他出站套接字，端口值是否应保持固定。
socket-binding-ref	用于确定此出站套接字连接到的端口的本地套接字绑定名称。
source-interface	用于出站套接字的源地址的接口名称。
source-port	用作出站套接字源端口的端口号。

A.9. 默认套接字绑定

下表显示了每个套接字绑定组的默认套接字绑定。

- *标准套接字*
- ***ha-sockets***
- *全套接字*
- ***full-ha-sockets***
- ***load-balancer-sockets***

表 A.15. 标准套接字

套接字绑定	端口	描述
ajp	8009	Apache JServ 协议.用于 HTTP 集群和负载平衡。

套接字绑定	端口	描述
http	8080	部署 Web 应用的默认端口。
https	8443	部署的 Web 应用程序和客户端之间的 SSL 加密连接。
management-http	9990	用于与管理层的 HTTP 通信。
management-https	9993	用于与管理层通信的 HTTPS。
txn-recovery-environment	4712	JTA 事务恢复管理器.
txn-status-manager	4713	JTA/JTS 事务管理器.

*表 **A.16. ha-sockets***

套接字绑定	端口	多播端口	描述
ajp	8009		Apache JServ 协议.用于 HTTP 集群和负载平衡。
http	8080		部署 Web 应用的默认端口。
https	8443		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
jgroups-mping		45700	多播.用于发现 HA 集群中的初始成员资格。
jgroups-tcp	7600		使用 TCP 在 HA 集群中单播对等发现.
jgroups-udp	55200	45688	使用 UDP 在 HA 集群中进行多播对等发现.
management-http	9990		用于与管理层的 HTTP 通信。
management-https	9993		用于与管理层通信的 HTTPS。
modcluster		23364	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。
txn-recovery-environment	4712		JTA 事务恢复管理器.
txn-status-manager	4713		JTA/JTS 事务管理器.

*表 **A.17. 全套接字***

套接字绑定	端口	描述
ajp	8009	Apache JServ 协议.用于 HTTP 集群和负载平衡。
http	8080	部署 Web 应用的默认端口。
https	8443	部署的 Web 应用程序和客户端之间的 SSL 加密连接。
IIOP	3528	用于 JTS 事务和其他 ORB 依赖服务的 CORBA 服务.
iiop-ssl	3529	SSL 加密的 CORBA 服务.
management-http	9990	用于与管理层的 HTTP 通信。
management-https	9993	用于与管理层通信的 HTTPS。
txn-recovery-environment	4712	JTA 事务恢复管理器.
txn-status-manager	4713	JTA/JTS 事务管理器.

*表 **A.18. full-ha-sockets***

名称	端口	多播端口	描述
ajp	8009		Apache JServ 协议.用于 HTTP 集群和负载平衡。
http	8080		部署 Web 应用的默认端口。
https	8443		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
IIOP	3528		用于 JTS 事务和其他 ORB 依赖服务的 CORBA 服务.
iiop-ssl	3529		SSL 加密的 CORBA 服务.
jgroups-mping		45700	多播.用于发现 HA 集群中的初始成员资格。
jgroups-tcp	7600		使用 TCP 在 HA 集群中单播对等发现.
jgroups-udp	55200	45688	使用 UDP 在 HA 集群中进行多播对等发现.
management-http	9990		用于与管理层的 HTTP 通信。

名称	端口	多播端口	描述
management-https	9993		用于与管理层通信的 HTTPS。
modcluster		23364	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。
txn-recovery-environment	4712		JTA 事务恢复管理器.
txn-status-manager	4713		JTA/JTS 事务管理器.

*表 **A.19. load-balancer-sockets***

名称	端口	多播端口	描述
http	8080		部署 Web 应用的默认端口。
https	8443		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
management-http	9990		用于与管理层的 HTTP 通信。
management-https	9993		用于与管理层通信的 HTTPS。
mcmp-management	8090		用于传输生命周期事件的 Mod-Cluster Management 协议(MCMP)连接的端口。
modcluster		23364	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。

A.10. 模块命令参数

*以下参数可以传递给 模块 **add management CLI** 命令：*

*表 **A.20.** 模块命令参数*

参数	描述
--absolute-resources	使用此参数指定要从其 **module.xml** 文件中引用的绝对文件系统路径列表。指定的文件不会复制到模块目录中。 有关分隔符的详情，请参阅 **--resource-delimiter**。
--allow-nonexistent-resources	使用此参数为 **--resources** 指定的资源创建空目录，这些资源不存在。如果资源不存在并且未使用此参数，则 **模块 add** 命令会失败。

参数	描述
--dependencies	使用此参数提供此模块依赖的模块名称的逗号分隔列表。
--export-dependencies	使用此参数指定导出的依赖项。 module add --name=com.mysql --resources=*/path/to*/mysql-connector-java-8.0.12.jar --export-dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
--main-class	使用此参数指定声明模块主方法的完全限定类名称。
--module-root-dir	如果您已 定义了要使用的外部 JBoss EAP 模块目录，而不是默认的 ***EAP_HOME*/modules/** 目录，可使用此参数。 module add --module-root-dir=*/path/to*/my-external-modules/ --name=com.mysql --resources=*/path/to*/mysql-connector-java-8.0.12.jar --dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api
--module-xml	使用此参数提供要用于此新 **模块的 module.xml** 的文件系统路径。此文件复制到模块目录。如果未指定此参数，模块目录中将生成 **module.xml** 文件。
--name	使用此参数提供要添加的模块的名称。此参数是必需的。
--properties	使用此参数提供以逗号分隔的 ***PROPERTY_NAME=PROPERTY_VALUE*** 对列表来定义模块属性。
--resource-delimiter	使用此参数为提供给 **--resources 或 absolute-resources** 参数的资源列表设置用户定义的文件路径分隔符。如果没有设置，文件路径分隔符将为 Linux 的冒号(**:**)，以及用于 Windows 的分号(**;**)。
--resources	通过提供文件系统路径列表，使用此参数指定此模块的资源。这些文件复制到此模块目录中，并从其 **module.xml** 文件中引用。如果您提供目录的路径，目录及其内容将复制到模块目录中。符号链接不会被保留；链接的资源复制到模块目录中。除非提供了 **--absolute-resources 或 --module-xml,** 否则此参数是必需的。 有关分隔符的详情，请参阅 **--resource-delimiter**。
--slot	使用此参数将模块添加到默认 **主** 插槽以外的插槽中。 module add --name=com.mysql --slot=8.0 --resources=*/path/to*/mysql-connector-java-8.0.12.jar --dependencies=javaee.api,sun.jdk,ibm.jdk,javax.api,javax.transaction.api

A.11. DEPLOYMENT SCANNER MARKER 文件

标记文件供部署扫描程序用于在 **JBoss EAP** 服务器实例的部署目录中标记应用的状态。标志文件的名称与部署相同，文件后缀表示应用部署的状态。

例如，成功部署 **test-application.war** 将具有名为 **test-application.war.deployed** 的标志文件。

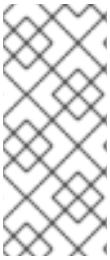
下表列出了可用的标志文件类型及其含义。

表 **A.21.** 标记文件类型

文件名后缀	Origin	描述
.deployed	系统生成的	表示内容已部署。如果删除了此文件，则其内容将被取消部署。
.dodeploy	用户生成的	表示应当部署或重新部署内容。
.failed	系统生成的	表示部署失败。标志文件包含有关失败原因的信息。如果删除了标志文件，则其内容将再次符合自动部署条件。
.isdeploying	系统生成的	表示部署正在进行中。此标志文件将在完成后删除。
.isundeploying	系统生成的	通过删除 a **.deployed** 文件触发，这表示内容正在取消部署。此标志文件将在完成后删除。
.pending	系统生成的	表示部署扫描器识别部署内容的需求，但目前存在一个问题阻止自动部署（例如，内容正在被复制）。*此标记充当全局部署路块，这意味着扫描程序不会指示服务器在该标志文件存在时部署或取消部署任何内容。*
.skipdeploy	用户生成的	禁用在存在时应用的自动部署。作为临时阻止自动部署爆炸式内容的方法，可以避免推送不完整的内容的风险。可以与压缩的内容一起使用，但扫描程序检测到已压缩内容的更改并等待完成。
.undeployed	系统生成的	表示内容已被取消部署。删除此标志文件不会影响内容重新部署。

A.12. DEPLOYMENT SCANNER ATTRIBUTES

部署扫描器包含以下可配置的属性：



注意

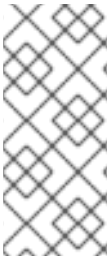
*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-deployment-scanner_2_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.22. Deployment Scanner Attributes***

名称	默认	描述
auto-deploy-exploded	false	允许自动部署展开式内容，而无需 a **.dodeploy** 标记文件。建议只在基本开发场景中使用，以防止开发人员或操作系统更改期间发生爆炸式应用部署。
auto-deploy-xml	true	允许自动部署 XML 内容，而无需 a **.dodeploy** 标记文件。
auto-deploy-zipped	true	允许自动部署压缩的内容，而无需 a **.dodeploy** 标记文件。
deployment-timeout	600	部署扫描程序的时间值（以秒为单位），以允许部署尝试然后被取消。
路径	部署	要扫描的实际文件系统路径。被视为绝对路径，除非指定了 **relative-to** 属性，在这种情况下，值被视为相对于该路径。
relative-to	jboss.server.base.dir	对定义为服务器配置中的路径的文件系统路径的引用。
runtime-failure-causes-rollback	false	部署运行时失败是否会导致回滚部署以及作为扫描操作一部分的所有其他（可能不相关）部署。
已启用 scan-enabled	true	通过扫描 **-间隔和启动时允许自动扫描** 应用程序.
scan-interval	5000	应扫描存储库以毫秒为单位的变化的时间间隔。值小于 **1** 会导致扫描仅在初始启动时发生。

A.13. 受管域 JVM 配置属性

*可以为主机、服务器组或服务器级别的受管域设置下列 **JVM** 配置选项：请注意，其中一些属性的有效值取决于您的 **JVM**。如需更多信息，请参阅您的 **JDK** 厂商文档。*



注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.23. JVM** 配置属性*

属性	描述
agent-lib	设置 **-agentlib java** 选项的值，该选项指定 Java 代理库。
agent-path	设置 **-agentpath java** 选项的值，该选项指定 Java 代理路径。
debug-enabled	是否启用 debug。此属性仅适用于服务器级别的 JVM 配置。
debug-options	指定启用 debug 时要使用的 JVM 选项。此属性仅适用于服务器级别的 JVM 配置。
env-classpath-ignored	是否忽略 **CLASSPATH** 环境变量。
environment-variables	指定键/值对环境变量.
heap-size	设置 **-Xms** 选项的值，该值指定 JVM 分配的初始堆大小。
java-agent	设置 **-javaagent java** 选项的值，该选项指定 Java 代理。
java-home	设置 **JAVA_HOME** 变量的值。
JVM-options	指定所需的任何其他 JVM 选项。
launch-command	在用于启动服务器进程的 **java 命令之前，指定要** 添加前缀的操作系统级命令。例如，您可以使用 **sudo** 命令以另一个用户身份运行 Java 进程。
max-heap-size	设置 **-Xmx** 选项的值，该值指定 JVM 分配的最大堆大小。
max-permgen-size	设置永久代的最大大小。*已弃用：JVM 不再提供单独的永久代空间。*
Permgen-size	设置初始永久生成大小。*已弃用：JVM 不再提供单独的永久代空间。*
stack-size	设置 **-Xss** 选项的值，用于指定 JVM 堆栈大小。
type	指定使用中的 JVM 的供应商。可用的选项有 **ORACLE**、**IBM**、**SUN** 或 **OTHER**。

A.14. 邮件子系统属性

下表描述了邮件 会话 的 **mail** 子系统中的属性和以下邮件服务器类型：

- ***imap***
- ***pop3***
- ***smtp***
- ***custom***

注意

这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 ***EAP_HOME/docs/schema/wildfly-mail_3_0.xsd*** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。

表 ***A.24.*** *邮件会话属性*

属性	描述
debug	是否启用 Jakarta Mail 调试。
来自	如果发送时未设置，则使用的默认"from"地址。
jndi-name	邮件会话应绑定到的 JNDI 名称。

表 ***A.25. IMAP*** *邮件服务器属性*

属性	描述
credential-reference	凭据（来自凭据存储），用于在服务器上进行身份验证。
outbound-socket-binding-ref	引用邮件服务器的出站套接字绑定。
password	要在服务器上进行身份验证的密码。
ssl	服务器是否需要 SSL。

属性	描述
tls	服务器是否需要 TLS。
username	要在服务器上进行身份验证的用户名。

*表 **A.26. POP3** 邮件服务器属性*

属性	描述
credential-reference	凭据（来自凭据存储），用于在服务器上进行身份验证。
outbound-socket-binding-ref	引用邮件服务器的出站套接字绑定。
password	要在服务器上进行身份验证的密码。
ssl	服务器是否需要 SSL。
tls	服务器是否需要 TLS。
username	要在服务器上进行身份验证的用户名。

*表 **A.27. SMTP** 邮件服务器属性*

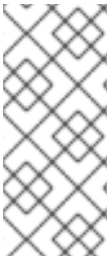
属性	描述
credential-reference	凭据，以存储服务器上进行身份验证。
outbound-socket-binding-ref	引用邮件服务器的出站套接字绑定。
password	要在服务器上进行身份验证的密码。
ssl	服务器是否需要 SSL。
tls	服务器是否需要 TLS。
username	要在服务器上进行身份验证的用户名。

*表 **A.28.** 自定义邮件服务器属性*

属性	描述
credential-reference	凭据（来自凭据存储），用于在服务器上进行身份验证。

属性	描述
outbound-socket-binding-ref	引用邮件服务器的出站套接字绑定。
password	要在服务器上进行身份验证的密码。
属性	此服务器的 Jakarta Mail 属性。
ssl	服务器是否需要 SSL。
tls	服务器是否需要 TLS。
username	要在服务器上进行身份验证的用户名。

***A.15.** 根日志记录器属性*



注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-logging_3_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能有所不同。*

*表 **A.29.** 根日志记录器属性*

属性	描述
filter	定义简单的过滤器类型。*弃用了 **filter-spec**。*
filter-spec	定义过滤器的表达式值。以下表达式定义了一个过滤器，它排除与模式 *不匹配* 的日志条目：not **（match("WFLY.*")）**
处理程序	根日志记录器使用的日志处理程序列表。
level	根日志记录器记录的日志消息级别最低。

注意

*为根日志记录器指定的 **filter-spec** 不会被 其他处理程序继承。相反，每个处理程序都必须指定 **filter-spec**。*

***A.16.** 日志类别属性*

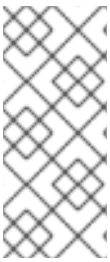
注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-logging_6_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能有所不同。*

*表 **A.30.** 日志类别属性*

属性	描述
类别	从中捕获日志消息的日志类别。
filter	定义简单的过滤器类型。*弃用了 **filter-spec**。*
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
处理程序	与日志记录器关联的日志处理程序列表。
level	日志类别记录的日志消息级别最低。
use-parent-handlers	如果设置为 **true**，此类别除任何其他分配的处理程序外，还将使用根日志记录器的日志处理程序。

A.17. 日志处理程序属性



注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-logging_6_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能有所不同。*

*表 **A.31.** 控制台日志处理程序属性*

属性	描述
autoflush	如果设为 **true**，则日志消息会在收到后立即发送到分配的处理程序。
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
filter	定义简单的过滤器类型。*弃用了 **filter-spec**。*

属性	描述
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。
目标	发送日志处理程序输出的系统输出流。可以是以下之一： • **system.err** ：日志处理程序输出转至系统错误流。 • **system.out** ：日志处理程序输出转至标准输出流。 • **控制台** ：日志手输出转至 **java.io.PrintWriter** 类。

*表 **A.32.** 文件日志处理程序属性*

属性	描述
附加	如果设置为 **true**，则 此处理程序编写的所有消息都会附加到 文件（如果它已存在）。如果设置为 **false**，则每次应用服务器启动时都会创建一个新文件。
autoflush	如果设为 **true**，则日志消息会在收到后立即发送到分配的处理程序。
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
file	代表将此日志处理程序的输出写入的文件的对象。它有两个配置属性： **relative-to** 和 **path**。
filter	定义简单的过滤器类型。*弃用了* ***filter-spec***。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。

属性	描述
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。

*表 **A.33.** 定期日志处理程序属性*

属性	描述
附加	如果设置为 **true，** 则 此处理程序编写的所有消息都会附加到 文件（如果它已存在）。如果设置为 **false**，则每次应用服务器启动时都会创建一个新文件。
autoflush	如果设为 **true**，则日志消息会在收到后立即发送到分配的处理程序。
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
file	代表此日志处理程序输出所写入的文件的对象。它有两个配置属性： **relative-to** 和 **path**。
filter	定义简单的过滤器类型。*弃用了* ***filter-spec***。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**。
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。
suffix	此字符串包含在附加到轮转日志的后缀中。**后缀** 的格式是点(**.**)，后跟一个日期字符串，它可以被 **SimpleDateFormat** 类解析。

*表 **A.34.** 日志处理程序属性大小*

属性	描述
附加	如果设置为 **true，** 则 此处理程序编写的所有消息都会附加到 文件（如果它已存在）。如果设置为 **false**，则每次应用服务器启动时都会创建一个新文件。
autoflush	如果设为 **true**，则日志消息将在收到后立即发送到分配的处理程序。

属性	描述
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
file	代表写入此日志处理程序的输出的文件的对象。它有两个配置属性： **relative-to** 和 **path**。
filter	定义简单的过滤器类型。*弃用了* ***filter-spec***。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。
max-backup-index	保存的最大轮转日志数。达到这个数字后，会重复使用最旧的日志。默认值为 **1**。 如果使用 **suffix** 属性，则轮转日志文件的后缀包含在轮转算法中。轮转日志文件时，删除 **名称**开头为 **+后缀** 的最旧文件，其余轮转日志文件的数值后缀递增，新轮转的日志文件为数字后缀 **1**。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。
rotate-on-boot	如果设为 **true**，则在服务器重启时创建新的日志文件。默认值为 **false**。
rotate-size	轮转日志文件前可以达到的最大大小。附加到数字中的单个字符表示大小单位：b 表示 **字节**，**k** 表示 KB，**m** 代表兆字节，**g** 表示千兆字节。例如，**50m** 表示 50 MB。
suffix	此字符串包含在附加到轮转日志的后缀中。**后缀** 的格式是点(**.**)，后跟一个日期字符串，它可以被 **SimpleDateFormat** 类解析。

*表 **A.35.** 定期大小日志处理程序属性*

属性	描述
附加	如果设置为 **true，则** 此处理程序编写的所有消息都会附加到 文件（如果它已存在）。如果设置为 **false**，则每次应用服务器启动时都会创建一个新文件。
autoflush	如果设为 **true**，则日志消息会在收到后立即发送到分配的处理程序。

属性	描述
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
file	代表写入此日志处理程序的输出的文件的对象。它有两个配置属性： **relative-to** 和 **path**。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。
max-backup-index	保存的最大轮转日志数。达到这个数字后，会重复使用最旧的日志。默认值为 **1**。 如果使用 **suffix** 属性，则轮转日志文件的后缀包含在轮转算法中。轮转日志文件时，删除 **名称**开头为 +**后缀** 的最旧文件，其余轮转日志文件的数值后缀递增，新轮转的日志文件为数字后缀 **1**。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。
rotate-on-boot	如果设为 **true**，则在服务器重启时创建新的日志文件。默认值为 **false**。
rotate-size	轮转日志文件前可以达到的最大大小。附加到数字中的单个字符表示大小单位：b 表示 **字节**，**k** 表示 KB，**m** 代表兆字节，**g** 表示千兆字节。例如，**50m** 表示 50 MB。
suffix	此字符串包含在附加到轮转日志的后缀中。**后缀** 的格式是点(**.**)，后跟一个日期字符串，它可以被 **SimpleDateFormat** 类解析。

*表 **A.36. Syslog Handler Attributes***

属性	描述
app-name	在以 RFC5424 格式格式化消息时使用的应用名称。默认情况下，应用程序名称为 **java**。
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
功能	RFC-5424 和 RFC-3164 定义的功能。

属性	描述
hostname	发送邮件的主机的名称。例如，运行应用服务器的主机的名称：
level	日志消息记录的最低日志级别。
port	syslog 服务器正在侦听的端口。
server-address	syslog 服务器的地址。
syslog-format	根据 RFC 规范格式化日志消息。
named-formatter	格式化 syslog 有效负载的消息。使用这个属性，您可以根据需要自定义消息。

*表 **A.37.** 套接字日志处理程序属性*

属性	描述
autoflush	每次写入后是否自动刷新.
block-on-reconnect	如果设为 **true，则** 写入方法在尝试重新连接时将阻止。只有在使用异步处理程序时，建议将其设置为 **true**。
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	此处理程序使用的字符编码
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not **（match("WFLY.*")）**
level	日志消息记录的最低日志级别。
named-formatter	处理程序上要使用的已定义格式器名称。
outbound-socket-binding-ref	对套接字连接的出站套接字绑定的引用。
协议	套接字应通过该协议进行通信的协议。允许的值有 **TCP**、**UDP** 或 **SSL_TCP**。
ssl-context	对定义的 SSL 上下文的引用。仅当 **协议** 设置为 **SSL_TCP** 时，才使用它。

*表 **A.38.** 自定义日志处理程序属性*

属性	描述
class	要使用的日志记录处理程序类。

属性	描述
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
编码	用于输出的字符编码方案。
filter	定义简单的过滤器类型。*弃用了* ***filter-spec***。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
formatter	此日志处理程序使用的日志格式器。
level	日志消息记录的最低日志级别。
module	日志记录处理程序所依赖的模块。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
named-formatter	处理程序上要使用的已定义格式器名称。
属性	用于日志记录处理程序的属性。

*表 **A.39. Async** 日志处理程序属性*

属性	描述
enabled	如果设为 **true**，处理程序将正常启用并正常运行。如果设置为 **false**，则处理日志消息时会忽略处理程序。
filter	定义简单的过滤器类型。*弃用了* ***filter-spec***。
filter-spec	定义过滤器的表达式值。以下表达式定义了一个不匹配模式的过滤器：not（**match("WFLY.*")**）
level	日志消息记录的最低日志级别。
name	日志处理程序的名称。*弃用，因为处理程序的地址包含名称。*
overflow-action	此处理程序在超出其队列长度时如何响应。这可以设置为 **BLOCK** 或 **DISCARD**。**BLOCK** 使日志记录应用等待直到队列中存在可用空间。这与非同步日志处理程序的行为相同。**DISCARD** 允许日志记录应用继续，但日志消息已被删除。

属性	描述
queue-length	此处理程序在等待子处理程序响应期间保存的日志消息的最大数量。
subhandlers	此异步处理程序传递其日志消息的日志处理程序列表。

A.18. 日志格式条件属性

*表 **A.40. Pattern Formatter** 格式字符的格式*

符号	描述
%c	日志事件的类别。
%p	日志条目的级别（INFO、DEBUG 等）。
%P	日志条目的本地化级别。
%d	当前日期/时间（**yyyy-MM-dd HH:mm:sss,SSS** 格式）。
%r	相对时间（日志初始化后的毫秒）。
%z	时区，必须在日期(**%d)**之前指定。例如：**%z{GMT}%d{HH:mm:ss,SSS}**。
%k	日志资源密钥（用于日志消息本地化）。
%m	日志消息（包括异常 trace）。
%s	简单的日志消息（无异常追踪）。
%e	异常堆栈跟踪（无扩展模块信息）。
%E	例外堆栈跟踪（包含扩展模块信息）。
%t	当前线程的名称。
%n	个换行字符。
%C	调用日志方法(slow)的代码类。
%F	调用日志方法(slow)的类的文件名。
%l	调用日志方法(slow)的代码的源位置。
%L	调用日志方法的代码行号(slow)。

符号	描述
%M	调用日志方法(slow)的代码方法。
%x	嵌套诊断上下文.
%X	消息诊断上下文.
%%	字面百分比(%)**字符**(Escaping)。

*表 **A.41. JSON Log Formatter Attributes***

属性	描述
date-format	日期时间格式模式。模式必须是有效的 **java.time.format.DateTimeFormatter.ofPattern（）** 模式。默认模式是 ISO-8601 扩展的偏移日期格式。
exception-output-type	指明日志消息的原因（如果可用）如何添加到 JSON 输出中。允许的值有： • 详细 • **formatted** • **detailed-and-formatted**
key-overrides	允许覆盖 JSON 属性的密钥名称。
meta-data	设置 JSON 格式器使用的元数据。
pretty-print	格式化时是否应使用大量打印.
print-details	是否应打印详细信息.详细信息包括源类名称、源文件名、源方法名称、源代码模块名称、源代码模块版本和源代码行号。  **注意** 打印详细信息可能会代价高昂，因为从调用者检索了这些值。
record-delimiter	用于表示记录末尾的值。如果设置为 null no 分隔符，则会在记录末尾使用。默认值为行源。
zone-id	格式化日期和时间的区域 ID。如果未定义，则使用系统默认值。

*表 **A.42. XML** 日志格式条件属性*

属性	描述
date-format	日期时间格式模式。模式必须是有效的 **java.time.format.DateTimeFormatter.ofPattern（）** 模式。默认模式是 ISO-8601 扩展的偏移日期格式。
exception-output-type	指明日志消息的原因（如果可用）如何添加到 XML 输出中。允许的值有： • 详细 • **formatted** • **detailed-and-formatted**
key-overrides	允许覆盖 XML 属性的密钥名称。
meta-data	设置 XML 格式要使用的元数据。属性添加到每一日志消息中。
namespace-uri	如果 print-namespace 属性为 true，则设置用于每个记录的命名空间 URI。请注意，如果没有定义 namespace-uri，并且有覆盖的密钥而没有写入命名空间，无论 **print-namespace** 属性是否设置为 true。
pretty-print	格式化时是否应使用大量打印.
print-details	是否应打印详细信息.详细信息包括源类名称、源文件名、源方法名称、源代码模块名称、源代码模块版本和源代码行号。 **注意** 打印详细信息可能会代价高昂，因为从调用者检索了这些值。
record-delimiter	用于表示记录末尾的值。如果这是 null，则记录末尾不会使用分隔符。默认值为行源。
zone-id	格式化日期和时间的区域 ID。如果未定义，则使用系统默认值。

A.19. 数据源连接 URL

*表 **A.43. 数据源连接 URL***

数据源	连接 URL
IBM DB2	JDBC:db2://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
MariaDB	JDBC:mariadb://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
MariaDB Galera Cluster	JDBC:mariadb://*SERVER_NAME*:*PORT*,*SERVER_NAME*:*PORT*/*DATABASE_NAME*

数据源	连接 URL
Microsoft SQL Server	jdbc:sqlserver://*SERVER_NAME*:*PORT*;DatabaseName=*DATABASE_NAME*
MySQL	JDBC:mysql://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
Oracle	JDBC:oracle:thin:@*SERVER_NAME*:*PORT*:*ORACLE_SID*
PostgreSQL	JDBC:postgresql://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
Sybase	JDBC:sybase:Tds:*SERVER_NAME*:*PORT*/*DATABASE_NAME*

A.20. DATASOURCE ATTRIBUTES

注意

此表中的属性名称会在管理模型中出现时列出，例如使用管理 ***CLI*** *时。请参阅位于* ***EAP_HOME/docs/schema/wildfly-datasources_5_0.xsd*** *的架构定义文件，以查看* ***XML*** *中出现的元素，因为管理模型可能会有所不同。*

表 ***A.44. Datasource Attributes***

属性	数据源类型	描述
allocation-retry	非 XA、XA	在引发异常之前，应尝试分配连接的次数。默认值为 **0**，因此第一次失败时会引发异常。
allocation-retry-wait-millis	非 XA、XA	在重试分配连接之间等待的时间，以毫秒为单位。默认值为 **0** ms。
allow-multiple-users	非 XA、XA	多个用户是否将通过 **getConnection（用户、密码）方法以及** 内部池类型是否为此行为访问数据源。
authentication-context	非 XA、XA	定义用于区分池中连接的 **javax.security.auth.Subject** 的 Elytron 身份验证上下文。
background-validation	非 XA、XA	是否应该在后台线程验证连接，而不是在使用前验证连接。后台验证通常不与 **validate-on-match** 一起使用，否则将进行冗余检查。通过后台验证，在验证到客户端之间连接可能发生错误，因此应用必须考虑这一可能性。
background-validation-millis	非 XA、XA	后台验证运行的频率（毫秒）。

属性	数据源类型	描述
blocking-timeout-wait-millis	非 XA、XA	在引发异常前等待连接时停止的最长时间，以毫秒为单位。请注意，这只在等待锁定连接时阻止，如果创建新连接的时间过长，永远不会抛出异常。
capacity-decrementer-class	非 XA、XA	定义用于在池中减少连接的策略的类。
capacity-decrementer-properties	非 XA、XA	要注入到类中的属性，该类定义用于减少池中连接的策略。
capacity-incrementer-class	非 XA、XA	定义池中递增连接的策略的类。
capacity-incrementer-properties	非 XA、XA	定义池中递增连接的策略的类中注入的属性。
check-valid-connection-sql	非 XA、XA	用于检查池连接的有效性的 SQL 语句。从池中获取托管连接时，可能会调用此项。
可连接	非 XA、XA	启用使用 CMR，这意味着本地资源可以可靠地参与 XA 事务。
connection-listener-class	非 XA、XA	Specifies class name extending **org.jboss.jca.adapters.jdbc.spi.listener.ConnectionListener**.此类侦听连接激活和传递，以便在连接返回到应用或池之前执行操作。指定的类必须使用两个资源 JAR 将一个模块中的 JDBC 驱动程序与 JDBC 驱动程序捆绑在一起（如将 JDBC 驱动程序安装为核心模块 ）或单独的全局模块中所示，如 Define Global Modules 所示。
connection-listener-property	非 XA、XA	要注入到 **connection-listener-class** 中指定的类的属性。注入的属性符合 JavaBeans 约定。例如，如果您指定了名为 **foo** 的属性，则连接监听器类需要有一个方法 **setFoo**，它接受 **String** 作为参数。
连接属性	Non-XA onlyly	要传递给 **Driver.connect(url, props)** 方法的任意字符串名称/值对连接属性。
connection-url	Non-XA onlyly	JDBC 驱动程序连接 URL。
credential-reference	非 XA、XA	凭据（来自凭据存储），用于对数据源进行身份验证。
datasource-class	Non-XA onlyly	JDBC 数据源类的完全限定名称。
driver-class	Non-XA onlyly	JDBC 驱动程序类的完全限定名称。

属性	数据源类型	描述
driver-name	非 XA、XA	定义数据源应使用的 JDBC 驱动程序。它是与已安装驱动程序名称匹配的符号名称。如果驱动程序部署为 JAR，则名称是部署的名称。
elytron-enabled	非 XA、XA	启用 Elytron 安全性来处理连接身份验证。如果未指定上下文，要使用的 Elytron **身份验证-context** 将为当前上下文。如需更多信息，请参阅**身份验证** 上下文。
enabled	非 XA、XA	是否应启用数据源。
enlistment-trace	非 XA、XA	是否应该记录加入追踪。默认情况下为 **false**。
exception-sorter-class-name	非 XA、XA	**org.jboss.jca.adapters.jdbc.ExceptionSorter** 的实例提供了一种方法来验证异常是否应该广播错误。
exception-sorter-properties	非 XA、XA	异常分类器属性。
flush-strategy	非 XA、XA	指定在出现错误时应如何清空池。有效值为： **FailingConnectionOnly** 仅删除失败的连接。这是默认的设置。 **InvalidIdleConnections** ValidatingManagedConnectionFactory.getInvalidConnections(...) 方法会移除共享相同凭证并返回到无效的连接和空闲连接。 **IdleConnections** 共享同一凭据的失败连接和空闲连接将被删除。 **安全** 共享同一凭据的失败连接和空闲连接将被删除。在返回到池时，共享相同凭据的活动连接将被销毁。 **EntirePool** 共享相同凭据的连接以及空闲和活动连接失败。不建议在生产环境中使用这个设置。 **AllInvalidIdleConnections** ValidatingManagedConnectionFactory.getInvalidConnections(...) 方法返回为无效的连接和空闲连接失败。 **AllIdleConnections** 失败的连接和所有空闲连接被删除。 **AllGracefully** 失败的连接和所有空闲连接被删除。返回到池后，活动连接将被销毁。 **AllConnections** 连接失败，以及所有空闲和活跃的连接被删除。不建议在生产环境中使用这个设置。

属性	数据源类型	描述
idle-timeout-minutes	非 XA、XA	连接在关闭前可能会闲置，以分钟为单位。如果没有指定，则默认为 **30** 分钟。实际的最长时间还取决于 IdleRemover 扫描时间，这是任何池的最小空闲时间分钟值的一半。
initial-pool-size	非 XA、XA	池应保留的初始连接数。
interleaving	XA Only	是否为 XA 连接启用交集.
jndi-name	非 XA、XA	数据源的唯一 JNDI 名称。
jta	Non-XA onlyly	启用 JTA 集成.
max-pool-size	非 XA、XA	池可以容纳的最大连接数。
mcp	非 XA、XA	**ManagedConnectionPool** 实施.例如：**org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool**
min-pool-size	非 XA、XA	池可以保存的最少连接数。
new-connection-sql	非 XA、XA	每当连接添加到连接池时要执行的 SQL 语句。
no-recovery	XA Only	是否应该将连接池从恢复中排除。
no-tx-separate-pool	XA Only	是否为每个上下文创建单独的子池。些 Oracle 数据源可能要求这样做，它们可能不允许 JTA 事务内部和外部使用 XA 连接。使用此选项将导致池总大小为 **max-pool-size** 的两倍，因为将创建两个实际池。
pad-xid	XA Only	是否 pad the Xid.
password	非 XA、XA	创建新连接时要使用的密码。
pool-fair	非 XA、XA	定义池是否应该公平。此设置是 **Semaphore** 类的一部分，用于在 Jakarta Connectors 中管理连接池，这在某些用例中提供了性能优势，在某些情况下不需要租用连接的顺序。
pool-prefill	非 XA、XA	是否应预先填充池。
pool-use-strict-min	非 XA、XA	是否应该严格考虑 **min-pool-size**。
prepared-statements-cache-size	非 XA、XA	在 Least Recently Used(LRU)缓存中每个连接准备的语句数量。

属性	数据源类型	描述
query-timeout	非 XA、XA	查询的超时，以秒为单位。默认值为没有超时。
reauth-plugin-class-name	非 XA、XA	重新身份验证插件实施的完全限定类名称，用于重新验证物理连接。
reauth-plugin-properties	非 XA、XA	重新身份验证插件的属性：
recovery-authentication-context	XA Only	定义用于区分池中连接的 **javax.security.auth.Subject** 的 Elytron 身份验证上下文。
recovery-credential-reference	XA Only	凭据（来自凭据存储），用于对数据源进行身份验证。
recovery-elytron-enabled	XA Only	启用 Elytron 安全性，以处理用于恢复的连接的身份验证。如果未指定 **身份验证-上下文，** 则使用的 Elytron **authentication-context** 将是当前的上下文。如需更多信息，**请参阅身份验证** 上下文。
restore-password	XA Only	用于连接资源以进行恢复的密码。
recovery-plugin-class-name	XA Only	恢复插件实施的完全限定类名称。
recovery-plugin-properties	XA Only	恢复插件的属性。
recovery-security-domain	XA Only	用于连接资源以进行恢复的安全域。
restore-username	XA Only	用于连接资源以进行恢复的用户名。
same-rm-override	XA Only	**javax.transaction.xa.XAResource.isSameRM(XAResource)** 类返回 **true** 或 **false**。
security-domain	非 XA、XA	处理身份验证的 JAAS security-manager 的名称。此名称与 JAAS 登录配置中的 application-policy/name 属性相关联。
set-tx-query-timeout	非 XA、XA	是否根据剩余时间设置查询超时，直到事务超时为止。如果不存在事务，则将使用任何配置的查询超时。
共享准备状态	非 XA、XA	当为应用提供的打包程序受应用代码封闭时，无论是 JBoss EAP 是否应该缓存，而不是关闭或终止的底层物理语句。默认值为 **false**。

属性	数据源类型	描述
spy	非 XA、XA	在 JDBC 层上启用间隔功能。这会记录到数据源的所有 JDBC 流量。注意日志记录类别 **jboss.jdbc.spy** 还必须设置为 **logging** 子系统中的日志级别 **DEBUG**。
stale-connection-checker-class-name	非 XA、XA	**org.jboss.jca.adapters.jdbc.StaleConnectionChecker** 的实例，提供 **isStaleConnection(SQLException)** 方法。如果此方法返回 **true**，则异常被嵌套在 **org.jboss.jca.adapters.jdbc.StaleConnectionException** 中。
stale-connection-checker-properties	非 XA、XA	过时的连接检查器属性。
statistics-enabled	非 XA、XA	是否启用运行时统计数据。默认值为 **false**。
track-statements	非 XA、XA	连接返回到池时，是否检查未克隆的语句，并且语句返回到准备的语句缓存。如果为 false，则语句不会被跟踪。有效值： • **True：** 跟踪语句和结果集，并在未关闭时发出警告。 • **false** ：不跟踪语句或结果集。 • **nowarn** ：语句会被跟踪，但没有发出警告（*默认*）。
tracking	非 XA、XA	是否跟踪连接处理跨越事务界限.
transaction-isolation	非 XA、XA	**java.sql.Connection** 事务隔离级别。有效值： • **TRANSACTION_READ_UNCOMMITTED** • **TRANSACTION_READ_COMMITTED** • **TRANSACTION_REPEATABLE_READ** • **TRANSACTION_SERIALIZABLE** • **TRANSACTION_NONE**
url-delimiter	非 XA、XA	用于高可用性(HA)数据源的 connection-url 中的 URL 分隔符。
url-property	XA Only	the **xa-datasource-property 值中的 URL 属性的属性**。

属性	数据源类型	描述
url-selector-strategy-class-name	非 XA、XA	实现 **org.jboss.jca.adapters.jdbc.URLSelectorStrategy** 的类。
use-ccm	非 XA、XA	启用缓存的连接管理器。
use-fast-fail	非 XA、XA	如果为 true，在连接无效时第一次尝试时会失败。如果为 false，请继续尝试直到池耗尽。
use-java-context	非 XA、XA	是否将数据源绑定到全局 JNDI 中.
use-try-lock	非 XA、XA	内部锁定的超时值。这尝试在超时前获得配置的秒数锁定，而不是立即失败（如果锁定不可用）。使用 **tryLock（）** 而不是 **lock（）。**
user-name	非 XA、XA	创建新连接时要使用的用户名。
valid-connection-checker-class-name	非 XA、XA	**org.jboss.jca.adaptors.jdbc.ValidConnectionChecker** 实施，它提供了一个 **SQLException.isValidConnection(Connection e)** 方法来验证连接。异常表示连接已被销毁。这会覆盖 **check-valid-connection-sql** 属性（如果存在）。
valid-connection-checker-properties	非 XA、XA	有效的连接检查器属性。
validate-on-match	非 XA、XA	连接工厂尝试匹配受管连接时是否执行连接验证。当客户端在使用前必须验证连接时，应使用此方法。validate-on-match 通常不与 **后台验证** 一起使用，或者将进行冗余检查。
wrap-xa-resource	XA Only	是否将 XAResource 包装到 **org.jboss.tm.XAResourceWrapper** 实例中。
xa-datasource-class	XA Only	**javax.sql.XADataSource** 实施类的完全限定名称。
xa-datasource-properties	XA Only	XA 数据源属性的字符串名称/值对。
xa-resource-timeout	XA Only	如果非零，这个值将传递给 **XAResource.setTransactionTimeout** 方法。

*表 **A.45. JDBC Driver Attributes***

属性	数据源类型	描述
datasource-class-info	非 XA、XA	**jdbc-driver** 的 **datasource-class** 和 **xa-datasource-class** 的可用属性。**datasource-class and xa-datasource-class** 属性定义实施 **javax.sql.DataSource 或 javax.sql .XADataSource** 类的完全限定类名称。定义的类可以具有各种属性的设置器。**datasource-class-info** 属性列出了可为类设置的这些属性。

A.21. DATASOURCE STATISTICS

*表 **A.46. Core Pool Statistics***

名称	描述
ActiveCount	活跃连接的数量。每个连接都可供应用使用，或者在池中可用。
AvailableCount	池中可用连接的数量。
AverageBlockingTime	阻止在池中获取专用锁定的平均时间。这个值以毫秒为单位。
AverageCreationTime	创建连接所需的平均时间。这个值以毫秒为单位。
AverageGetTime	获取连接所需的平均时间。这个值以毫秒为单位。
AveragePoolTime	池中连接的平均时间。这个值以毫秒为单位。
AverageUsageTime	使用连接的平均时间。这个值以毫秒为单位。
BlockingFailureCount	试图获得连接的失败数量。
CreatedCount	创建的连接数。
DestroyedCount	销毁的连接数量。
IdleCount	当前空闲的连接数。
InUseCount	当前使用的连接数。
MaxCreationTime	创建连接所花费的最长时间。这个值以毫秒为单位。
MaxGetTime	获取连接的最长时间。这个值以毫秒为单位。
MaxPoolTime	池中连接的最长时间。这个值以毫秒为单位。

名称	描述
MaxUsageTime	使用连接的最长时间。这个值以毫秒为单位。
MaxUsedCount	使用的最大连接数。
MaxWaitCount	同时等待连接的最大请求数。
MaxWaitTime	等待池中专用锁定的最长时间。这个值以毫秒为单位。
timedOut	连接超时的数量。
TotalBlockingTime	在池中等待专用锁定的总时间。这个值以毫秒为单位。
TotalCreationTime	创建连接所花费的总时间。这个值以毫秒为单位。
TotalGetTime	获取连接所花费的总时间。这个值以毫秒为单位。
TotalPoolTime	池中连接花费的总时间。这个值以毫秒为单位。
TotalUsageTime	使用连接所用的总时间。这个值以毫秒为单位。
WaitCount	必须等待的请求数以获取连接。
XACommitAverageTime	XAResource 提交调用的平均时间。这个值以毫秒为单位。
XACommitCount	XAResource 提交调用的数量。
XACommitMaxTime	XAResource 提交调用的最长时间。这个值以毫秒为单位。
XACommitTotalTime	所有 XAResource 提交调用的总时间。这个值以毫秒为单位。
XAEndAverageTime	XAResource 结束调用的平均时间。这个值以毫秒为单位。
XAEndCount	XAResource 端点调用的数量。
XAEndMaxTime	XAResource 结束调用的最长时间。这个值以毫秒为单位。
XAEndTotalTime	所有 XAResource 结束调用的总时间。这个值以毫秒为单位。
XAForgetAverageTime	XAResource 的平均时间忘记调用。这个值以毫秒为单位。
XAForgetCount	XAResource 忘记调用的数量。
XAForgetMaxTime	XAResource 忘记调用的最长时间。这个值以毫秒为单位。
XAForgetTotalTime	所有 XAResource 都忘记调用的总时间。这个值以毫秒为单位。

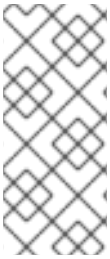
名称	描述
XAPrepareAverageTime	XAResource 准备调用的平均时间。这个值以毫秒为单位。
XAPrepareCount	XAResource 准备调用的数量。
XAPrepareMaxTime	XAResource 准备调用的最长时间。这个值以毫秒为单位。
XAPrepareTotalTime	所有 XAResource 准备调用的总时间。这个值以毫秒为单位。
XARecoverAverageTime	XAResource 恢复调用的平均时间。这个值以毫秒为单位。
XARecoverCount	XAResource 恢复调用的数量。
XARecoverMaxTime	XAResource 恢复调用的最长时间。这个值以毫秒为单位。
XARecoverTotalTime	所有 XAResource 恢复调用的总时间。这个值以毫秒为单位。
XARollbackAverageTime	XAResource 回滚调用的平均时间。这个值以毫秒为单位。
XARollbackCount	XAResource 回滚调用数量。
XARollbackMaxTime	XAResource 回滚调用的最长时间。这个值以毫秒为单位。
XARollbackTotalTime	所有 XAResource 回滚调用的总时间。这个值以毫秒为单位。
XAStartAverageTime	XAResource 开始调用的平均时间。这个值以毫秒为单位。
XAStartCount	XAResource start 调用的数量。
XAStartMaxTime	XAResource 开始调用的最长时间。这个值以毫秒为单位。
XAStartTotalTime	所有 XAResource 启动调用的总时间。这个值以毫秒为单位。

*表 **A.47. JDBC Statistics***

名称	描述
PreparedStatementCacheAccessCount	访问声明缓存的次数。
PreparedStatementCacheAddCount	添加至声明缓存的声明数。
PreparedStatementCacheCurrentSize	当前缓存在声明缓存中的已就绪和可调用语句的数量。
PreparedStatementCacheDeleteCount	从缓存中丢弃的声明数。

名称	描述
PreparedStatementCacheHitCount	使用缓存中的语句的次数。
PreparedStatementCacheMissCount	声明请求无法通过缓存中的语句满足的次数。

A.22. AGROAL DATASOURCE ATTRIBUTES



注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-agroal_1_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.48. Agroal Datasource Attributes***

属性	描述
可连接	是否在此数据源上启用 CMR（可分配资源）功能。这只适用于非 XA 数据源。
jndi-name	指定数据源的 JNDI 名称。
jta	是否启用 JTA 集成.这只适用于非 XA 数据源。
statistics-enabled	是否为此数据源启用统计信息。默认值为 **false**。

*表 **A.49. Agroal Datasource Connection Factory Attributes***

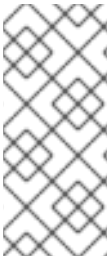
属性	描述
authentication-context	在 **elytron** 子系统中引用身份验证上下文.
连接属性	创建连接时要传递给 JDBC 驱动程序的属性。
credential-reference	用于进行身份验证的凭据（来自凭据存储）。
驱动程序	对 JDBC 驱动程序的唯一参考。
new-connection-sql	创建后要在连接上执行的 SQL 语句。
password	用于数据库基本身份验证的密码。
transaction-isolation	将 **java.sql.Connection** 事务隔离级别设置为使用。

属性	描述
url	JDBC 驱动程序连接 URL。
username	用于数据库进行基本身份验证的用户名。

*表 **A.50. Agroal Datasource Connection Pool Attributes***

属性	描述
background-validation	后台验证运行之间的时间（毫秒）。
blocking-timeout	在引发异常前等待连接时停止的最长时间，以毫秒为单位。
idle-removal	连接在删除前必须处于空闲状态的时间（以分钟为单位）。
Initial-size	池应保留的初始连接数。
leak-detection	以毫秒为单位，必须保存连接的时间，才能发出泄漏警告。
max-size	池中连接的最大数量。
Min-size	池应保留的最小连接数。

A.23. 事务管理器配置选项



注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-txn_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.51.** 事务子系统属性*

属性	描述
default-timeout	默认事务超时。默认值为 **300** 秒。您可以逐个事务以编程方式覆盖它。
enable-statistics	*弃用，而是启用统计信息。*
enable-tsm-status	是否启用用于进程外恢复的事务状态管理器(TSM)服务。不支持这个选项，因为不支持运行进程外恢复管理器从不同进程（而不是内存中）联系 **ActionStatusService**。

属性	描述
hornetq-store-enable-async-io	*弃用了* ***journal-store-enable-async-io***。
jdbc-action-store-drop-table	JDBC 操作存储是否应丢弃表.默认值为 **false**。
jdbc-action-store-table-prefix	用于在配置的 JDBC 操作存储中写入事务日志的表的可选前缀。
jdbc-communication-store-drop-table	JDBC 通信存储是否应丢弃表。默认值为 **false**。
jdbc-communication-store-table-prefix	用于在配置的 JDBC 通信存储中写入事务日志的表的可选前缀。
jdbc-state-store-drop-table	JDBC 状态存储是否应丢弃表。默认值为 **false**。
jdbc-state-store-table-prefix	用于在配置的 JDBC 状态存储中写入事务日志的表可选前缀。
jdbc-store-datasource	使用的非 XA 数据源的 JNDI 名称.数据源应当在 **datasources** 子系统中定义。
journal-store-enable-async-io	是否应该为日志存储启用 **AsyncIO**。默认值为 **false**。服务器应重新启动，以使此设置生效。
jts	是否使用 Java 事务服务(JTS)事务.默认值为 **false**，它只使用 JTA 事务。
maximum-timeout	如果事务的事务超时设为 **0（代表** 无限超时），事务管理器将改为使用此属性设置的值。默认值为 **31536000** 秒（365 天）。
node-identifier	事务管理器的节点标识符。如果未设置此选项，您将在服务器启动时看到警告。在以下情况下需要这个选项： • 用于 JTS 通信 • 当两个事务管理器访问共享资源管理器时 • 当两个事务管理器访问共享对象存储时 node-identifier 必须为每个事务管理器唯一，因为它需要在恢复期间强制实施数据完整性。node-identifier 还必须对 JTA 唯一，因为多个节点可能会与同一资源管理器交互或共享事务对象存储。
object-store-path	事务管理器对象存储存储数据的相对或绝对文件系统路径。默认情况下，相对于 **object-store-relative-to** 参数的值。如果 **object-store-relative-to** 设置为空字符串，则此值被视为绝对路径。

属性	描述
object-store-relative-to	引用域模型中的全局路径配置。默认值为 JBoss EAP 的数据目录，即属性 **jboss.server.data.dir** 的值，受管域的默认值为 ***EAP_HOME*/domain/data/**，或者单机服务器实例的 ***EAP_HOME*/standalone/data/**。对象存储 **object-store-path** 事务管理器属性的值相对于此路径。将此属性设置为空字符串，使 **object-store-path** 被视为绝对路径。
process-id-socket-binding	如果事务管理器应使用基于套接字的进程 ID，则要使用的套接字绑定配置名称。如果 **process-id-uuid** 为 **true**，则将 **未** 定义；否则，必须设置.
process-id-socket-max-ports	事务管理器为每个事务日志创建一个唯一标识符。为生成唯一标识符提供了两种不同的机制：一种基于套接字的机制，以及基于进程的进程标识符的机制。 对于基于套接字的标识符，将打开套接字，其端口号用于标识符。如果端口已在使用，则下一个端口将被探测到找到空闲端口为止。process-id-socket-max-ports 代表事务管理器在失败前将尝试的最大套接字数量。默认值为 **10**。
process-id-uuid	设置为 **true**，以使用进程标识符为每个事务创建唯一标识符。否则，将使用基于套接字的机制。默认值为 **true**。如需更多信息，请参阅 **process-id-socket-max-ports**。要启用 **process-id-socket-binding**，请将 **process-id-uuid** 设置为 **false**。
restore-listener	事务恢复进程是否应该侦听网络套接字。默认值为 **false**。
socket-binding	指定当 restore **-listener 设为 true 时，事务周期恢复** 监听器使用的套接字绑定的名称。
statistics-enabled	是否应启用统计数据。默认值为 **false**。
status-socket-binding	指定用于事务状态管理器的套接字绑定。不支持这个选项。
use-hornetq-store	*弃用了 **use-journal-store**。*
use-jdbc-store	使用 JDBC 存储编写事务日志。设置为 **true**，设为 enable，**设为 false**，以使用默认的日志存储类型。
use-journal-store	将 Apache ActiveMQ Artemis 日志存储机制用于事务日志，而不是基于文件的存储。默认情况下禁用此设置，但可以提高 I/O 性能。对于独立事务管理器的 JTS 事务，我们不建议这样做。更改此选项时，必须使用 **shutdown** 命令重新启动服务器，才能使更改生效。

*表 **A.52.** 日志存储属性*

属性	描述
expose-all-logs	是否公开所有日志.默认值为 **false**，这表示仅公开一个事务日志子集。
type	指定日志记录存储的实施类型。默认值为 **default**。

*表 **A.53.** 提交可标记资源属性*

属性	描述
batch-size	此 CMR 资源的批处理大小。默认值为 **100**。
立即清理	是否为此 CMR 资源执行立即清理。默认值为 **true**。
jndi-name	此 CMR 资源的 JNDI 名称。
name	用于存储 XID 的表名称。默认值为 **xids**。

A.24. IIOP 子系统属性

注意

*此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-iiop-openjdk_3_0.xsd** 的架构定义文件，以查看它们出现在 **XML** 中的元素，因为管理模型可能会有所不同。*

*表 **A.54. IIOP** 子系统属性*

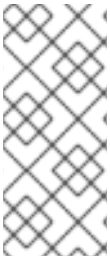
属性	描述
add-component-via-interceptor	指明 SSL 组件是否应由 IOR 拦截器添加。*弃用.*
auth-method	验证方法。有效值为 **none** 和 **username_password**。
authentication-context	安全初始化器设置为 **elytron** 时使用的身份验证上下文名称。
caller-propagation	指明是否应在 SAS 上下文中传播调用者身份。有效值为 **none** 且 **支持**。
client-quires	表示客户端 SSL 必需参数的值。有效值有 **None**、**ServerAuth**、**A 客户端Auth** 和 **MutualAuth**。*弃用：改为使用 **client-requires-ssl**。*
client-requires-ssl	指明来自服务器的 IIOP 连接是否需要 SSL。

属性	描述
client-ssl-context	用于创建客户端 SSL 套接字的 SSL 上下文名称。
客户端支持	表示客户端 SSL 支持的参数的值。有效值有 **None**、**ServerAuth**、**A 客户端Auth** 和 **MutualAuth**。*弃用：改为使用* ***client-requires-ssl***。
confidentiality	指明传输是否需要保密保护。有效值为 **none**、**受支持** 且 **必需** 值。*弃用：改为使用* ***server-requires-ssl***。
detect-misordering	指明传输是否需要进行错误排序检测。有效值为 **none**、**受支持** 且 ***必需*** 值。*弃用：改为使用* ***server-requires-ssl***。
detect-replay	指明传输是否需要重播检测。有效值为 **none**、**受支持** 且 **必需** 值。*弃用：改为使用* ***server-requires-ssl***。
export-corbaloc	指明根上下文是否应当导出为 **corbaloc::address:port/NameService**。
giop-version	要使用的 GIOP 版本。
high-water-mark	TCP 连接缓存参数.每次连接数超过这个值时，ORB 尝试重新声明连接。回收的连接数通过 **number-to-reclaim** 属性指定。如果没有设置此属性，则使用 OpenJDK ORB 默认值。
完整性	指明传输是否需要完整性保护。有效值为 **none**、**受支持** 且 **必需** 值。*弃用：改为使用* ***server-requires-ssl***。
number-to-reclaim	TCP 连接缓存参数.每次连接数超过 **high-water-mark** 属性时，ORB 尝试回收连接。回收的连接数量由此属性指定。如果没有设置，则使用 OpenJDK ORB 默认值。
persistent-server-id	服务器的持久 ID。持久对象引用在服务器的许多激活之间有效，它们使用此属性进行识别。因此，同一服务器的许多激活应将此属性设置为相同的值，在同一主机上运行的不同服务器实例应具有不同的服务器 ID。
属性	通用键/值属性列表。
realm	身份验证服务域名称。
必需	指明是否需要身份验证。
root-context	命名服务根上下文.
security	指明是否要安装安全拦截器。有效值为 **client**、**identity**、**elytron** 和 **none**。

属性	描述
security-domain	保存将用于建立 SSL 连接的密钥存储和信任存储的安全域的名称。
server-requires	表示服务器 SSL 必需参数的值。有效值有 **None**、**ServerAuth**、**A 客户端Auth** 和 **MutualAuth**。*弃用：改为使用* ***server-requires-ssl***。
server-requires-ssl	指明到服务器的 IIOP 连接是否需要 SSL。
server-ssl-context	用于创建服务器端 SSL 套接字的 SSL 上下文名称。
服务器支持	表示服务器 SSL 支持参数的值。有效值有 **None**、**ServerAuth**、**A 客户端Auth** 和 **MutualAuth**。*弃用：改为使用* ***server-requires-ssl***。
socket-binding	指定 ORB 端口的套接字绑定配置的名称。
ssl-socket-binding	指定 ORB SSL 端口的套接字绑定配置的名称。
support-ssl	指明是否支持 SSL。
事务	指明是否要安装交易拦截器。有效值为 **full**、**spec** 和 **none**。值 **full** 可启用 JTS，而 **spec** 值则启用拒绝传入事务上下文的非 JTS spec 兼容模式。
trust-in-client	指明是否需要在客户端中建立信任。有效值为 **none**、**受支持** 且 **必需** 值。*弃用：改为使用* ***server-requires-ssl***。
trust-in-target	指明传输是否需要建立目标信任。有效值为 **none** 且 **支持**。*弃用：改为使用* ***server-requires-ssl***。

A.25. 资源适配器属性

下表描述了资源适配器属性。



注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-resource-adapters_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.55.** 主要属性*

属性	描述
Archive	资源适配器存档。
beanvalidationgroups	应使用的 bean 验证组。
bootstrap-context	应使用的 bootstrap 上下文的唯一名称。
config-properties	自定义定义的配置属性.
module	从中加载资源适配器的模块。
statistics-enabled	是否启用运行时统计数据.
transaction-support	资源适配器的事务支持级别。有效值为 **NoTransaction**、**LocalTransaction** 或 **XATransaction**。
wm-elytron-security-domain	定义应使用的 Elytron 安全域的名称。
wm-security	为此资源适配器开启/关闭 **wm.security**。如果为 false，则所有 **wm-security-*** 参数都会被忽略，即使默认值也是如此。
wm-security-default-groups	应添加到所用 **Subject** 实例的默认组列表。
wm-security-default-principal	应添加到所用 **Subject** 实例的默认主体名称。
wm-security-domain	应使用的安全域的名称。
wm-security-mapping-groups	组映射列表.
wm-security-mapping-required	定义是否需要安全凭据的映射。
wm-security-mapping-users	用户映射列表.

注意

*如果您的资源适配器以及 **elytron -enabled** 设置为 **true** 的工作管理器使用 **bootstrap-context**，则必须将 **wm-elytron-security-domain** 属性而不是 **wm-security-domain** 属性用于安全域规格。*

*表 **A.56. admin-objects** 属性*

属性	描述
class-name	管理对象的完全限定类名称。

属性	描述
enabled	指定是否应启用管理对象。
jndi-name	管理对象的 JNDI 名称。
use-java-context	将此设置为 false 会将对象绑定到全局 JNDI。

*表 **A.57.** 连接定义属性*

属性	描述
allocation-retry	表示在引发异常之前，应尝试分配连接的次数。
allocation-retry-wait-millis	在重试分配连接之间等待的时间，以毫秒为单位。
authentication-context	定义用于区分池中连接的 **javax.security.auth.Subject** 的 Elytron 身份验证上下文。
authentication-context-and-application	表示提供的应用程序参数（如 from **getConnection（user、pw）或 Subject ）** 都用于区分池中的连接。在使用配置的验证上下文时，Elytron 在 **验证后会提供这些参数**。
background-validation	指定应在后台线程验证连接，而不是在使用前验证连接。更改此值需要重新启动服务器。
background-validation-millis	后台验证将运行的时间，以毫秒为单位。更改此值需要重新启动服务器。
blocking-timeout-wait-millis	在引发异常前等待连接时停止的最长时间，以毫秒为单位。请注意，这只在等待锁定连接时阻止，如果创建新连接的时间过长，永远不会抛出异常。
capacity-decrementer-class	定义用于在池中减少连接的策略的类。
capacity-decrementer-properties	属性注入类，该类定义池中减少连接的策略。
capacity-incrementer-class	定义池中递增连接的策略的类。
capacity-incrementer-properties	将属性注入类，以定义池中增加连接的策略。
class-name	受管连接工厂或 admin 对象的完全限定类名称。
可连接	启用使用 CMR。此功能意味着本地资源可以可靠地参与 XA 事务。
elytron-enabled	启用 Elytron 安全性来处理连接身份验证。如果未指定上下文，要使用的 Elytron **身份验证-context** 将是当前的上下文。如需更多信息，**请参阅身份验证** 上下文。

属性	描述
enabled	指定是否应该启用资源适配器。
加入	指定资源适配器是否应该使用 lazy enlist。
enlistment-trace	指定 JBoss EAP/IronJacamar 是否应该记录条目。默认情况下为 **false**。
flush-strategy	指定在出现错误时应如何清空池。有效值为： **FailingConnectionOnly** 仅删除失败的连接。这是默认的设置。 **InvalidIdleConnections** ValidatingManagedConnectionFactory.getInvalidConnections(...) 方法会移除共享相同凭证并返回到无效的连接和空闲连接。 **IdleConnections** 共享同一凭据的失败连接和空闲连接将被删除。 **安全** 共享同一凭据的失败连接和空闲连接将被删除。在返回到池时，共享相同凭据的活动连接将被销毁。 **EntirePool** 共享相同凭据的连接以及空闲和活动连接失败。不建议在生产环境中使用这个设置。 **AllInvalidIdleConnections** ValidatingManagedConnectionFactory.getInvalidConnections(...) 方法返回为无效的连接和空闲连接失败。 **AllIdleConnections** 失败的连接和所有空闲连接被删除。 **AllGracefully** 失败的连接和所有空闲连接被删除。返回到池后，活动连接将被销毁。 **AllConnections** 连接失败，以及所有空闲和活跃的连接被删除。不建议在生产环境中使用这个设置。
idle-timeout-minutes	连接在关闭前可能会闲置，以分钟为单位。实际的最长时间还取决于 **IdleRemover** 扫描时间，这是任何池的最小 **空闲时间分钟值** 的一半。更改此值需要重新启动服务器。
initial-pool-size	池应保留的初始连接数。
interleaving	指定是否为 XA 连接启用交集。
jndi-name	连接工厂的 JNDI 名称。
max-pool-size	池的最大连接数。不会在每个子池中创建更多连接。

属性	描述
mcp	**ManagedConnectionPool** 实施.例如：**org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool**。
min-pool-size	池的最小连接数。
no-recovery	指定是否应从恢复中排除连接池。
no-tx-separate-pool	Oracle 不像 JTA 事务内部和外部使用 XA 连接。要解决这个问题，您可以为不同的上下文创建单独的子池。
pad-xid	指定是否应该添加 Xid。
pool-fair	指定池使用应公平。
pool-prefill	指定是否应预先填充池。更改此值需要重新启动服务器。
pool-use-strict-min	指定 **min-pool-size** 是否被视为严格。
recovery-authentication-context	用于恢复的 Elytron 身份验证上下文。如果没有指定 **身份验证上下文**，则将使用当前上下文。
recovery-credential-reference	凭据（来自凭据存储），用于在恢复连接时进行身份验证。
recovery-elytron-enabled	表示将使用 Elytron 身份验证上下文进行恢复。默认值为 **false**。
restore-password	用于恢复的密码。
recovery-plugin-class-name	恢复插件实施的完全限定类名称。
recovery-plugin-properties	恢复插件的属性。
recovery-security-domain	用于恢复的安全域。
restore-username	用于恢复的用户名。
same-rm-override	无条件设置 **javax.transaction.xa.XAResource.isSameRM(XAResource)** 返回 true 或 false。
security-application	表示应用提供的参数（如 from **getConnection（用户、pw）** 用于区分池中的连接。
security-domain	定义用于区分池中连接的 **javax.security.auth.Subject** 的安全域。

属性	描述
security-domain-and-application	表示提供的应用提供的参数（如 from **getConnection（user、pw）或 Subject ）** 都用于区分池中的连接。
sharable	启用可隔离连接，如果受支持，则允许启用 lazy 关联。
tracking	指定 IronJacamar 是否应该跟踪连接跨事务边界处理。
use-ccm	启用使用缓存的连接管理器。
use-fast-fail	当设置为 **true** 时，第一次尝试无效的连接分配会失败。当设置为 **false** 时，请继续尝试直到所有潜在连接耗尽为止。
use-java-context	将此设置为 **false** 会将对象绑定到全局 JNDI。
validate-on-match	指定当连接工厂尝试匹配受管连接时，是否应进行连接验证。这通常专用于使用后台验证。
wrap-xa-resource	指定 **XAResource** 实例是否应嵌套在 **org.jboss.tm.XAResourceWrapper** 实例中。
xa-resource-timeout	该值以秒为单位传递给 **XAResource.setTransactionTimeout(**)。默认为 **0**。

A.26. 资源适配器统计信息

*表 **A.58.** 资源适配器统计信息*

名称	描述
ActiveCount	活跃连接的数量。每个连接可以由应用使用，或者在池中可用
AvailableCount	池中可用连接的数量。
AverageBlockingTime	阻止在池中获取专用锁定的平均时间。该值以毫秒为单位。
AverageCreationTime	创建连接所需的平均时间。该值以毫秒为单位。
CreatedCount	创建的连接数。
DestroyedCount	销毁的连接数量。
InUseCount	当前使用的连接数。
MaxCreationTime	创建连接所花费的最长时间。该值以毫秒为单位。

名称	描述
MaxUsedCount	使用的最大连接数。
MaxWaitCount	同时等待连接的最大请求数。
MaxWaitTime	等待池中专用锁定的最长时间。
timedOut	连接超时的数量。
TotalBlockingTime	在池中等待专用锁定的总时间。该值以毫秒为单位。
TotalCreationTime	创建连接所花费的总时间。该值以毫秒为单位。
WaitCount	必须等待连接的请求数。

A.27. UNDERTOW 子系统属性

*下表中列出了 **undertow** 子系统的各种元素的属性。*

注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-undertow_4_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

- *主要属性*
- *应用程序安全域属性*
- *缓冲区缓存属性*
- *字节缓冲区池属性*
- ***servlet** 容器属性*

- *过滤属性*
- *处理程序属性*
- *服务器属性*

表 A.59. 主要 undertow 属性

属性	默认	描述
default-security-domain	其他	Web 部署使用的默认安全域。
default-server	default-server	用于部署的默认服务器。
default-servlet-container	default	用于部署的默认 servlet 容器。
default-virtual-host	default-host	用于部署的默认虚拟主机。
instance-id	${jboss.node.name}	集群实例 ID。
statistics-enabled	false	是否启用统计数据.

应用程序安全域属性

应用程序安全域属性具有以下结构：

- ***application-security-domain***
  - *设置*
    - ***single-sign-on***

application-security-domain Attributes

表 A.60. application-security-domain Attributes

属性	默认	描述
enable-jacc	false	通过 JACC 启用授权。

属性	默认	描述
enable-jaspi	true	为关联的部署启用 JASPI 身份验证。
http-authentication-factory		引用映射的安全域的部署要使用的 HTTP 身份验证工厂。
integrated-jaspi	true	是否应使用集成的 JASPI。当在 JASPI 身份验证期间设置为 **true** 时，身份从部署引用的 **SecurityDomain** 加载。如果设置为 **false**，则会创建临时身份。
override-deployment-config	false	部署中的身份验证配置是否应该被工厂覆盖。
引用部署		当前引用此映射的部署。
security-domain		部署要使用的 **SecurityDomain**。

单点登录属性

***表 A.61.** 单点登录属性*

属性	默认	描述
client-ssl-context		引用用于保护 back-channel 注销连接的 SSL 上下文。
cookie-name	JSESSIONIDSSO	Cookie 的名称。
credential-reference		凭据引用用于解密私钥条目。
domain		将使用的 Cookie 域。
http-only	false	设置 Cookie httpOnly 属性.
key-alias		用于签名并验证 back-channel 注销连接的私钥条目别名。
key-store		对包含私钥条目的密钥存储的引用.
路径	/	Cookie 路径.
安全	false	设置 Cookie 安全属性.

缓冲区缓存属性

***表 A.62. buffer-cache** 属性*

属性	默认	描述
buffer-size	1024	缓冲区的大小。更小的缓冲区允许更有效地使用空间。
buffers-per-region	1024	每个区域的缓冲区数量。
Max-regions	10	区域的最大数量.这控制可用于缓存的最大内存量。

字节缓冲区池属性

表 A.63. byte-buffer-pool Attributes

属性	默认	描述
buffer-size		每个缓冲区片段的大小（以字节为单位）。如果没有指定，则根据系统的可用 RAM 设置大小： ● RAM 小于 64 MB 的 512 字节 ● 1024 bytes (1 KB) for 64 MB - 128 MB RAM ● 16384 字节(16 KB)，用于 128 MB RAM 有关此属性的性能调优建议，请参阅 JBoss EAP *性能调优指南中的配置* *https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/performance_tuning_guide/#io_buffer_pools 缓冲池*。
direct		表示此缓冲区是否为直接还是堆池的布尔值。如果没有指定，则该值会根据系统的可用 RAM 设置： ● 如果可用 RAM 是 < 64MB，则值设为 **false** ● 如果可用 RAM 是 >= 64MB，则值被设置为 **true** 请注意，直接池也有对应的堆池。
leak-detection-percent	0	应使用泄漏检测器分配的缓冲区百分比。
max-pool-size		要在池中保留的最大缓冲区数量。缓冲区仍会被分配超过这个限制，但如果池已满，则不会保留。
thread-local-cache-size	12	每个线程缓存的大小。这是最大大小，缓存将使用智能大小调整来仅在线程实际分配缓冲区时保留线程上的缓冲。

***servlet** 容器属性*

***servlet** 容器组件具有以下结构：*

- ***servlet-container***
  - ***mime-mapping***
  - *设置*
    - ***crawler-session-management***
    - ***jsp***
    - ***persistent-sessions***
    - ***session-cookie***
    - ***Websockets***
  - ***welcome-file***

***servlet-container** 属性*

*表 **A.64. servlet-container** 属性*

属性	默认	描述
allow-non-standard-wrappers	false	是否可以使用不扩展标准打包程序类的请求和响应封装器。
default-buffer-cache	default	用于缓存静态资源的缓冲区缓存。
default-cookie-version	0	用于应用创建的 Cookie 的默认 cookie 版本。
default-encoding		用于所有已部署应用的默认编码。

属性	默认	描述
default-session-timeout	30	容器中部署的所有应用的默认会话超时时间（以分钟为单位）。
directory-listing		如果应该为默认 servlet 启用目录列表：
disable-caching-for-secured-pages	true	是否为受保护的 paged 设置标头来禁用缓存。禁用此功能可能会导致安全问题，因为敏感页面可以被中间人缓存。
disable-file-watch-service	false	如果设为 **true**，则不会使用文件监视服务来监控爆炸式部署中的更改。此属性覆盖了 **io.undertow.disable-file-system-watcher** 系统属性。
disable-session-id-reuse	false	如果设为 **true**，则永远不会重复使用未知会话 ID 并生成新的会话 ID。如果设置为 **false**，则只有另一部署的会话管理器中存在会话 ID，以允许在同一服务器上的应用之间共享相同的会话 ID，才会重复使用会话 ID。
eager-filter-initialization	false	是否在部署时调用过滤器 init（），而不是在第一次请求时调用。
ignore-flush	false	忽略 servlet 输出流上的刷新。在大多数情况下，这些都毫无理由影响性能。
max-sessions		一次可以处于活跃状态的会话数量上限。
主动验证	true	是否应该使用主动身份验证。如果这是 **true**，则如果存在凭据，用户将始终进行身份验证。
session-id-length	30	会话 ID 越长，安全性越高。此值指定生成的会话 ID 的长度，以字节为单位。系统将生成的会话 ID 编码为 Base64 字符串，并将结果作为会话 ID Cookie 提供给客户端。由于此处理，服务器向客户端发送一个 Cookie 值，比原先生成的会话 ID 大约为 33%。例如，会话 ID 长度为 30 时，cookie 值长度为 40。
stack-trace-on-error	local-only	如果错误时应生成带有堆栈 trace 的错误页面。值是 **全部**、**无** 值和 **仅本地值.**
use-listener-encoding	false	使用侦听器中定义的编码。

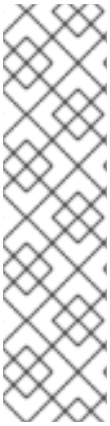
miME-mapping Attributes

*表 **A.65. miME-mapping Attributes***

属性	默认	描述
value		此映射的 mime 类型。

crawler-session-management Attributes

*为爬虫 **bots** 配置特殊会话处理。*



注意

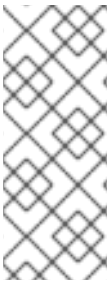
*在使用管理 **CLI** 管理 **crawler-session-management** 元素时，它位于 **servlet-container** 元素 的设置 下。例如：*

```
/subsystem=undertow/servlet-container=default/setting=crawler-session-management:add
/subsystem=undertow/servlet-container=default/setting=crawler-session-management:read-resource
```

*表 **A.66. crawler-session-management Attributes***

属性	默认	描述
session-timeout		由 crawlers 拥有的会话的会话超时（以秒为单位）。
user-agents		用于匹配爬虫的用户代理的正则表达式。

***JSP** 属性*



注意

*在使用管理 **CLI** 管理 **jsp** 元素时，它位于 **servlet-container** 元素 的设置 下。例如：*

```
/subsystem=undertow/servlet-container=default/setting=jsp:read-resource
```

*表 **A.67. JSP** 属性*

属性	默认	描述
check-interval	0	使用后台线程检查 JSP 更新的间隔。这不适用于大多数使用文件系统通知 API 处理 JSP 更改通知的部署。这只有在文件 watch 服务被禁用时才生效。
开发	false	启用允许即时重新加载 JSP 的开发模式。

属性	默认	描述
disabled	false	启用 JSP 容器。
display-source-fragment	true	发生运行时错误时，尝试显示对应的 JSP 源片段。
dump-smap	false	将 SMAP 数据写入文件.
error-on-use-bean-invalid-class-attribute	false	在 useBean 中使用错误类时启用错误。
generate-strings-as-char-arrays	false	生成字符串常量作为 char 数组。
java-encoding	UTF8	指定用于 Java 源的编码。
keep-generated	true	保留生成的 servlet。
mapping-file	true	映射到 JSP 源.
modification-test-interval	4	两次测试用于更新的最短时间，以秒为单位。
optimize-scriptlets	false	如果 JSP 脚本let 应该被优化来删除字符串串联。
recompile-on-fail	false	对每个请求重试失败的 JSP 编译。
scratch-dir		指定其他工作目录。
SMAP	true	启用 SMAP.
source-vm	1.8	用于编译的源虚拟机级别.
tag-pooling	true	启用标签池。
target-vm	1.8	用于编译的目标虚拟机级别.
trim-spaces	false	从生成的 servlet 中修剪一些空格。
x-powered-by	true	启用 x 驱动型 JSP 引擎的广告。

***persistent-sessions** 属性*

注意

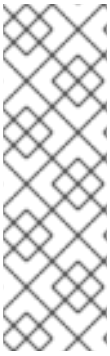
*在使用管理 **CLI** 管理 **persistent-sessions** 元素时，它位于 **servlet-container** 元素 的设置 下。例如：*

```
/subsystem=undertow/servlet-container=default/setting=persistent-sessions:add
/subsystem=undertow/servlet-container=default/setting=persistent-sessions:read-resource
```

*表 **A.68. persistent-sessions** 属性*

属性	默认	描述
路径		持久会话数据目录的路径。如果为空，会话将存储在内存中。
relative-to		路径相对于的目录。

***session-cookie** 属性*



注意

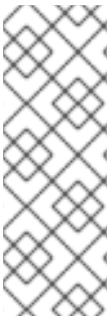
*在使用管理 **CLI** 管理 **session-cookie** 元素时，它位于 **servlet-container** 元素 的设置下。例如：*

```
/subsystem=undertow/servlet-container=default/setting=session-cookie:add
/subsystem=undertow/servlet-container=default/setting=session-cookie:read-resource
```

*表 **A.69. session-cookie** 属性*

属性	默认	描述
注释		Cookie 注释.
domain		Cookie 域.
http-only		Cookie 是否仅 http .
max-age		Cookie 最长期限.
name		Cookie 的名称。
安全		Cookie 是否安全。

***Websockets** 属性*



注意

*在使用管理 **CLI** 管理 **websockets** 元素时，它位于 **servlet-container** 元素 的设置下。例如：*

```
/subsystem=undertow/servlet-container=default/setting=websockets:read-resource
```

*表 **A.70. Websockets** 属性*

属性	默认	描述
buffer-pool	default	用于 websocket 部署的缓冲区池。
deflater-level	0	配置 DEFLATE 算法的压缩级别。
dispatch-to-worker	true	回调是否应该分配给 worker 线程。如果这为 **false**，则它们在 IO 线程中运行，速度更快，但必须小心不要执行阻止操作。
per-message-deflate	false	启用 websocket 按消息压缩扩展。
worker	default	用于 websocket 部署的 worker。

***welcome-file** 属性*

定义一个欢迎文件，并且没有选项。

过滤属性

*这些组件可以在 **/subsystem=undertow/configuration=filter** 找到。*

***custom-filter** 过滤器*

*表 **A.71. custom-filter** 属性*

属性	默认	描述
class-name		HttpHandler 类名称.
module		从其中加载类的模块名称。
parameters		过滤参数.

错误页面过滤器

错误页面

表 A.72. 错误页面属性

属性	默认	描述
Code		错误页面代码.
路径		错误页面路径.

***expression-filter** 过滤器*

*从 **Undertow** 表达式语言解析的过滤器。*

*表 A.73. **expression-filter** 属性*

属性	默认	描述
expression		定义过滤器的表达式。
module		用于加载过滤器定义的模块。

***gzip** 过滤器*

*定义 **gzip** 过滤器，并且没有属性。*

***MoD-cluster** 过滤器*

***mod-cluster** 过滤器组件具有以下结构：*

- ***Mod-cluster***
  - ***balancer***
    - ***load-balancing-group***
    - *节点*

- *context*

表 A.74. MoD-cluster Attributes

属性	默认	描述
advertise-frequency	10000	mod_cluster 在网络上公告自身的频率（以毫秒为单位）。
advertise-path	/	mod_cluster 注册的路径。
advertise-protocol	http	正在使用的协议。
advertise-socket-binding		用于公告的多播组。
break-node-timeout	60000	从表中删除中断的节点前必须等待的时间。
cached-connections-per-thread	5	无限期保留的连接数。
connection-idle-timeout	60	连接在关闭前可以闲置的时间。池大小到配置的最小值后，连接不会超时，这由 cached **-connections-per-thread** 配置。
connections-per-thread	10	每个 IO 线程将维护到后端服务器的连接数量。
enable-http2	false	负载均衡器是否应该尝试将后端连接升级到 HTTP/2。如果不支持 HTTP/2，则将正常使用 HTTP 或 HTTPS。
failover-strategy	LOAD_BALANCED	确定如何在会话的节点不可用时选择故障切换节点的属性。
health-check-interval	10000	对后端节点进行健康检查的频率。
http2-enable-push	true	是否应该为 HTTP/2 连接启用 push。
http2-header-table-size	4096	用于 HPACK 压缩的标头表大小，以字节为单位。此内存量将分配给每个连接以进行压缩。较大的值使用更多内存，但可能会提供更好的压缩。
http2-initial-window-size	65535	流控制窗口大小（以字节为单位），它控制客户端向服务器发送数据的速度。
http2-max-concurrent-streams		在单个连接上随时处于活跃状态的 HTTP/2 流的最大数量。
http2-max-frame-size	16384	最大 HTTP/2 帧大小，以字节为单位。

属性	默认	描述
http2-max-header-list-size		服务器准备接受的请求标头的最大大小，以字节为单位。
management-access-predicate		应用到传入请求的 predicate，以确定它们是否可以执行 mod 集群管理命令。通过将管理限制为源自 management- **socket-binding** 的请求来提供额外的安全性。
management-socket-binding		mod_cluster 管理端口的套接字绑定。使用 mod_cluster 时应定义两个 HTTP 侦听器，一个用于处理请求，另一个则绑定到内部网络来处理群集命令。此套接字绑定应当与内部监听器对应，并且不应公开访问。
max-ajp-packet-size	8192	AJP 数据包的最大大小，以字节为单位。增加此设置可让 AJP 处理具有大量标头的请求和响应。这在负载均衡器和后端服务器之间必须相同。
max-request-time	-1	向后端节点的请求在终止前可以等待的最长时间。
max-retries	1	如果请求失败，则尝试重试请求的次数。 **注意** 如果请求不被视为幂等，则只有在代理可以确定它没有发送到后端服务器时，才会重试请求。
request-queue-size	10	如果在请求被拒绝 503 之前连接池已满，可以排队的请求数。
security-key		用于 mod_cluster 组的安全键。所有成员必须使用相同的安全密钥。
security-realm		提供 SSL 配置的安全域。*弃用：使用**ssl-context** 属性直接引用配置的 SSLContext。*
ssl-context		对过滤器使用的 **SSLContext** 的引用。
use-alias	false	是否执行别名检查。
worker	default	用于发送公告通知的 XNIO 工作程序。

*表 **A.75.** 负载均衡器属性*

属性	默认	描述
max-attempts		将请求发送到后端服务器的尝试数量。
sticky-session		如果启用了粘性会话.
sticky-session-cookie		会话 Cookie 名称。
sticky-session-force		如果为 **true**，则如果请求无法路由到 sticky 节点，则返回错误，否则它将路由到另一节点。
sticky-session-path		粘性会话 Cookie 的路径。
sticky-session-remove		如果请求无法路由到正确的主机，则删除会话 Cookie。
wait-worker		等待可用 worker 的秒数。

load-balancing-group Attributes

定义负载平衡组，无选项。

*表 **A.76.** 节点属性*

属性	默认	描述
Alias		节点别名。
cache-connections		无限期保持活动的连接数。
已选择		已选定计数。
flush-packets		如果收到数据，应立即清空。
load		此节点的当前负载。
load-balancing-group		此节点所属的负载平衡组。
max-connections		每个 IO 线程的最大连接数。
open-connections		当前打开的连接数。
ping		节点 ping。
queue-new-requests		如果收到请求，并且没有 worker 立即可用（如果它排队）。

属性	默认	描述
读取		从节点读取的字节数。
request-queue-size		请求队列的大小。
status		此节点的当前状态。
timeout		请求超时。
ttl		如果连接的数量大于 **cache-connections**，则在关闭前，时间连接将保持活动状态且无请求。
uri		负载平衡器用于连接节点的 URI。
written		传输到节点的字节数。

*表 **A.77.** 上下文属性*

属性	默认	描述
requests		针对此上下文的请求数。
status		此上下文的状态。

request-limit Filters

*表 **A.78. request-limit** 属性*

属性	默认	描述
max-concurrent-requests		并发请求的最大数量.
queue-size		在队列开始被拒绝前的请求数。

response-header Filters

通过响应标头过滤器，您可以添加自定义标头。

*表 **A.79. response-header Attributes***

属性	默认	描述
header-name		标头名称。
header-value		标头值。

重写过滤器

*表 **A.80.** 重写属性*

属性	默认	描述
重定向	false	是否要进行重定向而不是重写.
目标		定义目标的表达式。如果您要重定向到固定目标，请在值旁边放置单引号。

处理程序属性

*这些组件可以在 **/subsystem=undertow/configuration=handler** 找到。*

文件属性

*表 **A.81.** 文件属性*

属性	默认	描述
cache-buffer-size	1024	缓冲区的大小.
cache-buffers	1024	缓冲区数量.
case-sensitive	true	是否使用区分大小写的文件处理.请注意，仅当底层文件系统不区分大小写时，此选项 **针对** 大小写敏感才能使用。
directory-listing	false	是否启用目录列表.
follow-symlink	false	是否启用以下符号链接：
路径		文件处理程序将提供资源的文件系统中的路径。
safe-symlink-paths		符号链接目标安全的路径.

*将 **WebDAV** 用于静态资源*

*早期版本的 **JBoss EAP** 允许通过 **Web davServlet** 将 **Web DAV** 与 **Web** 子系统搭配使用，以托管静态资源，并且启用额外的 **HTTP** 方法来访问和操作这些文件。在 **JBoss EAP 7** 中，**undertow** 子系统提供了使用文件处理程序提供静态文件的机制，但 **undertow** 子系统不支持 **WebDAV**。如果要将 **WebDAV** 与 **JBoss EAP 7** 搭配使用，您可以编写自定义 **WebDAVservlet**。*

***reverse-proxy** 属性*

***reverse-proxy** 处理器组件具有以下结构：*

- *reverse-proxy*
  - *主机*

表 A.82. reverse-proxy 属性

属性	默认	描述
cached-connections-per-thread	5	无限期保留的连接数。
connection-idle-timeout	60	连接在关闭前可以闲置的时间。当池大小到配置的最小值（如 cached-connections-per-thread 配置）后，连接不会超时。
connections-per-thread	40	每个 IO 线程将维护到后端服务器的连接数量。
max-request-time	-1	代理请求在被终止前可以激活的最长时间。默认值为无限。
max-retries	1	如果请求失败，则尝试重试请求的次数。  **注意** 如果请求不被视为幂等，则只有在代理可以确定它没有发送到后端服务器时，才会重试请求。
problem-server-retry	30	尝试重新连接到停机的服务器前等待的时间（以秒为单位）。
request-queue-size	10	如果在请求被拒绝 503 之前连接池已满，可以排队的请求数。
session-cookie-names	JSESSIONID	会话 Cookie 名称的逗号分隔列表。通常这仅仅是 JSESSIONID。

表 A.83. 主机属性

属性	默认	描述
enable-http2	false	如果为 **true**，代理将尝试使用 HTTP/2 连接到后端。如果不支持，它将回退到 HTTP/1.1。
instance-id		用于启用粘性会话的实例 ID 或 JVM 路由。
outbound-socket-binding		此主机的出站套接字绑定.

属性	默认	描述
路径	/	如果主机使用非 root 资源，则可选路径。
scheme	http	使用的方案类型。
security-realm		为主机连接提供 SSL 配置的安全域。
ssl-context		引用此处理程序要使用的 SSLContext。

服务器属性

服务器组件有以下结构：

- ***server***
  - ***ajp-listener***
  - *主机*
    - ***filter-ref***
    - *位置*
      - ***filter-ref***
    - *设置*
      - ***access-log***
      - ***console-access-log***
      - ***http-invoker***

- *single-sign-on*
  - *http-listener*
  - *https-listener*

服务器属性

表 A.84. 服务器属性

属性	默认	描述
default-host	default-host	服务器的默认虚拟主机.
servlet-container	default	服务器的默认 servlet 容器。

ajP-listener Attributes

表 A.85. ajP-listener Attributes

属性	默认	描述
allow-encoded-slash	false	如果请求附带编码的字符，如 **%2F**，则请求是否将被解码。
allow-equals-in-cookie-value	false	是否允许非转义的 cookie 值中的字符等于字符。未引用的 Cookie 值可能不包含等同字符。如果值在等号之前结束。Cookie 值的其余部分将被丢弃。
allow-unescaped-characters-in-url	false	是否允许 URL 中的非转义字符。如果设置为 **true**，侦听器会处理包含非转义、非 ASCII 字符的任何 URL。如果设置为 **false**，侦听器将拒绝包含非转义、非 ASCII 字符且带有 **HTTP Bad Request 400** 响应代码的任何 URL。
always-set-keep-alive	true	是否会在响应中添加 Connection: keep-alive 标头，即使该规范没有严格要求。
buffer-pipelined-data	false	是否缓冲管道请求。
buffer-pool	default	AJP 侦听器的缓冲区池.
decode-url	true	如果这是 **true**，解析器将使用选定的字符编码解码 URL 和查询参数，默认为 UTF-8。如果这是误，则不会解码。这将允许后续处理程序将它们解码到所需的任何 charset 中。

属性	默认	描述
disallowed-methods	["TRACE"]	不允许的 HTTP 方法的逗号分隔列表。
enabled	true	如果启用了监听程序。*弃用：启用的属性可能会导致实施配置一致性出现问题。*
max-ajp-packet-size	8192	AJP 数据包支持的最大值。如果修改了这个值，它会在负载均衡器和后端服务器上增加。
max-buffered-request-size	16384	缓冲请求的最大大小（以字节请求为单位）通常不会被缓冲，最常见的情况是在对 POST 请求执行 SSL 重新协商时，并且必须完全缓冲后数据才能执行重新协商。
max-connections		并发连接的最大数量。如果没有在服务器配置中设置值，则并发连接数的限制为 **Integer.MAX_VALUE**。
max-cookies	200	要解析的最大 Cookie 数。这用于防止哈希漏洞。
max-header-size	1048576	HTTP 请求标头的最大字节大小。
max-headers	200	要解析的最大标头数。这用于防止哈希漏洞。
max-parameters	1000	要解析的参数的最大数量。这用于防止哈希漏洞。这适用于查询参数和 POST 数据，但不适用于累积数据。例如，您可以具有 max 参数 * 2 个总参数。
max-post-size	10485760	将接受的 post 的最大大小
no-request-timeout	60000	容器关闭连接前可以闲置的时长，以毫秒为单位。
read-timeout		配置套接字的读取超时，以毫秒为单位。如果给定的时间没有成功读取，则套接字的下一个读取将引发 **ReadTimeoutException**。
receive-buffer		接收缓冲区的大小。
record-request-start-time	false	是否记录请求的开始时间，以允许记录请求时间。这会对性能产生轻微但可观的影响。
redirect-socket		如果此侦听器支持非 SSL 请求，并且收到匹配项需要 SSL 传输的请求，则是否自动将请求重定向到此处指定的套接字绑定端口。

属性	默认	描述
request-parse-timeout		解析请求的最长时间，以毫秒为单位。
resolve-peer-address	false	启用主机 DNS 查找。
scheme		侦听器方案可以是 HTTP 或 HTTPS。默认情况下，方案将从传入的 AJP 请求中获取。
安全	false	如果 **情况如此**，则源自此侦听器的请求会标记为安全，即使请求不使用 HTTPS。
send-buffer		发送缓冲区大小。
socket-binding		AJP 侦听器的套接字绑定.
tcp-backlog		配置具有指定积压的服务器.
tcp-keep-alive		将通道配置为以依赖实现的方式发送 TCP keep-alive 消息。
url-charset	UTF-8	URL charset.
worker	default	侦听器的 XNIO 工作程序.
write-timeout		为套接字配置写入超时，以毫秒为单位。如果给定的时间没有成功写入，则套接字的下一次写入将引发 **WriteTimeoutException**。

主机属性

*表 **A.86.** 主机属性*

属性	默认	描述
Alias		以逗号分隔的主机别名列表。
default-response-code	404	如果设置，这是在服务器上不存在请求的上下文时发回的响应代码。
default-web-module	ROOT.war	默认 Web 模块.
disable-console-redirect	false	如果设为 **true**，则不会为此主机启用/**console** 重定向。
queue-requests-on-start	true	如果设为 **true**，则在此主机的启动时应将请求排队。如果设为 **false**，则返回默认的响应代码。

filter-ref Attributes

*表 **A.87. filter-ref Attributes***

属性	默认	描述
predicate		predicates 提供了一种基于交换做出正确/假决策的简单方法。许多处理程序都要求有条件地应用它们，谓词则提供指定条件的一般方式。
priority	1	定义过滤顺序。较低数字指示服务器早期包含在处理程序链中，而与其他上下文上方的其他链相比。值范围为 **1**，表示将首先处理过滤器到 **2147483647**，从而导致最后处理的过滤器。

位置属性

*表 **A.88.** 位置属性*

属性	默认	描述
handler		此位置的默认处理程序.

filter-ref Attributes

*表 **A.89. filter-ref Attributes***

属性	默认	描述
predicate		predicates 提供了一种基于交换做出正确/假决策的简单方法。许多处理程序都要求有条件地应用它们，谓词则提供指定条件的一般方式。
priority	1	定义过滤顺序。它应设置为 1 或更多。较高的数字指示服务器早于同一上下文下的其他链包含在处理程序链中。

***access-log** 属性*

注意

*在使用管理 **CLI** 管理 **access-log** 元素时，它位于 主机 元素的 设置 下。例如：*

```
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:add
/subsystem=undertow/server=default-server/host=default-host/setting=access-log:read-resource
```

*表 **A.90. access-log** 属性*

属性	默认	描述
目录	${jboss.server.log.dir}	要保存日志的目录。
Extended	false	日志是否使用扩展日志文件格式。
pattern	common	访问日志模式。如需有关可用于此属性的选项的详细信息，请参阅 JBoss EAP *开发指南中的提供* Undertow 处理程序。  **注意** 如果您将 **模式** 设置为打印处理请求的时间，您必须在适当的监听器上启用 **record-request-start-time** 属性；否则，访问日志中将无法正确记录时间。例如： `/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=record-request-start-time,value=true)`
predicate		确定是否应记录请求的 predicate。
prefix	access_log.	日志文件名称的前缀。
relative-to		路径相对于的目录。
rotate	true	是否每天轮转访问日志。
suffix	log	日志文件名称的后缀。
use-server-log	false	日志是否应写入到服务器日志，而不是单独的 文件。
worker	default	用于日志记录的 worker 的名称。

***console-access-log** 属性*

*表 **A.91. console-access-log** 属性*

属性	默认	描述

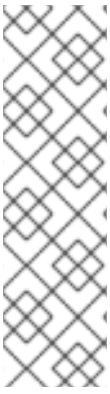
属性	默认	描述
属性	{remote-host={},remote-user={},date-time={},request-line={},response-code={},bytes-sent={}}	指定要包含在控制台访问日志输出中的日志数据，或自定义默认数据。
include-host-name	false	指定是否将主机名包含在 JSON 结构化输出中。如果设置为 **true**，结构化数据中的键为"hostName"，其值是配置 console-access-log 的主机的名称。
metadata		指定要包含在控制台访问日志输出中的自定义元数据。
predicate		确定是否应记录请求的 predicate。
worker	default	用于日志记录的 worker 的名称。

HTTP-invoker Attributes

*表 **A.92. HTTP-invoker Attributes***

属性	默认	描述
http-authentication-factory		用于身份验证的 HTTP 身份验证工厂。
路径	wildfly-services	安装服务的路径。
security-realm		用于身份验证的传统安全域。

单点登录属性



注意

*在使用管理 **CLI** 管理 单点登录元素时，它位于 主机 元素的 设置 下。例如：*

```
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:add
/subsystem=undertow/server=default-server/host=default-host/setting=single-sign-on:read-resource
```

重要

*虽然分布式单点登录与应用透视图不同于先前版本的 **JBoss EAP**，但在 **JBoss EAP 7** 中，身份验证信息的缓存和发布将有所不同。对于 **JBoss EAP 7**，运行 **ha** 配置文件时，默认情况下，每一主机都有自己的 **Infinispan** 缓存，它将存储相关的会话和 **SSO Cookie** 信息。此缓存基于 **Web** 缓存容器的默认缓存。**JBoss EAP** 还将处理在所有主机的单个缓存之间传播的信息。*

*表 **A.93.** 单点登录属性*

属性	默认	描述
cookie-name	JSESSIONIDSSO	Cookie 的名称。
domain		将使用的 Cookie 域。
http-only	false	设置 Cookie httpOnly 属性.
路径	/	Cookie 路径.
安全	false	设置 Cookie 安全属性.

HTTP-listener Attributes

*表 **A.94. HTTP-listener Attributes***

属性	默认	描述
allow-encoded-slash	false	如果请求附带编码的字符，如 **%2F**，则请求是否将被解码。
allow-equals-in-cookie-value	false	是否允许非转义的 cookie 值中的字符等于字符。未引用的 Cookie 值可能不包含等同字符。如果值在等号之前结束。Cookie 值的其余部分将被丢弃。
allow-unescaped-characters-in-url	false	是否允许 URL 中的非转义字符。如果设置为 **true**，侦听器会处理包含非转义、非 ASCII 字符的任何 URL。如果设置为 **false**，侦听器将拒绝包含非转义、非 ASCII 字符且带有 **HTTP Bad Request 400** 响应代码的任何 URL。
always-set-keep-alive	true	是否会在响应中添加 Connection: keep-alive 标头，即使该规范没有严格要求。
buffer-pipelined-data	false	是否缓冲管道请求。
buffer-pool	default	侦听器的缓冲区池。

属性	默认	描述
certificate-forwarding	false	是否启用证书转发。如果启用此项，侦听器将从 **SSL_CLIENT_CERT 属性获取** 证书。只有在代理后面时才启用此功能，代理会被配置为始终设置这些标头。
decode-url	true	解析器是否使用选定的字符编码解码 URL 和查询参数，默认为 UTF-8。如果这是误，则不会解码。这将允许后续处理程序将它们解码到所需的任何 charset 中。
disallowed-methods	["TRACE"]	不允许的 HTTP 方法的逗号分隔列表。
enable-http2	false	是否为此侦听器启用 HTTP/2 支持。
enabled	true	是否启用监听器。*弃用：启用的属性可能会导致实施配置一致性出现问题。*
http2-enable-push	true	是否为此连接启用了服务器推送。
http2-header-table-size	4096	用于 HPACK 压缩的标头表的大小（以字节为单位）。此内存量将分配给每个连接以进行压缩。较大的值使用更多内存，但可能会提供更好的压缩。
http2-initial-window-size	65535	流控制窗口大小（以字节为单位），它控制客户端向服务器发送数据的速度。
http2-max-concurrent-streams		在单个连接上随时处于活跃状态的 HTTP/2 流的最大数量。
http2-max-frame-size	16384	最大 HTTP/2 帧大小，以字节为单位。
http2-max-header-list-size		服务器准备接受的最大请求标头大小。
max-buffered-request-size	16384	缓冲请求的最大大小（以字节请求为单位）通常不会被缓冲，最常见的情况是在对 POST 请求执行 SSL 重新协商时，并且必须完全缓冲后数据才能执行重新协商。
max-connections		并发连接的最大数量。如果没有在服务器配置中设置值，则并发连接数的限制为 **Integer.MAX_VALUE**。
max-cookies	200	要解析的最大 Cookie 数。这用于防止哈希漏洞。
max-header-size	1048576	HTTP 请求标头的最大字节大小。
max-headers	200	要解析的最大标头数。这用于防止哈希漏洞。

属性	默认	描述
max-parameters	1000	要解析的参数的最大数量。这用于防止哈希漏洞。这适用于查询参数和 POST 数据，但不适用于累积数据。例如，您可以具有 max 参数 * 2 个总参数。
max-post-size	10485760	将接受的 post 的最大大小。
no-request-timeout	60000	容器关闭连接前可以闲置的时长，以毫秒为单位。
proxy-address-forwarding	false	是否启用 x-forwarded-host 和类似的标头，并设置远程 IP 地址和主机名。
proxy-protocol	false	是否使用 PROXY 协议传输连接信息。如果设置为 **true**，侦听器将使用 PROXY 协议版本 1，如PROXY 协议版本 1 和 2 规范中所 定义。只有在支持同一协议的负载均衡器后面的监听程序才会启用这个选项。
read-timeout		配置套接字的读取超时，以毫秒为单位。如果给定的时间没有成功读取，则套接字的下一个读取将引发 **ReadTimeoutException**。
receive-buffer		接收缓冲区的大小。
record-request-start-time	false	是否记录请求的开始时间，以允许记录请求时间。这会对性能产生轻微但可观的影响。
redirect-socket		如果此侦听器支持非 SSL 请求，并且收到匹配项需要 SSL 传输的请求，则是否自动将请求重定向到此处指定的套接字绑定端口。
request-parse-timeout		解析请求的最长时间，以毫秒为单位。
require-host-http11	false	它要求所有 HTTP/1.1 请求都有一个 **Host** 标头。如果请求不包含此标头，它将被拒绝，并显示 403 错误。
resolve-peer-address	false	启用主机 DNS 查找。
安全	false	如果 **情况如此**，源自此侦听器的请求会标记为安全，即使请求没有使用 HTTPS。
send-buffer		发送缓冲区大小。
socket-binding		侦听器的套接字绑定
tcp-backlog		配置具有指定积压的服务器.
tcp-keep-alive		将通道配置为以依赖实现的方式发送 TCP keep-alive 消息。

属性	默认	描述
url-charset	UTF-8	URL charset.
worker	default	侦听器的 XNIO 工作程序.
write-timeout		为套接字配置写入超时，以毫秒为单位。如果给定的时间没有成功写入，则套接字的下一次写入将引发 **WriteTimeoutException**。

https-listener Attributes

*表 **A.95. https-listener Attributes***

属性	默认	描述
allow-encoded-slash	false	如果请求附带编码的字符，如 **%2F**，则请求是否将被解码。
allow-equals-in-cookie-value	false	是否允许非转义的 cookie 值中的字符等于字符。未引用的 Cookie 值可能不包含等同字符。如果值在等号之前结束。Cookie 值的其余部分将被丢弃。
allow-unescaped-characters-in-url	false	是否允许 URL 中的非转义字符。如果设置为 **true**，侦听器会处理包含非转义、非 ASCII 字符的任何 URL。如果设置为 **false**，侦听器将拒绝包含非转义、非 ASCII 字符且带有 **HTTP Bad Request 400** 响应代码的任何 URL。
always-set-keep-alive	true	是否会在响应中添加 Connection: keep-alive 标头，即使该规范没有严格要求。
buffer-pipelined-data	false	是否缓冲管道请求。
buffer-pool	default	侦听器的缓冲区池。
certificate-forwarding	false	是否启用证书转发。如果启用此项，侦听器将从 **SSL_CLIENT_CERT 属性获取** 证书。只有在代理后面时才启用此功能，代理会被配置为始终设置这些标头。
decode-url	true	解析器是否使用选定的字符编码解码 URL 和查询参数，默认为 UTF-8。如果这是误，则不会解码。这将允许后续处理程序将它们解码到所需的任何 charset 中。
disallowed-methods	["TRACE"]	不允许的 HTTP 方法的逗号分隔列表。
enable-http2	false	启用对此监听器的 HTTP/2 支持。

属性	默认	描述
enable-spdy	false	启用对此监听器的 SPDY 支持。*弃用：SPDY 已被 HTTP/2 替代。*
enabled	true	如果启用了监听程序。*弃用：启用的属性可能会导致实施配置一致性出现问题。*
enabled-cipher-suites		配置已启用的 SSL 密码.*弃用：引用 SSLContext 的位置，应使用要支持的加密套件进行配置。*
enabled-protocols		配置 SSL 协议.*弃用：引用 SSLContext 的位置，应使用要支持的加密套件进行配置。*
http2-enable-push	true	如果为此连接启用了服务器推送。
http2-header-table-size	4096	用于 HPACK 压缩的标头表的大小（以字节为单位）。此内存量将分配给每个连接以进行压缩。较大的值使用更多内存，但可能会提供更好的压缩。
http2-initial-window-size	65535	流控制窗口大小（以字节为单位），它控制客户端向服务器发送数据的速度。
http2-max-concurrent-streams		在单个连接上随时处于活跃状态的 HTTP/2 流的最大数量。
http2-max-frame-size	16384	最大 HTTP/2 帧大小，以字节为单位。
http2-max-header-list-size		服务器准备接受的最大请求标头大小。
max-buffered-request-size	16384	缓冲请求的最大大小（以字节请求为单位）通常不会被缓冲，最常见的情况是在对 POST 请求执行 SSL 重新协商时，并且必须完全缓冲后数据才能执行重新协商。
max-connections		并发连接的最大数量。如果没有在服务器配置中设置值，则并发连接数的限制为 **Integer.MAX_VALUE**。
max-cookies	100	要解析的最大 Cookie 数。这用于防止哈希漏洞。
max-header-size	1048576	HTTP 请求标头的最大字节大小。
max-headers	200	要解析的最大标头数。这用于防止哈希漏洞。
max-parameters	1000	要解析的参数的最大数量。这用于防止哈希漏洞。这适用于查询参数和 POST 数据，但不适用于累积数据。例如，您可以具有 max 参数 * 2 个总参数。

属性	默认	描述
max-post-size	10485760	将接受的 post 的最大大小。
no-request-timeout	60000	容器关闭连接前可以闲置的时长，以毫秒为单位。
proxy-address-forwarding	false	启用处理 x-forwarded-host 标头和其他 x-forwarded-* 标头，并使用此标头信息设置远程地址。这应该只在设置这些标头的可信代理后面使用，否则远程用户可能会欺骗其 IP 地址。
proxy-protocol	false	是否使用 PROXY 协议传输连接信息。如果设置为 **true**，侦听器将使用 PROXY 协议版本 1，如 PROXY 协议版本 1 和 2 规范中所 定义。只有在支持同一协议的负载均衡器后面的监听程序才会启用这个选项。
read-timeout		配置套接字的读取超时，以毫秒为单位。如果给定的时间没有成功读取，则套接字的下一个读取将引发 **ReadTimeoutException**。
receive-buffer		接收缓冲区的大小。
record-request-start-time	false	是否记录请求的开始时间，以允许记录请求时间。这会对性能产生轻微但可观的影响。
request-parse-timeout		解析请求的最长时间，以毫秒为单位。
require-host-http11	false	要求所有 HTTP/1.1 请求都具有 'Host' 标头。如果请求不包含此标头，它将使用 403 拒绝。
resolve-peer-address	false	启用主机 DNS 查找。
安全	false	如果 **情况如此**，源自此侦听器的请求会标记为安全，即使请求没有使用 HTTPS。
security-realm		侦听器的安全域.*弃用：使用**ssl-context** 属性直接引用配置的 SSLContext。*
send-buffer		发送缓冲区大小。
socket-binding		侦听器的套接字绑定.
ssl-context		引用此侦听器要使用的 SSLContext。
ssl-session-cache-size		活跃 SSL 会话的最大数量.*弃用：现在可在 Elytron 安全上下文中配置。*
ssl-session-timeout		SSL 会话的超时时间，以秒为单位。*弃用：现在可在 Elytron 安全上下文中配置。*

属性	默认	描述
tcp-backlog		配置具有指定积压的服务器.
tcp-keep-alive		将通道配置为以依赖实现的方式发送 TCP keep-alive 消息。
url-charset	UTF-8	URL charset.
verify-client	NOT_REQUESTED	SSL 频道所需的 SSL 客户端身份验证模式.*弃用：在引用 SSLContext 时，应当直接为客户端验证模式配置它。*
worker	default	侦听器的 XNIO 工作程序.
write-timeout		为套接字配置写入超时，以毫秒为单位。如果给定的时间没有成功写入，则套接字的下一次写入将引发 **WriteTimeoutException**。

A.28. UNDERTOW 子系统统计信息

*表 **A.96. ajp-listener Statistics***

名称	描述
bytes-received	此侦听器收到的字节数。
bytes-sent	在这个监听器上发出的字节数。
error-count	此侦听器发送的 500 个响应数。
max-processing-time	对此监听器的请求花费的最大处理时间。
processing-time	此监听器发送的所有请求的总处理时间。
request-count	此侦听器服务的请求数。

*表 **A.97. HTTP-listener Statistics***

名称	描述
bytes-received	此侦听器收到的字节数。
bytes-sent	在这个监听器上发出的字节数。

名称	描述
error-count	此侦听器发送的 500 个响应数。
max-processing-time	对此监听器的请求花费的最大处理时间。
processing-time	此监听器发送的所有请求的总处理时间。
request-count	此侦听器服务的请求数。

*表 **A.98. HTTPS-listener Statistics***

名称	描述
bytes-received	此侦听器收到的字节数。
bytes-sent	在这个监听器上发出的字节数。
error-count	此侦听器发送的 500 个响应数。
max-processing-time	对此监听器的请求花费的最大处理时间。
processing-time	此监听器发送的所有请求的总处理时间。
request-count	此侦听器服务的请求数。

A.29. HTTP 方法的默认行为

*与之前的 **JBoss EAP** 发行版中的 **Web** 子系统相比，**JBoss EAP 7.3** 中的 **undertow** 子系统具有 **HTTP** 方法的不同默认行为。下表概述了 **JBoss EAP 7.3** 中的默认行为。*

*表 **A.99. HTTP** 方法默认行为*

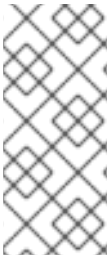
HTTP 方法	JSP	静态 HTML	按文件处理程序的静态 HTML
GET	确定	确定	确定
POST	确定	NOT_ALLOWED	确定
HEAD	确定	确定	确定
PUT	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
TRACE	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED

HTTP 方法	JSP	静态 HTML	按文件处理程序的静态 HTML
删除	NOT_ALLOWED	NOT_ALLOWED	NOT_ALLOWED
选项	NOT_ALLOWED	确定	NOT_ALLOWED

注意

*对于 **servlets**，默认行为取决于其实施，但 **TRACE** 方法除外，其默认行为为 **NOT_ALLOWED**。*

A.30. 远程子系统属性

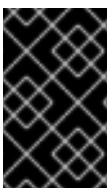


注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-remoting_4_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.100.** 远程属性*

属性	默认	描述
worker-read-threads	1	为远程 worker 创建的读取线程数量。
worker-task-core-threads	4	远程 worker 任务线程池的内核线程数量。
worker-task-keepalive	60	保持非核心重新移动 worker 任务线程的毫秒数。
worker-task-limit	16384	拒绝前允许的最大 worker 任务数量。
worker-task-max-threads	16	远程 worker 任务线程池的最大线程数。
worker-write-threads	1	为远程 worker 创建的写入线程数量。

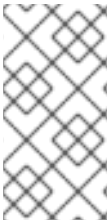


重要

*以上远程 元素 属性已弃用。现在，这些属性应当使用 **the io** 子系统进行配置。*

*表 **A.101.** 端点属性*

属性	默认	描述
auth-realm		如果未指定身份验证 CallbackHandler，则要使用的身份验证域。
authentication-retries	3	指定客户端在关闭连接前可以重试身份验证的次数。
authorize-id		SASL 授权 ID。如果未指定身份验证 CallbackHandler，并且选定的 SASL 机制需要用户名，则用作身份验证用户名以使用。
buffer-region-size		分配的缓冲区区域的大小。
heartbeat-interval	2147483647	用于连接 heartbeat 的时间间隔，以毫秒为单位。如果连接在该时间的出站方向中闲置，将会发送 ping 消息，这将触发对应的回复消息。
max-inbound-channels	40	频道中并发入站消息的最大数量。
max-inbound-message-size	9223372036854775807	允许的最大入站消息大小。超过这个大小的消息会导致在读侧以及写操作端抛出异常。
max-inbound-messages	80	支持连接的最大入站频道数。
max-outbound-channels	40	频道中并发出站消息的最大数量。
max-outbound-message-size	9223372036854775807	要发送的最大出站消息大小。不会传输大于这个值的消息；试图传送的信息将导致写操作出现异常。
max-outbound-messages	65535	支持连接的最大出站频道数。
receive-buffer-size	8192	此端点通过连接接受的最大缓冲区的大小。
receive-window-size	131072	连接频道接收方向的最大窗口大小，以字节为单位。
sasl-protocol	**remote**	创建 **SaslServer** 或 **SaslClient** 时，默认指定的协议是 **远程** 的。此属性可用于覆盖此协议。
send-buffer-size	8192	此端点通过连接传输的最大缓冲区的大小。
server-name		连接的服务器端将其名称传递到初始问候中的客户端，默认情况下，该名称会自动从连接的本地地址发现，或者可以使用此名称来覆盖。
transmit-window-size	131072	连接通道传输方向的最大窗口大小，以字节为单位。
worker	**default**	使用 worker



注意

*在使用管理 **CLI** 更新 端点 元素时，它位于远程 元素 的配置 下。例如：**/subsystem=remoting/configuration=endpoint/**。*

连接器属性

连接器组件具有以下结构：

- ***connector***
  - *属性*
  - ***security***
    - ***SASL***
      - *属性*
      - ***sasl-policy***
        - ***policy***

*表 **A.102.** 连接器属性*

属性	默认	描述
authentication-provider		**authentication-provider** 元素包含用于传入连接的身份验证提供商名称。
sasl-authentication-factory		参考 SASL 身份验证工厂来保护此连接器。
sasl-protocol	**remote**	传递到用于身份验证的 SASL 机制的协议。
security-realm		用于为此连接器进行身份验证的关联安全域。

属性	默认	描述
server-name		要在初始消息交换中发送的服务器名称，以及用于基于 SASL 的身份验证的服务器名称。
socket-binding		要关联的套接字绑定的名称（或名称）。
ssl-context		引用要用于此连接器的 SSL 上下文。

*表 **A.103.** 属性属性*

属性	默认	描述
value		属性值。

安全属性

*安全 组件允许您配置连接器的安全性，但不包含直接配置属性。它可使用其嵌套组件（如 **sasl**）进行配置。*

*表 **A.104.** SASL 属性*

属性	默认	描述
include-mechanisms		可选的嵌套 **include-mechanisms** 元素包含允许的 SASL 机制名称白名单。不允许在此列表中不存在的机制。
qop		可选的嵌套 **qop** 元素包含以逗号分隔的保护质量值列表，以降低偏好顺序。 这个列表的保护质量值为： • **auth** ：只进行身份验证 • **auth-int** ：身份验证，以及完整性保护 • **auth-conf** ：身份验证，以及完整性保护和保密性保护
reuse-session	false	可选的嵌套 **reuse-session** 布尔值元素指定服务器是否应该尝试重复利用以前通过的会话信息。机制可能支持或可能不支持此类重复使用，而其他一些因素也可能阻止此类重复使用。
server-auth	false	可选的嵌套 **server-auth** 布尔值元素指定服务器是否应该向客户端进行身份验证。并非所有机制都可能支持此设置。

属性	默认	描述
强度		可选的嵌套 **强度** 元素包含以逗号分隔的密码强度列表，以降低首选顺序。 这个列表的密码强度值为： • **high** • **中** • **低**

SASL-policy 属性

***sasl-policy** 组件允许您指定一个可选策略来缩小可用的一组机制，但不包含直接配置属性。它可使用其嵌套组件（如策略）进行配置。*

*表 **A.105.** 策略属性*

属性	默认	描述
forward-secrecy	true	可选的 nested **forward-secrecy** 元素包含一个布尔值，指定是否需要在会话之间实施转发保密的机制。转发保密意味着拆分一个会话不会自动提供拆分到将来的会话的信息。
no-active	true	可选的嵌套 **no-active** 元素包含一个布尔值，指定是否不允许受到活动（非字典）攻击的机制。**不允许**，为 **true**。
no-anonymous	true	可选的嵌套 **no-anonymous** 元素包含一个布尔值，指定是否允许接受匿名登录的机制。**不允许**，为 **true**。
no-dictionary	true	可选的嵌套 **no-dictionary** 元素包含一个布尔值，指定是否允许受被动字典攻击的机制。**不允许**，为 **true**。
no-plain-text	true	可选的嵌套 **no-plain-text** 元素包含一个布尔值，用于指定是否不允许出现简单的纯被动攻击（例如，**PLAIN**）的机制。**不允许**，为 **true**。
pass-credentials	true	可选的嵌套 **pass-credentials** 元素包含一个布尔值，用于指定是否需要传递客户端凭据的机制。

HTTP 连接器属性

***http-connector** 组件有以下结构：*

- ***http-connector***
  - *属性（也称连接器）*
  - *安全（同样是连接器）*
    - ***SASL（作为连接器）***
      - *属性（也称连接器）*
      - ***SASL-policy（作为连接器）***
        - *策略（也称连接器）*

*表 **A.106. HTTP-connector** 属性*

属性	默认	描述
authentication-provider		**authentication-provider** 元素包含用于传入连接的身份验证提供商名称。
connector-ref		要连接的 **undertow** 子系统中的连接器名称（或名称）。
sasl-authentication-factory		参考 SASL 身份验证工厂来保护此连接器。
sasl-protocol	**remote**	传递到用于身份验证的 SASL 机制的协议。
security-realm		用于为此连接器进行身份验证的关联安全域。
server-name		要在初始消息交换中发送的服务器名称，以及用于基于 SASL 的身份验证的服务器名称。

出站连接属性

***outbound-connection** 组件具有以下结构：*

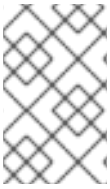
- ***outbound-connection***
  - *属性*

*表 **A.107.** 出站连接属性*

属性	默认	描述
uri		出站连接的连接 URI。

*表 **A.108.** 属性属性*

属性	默认	描述
value		属性值。



注意

*上述 属性 与创建连接期间要使用的 **XNIO** 选项相关。*

远程出站连接

***remote-outbound-connection** 组件具有以下结构：*

- ***remote-outbound-connection***
  - *属性（与出站连接相同）*

*表 **A.109.** remote-outbound-connection Attributes*

属性	默认	描述
authentication-context		引用身份验证上下文实例，其中包含出站连接的配置。
outbound-socket-binding-ref		**出站-socket-binding** 的名称，用于确定连接的目标地址和端口。

属性	默认	描述
协议	**http-remoting**	用于远程连接的协议。默认为 **http-remoting**。*弃用：出站安全设置应迁移到 身份验证上下文 定义。*
security-realm		引用用于获取密码和 SSL 配置的安全域。*弃用：出站安全设置应迁移到 身份验证上下文 定义。*
username		对远程服务器进行身份验证时使用的用户名。*弃用：出站安全设置应迁移到 身份验证上下文 定义。*

本地出站连接属性

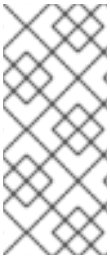
***local-outbound-connection** 组件具有以下结构：*

- *local-outbound-connection*
  - *属性（与出站连接相同）*

*表 **A.110. local-outbound-connection Attributes***

属性	默认	描述
outbound-socket-binding-ref		**出站-socket-binding** 的名称，用于确定连接的目标地址和端口。

A.31. IO 子系统属性



注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。查看位于 **EAP_HOME/docs/schema/wildfly-io_3_0.xsd** 中的架构定义文件，以查看它们出现在 **XML** 中的元素，因为管理模型可能会有所不同。*

*表 **A.111.** 工作程序属性*

属性	默认	描述
io-threads		为 worker 创建的 I/O 线程数量。如果没有指定，线程数被设置为 CPU 的数量 [1] 2。

属性	默认	描述
stack-size	0	堆栈大小（以字节为单位），以尝试用于 worker 线程。
task-keepalive	60000	非核心任务线程保持活动状态的毫秒数。
task-core-threads	2	内核任务线程池的线程数。
task-max-threads		worker 任务线程池的最大线程数。如果没有指定，则最大线程数被设置为 CPU * 16 的最大数量，采用 **MaxFileDescriptorCount** JMX 属性（如果设置）。

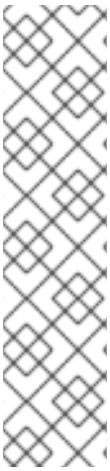
*表 **A.112. buffer-pool** 属性*

属性	默认	描述
buffer-size		每个缓冲区片段的大小（以字节为单位）。如果没有指定，则根据系统的可用 RAM 设置大小： • RAM 小于 64 MB 的 512 字节 • 1024 bytes (1 KB) for 64 MB - 128 MB RAM • 16384 字节(16 KB)，用于 128 MB RAM 有关此属性的性能调优建议，请参阅 JBoss EAP *性能调优指南中的配置* *https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.3/html-single/performance_tuning_guide/#io_buffer_pools 缓冲池*。
buffers-per-slice		将更大的缓冲区分成多少个片段或部分。这比分配多个单独的缓冲区更高效内存。如果没有指定，则根据系统的可用 RAM 设置分片数： • 10，内存小于 128 MB • 20 内存超过 128 MB
direct-buffers		缓冲区池是否使用直接缓冲区，在很多情况下，使用 NIO 时速度更快。请注意，有些平台不支持直接缓冲区。

A.32. JAKARTA SERVER FACES 模块模板

*以下是安装用于为 **JBoss EAP** 安装不同 **Jakarta Server Faces** 版本时所需的各种 **Jakarta Server Faces** 模块的示例模板：有关完整说明，请参阅安装 **Jakarta Server Faces** 实施。*

*示例：**Mojarra Jakarta Server Faces implementation JAR module.xml***



注意

确保对模板中的以下可替换变量使用适当的值：

- ***IMPL_NAME***
- ***VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="com.sun.jsf-impl:IMPL_NAME-VERSION">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <dependencies>
    <module name="javax.faces.api:IMPL_NAME-VERSION"/>
    <module name="javaee.api"/>
    <module name="javax.servlet.jstl.api"/>
    <module name="org.apache.xerces" services="import"/>
    <module name="org.apache.xalan" services="import"/>
    <module name="org.jboss.weld.core"/>
    <module name="org.jboss.weld.spi"/>
    <module name="javax.xml.rpc.api"/>
    <module name="javax.rmi.api"/>
    <module name="org.omg.api"/>
  </dependencies>

  <resources>
    <resource-root path="impl-VERSION.jar"/>
  </resources>
</module>
```

*示例：**MyFaces Jakarta Server Faces implementation JAR module.xml***

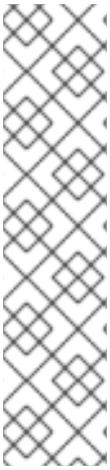
注意

确保对模板中的以下可替换变量使用适当的值：

- **IMPL_NAME**
- **VERSION**

```
<module xmlns="urn:jboss:module:1.8" name="com.sun.jsf-impl:IMPL_NAME-VERSION">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <dependencies>
    <module name="javax.faces.api:IMPL_NAME-VERSION">
      <imports>
        <include path="META-INF/**"/>
      </imports>
    </module>
    <module name="javaee.api"/>
    <module name="javax.servlet.jstl.api"/>
    <module name="org.apache.xerces" services="import"/>
    <module name="org.apache.xalan" services="import"/>

    <!-- extra dependencies for MyFaces -->
    <module name="org.apache.commons.collections"/>
    <module name="org.apache.commons.codec"/>
    <module name="org.apache.commons.beanutils"/>
    <module name="org.apache.commons.digester"/>

    <!-- extra dependencies for MyFaces 1.1
    <module name="org.apache.commons.logging"/>
    <module name="org.apache.commons.el"/>
    <module name="org.apache.commons.lang"/> -->
    <module name="javax.xml.rpc.api"/>
    <module name="javax.rmi.api"/>
    <module name="org.omg.api"/>
  </dependencies>

  <resources>
    <resource-root path="IMPL_NAME-impl-VERSION.jar"/>
  </resources>
</module>
```

示例：**Mojarra Jakarta Server Faces API JAR module.xml**



注意

确保对模板中的以下可替换变量使用适当的值：

- ***IMPL_NAME***
- ***VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="javax.faces.api:IMPL_NAME-VERSION">
  <dependencies>
    <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
    <module name="javax.enterprise.api" export="true"/>
    <module name="javax.servlet.api" export="true"/>
    <module name="javax.servlet.jsp.api" export="true"/>
    <module name="javax.servlet.jstl.api" export="true"/>
    <module name="javax.validation.api" export="true"/>
    <module name="org.glassfish.javax.el" export="true"/>
    <module name="javax.api"/>
    <module name="javax.websocket.api"/>
  </dependencies>

  <resources>
    <resource-root path="jsf-api-VERSION.jar"/>
  </resources>
</module>
```

*示例：**MyFaces Jakarta Server Faces API JAR module.xml***

注意

确保对模板中的以下可替换变量使用适当的值：

- ***IMPL_NAME***
- ***VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="javax.faces.api:IMPL_NAME-VERSION">
  <dependencies>
    <module name="javax.enterprise.api" export="true"/>
    <module name="javax.servlet.api" export="true"/>
    <module name="javax.servlet.jsp.api" export="true"/>
    <module name="javax.servlet.jstl.api" export="true"/>
```

```
        <module name="javax.validation.api" export="true"/>
        <module name="org.glassfish.javax.el" export="true"/>
        <module name="javax.api"/>

        <!-- extra dependencies for MyFaces 1.1
        <module name="org.apache.commons.logging"/>
        <module name="org.apache.commons.el"/>
        <module name="org.apache.commons.lang"/> -->
    </dependencies>

    <resources>
        <resource-root path="myfaces-api-VERSION.jar"/>
    </resources>
</module>
```

*示例：**Mojarra Jakarta Server Faces Injection JAR module.xml***

注意

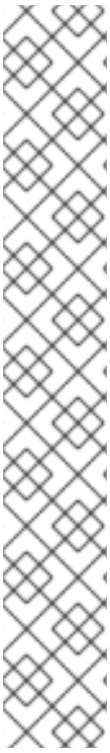
确保对模板中的以下可替换变量使用适当的值：

- ***IMPL_NAME***
- ***VERSION***
- ***INJECTION_VERSION***
- ***WELD_VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="org.jboss.as.jsf-injection:IMPL_NAME-VERSION">
    <properties>
        <property name="jboss.api" value="private"/>
    </properties>

    <resources>
        <resource-root path="wildfly-jsf-injection-INJECTION_VERSION.jar"/>
        <resource-root path="weld-core-jsf-WELD_VERSION.jar"/>
    </resources>

    <dependencies>
        <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
        <module name="java.naming"/>
        <module name="java.desktop"/>
        <module name="org.jboss.as.jsf"/>
        <module name="org.jboss.as.web-common"/>
```

```
        <module name="javax.servlet.api"/>
        <module name="org.jboss.as.ee"/>
        <module name="org.jboss.as.jsf"/>
        <module name="javax.enterprise.api"/>
        <module name="org.jboss.logging"/>
        <module name="org.jboss.weld.core"/>
        <module name="org.jboss.weld.api"/>

        <module name="javax.faces.api:IMPL_NAME-VERSION"/>
    </dependencies>
</module>
```

*示例：**MyFaces Jakarta Server Faces Injection JAR module.xml***



注意

确保对模板中的以下可替换变量使用适当的值：

- ***IMPL_NAME***
- ***VERSION***
- ***INJECTION_VERSION***
- ***WELD_VERSION***

```
<module xmlns="urn:jboss:module:1.8" name="org.jboss.as.jsf-injection:IMPL_NAME-VERSION">
    <properties>
        <property name="jboss.api" value="private"/>
    </properties>

    <resources>
        <resource-root path="wildfly-jsf-injection-INJECTION_VERSION.jar"/>
        <resource-root path="weld-jsf-WELD_VERSION.jar"/>
    </resources>

    <dependencies>
        <module name="com.sun.jsf-impl:IMPL_NAME-VERSION"/>
        <module name="javax.api"/>
        <module name="org.jboss.as.web-common"/>
        <module name="javax.servlet.api"/>
        <module name="org.jboss.as.jsf"/>
        <module name="org.jboss.as.ee"/>
        <module name="org.jboss.as.jsf"/>
        <module name="javax.enterprise.api"/>
```

```
        <module name="org.jboss.logging"/>
        <module name="org.jboss.weld.core"/>
        <module name="org.jboss.weld.api"/>
        <module name="org.wildfly.security.elytron"/>

        <module name="javax.faces.api:IMPL_NAME-VERSION"/>
    </dependencies>
</module>
```

*示例：**MyFaces commons-digester JAR module.xml***

注意

*务必为模板中的 **VERSION** 可替换变量使用适当的值。*

```
<module xmlns="urn:jboss:module:1.5" name="org.apache.commons.digester">
    <properties>
        <property name="jboss.api" value="private"/>
    </properties>

    <resources>
        <resource-root path="commons-digester-VERSION.jar"/>
    </resources>

    <dependencies>
        <module name="javax.api"/>
        <module name="org.apache.commons.collections"/>
        <module name="org.apache.commons.logging"/>
        <module name="org.apache.commons.beanutils"/>
    </dependencies>
</module>
```

A.33. JGROUPS SUBSYSTEM ATTRIBUTES

*下表中列出了 **jgroups** 子系统的各种元素的属性。*

- *主要属性*
- *频道属性*
- *堆栈属性*

注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-jgroups_5_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.113.** 主 **jgroups** 属性*

属性	默认	描述
default-channel	EE	默认 JGroups 频道。
default-stack		默认 JGroups 协议堆栈.

频道属性

频道元素具有以下结构：

- *频道*
  - ***fork***
    - ***protocol***
  - ***protocol***

频道属性

*表 **A.114.** 频道属性*

属性	默认	描述
cluster		JGroups 频道的集群名称。如果未定义，则使用频道的名称。
module	org.wildfly.clustering.server	要从中加载频道服务的模块。
堆栈		JGroups 通道的协议堆栈。
statistics-enabled	false	是否启用统计数据.

属性	默认	描述
stats-enabled	false	是否启用统计数据.*弃用：改为使用启用* ***statistics-enabled*** *属性。*

堆栈属性

stack *元素具有以下结构：*

- *堆栈*
  - ***protocol***
  - ***relay***
    - ***remote-site***
  - ***transport***
    - ***thread-pool***

堆栈属性

表 ***A.115.*** *堆栈属性*

属性	默认	描述
statistics-enabled	false	指明堆栈中的所有协议是否都会收集统计信息。

协议属性

有关常用协议的列表， *请参阅* ***JGroups*** *协议* *部分。*

表 ***A.116.*** *协议属性*

属性	默认	描述
module	org.jgroups	用于解析协议类型的模块。

属性	默认	描述
属性		此协议的属性.
statistics-enabled	false	指明此协议是否收集统计数据，覆盖堆栈配置。

转发属性

***表 A.117.** 转发属性*

属性	默认	描述
module	org.jgroups	用于解析协议类型的模块。
属性		此协议的属性.
站点		本地站点的名称。
statistics-enabled	false	指明此协议是否收集统计数据，覆盖堆栈配置。

远程站点属性

***表 A.118.** 远程站点属性*

属性	默认	描述
channel		用于与此远程站点通信的桥接频道的名称。
cluster		到这个远程站点的桥接频道的集群名称。*弃用：使用明确定义的频道*。
堆栈		从中创建网桥到此远程站点的堆栈。*弃用：使用明确定义的频道*。

传输属性

***表 A.119.** 传输属性*

属性	默认	描述
default-executor		用于处理传入消息的线程池 executor。*弃用：改为配置预定义的默认线程池。*
diagnostics-socket-binding		此协议层的诊断套接字绑定规格，用于指定用于通信的 IP 接口和端口。

属性	默认	描述
机器		此节点的机器或主机标识符。由 Infinispan 的拓扑感知型哈希使用。
module	org.jgroups	解析协议类型的模块。
OOB-executor		用于处理传入的带外消息的线程池执行器。*弃用：改为配置预定义的* ***oob*** *线程池。*
属性		此传输的属性.
rack		此节点的机架，如服务器机架标识符。由 Infinispan 的拓扑感知型哈希使用。
共享	false	如果为 **true**，则底层传输由使用此堆栈的所有渠道共享。*deprecated：改为配置频道的分叉。*
站点		此节点的站点（如数据中心）标识符。由 Infinispan 的拓扑感知型哈希使用。
socket-binding		此协议层的套接字绑定规格，用于指定用于通信的 IP 接口和端口。
statistics-enabled	false	指明此协议是否收集统计数据，覆盖堆栈配置。
thread-factory		用于处理异步传输特定任务的线程工厂。*弃用：改为配置预定义的内部线程池。*
timer-executor		用于处理协议相关计时任务的线程池 executor。*弃用：改为配置预定义的定时器线程池。*

thread-pool 属性

*表 **A.120. thread-pool 属性***

属性	默认	描述
keepalive-time	5000L	池线程闲置时应保持运行中的毫秒数。如果未指定，则线程将运行直到 executor 关闭为止。
max-threads	4	最大线程池大小。
Min-threads	2	内核线程池大小，小于 **max-threads**。如果未定义，则核心线程池大小与 **max-threads** 相同。
queue-length	500	队列长度。

A.34. JGROUPS PROTOCOLS

协议	协议类型	描述
ASYM_ENCRYPT	Encryption	使用机密密钥存储在群集上的协调器中，用于加密群集成员之间的消息。
AUTH	身份验证	为群集成员提供身份验证层。
azure.AZURE_PING	发现	支持使用 Microsoft Azure 的 blob 存储进行节点发现。
FD_ALL	故障检测	基于简单的心跳协议提供故障检测.
FD_SOCK	故障检测	根据群集成员之间创建的 TCP 套接字环提供故障检测。
JDBC_PING	发现	通过使用成员在其中写入其地址的共享数据库发现群集成员。
MERGE3	merge	在集群分割时将子集群合并在一起。
MFC	流控制	在发送方和所有群集成员之间提供多播流控制。
MPING	发现	发现具有 IP 多播的群集成员。
pbcast.GMS	Group Membership	处理组成员资格，包括加入群集的新成员、保留现有成员的请求，以及针对已崩溃成员的 **SUSPECT** 消息。
pbcast.NAKACK2	消息传输	确保消息可靠性和顺序，保证一个发件人发送的所有消息按照发送者发送的顺序收到。
Pbcast.STABLE	消息稳定性	删除所有成员已看到的消息。
PING	发现	首次发现成员，并支持动态发现群集成员。
SASL	身份验证	使用 SASL 机制为群集成员提供身份验证层。
SYM_ENCRYPT	Encryption	使用共享密钥存储加密群集成员之间的消息。
S3_PING	发现	使用 Amazon S3 发现初始成员.
TCPGOSSIP	发现	使用外部 Gossip 路由器发现群集成员。
TCPPING	发现	包含用于组成群集的群集成员地址的静态列表。
UFC	流控制	提供发送方和所有群集成员之间的单播流控制

协议	协议类型	描述
UNICAST3	消息传输	确保单播消息的消息可靠性和顺序，保证一个发件人发送的所有消息将按照发送的顺序接收。
VERIFY_SUSPECT	故障检测	验证可疑成员是否因 ping 成员在被驱除前的最后一个时间而结束。

通用协议属性

所有协议都有权访问以下属性：

*表 **A.121.** 协议属性*

属性	默认	描述
module	org.jgroups	用于解析协议类型的模块。
属性		此协议的属性.
statistics-enabled	false	是否启用统计数据.

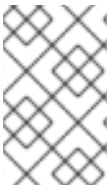
身份验证协议

*身份验证协议用于执行身份验证，主要负责确保只有经过身份验证的成员才能加入群集。这些协议位于 **GMS** 协议之下，因此它们可以侦听加入群集的请求。*

- ***AUTH***
- ***SASL***

***AUTH** 属性*

*虽然 **AUTH** 协议不包含额外的属性，但它必须将令牌定义为子元素。*



注意

*在定义此协议时，使用 **auth-protocol** 元素而不是 **protocol** 元素。*

令牌类型

*在使用 **Elytron** 实现安全性时，建议使用以下身份验证令牌之一：这些身份验证令牌专门设计为用于*

Elytron，可能不用于传统安全配置。

表 A.122. Elytron 令牌类型

令牌	描述
cipher-token	转换共享 secret 的身份验证令牌。RSA 是用于转换的默认算法。
digest-token	转换共享 secret 的身份验证令牌。SHA-256 是用于转换的默认算法。
plain-token	身份验证令牌，没有额外的转换到共享 secret。

以下身份验证令牌继承自 JGroups，并有资格在任何需要身份验证的配置中使用。

表 A.123. JGroups Token Types

令牌	描述
MD5Token	一个身份验证令牌，其中共享 secret 使用 MD5 或 SHA 哈希进行加密。MD5 是用于加密的默认算法。
SimpleToken	身份验证令牌，没有额外的转换到共享 secret。此令牌不区分大小写，在决定字符串是否匹配时不考虑大小写。
X509Token	一个身份验证令牌，其中共享 secret 使用 X509 证书加密。

SASL 属性

表 A.124. SASL 属性

属性	默认	描述
client_callback_handler		当节点充当客户端时要使用的 **CallbackHandler** 类名称。
client_name		节点充当客户端时要使用的名称。如果使用 JAAS 登录模块，此名称也将用于获取该主题。
client_password		节点充当客户端时使用的密码。如果使用 JAAS 登录模块，此密码也将用于获取该主题。
login_module_name		用作创建 SASL 客户端和服务器的 JAAS 登录模块的名称。此属性仅需要某些 **机制值**，如 GSSAPI。

属性	默认	描述
Mech		SASL 身份验证机制的名称。这个名称可以是本地 SASL 提供商支持的任何机制，而 JDK 默认提供 **CRAM-MD5**、**DIGEST-MD5**、**GSSAPI** 和 **NTLM**。
sasl_props		已定义机制 **的属性**.
server_callback_handler		当节点充当服务器时要使用的 **CallbackHandler** 类名称。
server_name		完全限定服务器名称。
timeout	5000	等待响应质询的毫秒数。

发现协议

以下协议用于查找群集的初始成员身份，然后可用于确定当前的协调员。如下为发现协议的列表。

- ***AZURE_PING***
- ***JDBC_PING***
- ***MPING***
- ***PING***
- ***S3_PING***
- ***TCPGOSSIP***
- ***TCPPING***

AZURE_PING Attributes

表 A.125. AZURE_PING Attributes

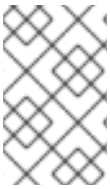
*表 **A.125. AZURE_PING Attributes***

属性	默认	描述
Container		用于 **PING** 数据的 blob 容器的名称。这必须是有效的 DNS 名称。
storage_access_key		存储帐户的机密访问密钥。
storage_account_name		包含 blob 容器的 Microsoft Azure 存储帐户的名称。

JDBC_PING Attributes

*表 **A.126. JDBC_PING Attributes***

属性	默认	描述
data-source		要使用的数据源引用，而不使用连接和 JNDI 查找属性。



注意

*在定义 **JDBC_PING** 协议时，使用 **jdbc-**协议 元素而不是 协议 元素。*

S3_PING Attributes

*表 **A.127. S3_PING Attributes***

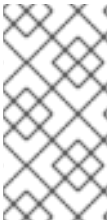
属性	默认	描述
access_key		用于访问 S3 存储桶的 Amazon S3 访问密钥。
主机	s3.amazonaws.com	S3 Web 服务的目的地。
位置		要使用的 Amazon S3 存储桶的名称。bucket 必须存在，并且使用唯一名称。
pre_signed_delete_url		用于 DELETE 操作的预签名 URL。

属性	默认	描述
port	• 443（如果 **use_ssl** 为 **true**） • 80，如果 **use_ssl** 为 **false**.	Web 服务正在侦听的端口。
pre_signed_put_url		用于 PUT 操作的预签名 URL。
prefix		如果设置了，则将存储桶名称定义为 ***PREFIX-LOCATION***。如果设置，并且存储桶在指定的 ***PREFIX-LOCATION*** 中不存在，则存储桶名称将变为 ***PREFIX***，后跟随机 UUID。
secret_access_key		用于访问 S3 存储桶的 Amazon S3 机密访问密钥。
use_ssl	true	确定在联系主机和端口组合时是否使用 SSL。

***TCPGOSSIP** 属性*

*表 **A.128. TCPGOSSIP** 属性*

属性	默认	描述
socket-binding		此协议层的套接字绑定规格。*弃用：改为使用* ***socket-bindings***。
socket-bindings		此协议的出站套接字绑定。



注意

*定义 **TCPGOSSIP** 协议时，将使用 **socket-discovery-protocol** 元素而不是 协议 元素。*

TCPPING Attributes

表 A.129. TCPPING Attributes

属性	默认	描述
socket-binding		此协议层的套接字绑定规格。*弃用：改为使用* ***socket-bindings****。*
socket-bindings		此协议的出站套接字绑定。

注意

在定义 ***TCPPING*** *协议时，使用* ***socket-discovery-protocol*** *元素而不是 协议 元素。*

加密协议

下列协议用于保护通信堆栈。加密基于群集所有成员拥有的共享 ***secret*** *密钥。当使用* ***SYM_ENCRYPT*** *时，这个密钥可以从共享密钥存储获取，或使用* ***ASYM_ENCRYPT*** *时从公钥/私钥交换获取。在定义以下任何协议时，会在生成的* ***XML*** *中创建* ***encrypt-protocol*** *元素：*

注意

如果使用 ***ASYM_ENCRYPT****，则同一堆栈必须定义* ***AUTH*** *协议。使用* ***SYM_ENCRYPT*** *时，* ***AUTH*** *协议是可选的。*

- ***ASYM_ENCRYPT***
- ***SYM_ENCRYPT***

ASYM_ENCRYPT Attributes

表 A.130. ASYM_ENCRYPT Attributes

属性	默认	描述
key-alias		来自指定的密钥存储的加密密钥别名。
key-credential-reference		从密钥存储获取加密密钥所需的凭据。
key-store		对包含加密密钥的密钥存储的引用。

SYM_ENCRYPT Attributes

表 A.131. SYM_ENCRYPT Attributes

属性	默认	描述
key-alias		来自指定的密钥存储的加密密钥别名。
key-credential-reference		从密钥存储获取加密密钥所需的凭据。
key-store		对包含加密密钥的密钥存储的引用。

故障检测协议

以下协议用于探测群集成员，以确定它们是否仍然处于活动状态。除了通用属性外，这些协议没有任何额外属性。

- **FD_ALL**
- **FD_SOCK**
- **VERIFY_SUSPECT**

流控制协议

以下协议负责流控制，或者调整消息发送器到最慢接收者的速率。如果发送方持续以快于接收方的速度发送消息，接收方将排队或丢弃消息，从而导致重新传输。除了通用属性外，这些协议没有任何额外属性。

- **MFC -** 多播流控制
- **UFC -** 单播流控制

组成员资格协议

pbcast.GMS 协议负责加入群集的新成员、现有成员离开群集，以及怀疑已崩溃的成员。除了通用属性外，此协议没有任何额外属性。

合并协议

如果集群被分割，则 **MERGE3** 协议负责将子集群合并在一起。虽然此协议负责将群集成员重新合并在

一起，但这不会合并群集的状态。应用负责处理回调以合并状态。除了通用属性外，此协议没有任何额外属性。

消息稳定性

pbcast.STABLE 协议负责收集群集所有成员已看到的消息。此协议启动包含给定成员的消息编号的稳定消息，称为摘要。旦群集的所有成员收到其他人的摘要，则消息可能会从重新传输表中移除。除了通用属性外，此协议没有任何额外属性。

可靠的消息传输

以下协议为发送到集群中所有节点的消息提供可靠的消息发送和 **FIFO** 属性。可靠发送意味着发送方发送的任何邮件都不会丢失，因为所有消息都是编号的，如果未收到序列号，则会发送重新传输请求。除了通用属性外，这些协议没有任何额外属性。

- **pbcast.NAKACK2**
- **pbcast.UNICAST3**

弃用的协议

以下协议已弃用，并被仅包含类名称的协议所取代。例如，协议名称应为 **ASY M_ENCRYPT**，而不是指定 **org.jgroups.protocols.ASY M_ENCRYPT**。

- **org.jgroups.protocols.ASYM_ENCRYPT**
- **org.jgroups.protocols.AUTH**
- **org.jgroups.protocols.JDBC_PING**
- **org.jgroups.protocols.SYM_ENCRYPT**
- **org.jgroups.protocols.TCPGOSSIP**
- **org.jgroups.protocols.TCPPING**

A.35. MICROPROFILE CONFIG SMALLRYE SUBSYSTEM ATTRIBUTES

注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-microprofile-config-smallrye_1_0.xsd** 的架构定义文件，以查看它们在 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.132. MicroProfile Config SmallRye Subsystem: config-source Attributes***

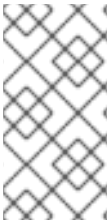
属性	描述
class	要加载的 **ConfigSource** 类。**ConfigSource** 为配置值提供源。
dir	要扫描的目录。此目录包含属性文件，其中每个文件名是 属性的名称，文件内容是其值。
ordinal	**ConfigSource** 的序数值或优先级。带有更高序值的 ConfigSource 提供的值会覆盖由具有较低序值的 ConfigSources 提供的值。
属性	直接存储在 **ConfigSource** 中的属性。

*表 **A.133. MicroProfile Config SmallRye Subsystem: config-source-provider Attributes***

属性	描述
class	要加载的 **ConfigSourceProvider** 类。**ConfigSourceProvider** 注册多个 **ConfigSource** 实例的实现。

A.36. APACHE HTTP 服务器 MOD_CLUSTER 指令

***mod_cluster** 连接器是基于 **Apache HTTP Server** 的负载平衡器。它使用通信通道将来自 **Apache HTTP** 服务器的请求转发到一组应用服务器节点。以下指令可以设置为配置 **mod_cluster**：*



注意

*不需要使用**ProxyPass** 指令，因为 **mod_cluster** 会自动配置必须转发到 **Apache HTTP** 服务器的 **URL**。*

*表 **A.134. mod_cluster Directives***

指令	描述	值
CreateBalancers	定义如何在 Apache HTTP 服务器 VirtualHosts 中创建均衡器。这允许指令，如： **ProxyPass /balancer://mycluster1/**。	• **0** ：创建 Apache HTTP 服务器中定义的所有虚拟主机 • **1** ：不要创建平衡器（至少需要一个 *ProxyPass* 或 *ProxyMatch* 来定义平衡器名称） • **2** ：仅创建主服务器 *（默认）*
UseAlias	检查别名是否与服务器名称对应。	• **0** ：忽略别名 *（默认）* • **1** ：检查别名
LBstatusRecalTime	负载平衡逻辑重新计算节点状态的时间间隔（以秒为单位）。	默认：5 秒
WaitBeforeRemove	httpd 忘记移除节点前的时间（以秒为单位）。	默认：10 秒
ProxyPassMatch/ProxyPass	ProxyPassMatch 和 ProxyPass 是 *mod_proxy 指令，在使用时，而不是后端* URL，可防止路径中的 reverse-proxy。这用于允许 Apache HTTP 服务器提供静态内容。例如： **ProxyPassMatch ^(/.*\.gif)$!**此示例允许 Apache HTTP 服务器直接提供 .gif 文件。	

注意

*由于 **JBoss EAP 7** 中会话的性能优化，不支持配置热机节点。*

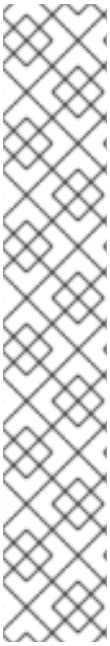
mod_manager

***mod_manager** 指令的上下文是 **VirtualHost**，除非另有说明。服务器配置上下文表示该指令必须在 **VirtualHost** 配置之外。如果没有，则会显示错误消息，并且 **Apache HTTP** 服务器不会启动。*

*表 **A.135. mod_manager Directives***

指令	描述	值
EnableMCPMReceive	允许 VirtualHost 从节点接收 *MCPM*。在 Apache HTTP 服务器配置中包含 *EnableMCPMReceive*，以允许 mod_cluster 正常工作。将它保存到配置广告的 VirtualHost 中。	
MemManagerFile	*mod_manager* 用于存储配置的名称的基本名称，为共享内存或锁定文件生成密钥。这必须是绝对路径名称；在需要时创建目录。建议将这些文件放在本地驱动器中，而不是 NFS 共享中。上下文：服务器配置	**$server_root/logs/**
Maxcontext	mod_cluster 支持的最大上下文数。上下文：服务器配置	默认： **100**
Maxnode	mod_cluster 支持的最大节点数。上下文：服务器配置	默认： **20**
Maxhost	mod_cluster 支持的最大主机数量或别名。它还包括最大均衡器数量。上下文：服务器配置	默认： **20**
Maxsessionid	存储的活动 *sessionid* 的数量，以在 *mod_cluster-manager* 处理程序中提供活动会话的数量。当 mod_cluster 在 5 分钟内没有从会话接收任何信息时，会话不活跃。上下文：服务器配置.此字段仅用于演示和调试目的。	**0** ：逻辑没有激活。
MaxMCMPMaxMessSize	来自其他 Max 指令的 MCMP 消息的最大大小	从其他 Max 指令计算.Min: **1024**
ManagerBalancerName	JBoss EAP 实例不提供负载平衡器名称时要使用的负载平衡器名称。	*mycluster*
PersistSlots	告诉 mod_slotmem 在文件中持久保留节点、别名和上下文。上下文：服务器配置	å...[3]
CheckNonce	使用 *mod_cluster-manager* 处理程序时切换检查 *nonce*。	On/off Default: on - Nonce选中
AllowDisplay	在 *mod_cluster-manager* 主页上切换其他显示。	on/off Default: off - 仅显示 version

指令	描述	值
AllowCmd	允许使用 *mod_cluster-manager* URL 的命令.	默认： on - 命令允许
ReduceDisplay	减少主 *mod_cluster-manager* 页面上显示的信息，以便在页面中显示更多节点。	On/off Default: off - 显示完整信息
SetHandler mod_cluster-manager	显示 mod_cluster 从群集中看到的节点的信息。此信息包括通用信息，并额外计算活动会话数。 <Location /mod_cluster-manager> SetHandler mod_cluster-manager Require ip 127.0.0.1 </Location>	开/off 默认： off



注意

*访问 **httpd.conf** 中定义的位置时：*

- *传输：与发送到后端服务器的 **POST** 数据相对应。*
- *连接： 与请求 **mod_cluster** 状态页面时处理的请求数相对应。*
- ***Num_sessions**：与会话 **mod_cluster** 报告数量相符（过去 **5** 分钟内有请求）。**Maxsessionid** 为零时不会出现此字段，且仅用于演示和调试目的。*

A.37. MODCLUSTER SUBSYSTEM ATTRIBUTES

***modcluster** 子系统具有以下结构：*

- *proxy*
  - *表 A.137 “load-provider=dynamic 配置选项”*
    - *custom-load-metric*
    - *load-metric*
  - ***load-provider=simple***
  - *ssl*

***load-provider=dynamic** 资源允许您配置因素，如 **CPU**、会话、堆、内存和权重，以确定负载平衡行为。*

***load-provider=simple** 资源仅允许将静态常量设置为 **factor** 属性。当用户不需要动态或复杂的规则来平衡传入的 **HTTP** 请求时，这有助于帮助。*

注意

*这些表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/jboss-as-mod-cluster_3_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。*

*表 **A.136.** 代理配置选项*

属性	默认	描述
advertise	true	是否启用基于多播的广告机制.
advertise-security-key		它是 httpd 实例和 JBoss EAP 服务器之间共享的机密，侦听来自 httpd 实例的广播。
advertise-socket		反向代理中要注册的负载均衡器的名称。

属性	默认	描述
auto-enable-contexts	true	如果设置为 **false**，则上下文会将反向代理注册为禁用状态。您可以使用 **enable-context** 操作或 mod_cluster_manager 控制台来启用上下文。
balancer		反向代理中要注册的负载均衡器的名称。如果没有设置，则会使用 **ManagerBalancerName** 指令在 Apache HTTP 服务器端配置该值，该指令默认为 **mycluster**。
连接器		mod_cluster 反向代理将连接的 Undertow 侦听器的名称。
excluded-contexts		要排除与反向代理注册之外的上下文列表。如果未指明主机，则假定主机为 **localhost**。**ROOT** 表示 Web 应用的根上下文。
flush-packets	false	是否启用数据包刷新到 Web 服务器.
flush-wait	-1	在 httpd 中清空数据包前等待的时间。Max 值为 **2.147,483,647**。
监听程序		将注册到反向代理的 Undertow 侦听器的名称。
load-balancing-group		如果设置，请求将发送到负载平衡器上的指定负载平衡组。
max-attempts	1	反向代理在放弃前会尝试向 worker 发送给定请求的次数。
node-timeout	-1	代理到 worker 连接的超时时间（以秒为单位）。这是 mod_cluster 在返回错误前等待后端响应的时间。如果未定义 **node-timeout** 属性，则使用 httpd **ProxyTimeout** 指令。**如果未定义**，则使用 httpd **Timeout** 指令，默认为 300 秒。
ping	10	等待 ping 回答的时间（以秒为单位）。
代理		要注册到 **socket-binding-group** 中的 outbound-**socket-binding** 定义的 mod_cluster 代理列表。
proxy-list		代理列表.格式为 ***HOST_NAME*:*PORT***，用逗号分隔。*弃用了 代理。*
proxy-url	/	MCMP 请求的基本 URL。

属性	默认	描述
session-draining-strategy	DEFAULT	取消部署 Web 应用程序期间使用的会话排空策略。有效值为 **DEFAULT**、**ALWAYS** 或 **NEVER**。 **DEFAULT** 只有 Web 应用不可分发时，才可排空会话，然后再取消部署。 **ALWAYS** 在 Web 应用取消部署之前始终排空会话，即使是对于可分布式 Web 应用也是如此。 **NEVER** 在 Web 应用取消部署前，请勿排空会话。
load-provider=simple		如果没有动态负载提供程序，要使用的负载供应商。它为每个群集成员分配 **1** 的负载因子，并在不应用负载平衡算法的情况下平均分配工作。
SMAX	-1	httpd 中的软最大空闲连接数.
socket-timeout	20	在超时前等待 httpd 代理到 MCMP 命令响应的秒数，并将代理标记为 error。
ssl-context		引用 mod_cluster 使用的 **SSLContext**.
status-interval	10	从应用服务器发送 STATUS 消息到反向代理的秒数。允许的值介于 **1** 和 **2,147,483,647** 之间。
sticky-session	true	如果可能，后续对给定会话的请求是否应该路由到同一节点。
sticky-session-force	false	如果负载均衡器无法将请求路由到其卡住的节点，反向代理是否应该返回错误。如果禁用粘性会话，则忽略此设置。
sticky-session-remove	false	删除有关故障转移的会话信息。
stop-context-timeout	10	等待上下文处理待处理请求、分布式上下文或销毁非分布式上下文的最大时间（以秒为单位）。
ttl	-1	闲置连接超过 smax 的时间（以秒为单位）。允许的值介于 **-1** 和 **2,147,483,647** 之间。
worker-timeout	-1	在 httpd 中等待超时，以便可用的工作程序处理请求。允许的值介于 **-1** 和 **2,147,483,647** 之间。

*表 **A.137. load-provider=dynamic** 配置选项*

属性	默认	描述
衰减	2	衰减.
history	9	历史.
初始负载	0	节点报告的初始负载。有效范围为 **0-100**，**0** 表示最大负载。 此属性有助于逐渐增加新加入节点的负载值，以避免在加入集群时出现过载。 您可以通过将值设置为 **-1** 来禁用此行为。禁用后，节点将报告负载值 **100**，表示加入集群时它没有负载。

*表 **A.138. custom-load-metric Attribute** 选项*

属性	默认	描述
capacity	1.0	指标的容量。
class		自定义指标的类名称。
属性		指标的属性。
weight	1	指标的权重。

*表 **A.139.** 负载指标属性选项*

属性	默认	描述
capacity	1.0	指标的容量。
属性		指标的属性。
type		指标的类型。有效值包括 **cpu**、**mem**、**堆**、**sessions**、**ce-traffic**、**send-traffic**、**request** 或 **busyness**。
weight	1	指标的权重。

*表 **A.140. SSL** 属性选项*

属性	默认	描述
ca-certificate-file		证书颁发机构.
ca-revocation-url		证书颁发机构撤销列表.
certificate-key-file	${user.home}/.keystore	证书的密钥文件。
cipher-suite		允许的密码套件.
key-alias		密钥别名.
password	changeit	密码.
协议	TLS	已启用的 SSL 协议.

A.38. MOD_JK WORKER PROPERTIES

***Worker .properties** 文件定义 **mod_jk** 将客户端请求传递给的 **worker** 的行为。**Worker .properties** 文件定义不同应用服务器所处的位置，以及跨它们平衡工作负载的方式。*

*属性的一般结构为 **worker**。**WORKER_NAME.DIRECTIVE.WORKER_NAME** 是一个唯一名称，必须与 **JBoss EAP undertow** 子系统中配置的 **instance-id** 匹配。**DIRECTIVE** 是应用到 **worker** 的设置。*

Apache mod_jk Load Balancers 的配置参考

模板指定默认的每个负载均衡器设置。您可以覆盖负载均衡器设置本身中的模板。

*表 **A.141.** 全局属性*

属性	描述
worker.list	*mod_jk* 将使用的以逗号分隔的 worker 名称列表。

*表 **A.142.** 强制指令*

属性	描述
type	worker 的类型。默认类型是 **ajp13**。其他可能的值有 **ajp14**、**lb**、**status**。有关这些指令的更多信息，请参阅 Apache Tomcat 连接器参考，网址为 https://tomcat.apache.org/connectors-doc/reference/workers.html。

*表 **A.143.** 负载平衡指令*

属性	描述
balance_workers	指定负载均衡器必须管理的 worker 节点。对于同一个负载均衡器，您可以多次使用该指令。它由以逗号分隔的 worker 节点名称列表组成。
sticky_session	指定来自同一会话的请求是否始终路由到同一 worker。默认值为 **1**，表示已启用粘性会话。要禁用粘性会话，请将它设置为 **0**。除非所有请求真正无状态，否则通常应启用粘性会话。

*表 **A.144.** 连接指令*

属性	描述
主机	后端服务器的主机名或 IP 地址。后端服务器必须支持 **ajp** 协议堆栈。默认值为 **localhost**。
port	侦听定义的协议请求的后端服务器实例的端口号。默认值为 **8009**，这是 AJP13 worker 的默认侦听端口。AJP14 worker 的默认值为 **8011**。
ping_mode	为网络状态探测连接的条件。该探测使用空 AJP13 数据包进行 CPing，并需要一个 CPong 来响应。使用指令标志的组合来指定条件。标志不用逗号或任何空格分隔。ping_mode 可以是 C、P、I 和 A 的任意组合。 • c - 连接.在连接到服务器后一次性探测连接。使用 connect_timeout 的值指定超时。否则，使用 ping_timeout 的值。 • P - 前置.在将每个请求发送到服务器之前，探测连接。使用 prepost_timeout 指令指定超时。否则，使用 ping_timeout 的值。 • i - 间隔.以 connection_ping_interval 指定的间隔（如果存在）探测连接。否则，使用 ping_timeout 的值。 • A - 所有.CPI 快捷方式，用于指定所有连接探测都被使用。
ping_timeout, connect_timeout, prepost_timeout, connection_ping_interval	上方连接探测设置的超时值。该值以毫秒为单位指定，**ping_timeout** 的默认值为 **10000**。
lbfactor	指定单个后端服务器实例的负载平衡因素。这可用于为更强大的服务器提供更多工作负载。要为 worker 提供 3 次默认负载，请将其设置为 **3** ：worker**.my_worker.lbfactor=3**

*以下示例演示了在侦听端口 **8009** 的两个工作程序节点 **node1** 和 **node 2** 之间通过粘性会话进行负载平衡。*

*示例： **worker.properties** 文件*

```
# Define list of workers that will be used for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host= node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

Apache mod_jk 的更多配置详情已超出本文档的范围，请参阅 **Apache** 文档。

A.39. 安全管理器子系统属性

security-manager 子系统本身没有可配置的属性，但它有一个含有可配置属性的子资源：**deployment-permissions=default**。

注意

此表中的属性名称会在管理模型中出现时列出，例如使用管理 **CLI** 时。请参阅位于 **EAP_HOME/docs/schema/wildfly-security-manager_1_0.xsd** 的架构定义文件，以查看 **XML** 中出现的元素，因为管理模型可能会有所不同。

表 **A.145. Deployment-permissions** 配置选项

属性	描述
最大权限	可授予部署或 JAR 的最大权限集。
最低权限	授予部署或 JAR 的最小权限集。

A.40. 从 JBOSS 核心服务安装 OPENSSL

JBoss Core Services OpenSSL 文件可以从 ZIP 或 RPM 分发包安装。根据您选择的安装方法，按照以下步骤操作。

注意

在 Red Hat Enterprise Linux 8 中，支持标准系统 OpenSSL，因此不再需要从 JBoss Core Services 安装 OpenSSL。

使用 JBoss 核心服务 OpenSSL ZIP 文件发布

注意

ZIP 存档中 libs/ 目录的路径为 jbcs-openssl-VERSION/openssl/lib(64)，用于 Linux，jbcs-openssl-VERSION/openssl/bin for Windows。

1. 从与您操作系统和架构相关的软件下载页面下载 OpenSSL 软件包。

2. 将下载的 ZIP 文件提取到您的安装目录。

3. 通知 JBoss EAP 在哪里查找 OpenSSL 库。

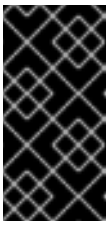
   您可以使用以下任一方法执行此操作：在以下每个命令中，请确保将 JBCS_OPENSSL_PATH 替换为 JBoss Core Services OpenSSL 库的路径，例如 /opt/rh/jbcs-httpd24/root/usr/lib64。

   - 您可以使用以下参数，将 OpenSSL 路径添加到 standalone.conf 或 domain.conf 配置文件中的 JAVA_OPTS 变量：

```
JAVA_OPTS="$JAVA_OPTS -Dorg.wildfly.openssl.path=JBCS_OPENSSL_PATH
```

- 您可以使用以下管理 **CLI** 命令定义指定 **OpenSSL** 路径的系统属性。

```
/system-property=org.wildfly.openssl.path:add(value=JBCS_OPENSSL_PATH)
```



重要

无论您使用哪一种方法，您必须执行服务器重启，使 **JAVA_OPTS** 值或 **system** 属性生效。服务器重新加载是不够的。

使用 **JBoss** 核心服务 **OpenSSL RPM** 分发

1. 确保该系统已注册到 **JBoss Core Services** 频道：

   a. 为您的操作系统版本和架构确定 **JBoss Core Services CDN** 存储库名称：

   - **RHEL 6** ：**jb-coreservices-1-for-rhel-6-server-rpms**
   - **RHEL 7** ：**jb-coreservices-1-for-rhel-7-server-rpms**

   b. 在系统中启用存储库：

   ```
   # subscription-manager repos --enable REPO_NAME
   ```

   c. 确保看到以下信息：

   ```
   Repository REPO_NAME is enabled for this system.
   ```

2. 从此频道安装 **OpenSSL:**

   ```
   # yum install jbcs-httpd24-openssl
   ```

3. 安装完成后，**JBCS OpenSSL** 库将在 **/opt/rh/jbcs-httpd24/root/usr/lib64** 中可用，或者仅 **/opt/rh/jbcs-httpd24/root/usr/lib on x86** 架构。

4. 通知 **JBoss EAP** 在哪里查找 **OpenSSL** 库。

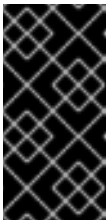
您可以使用以下任一方法执行此操作：在以下每个命令中，请确保将 **JBCS_OPENSSL_PATH** 替换为 **JBoss Core Services OpenSSL** 库的路径，例如 **/opt/rh/jbcs-httpd24/root/usr/lib64**。

- 您可以为服务配置文件中的 **eap7- standalone** 或 **eap7- domain** 设置更新 **WILDFLY_OPTS** 变量。

```
WILDFLY_OPTS="$WILDFLY_OPTS -
Dorg.wildfly.openssl.path=JBCS_OPENSSL_PATH"
```

- 您可以使用以下管理 **CLI** 命令定义指定 **OpenSSL** 路径的系统属性。

```
/system-property=org.wildfly.openssl.path:add(value=JBCS_OPENSSL_PATH)
```



重要

无论您使用哪一种方法，您必须执行服务器重启，使 **WILDFLY_OPTS** 值或系统属性生效。服务器重新加载是不够的。

A.41. 配置 JBOSS EAP 使用 OPENSSL

您可以通过多种方式将 **JBoss EAP** 配置为使用 **OpenSSL**：

- 您可以重新配置 **elytron** 子系统，使其默认在所有情形中都使用 **OpenSSL** 优先级。

注意

虽然 **elytron** 子系统中安装了 **OpenSSL**，但它不是默认的 **TLS** 提供程序。

```
/subsystem=elytron:write-attribute(name=initial-providers, value=combined-providers)
/subsystem=elytron:undefine-attribute(name=final-providers)

reload
```

- 在 **elytron** 子系统中，也可以在 **ssl-context** 资源上指定 **OpenSSL** 提供程序。这样，可以分案例选择 **OpenSSL** 协议，而不使用默认优先级。

  要创建 **ssl-context** 资源并在基于 **Elytron** 的 **SSL/TLS** 配置中使用 **OpenSSL** 库，请使用以下命令：

  ```
  /subsystem=elytron/server-ssl-context=httpsSSC:add(key-manager=localhost-manager,
  trust-manager=ca-manager, provider-name=openssl)

  reload
  ```

- 在旧 安全 子系统 **SSL/TLS** 配置中使用 **OpenSSL** 库：

  ```
  /core-service=management/security-realm=ApplicationRealm/server-identity=ssl:write-
  attribute(name=protocol,value=openssl.TLSv1.2)

  reload
  ```

  可以使用的不同 **OpenSSL** 协议：

  - **openssl.TLS**
  - **openssl.TLSv1**
  - **openssl.TLSv1.1**
  - **openssl.TLSv1.2**

JBoss EAP 将自动尝试搜索系统上的 **OpenSSL** 库并使用它们。您还可以在 **JBoss EAP** 启动过程中使用 **org.wildfly.openssl.path** 属性来指定自定义 **OpenSSL** 库位置。仅支持 **OpenSSL** 库版本 **1.0.2** 或 **JBoss Core Services** 提供的更高版本。

如果正确加载 **OpenSSL**，您将在 **JBoss EAP** 启动过程中在 **server.log** 中看到一条消息，类似于：

```
15:37:59,814 INFO [org.wildfly.openssl.SSL] (MSC service thread 1-7) WFOPENSSL0002 OpenSSL
Version OpenSSL 1.0.2k-fips 23 Mar 2017
```

A.42. 为 JAVA 8 提供的平台模块

- ***java.base*** *：此依赖项始终包含在提供的模块加载器中*
- ***java.compiler***
- ***java.datatransfer***
- ***java.desktop***
- ***java.instrument***
- ***java.jnlp***
- ***java.logging***
- ***java.management***
- ***java.management.rmi***
- ***java.naming***
- ***java.prefs***
- ***java.rmi***
- ***java.scripting***
- ***java.se*** *：此模块别名聚合了以下一组基本模块：*

- *java.compiler*
- *java.datatransfer*
- *java.desktop*
- *java.instrument*
- *java.logging*
- *java.management*
- *java.management.rmi*
- *java.naming*
- *java.prefs*
- *java.rmi*
- *java.scripting*
- *java.security.jgss*
- *java.security.sasl*
- *java.sql*

    - *java.sql.rowset*
    - *java.xml*
    - *java.xml.crypto*
- *java.security.jgss*
- *java.security.sasl*
- *java.smartcardio*
- *java.sql*
- *java.sql.rowset*
- *java.xml*
- *java.xml.crypto*
- *javafx.base*
- *javafx.controls*
- *javafx.fxml*
- *javafx.graphics*

- *javafx.media*
- *javafx.swing*
- *javafx.web*
- *jdk.accessibility*
- *jdk.attach*
- *jdk.compiler*
- *jdk.httpserver*
- *jdk.jartool*
- *jdk.javadoc*
- *jdk.jconsole*
- *jdk.jdi*
- *jdk.jfr*
- *jdk.jsobject*
- *jdk.management*

- *jdk.management.cmm*
- *jdk.management.jfr*
- *jdk.management.resource*
- *jdk.net*
- *jdk.plugin.dom*
- *jdk.scripting.nashorn*
- *jdk.sctp*
- *jdk.security.auth*
- *jdk.security.jgss*
- *jdk.unsupported*
- *jdk.xml.dom*

修订到 2022 年 2 月 18:20:46 +1000