xigemaAS

版本 9.1.0.1

xigemaAS 手册

http://www.vsettan.com.cn

目 录

1 欢迎使用 xigemaAS 快速使用手册

xigemaAS 快速使用手册基于 xigemaAS 文档，旨在帮助 xigemaAS 用户快速熟悉该产品。

1.1 xigemaAS 概述

xigemaAS 快速使用手册主要以具体用例的形式描述如何部署并访问应用，从而帮助用户快速理解并使用该产品。

本手册将包括以下用例的使用说明：

- jsp 和 servlet 组件相关用例
- 数据库连接池用例
- EJB 用例
- JMS 用例

👉 注：这些用例均位于安装目录中的 wlp\samples 目录。

关于xigemaAS 的详细介绍及使用信息，请参见 *xigemaAS 手册*。

1.2 开始使用 xigemaAS

本部分介绍通过相关准备工作后，如何使用 xigemaAS 安装包中提供的用例程序。

1.2.1 准备

在使用提供的用例程序之前，应执行相关准备工作。

这些准备工作包括：

1. 启动 xigemaAS 管理中心
2. 登录 xigemaAS 管理中心
3. 部署应用服务器
4. 如何获取应用的访问地址

启动 xigemaAS 管理中心

在已有 xigemaAS 安装的前提下，从命令行窗口启动 xigemaAS 管理中心。

按以下步骤启动 xigemaAS 管理中心。

- 进入 xigemaAS 的 wlp/bin 目录。

  类 UNIX 系统环境下执行以下命令启动 xigemaAS 管理中心：

```
./server run mc
```

或

```
./server start mc
```

Windows 系统环境下执行以下命令启动 xigemaAS 管理中心：

```
server run mc
```

或

```
server start mc
```

xigemaAS 管理中心启动成功，如下图所示：

若命令窗口显示"**服务器** `mc` **已准备就绪，可开始运行** `xigemaAS`"，则说明xigemaAS 管理中心已启动成功。

登录 xigemaAS 管理中心

启动成功后，可以从浏览器登录 xigemaAS 管理中心。

表 1: xigemaAS 管理中心支持以下浏览器

浏览器			
xigemaAS 管理中心支持的最低版本	Internet Explorer 10	Google Chrome 39.0	Firefox 38.0.5

若使用其它浏览器登录 xigemaAS 管理中心，请确保该浏览器支持 Websocket 协议。

- 访问 xigemaAS 管理中心，在浏览器地址栏输入 `http://hostname:port/mc` 进入 xigemaAS 管理中心登录页面。首次登录需要输入默认的用户名和密码 (admin/admin)。

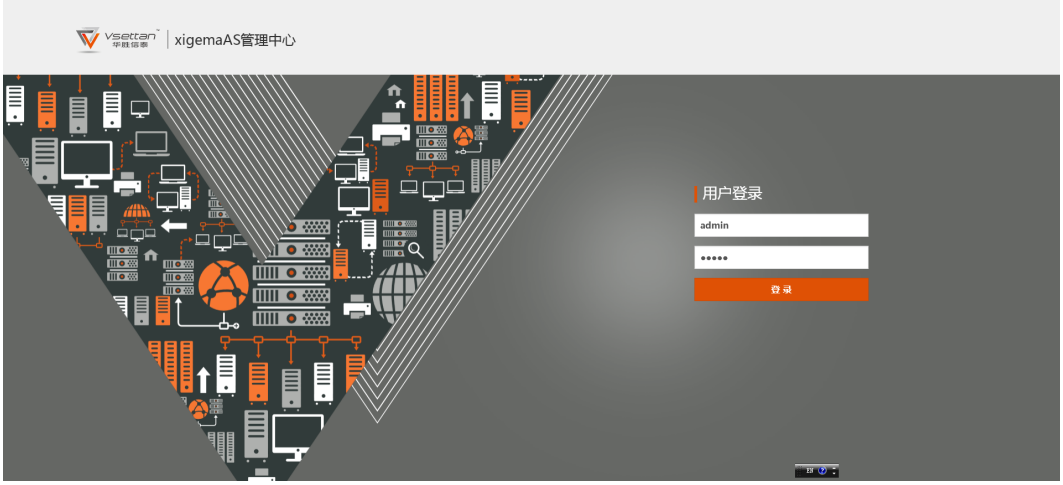


图 1: xigemaAS 管理中心登录页面

注：HTTP 端口号默认值为9060，HTTPS 端口号默认值为9043。

- 首次登录 xigemaAS 管理中心，登录后弹出 xigemaAS 管理中心所在主机 IP 地址配置引导页面。由 xigemaAS 管理中心管理的所有服务器通过在此配置的 IP 地址进行通信。配置完成后，进入 xigemaAS 管理中心。

图 2: xigemaAS 管理中心 IP 配置引导页面

注：进入 xigemaAS 管理中心后，若需要变更 xigemaAS 管理中心 IP 地址，可以单击**系统管理 > 参数配置**，进入参数配置页面变更 IP 地址。详细操作和注意事项请参见参数配置《xigemaAS 管理中心》中的“参数配置”章节 。

部署应用服务器

在注册节点、在节点上安装 xigemaAS、创建应用服务器之后才能在应用服务器上部署应用。可选择任意节点下的任意应用服务器部署应用。

在本手册中，选择默认节点 `defaultNode` 下的默认应用服务器 `defaultServer` 作为应用部署的目标服务器，因此，无需创建新的节点和服务器。可以直接前往该部分内容：如何获取用例的访问地址（见第 7 页）。

然而，如果需要创建新的节点和服务器，请参考以下内容：

1. 注册节点（见第 4 页）

2. 在节点上安装 *xigemaAS*（见第 5 页）
3. 创建应用服务器（见第 6 页）

注册节点

注册节点是使用 xigemaAS 管理中心进行节点管理，以及其他模块管理的第一步。

请确认已满足以下前置条件：

- 若使用 xigemaAS 管理中心选择 SSH 代理方式注册节点，则需要满足以下环境要求：
  - 若使用证书认证方式，请确保已为 xigemaAS 管理中心生成了 SSH 公、私钥证书文件，并已将 SSH 公钥证书文件添加到待注册节点的受信文件中；
  - 已安装 curl 工具，并且当前 SSH 用户具备 curl 命令的执行权限；
  - xigemaAS 管理中心与当前节点所在主机的网络连通。
- 若使用 xigemaAS 管理中心选择 Agent 代理方式注册节点，详细信息和具体操作请参阅安装 *Agent*。
- 注册本机节点，节点所在主机可以是 Windows 系统或类 UNIX 系统。

当所有前置条件已满足，请在已登录 xigemaAS 管理中心主页状态下执行以下操作注册节点。

1. 单击**节点 > 注册**，进入注册节点页面。
2. 通过 xigemaAS 管理中心可以**“创建新的认证”**或**“使用已有认证”**注册节点。若使用已有认证信息注册节点，可以在**“认证信息”**下拉列表中选择已注册的节点，选择完成后，当前节点将直接使用已选节点的认证信息；若选择创建新的认证信息，请执行以下操作注册节点。

   xigemaAS 管理中心支持本机节点和远程节点的注册。

   a. **远程节点**：从**“代理类型”**栏选中 **SSH** 项，在远程主机上注册节点。xigemaAS 管理中心默认注册远程节点，须填写认证信息和基本信息。

      - 在**“认证信息”**面板填写以下信息：

图 3: 认证信息面板

      - 选择并填写节点所在主机的计算机名，或 IPv4 地址。
      - SSH 用户名，即登录节点所在主机的用户名。
      - SSH 端口号，即节点所在主机用于 SSH 通信的端口号，默认值为22。
      - 选择认证方式。默认选择“用户名/密码”方式认证，需要填写登录节点所在主机使用的密码。若选择“证书”方式认证，需要填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径。

- 在**“基本信息”**面板填写以下信息：

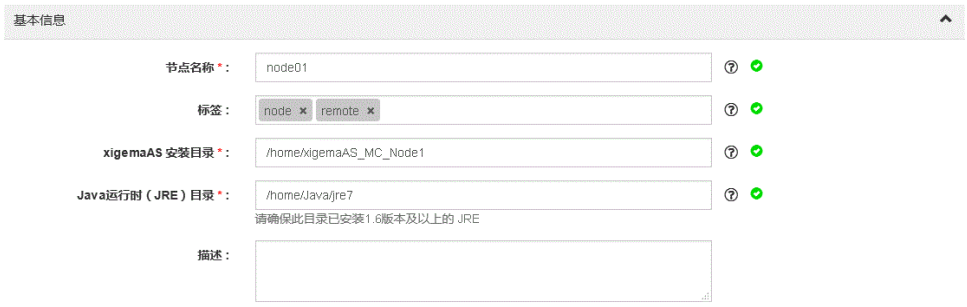


图 4: 基本信息面板

  - 节点名称。节点名称是节点在 xigemaAS 管理中心的唯一标识，保存后不可修改。
  - xigemaAS 安装目录。
  - Java 运行时（JRE）目录。

    注：保存后，在该节点下创建的服务器，将使用该目录作为其环境变量 JAVA_HOME 的值。修改此值不会修改已创建的服务器的 JAVA_HOME，但会作为此后创建服务器的 JAVA_HOME 路径。
  - 标签及描述，用于标识和分类管理节点。

b. **本机节点**：从**“代理类型”**栏选择**本机**项，在**“基本信息”**面板填写基本信息。注册 xigemaAS 管理中心所在主机上的节点，无需进行远程认证。

3. 单击**保存**按钮，保存节点注册信息。若提示 **“保存成功”** ，说明该节点注册成功。若存在无效信息，则无法保存。请参阅注册节点 （《xigemaAS 管理中心》文档中的）“注册节点”中界面元素的含义描述和操作规范，修改相应的节点信息，再进行保存操作以完成节点的注册。

   节点注册成功后，可继续注册新的节点或返回节点列表查看已注册节点。在节点列表中通过节点名称、主机名、xigemaAS 安装状态、标签可以查询已注册的节点。

在节点上安装 xigemaAS

节点注册完成后，需要在节点上安装 xigemaAS，安装成功后方可创建服务器。

1. 单击**节点**，进入节点管理页面。
2. 在节点列表中选择要安装 xigemaAS 的节点，单击**安装 AS** 按钮。提示**“xigemaAS 安装成功”**，则该节点 xigemaAS 安装状态更新为“已安装”。

   安装过程中，xigemaAS 管理中心自动进行该节点的环境检查。环境检查是否通过，直接影响该节点的 xigemaAS 能否安装成功。

   也可以在安装 xigemaAS 前，单击**环境检查**按钮以手动进行环境检查。有关环境检查的具体检查内容和操作请参阅在节点上安装 *xigemaAS* 《xigemaAS 管理中心》文档中的“在节点上安装 xigemaAS” 。

   xigemaAS 管理中心支持批量安装 xigemaAS，同时在安装过程中提供安装详情提示，以查看当前操作进度及详情。

   注：若所选节点上已安装 xigemaAS 、正在安装 xigemaAS、或正在卸载 xigemaAS，则不能重复执行安装操作。

创建应用服务器

在节点上成功安装 xigemaAS 后，可以通过 xigemaAS 管理中心在该节点上创建一个或多个应用服务器。

目标节点上已安装 xigemaAS，方能创建应用服务器。

1. 单击**服务器 > 应用服务器 > 创建**，进入**“应用服务器创建”**页面。
2. 在**“基本信息”**面板输入要创建的应用服务器的基本信息。

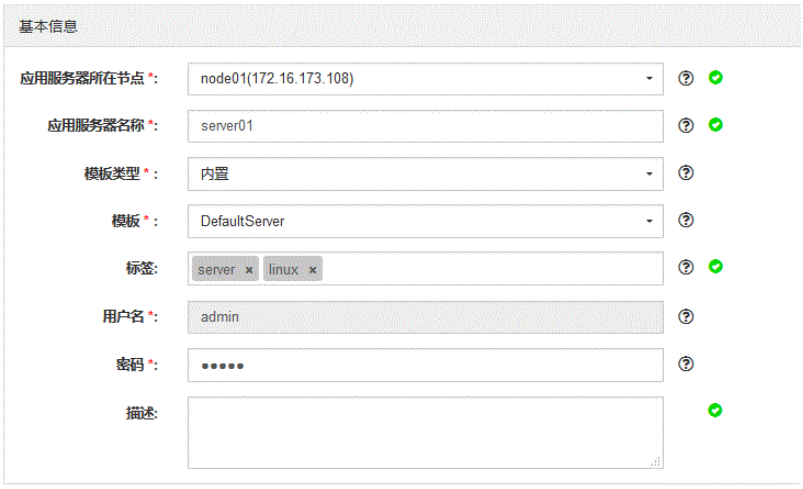


图 5: 应用服务器基本信息面板

a. 在**应用服务器所在节点**栏，可以从下拉列表中选择创建应用服务器的目标节点，也可以直接输入目标节点的名称，从而快速找到目标节点。
b. 在**应用服务器名称**栏输入要创建的应用服务器的名称。
c. 在**模板类型**栏选择应用服务器模板类型，默认模板类型为内置模板。选择完成后在**模板**栏选择相应类型的模板，默认模板为 DefaultServer。
d. 可选： 在**标签**、**描述**栏分别输入该应用服务器的标签和描述信息，用于标识和分类管理应用服务器。

3. 可选： 在**“系统属性”**面板编辑当前应用服务器的系统属性。

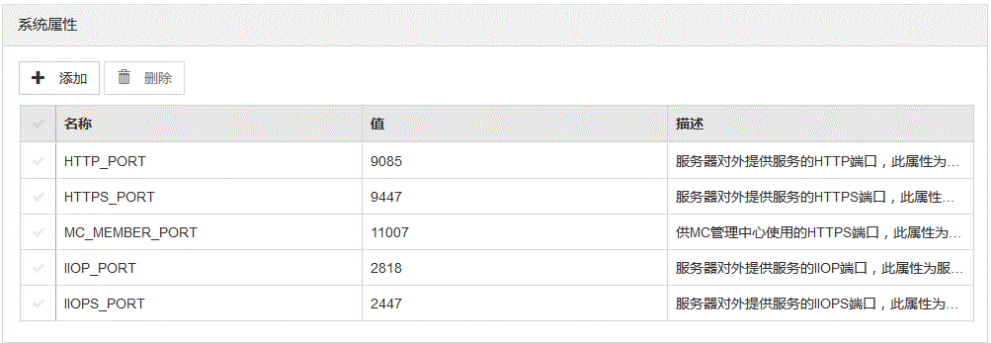


图 6: 应用服务器系统属性面板

a. 单击**添加**或**删除**按钮，添加或删除当前应用服务器的系统属性。其中，默认系统属性包括 HTTP_PORT、HTTPS_PORT、MC_MEMBER_PORT、IIOP_PORT、IIOPS_PORT，默认系统属性不可删除。

b. 单击系统属性的名称、值、描述，可对该系统属性进行编辑。默认系统属性仅支持属性值的编辑，名称和描述不可编辑。

4. 单击**保存 > 确定**，若提示 “**应用服务器创建成功**” ，说明已成功创建应用服务器。

   应用服务器创建成功后，可继续创建新的应用服务器，或返回应用服务器列表查看已创建的应用服务器。在应用服务器列表中通过应用服务器名称、应用服务器状态、标签、部署在应用服务器上的应用名称、应用服务器所在节点名称以及所属集群名称可以查询已创建的应用服务器。

有关创建应用服务器时界面元素的含义描述和操作规范，请参阅创建应用服务器 《xigemaAS 管理中心》文档中的“创建应用服务器”。

应用服务器创建成功后，可以作为应用的部署目标。一个应用可以选择多个应用服务器或集群作为部署目标。同时，当 Web 服务器添加到 xigemaAS 管理中心后，可以为 Web 应用和企业应用配置负载均衡。

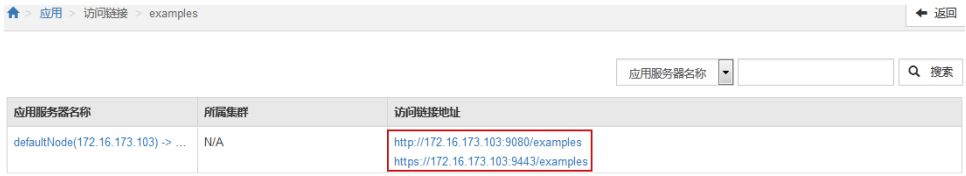
如何获取用例的访问地址

在应用部署完毕后，可在应用的信息中获得应用的访问地址。

本部分以示例说明如何获取用例的访问地址。在下一章“使用用例程序”中，统一按照以下方法获取访问地址。

1. 单击**应用**，进入**“应用”**页面。在某应用所在行中，单击其**链接**图标。

2. 获取访问地址。



1.2.2 使用用例程序

应用服务器创建成功后，可以作为应用的部署目标。一个应用可以选择多个应用服务器或集群作为部署目标。同时，当 Web 服务器添加到 xigemaAS 管理中心后，可以为 Web 应用和企业应用配置负载均衡。在本手册中，选择默认节点 `defaultNode` 下的默认应用服务器 `defaultServer` 作为应用部署的目标服务器。

请确认已满足以下前置条件：

- 应用部署所在应用服务器已成功创建，即存在部署目标。
- 所要部署的应用已放置在本地主机目录或者 xigemaAS 管理中心所在主机目录。

用例1： jsp 和 servlet 用例

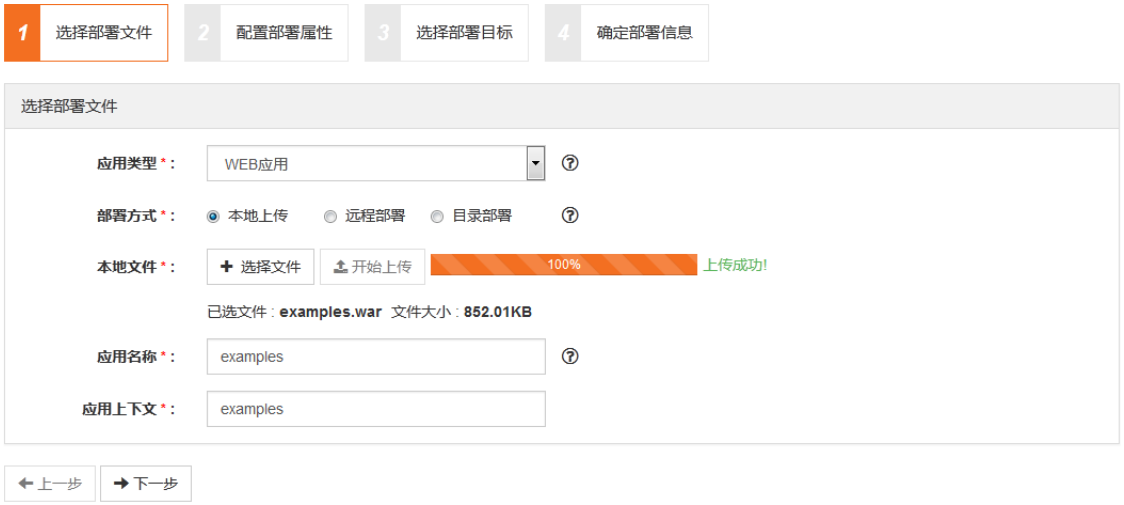
在应用服务器上部署并访问产品提供的 jsp 和 servlet 用例。

部署 wlp\samples 目录下的 example.war 应用。example.war 应用包括 jsp 和 servlet 用例。

按照以下步骤部署和访问该用例。

1. 单击**应用 > 部署**。进入**“选择部署文件”**页面。

图 7: 选择部署文件面板



a. 选择**应用类型**。本次部署的应用类型是 Web 应用，需要填写**应用上下文**。

xigemaAS 管理中心支持的应用类型包括：

- Web 应用
- 企业应用
- EJB 应用
- 连接器应用
- OSGi 应用

b. 选择应用**部署方式**。本次的部署方式为本地上传，具体操作：单击**选择文件**，从目录 wlp\samples\examples 下选择文件 examples.war。单击**开始上传**。完成后，右侧提示“上传成功”。

对于 Web 应用和企业应用，xigemaAS 管理中心支持本地上传、远程部署以及目录部署三种方式；针对其他应用，仅支持本地上传和远程部署。

根据所选部署方式，选择应执行的操作：

- 本地上传或远程部署：

  1. 单击**选择文件 > 开始上传**，若提示**上传成功**，则所选文件已上传。

- 目录部署：

  在部署Web 应用和企业应用时，如果选择**目录部署**，

  1. 在下方显示应用目录输入框，输入应用所在目录，如 D:\installAS\wlp\samples\examples。该应用必须是解压后文件。
  2. 填写**应用上下文**和**应用名称**。

  注：对于目录部署：

  - xigemaAS 管理中心运行用户必须对应用所在目录有读权限。
  - 如果多点部署（如集群），要求应用存在于各个部署服务器的同一目录下。

2. 配置部署属性。本次部署默认不配置部署属性。单击**下一步**。

   如果选择填写部署属性，参考以下可选步骤：

   a. 可选： 选择私有共享库、公有共享库、以及类加载优先顺序。若应用类型为 OSGi 应用，选择共享库和类加载优先顺序。

   b. 可选： 在**标签**、**描述**栏分别输入该应用的标签和描述信息，用于标识和分类管理应用。

3. 选择部署目标，设置负载均衡。本次部署选择一个部署目标。

   a. 在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。

图 8: 选择部署目标面板

   b. 可选： 若当前应用类型为 Web 应用或企业应用，可以为应用配置负载均衡。在**“负载均衡设置”**面板输入 Web 服务器端应用上下文，选择配置负载均衡的虚拟主机。

图 9: 设置负载均衡面板

4. 确认部署信息。确认部署应用相关信息是否填写正确。若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若提示 **“应用已部署”** ，则该应用已成功部署。

   应用部署成功后，可继续部署新的应用或返回应用列表查看已部署的应用。在应用列表中通过应用名称、应用类型、应用状态、标签可以查询已部署的应用。

部署应用成功后，用户可以访问该应用。

1. 通过 http://localhost:9080/examples 访问
2. 应用页面如下图所示。通过单击下列各个链接可访问具体用例。

用例2：数据库连接池

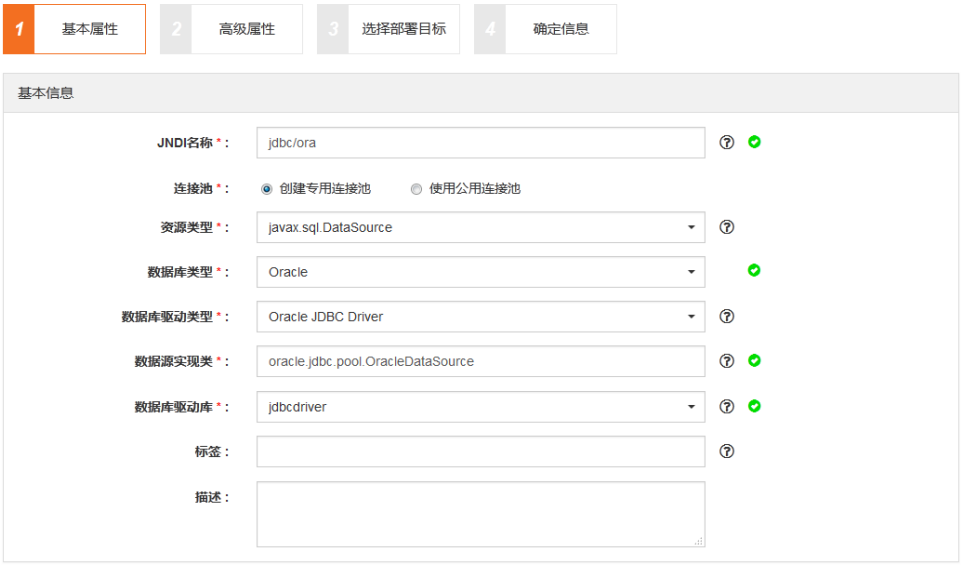
在应用服务器上部署并访问产品提供的数据库连接池用例。

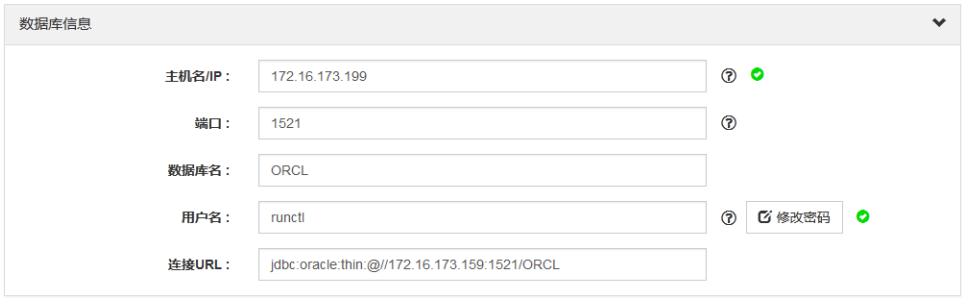
本用例的数据源以 Oracle 数据库为例。数据库的 **JDBC** 驱动 ojdbc6.jar 位于 samples/driver 目录下；数据库连接池用例 DataBase.war 位于 samples\jdbc目录下。

本部分描述如何使用 xigemaAS 管理中心 部署并使用数据库连接池。

1. 添加共享库。
   a. 单击**资源管理 > 共享库 > 新增**。
   b. 在**“共享库新增”**页面，
      1. 填写必要的基本信息。**共享库名称**为：jdbcdriver。
      2. 添加资源文件 ojdbc6.jar。单击 **新增 > 本地上传 > 选择文件**，从目录 samples/driver 下选择 ojdbc6.jar，并单击 **开始上传 > 确定**。
      3. 可以继续添加其他信息。完成后，单击**保存 > 确定**。

      共享库新增完毕。
2. 添加数据源。
   a. 单击**资源管理 > 数据源 > JDBC > 新增**。
   b. 在**“JDBC数据源新增”**页面的基本属性内容中，填写必要的基本信息和数据库信息。JNDI名称为：jdbc/ora。其他信息的填写如下图所示。





c. 继续配置高级属性，并选择部署目标。单击**下一步**。

> 注：关于高级属性和部署目标的操作规范、字段描述信息，参见《xigemaAS 管理中心使用手册》。

d. 在**“确定信息”**页，确认基本属性信息后，单击**完成 > 确定**。

JDBC 数据源添加完成。

3. 部署应用。如果需要以下各步骤的详细说明信息，可以参考用例*1*： *jsp* 和 *servlet* 用例 （见第 7 页）。

a. 单击**应用 > 部署**，进入**“选择部署文件”**页面。

1. 选择**应用类型**。本次部署的应用类型是 Web 应用，需要填写**应用上下文**。
2. 选择应用**部署方式**。本次的部署方式为本地上传，具体操作：单击**选择文件**，从目录 samples\jdbc 下选择文件 DataBase.war，然后单击**开始上传**。完成后，右侧提示“上传成功”。
3. 单击**下一步**。

b. 配置部署属性。本次部署默认不配置部署属性。单击**下一步**。

c. 选择部署目标，设置负载均衡。本次部署选择一个部署目标。

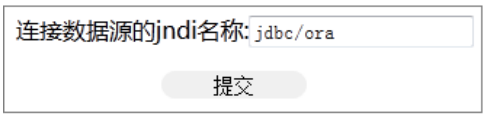
d. 在**“确定信息”**页，确认基本属性信息后，单击**完成 > 确定**。

在**“应用”**页面，从已部署的应用列表中可以看到该应用的名称。

4. 访问应用。

a. 通过 http://localhost:9080/Database 访问此用例程序。

b. 可以检查应用程序的功能。输入JNDI名称 jdbc/ora，单击**提交**。



用例3：EJB 用例

在应用服务器上部署并访问产品提供的 EJB 用例程序。

EJB 用例位于 samples\ejb 目录下，其中包含：

- session\EjbEarTest.ear：会话 bean的应用示例
- entity\EntityEjbTest.ear：实体 bean 的应用示例

本部分内容分别介绍会话 bean 和实体 bean 的部署。

部署会话 bean

部署 ejb\session 目录下的会话 bean 应用 EjbEarTest.ear。

1. 单击**应用 > 部署**，进入**“选择部署文件”**页面。

   注：如果需要以下各步骤的详细说明信息，请参考用例*1*： *jsp* 和 *servlet* 用例 （见第 7 页）。

   a. 选择**应用类型**。本次部署的应用类型是 企业应用，需要填写**应用名称**。
   b. 选择**部署方式**。本次的部署方式为本地上传，具体操作：单击**选择文件**，从目录 ejb\session 下选择文件 EjbEarTest.ear，然后单击**开始上传**。完成后，右侧提示“上传成功”。
   c. 单击**下一步**。
2. 配置部署属性。本次部署默认不配置部署属性。单击**下一步**。
3. 选择部署目标，设置负载均衡。本次部署选择一个部署目标。单击**下一步**。

   在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。
4. 确认部署信息。确认部署应用相关信息是否填写正确。确认无误后，单击**完成 > 确定**。

在**“应用”**页面，从已部署的应用列表中可以看到该应用的名称。

访问会话 bean：

- 访问地址 http://localhost:9080/WebCallStatefullEjb：访问有状态会话 bean。
- 访问地址 http://localhost:9080/WebCallStatelessEjb：访问无状态 会话 bean。

部署实体 bean

部署 ejb\entity 目录下的实体 bean 应用 EntityEjbTest.ear。

1. 新增数据源。

   添加共享库 ojdbc6.jar。新增数据源，JNDI 名称为 jdbc/data。具体填写方法，参考用例*2*：数据库连接池（见第 10 页）。如果已经添加共享库 ojdbc6.jar，则无需重复添加。
2. 修改应用服务器的配置。在 server.xml 文件中添加功能部件 jpa-2.1， 用于启用对应用程序的支持。

   jdbc/data 所在的应用服务器的名称是 defaultServer。修改此应用服务器的 server.xml 文件。

   单击**应用服务器 > defaultServer > server.xml 配置**，单击页面右上角的**编辑**按钮，进入以下配置页面：

   xigemaAS 管理中心提供源窗格和设计窗格两种方式编辑 `server.xml` 配置文件

   - 源：
     1. 单击**“源”**，在功能部件管理器中查看并编辑 `server.xml` 配置文件。
     2. 在 featureManager 模块中添加一行内容：`<feature>jpa-2.1</feature>`。
   - 设计：

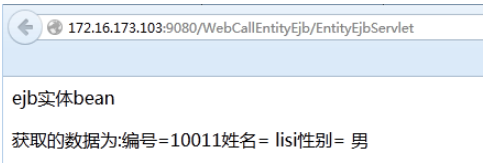
1. 单击**设计**（默认选项）。在左侧的元素列表中，单击**功能部件管理器**，然后在右侧面板单击**添加子代**。
2. 在**“添加子代”**页面中，单击**添加 > 选择**，从值列表中选择要添加的功能部件：jpa-2.1。

注：server.xml 配置文件修改后，需要重启应用服务器。

3. 部署 实体 bean 应用。关于更多的填写说明，参考 用例*1*： *jsp* 和 *servlet* 用例 （见第 7 页）。
   a. 单击**应用 > 部署**。进入**“选择部署文件”**页面。
      1. 选择**应用类型**。本次部署的应用类型是 Web 应用，需要填写**应用上下文**。应用上下文名称为：`WebCallEntityEjb`。
      2. 选择**部署方式**。本次的部署方式为本地上传，具体操作：单击**选择文件**，从目录 ejb\entity 下选择文件 EntityEjbTest.ear。单击 **开始上传**。完成后，右侧提示“上传成功”。
      3. 单击**下一步**。
   b. 配置部署属性。本次部署默认不配置部署属性。单击**下一步**。
   c. 选择部署目标。在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。单击**下一步**。
   d. 确认部署信息。确认部署应用相关信息是否填写正确。确认无误后，单击**完成 > 确定**。

部署应用成功后，用户可以访问该应用。

通过 http://localhost:9080/WebCallEntityEjb 访问实体 bean。访问成功后，将显示如下信息：



用例4：JMS 用例程序

在应用服务器上部署并使用产品提供的 JMS 用例程序。

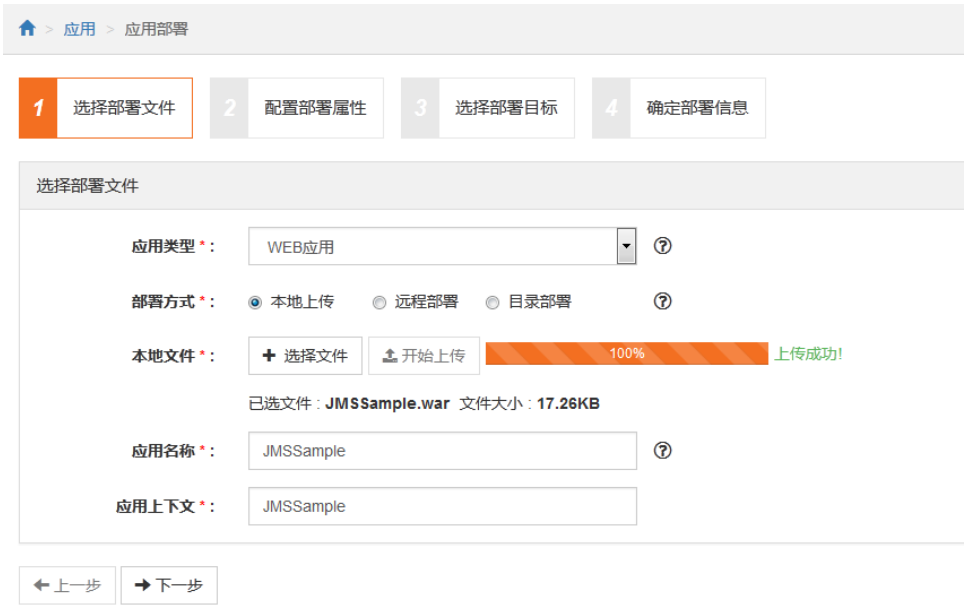
本示例中的 JMS Provider 以 ActiveMQ 为例。在本示例中，将在目标应用服务器上部署 Web 应用 JMSSample.war 和连接器应用 activemq-rar-5.10.0.rar。应用部署完成后，测试配置激活规范前后 ActiveMQ 消息发送中的变化。

部署应用

部署 Web 应用 JMSSample.war 和连接器应用 ActiveMQ。

JMS 用例程序 位于 samples\jms 目录下。按以下步骤进行部署。

1. 创建应用服务器。
   使用创建应用服务器 on page 6 中创建的应用服务器。
2. 部署 Web 应用 JMSSample.war。
   a. 单击**应用 > 部署**。进入**“选择部署文件”**页面。
      部署该应用的相关步骤说明，参考 用例*1*： *jsp* 和 *servlet* 用例 on page 7。



1. 选择**应用类型**。本次部署的应用类型是 Web 应用，需要填写**应用上下文**。
2. 选择**部署方式**。本次的部署方式为本地上传，具体操作：单击**选择文件**，从目录 jms 下选择文件 JMSSample.war。单击 **开始上传**。完成后，右侧提示“上传成功”。
3. 单击**下一步**。

b. 配置部署属性。本次部署默认不配置部署属性。单击**下一步**。

c. 选择部署目标。在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。单击**下一步**。

d. 确认部署信息。确认部署应用相关信息是否填写正确。确认无误后，单击**完成 > 确定**。

3. 部署连接器应用 activemq-rar-5.10.0.rar。

参考上述步骤*2* 部署该应用。其中，应用类型 为连接器应用。从该应用所在地址进行本地上传。

Note: 若 xigemaAS 管理中心 与 ActiveMQ 不在同一台机器上，需修改 serverUrl ，具体步骤如下所示；但若在同一台机器上，则不需要修改。

1. 单击应用列表中的应用名称 `activemq-rar-5.10.0`，单击**资源适配器属性**项，进入**“资源适配器属性”**页面。

2. ，修改属性值 serverUrl。

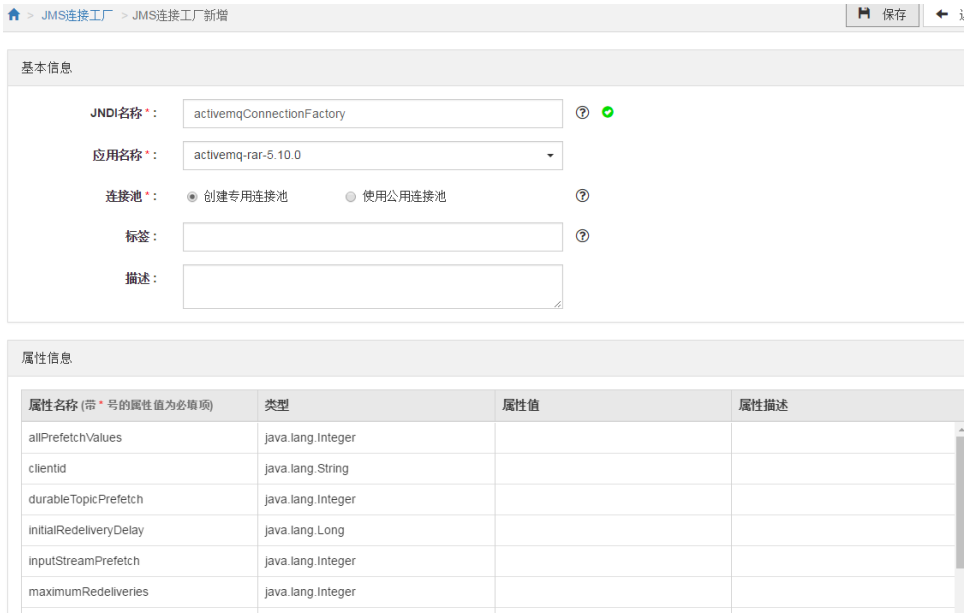
测试 ActiveMQ

本部分介绍测试 JMS 队列、测试激活规范和测试 JMS 主题。

在部署 Web 应用和连接器应用后，对 ActiveMQ 进行相关测试。测试前需进行以下准备工作：创建连接工厂、队列连接工厂和队列。

1. 新建连接工厂。连接工厂名称为 activemqConnectionFactory。

   a. 单击**资源管理 > JMS > 连接工厂 > 新增**，进入**“JMS连接工厂新增”**页面。

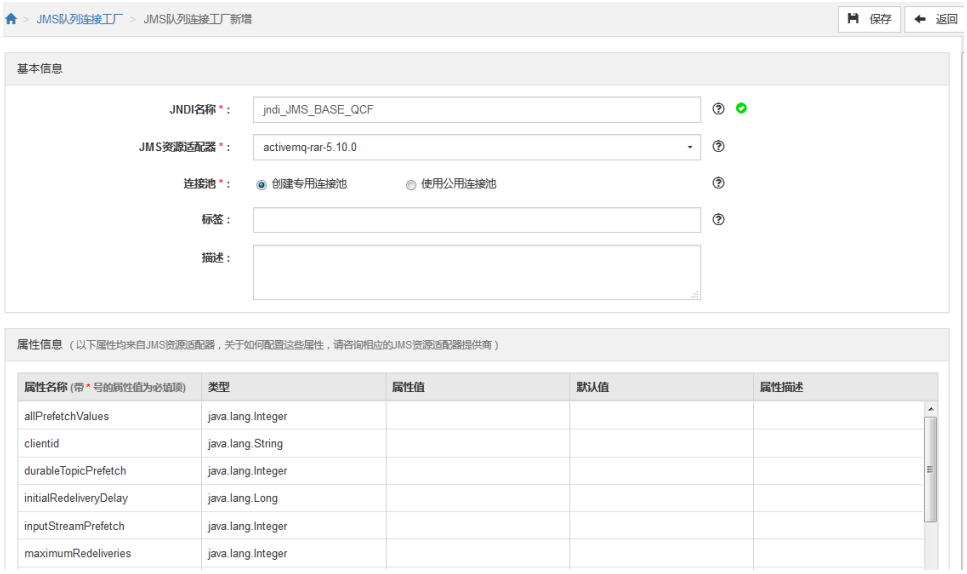


   b. 填写必要信息。单击**保存**。

   创建连接工厂成功。

2. 新建队列连接工厂。队列连接工厂名称为 jndi_JMS_BASE_QCF。

   a. 单击**资源管理 > JMS > 队列连接工厂 > 新增**，进入**“JMS队列连接工厂新增”**页面。



b. 填写必要信息。单击**保存**。

创建队列连接工厂成功。

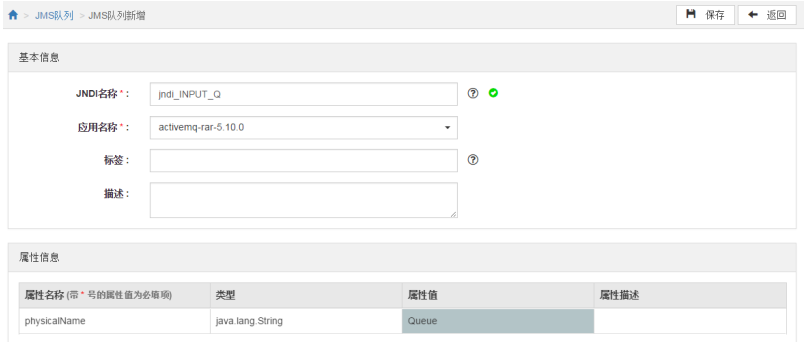
3. 新建队列。

新建以下三个队列：

JNDI	队列名称	描述
jndi_INPUT_Q	Queue	用于测试 JMS 队列。
jndi/MDBQ	mdbq	用于测试激活规范
jndi/MDBREPLYQ	mdbreplyq	用于测试 JMS 主题

a. 单击**资源管理 > JMS > 队列 > 新增**，进入“JMS队列新增”页面。

b. 填写必要信息。首先创建上述列表中的第一个队列，其 JNDI名称为`jndi_INPUT_Q`，属性值为`Queue`。



c. 单击**保存**。

队列 `jndi_INPUT_Q` 创建完成。请重复以上步骤，分别创建队列 `jndi/MDBQ` 和 `jndi/MDBREPLYQ`。

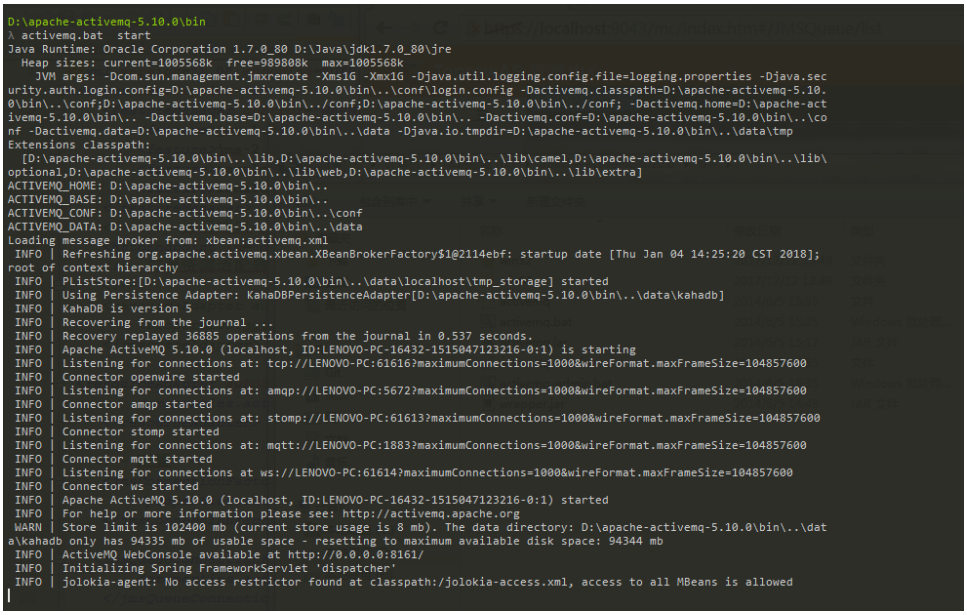
测试 JMS 队列

测试 JMS 队列。

1. 启动 ActiveMQ。

   a. 进入到 apache-activemq-5.10.0\bin 目录下，在命令行中输入以下命令：

```
activemq.bat start
```



   如上图所示，ActiveMQ 已启动。

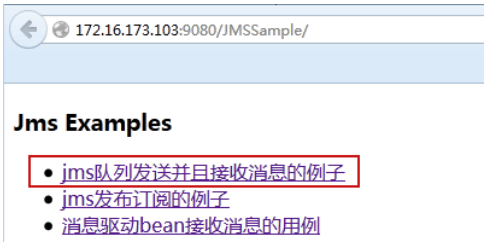
2. 登录 ActiveMQ。

   访问 ActiveMQ 首页：*http://localhost:8161*。单击链接 Manager ActiveNQ broker，在弹出页面输入用户名/密码：admin/admin。

   登录 ActiveMQ 成功。

3. 访问应用页面。

   a. 通过应用链接 http://ip:9080/JMSSample 访问应用首页。

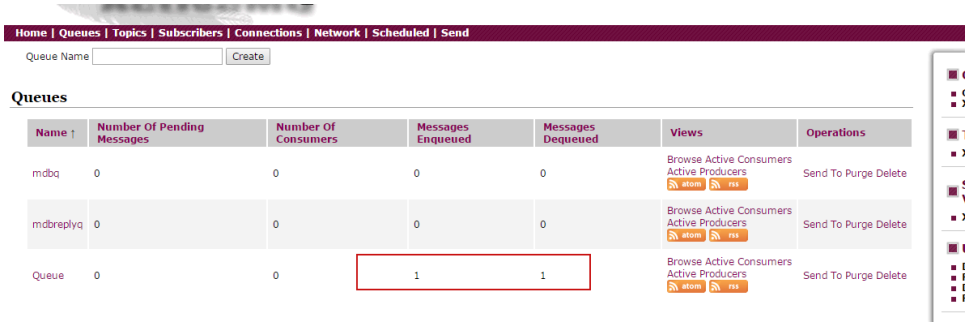


b. 单击第一个链接，测试JMS 队列。显示信息如下：

```
SendAndReceive Started
Message sent successfully
Received Message Successfully :ActiveMQTextMessage {commandId = 15, responseRequired = true,
messageId = ID:lenovo-PC-56799-1569229746315-25:1:2:1:1, originalDestination = null,
originalTransactionId = null, producerId = ID:lenovo-PC-56799-1569229746315-25:1:2:1, destination =
queue://Queue, transactionId = null, expiration = 0, timestamp = 1569230629285, arrival = 0,
brokerInTime = 1569230629285, brokerOutTime = 1569230629311, correlationId = null, replyTo = null,
persistent = true, type = null, priority = 4, groupID = null, groupSequence = 0, targetConsumerId =
null, compressed = false, userID = null, content = org.apache.activemq.util.ByteSequence@19e9298c,
marshalledProperties = null, dataStructure = null, redeliveryCounter = 0, size = 0, properties =
null, readOnlyProperties = true, readOnlyBody = true, droppable = false, jmsXGroupFirstForConsumer =
false, text = Liberty Sample Message}
SendAndReceive Completed
```

Queue 队列发送且接收消息成功。

4. 查看队列 Queue 的已发送和已接收消息。



测试激活规范

本部分内容使用“消息驱动bean接收消息的用例”来测试激活规范配置。

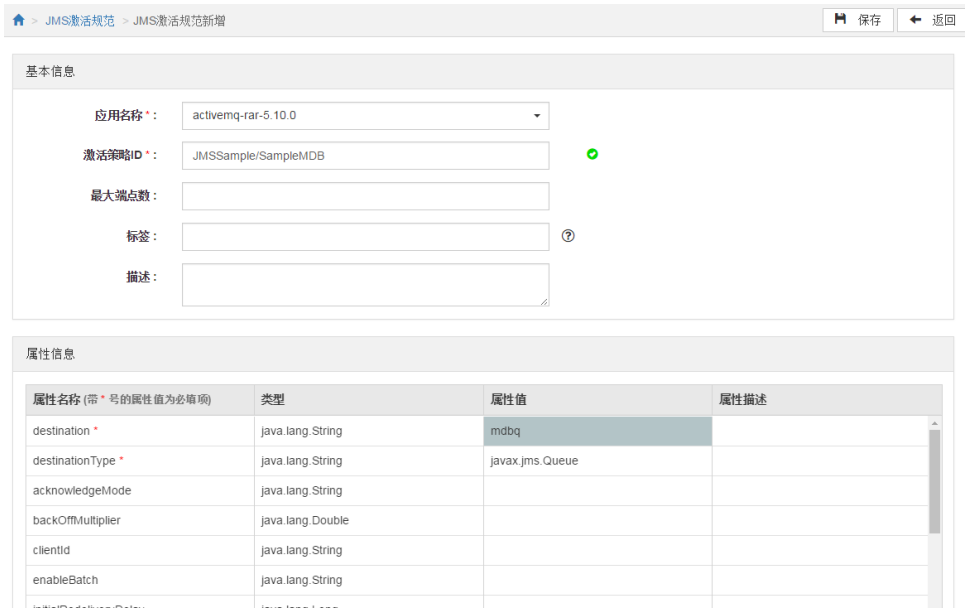
在配置激活规范之前，访问链接 ####bean######## 失败。显示消息如下：

```
MDBRequestResponse Started
Message sent successfully
Something unexpected happened, check the logs or restart the server
```

因此，需要配置激活规范。

请按以下步骤配置激活规范。

1. 单击**资源管理 > JMS > 激活规范 > 新增**，进入**“激活规范新增”**页面。



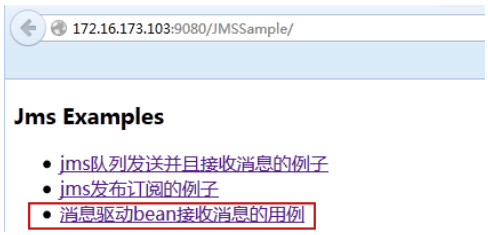
a. 填写基本信息。应用名称为`activemq-rar-5.10.0`，激活策略ID为 `JMSSample/SampleMDB`

b. 在**“属性信息”**中，填写必填项的属性值。destination 的值为 `mdbq`，destinationType的属性值为`javax.jms.Queue`。

c. 单击**保存**。

2. 重启应用服务器。单击**应用服务器**，然后单击应用所在服务器 `defaultServer` 的 重启图标。应用服务器重启成功。

3. 通过应用链接 http://ip:9080/JMSSample 访问应用首页。单击第三个链接。如下图所示：



消息接收成功。显示信息如下：

此时也可以查看服务器日志中显示发送的信息。具体操作：单击 **应用服务器 > defaultServer （应用所在服务器的名称） > 日志下载**。

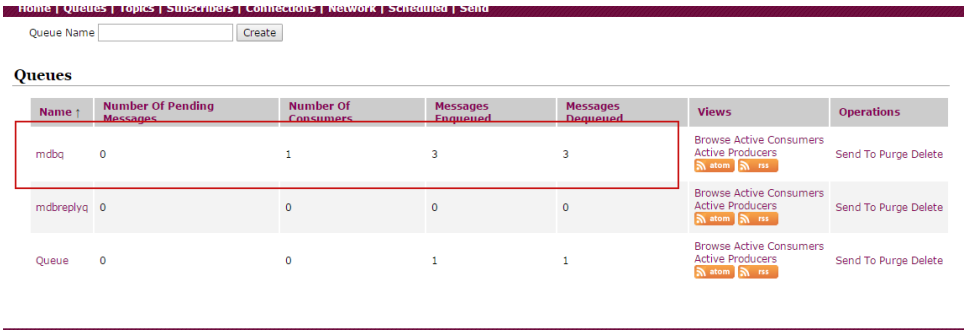
```
I J2CA8801I: 激活规范 JMSSample/SampleMDB 和消息驱动的 bean 应用程序 JMSSample#JMSSample.war#SampleMDB 的消息端点已激活。
A CWWKF0012I: 服务器已安装下列功能部件：[jsp-2.3, ejbLite-3.2, servlet-3.1, ssl-1.0, jndi-1.0, ejbHome-3.2, jca-1.7,
ejbPersistentTimer-3.2, jms-2.0, j2eeManagement-1.1, jdbc-4.1, el-3.0, ejbRemote-3.2, restConnector-2.0, mdb-3.2, mcMember-1.0, ejb-3.2,
json-1.0, distributedMap-1.0, jpa-2.1]。
I CWWKF0008I: 已在 24.546 秒内完成功能部件更新。
A CWWKF0011I: 服务器 defaultServer 已准备就绪，可开始运行xigemaAS。
O 2019-09-24 17:24:15,835 [ecutor-thread-3] INFO  ActiveMQEndpointWorker          - Establishing connection to broker
[tcp://localhost:61616]
O 2019-09-24 17:24:16,075 [ecutor-thread-3] INFO  ActiveMQEndpointWorker          - Successfully established connection to broker
[tcp://localhost:61616]
I J2CA8050I: 应使用认证别名，而非在 jndi_JMS_BASE_QCF 上定义用户名和密码。
O Message Received in MDB !!!ActiveMQTextMessage {commandId = 5, responseRequired = true, messageId = ID:lenovo-PC-58207-1569316745383-
3:1:1:1:1, originalDestination = null, originalTransactionId = null, producerId = ID:lenovo-PC-58207-1569316745383-3:1:1:1, destination
= queue://mdbq, transactionId = null, expiration = 0, timestamp = 1569316836276, arrival = 0, brokerInTime = 1569316836277,
brokerOutTime = 1569317056080, correlationId = null, replyTo = null, persistent = true, type = null, priority = 4, groupID = null,
groupSequence = 0, targetConsumerId = null, compressed = false, userID = null, content = org.apache.activemq.util.ByteSequence@77c5bfbb,
marshalledProperties = org.apache.activemq.util.ByteSequence@6973748c, dataStructure = null, redeliveryCounter = 0, size = 0, properties
= {COLOR=BLUE}, readOnlyProperties = true, readOnlyBody = true, droppable = false, jmsXGroupFirstForConsumer = false, text = MDB Test -
Message to MDB}
```

4. 查看消息发送和接收情况。

   a. 登录 ActiveMQ。具体登录方法，参见登录 *ActiveMQ*。

   b. 单击菜单栏中的 **Queues** 项。

   队列 mdbq 信息发送和接收情况如下：



测试JMS主题

本部分测试 JMS 主题。

1. 创建主题连接工厂。

   a. 单击**资源管理 > JMS > 主题连接工厂 > 新增**，进入**“JMS主题连接工厂新增”**页面。如下图所示：

   b. 填写信息。主题连接工厂的 JNDI 名称为：jmsTCF。

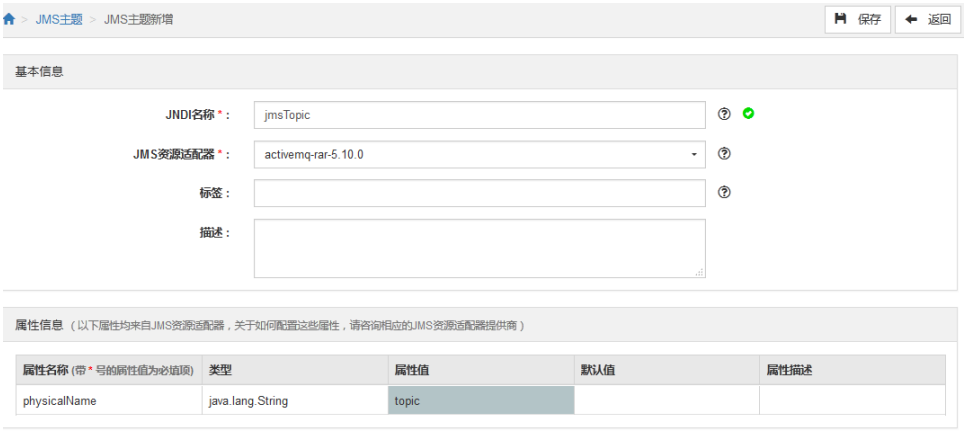
   c. 单击**保存 > 确定**。

主题连接工厂 jmsTCF 创建成功。

2. 创建主题。

   a. 单击**资源管理 > JMS > 主题 > 新增**，进入“JMS主题新增”页面。

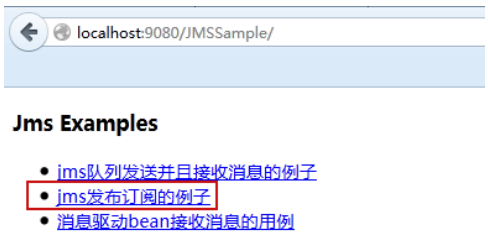
   如下图所示：



   b. 填写必要信息。主题的 JNDI 名称为：jmsTopic，physicalName 的属性值为：topic。

   c. 单击**保存 > 确定**。

3. 通过应用链接 http://ip:9080/JMSSample 访问应用首页。单击第二个链接。如下图所示：



   消息发布订阅成功。显示信息如下：

```
NonDurableSubscriber Started
Received message for non-durable subscriber ActiveMQTextMessage {commandId = 6, responseRequired = true, messageId =
ID:lenovo-PC-59247-1569390368181-3:1:1:1:1, originalDestination = null, originalTransactionId = null, producerId = ID:lenovo-
PC-59247-1569390368181-3:1:1:1, destination = topic://topic, transactionId = null, expiration = 0, timestamp =
1569390467569, arrival = 0, brokerInTime = 1569390467570, brokerOutTime = 1569390467571, correlationId = null, replyTo =
null, persistent = true, type = null, priority = 4, groupID = null, groupSequence = 0, targetConsumerId = null, compressed =
false, userID = null, content = org.apache.activemq.util.ByteSequence@34a0cdbf, marshalledProperties = null, dataStructure =
null, redeliveryCounter = 0, size = 0, properties = null, readOnlyProperties = true, readOnlyBody = true, droppable = false,
jmsXGroupFirstForConsumer = false, text = Liberty PubSub Message}
NonDurableSubscriber Completed
```

4. 查看 ActiveMQ 页面。

   a. 登录 ActiveMQ。具体登录方法，参见登录 *ActiveMQ*。

   b. 单击菜单栏中的**Topics**。

   主题 topic 信息发送和接收情况如下：


Topic Name
Create
Topics
Name ↑
Number Of Consumers
Messages Enqueued
Messages Dequeued
Operations
ActiveMQ.Advisory.Connection
0
2
0
Send To Active Subscribers
Active Producers
Delete
ActiveMQ.Advisory.Consumer.Topic.topic
0
2
0
Send To Active Subscribers
Active Producers
Delete
ActiveMQ.Advisory.Producer.Topic.topic
0
2
0
Send To Active Subscribers
Active Producers
Delete
ActiveMQ.Advisory.Topic
0
1
0
Send To Active Subscribers
Active Producers
Delete
topic
0
1
1
Send To Active Subscribers
Active Producers
Delete

2 欢迎使用 xigemaAS

为方便开发者使用并提高工作效率，对 xigemaAS 进行了优化。xigemaAS 是小型应用程序服务器，通过简化方法来配置服务器，并且在开发和运行不需要传统企业应用程序服务器的完整 Java™ EE 环境时，它的超短重新启动时间、小规模、动态行为和易于使用等特点使得它成为一个很好的选择。

xigemaAS 的主要优势包括但不限于以下功能：

- 供开发之用的免费顺畅下载
- 超轻量级模块化运行时环境
- 超短启动时间：简单的 Web 应用程序启动时间不到 5 秒
- 简化的配置，缩短了投产时间
- Web 应用程序的 Java™ EE 和 OSGi 应用程序部署支持
- LDAP 注册表支持
- EJB Lite 和 CDI，用于实现对 Java™ EE Web 概要文件的完全支持
- Web Service 支持，通过 JAX-WS Java™ EE API 实现
- 对单服务器消息传递提供程序、JMS、消息驱动的 bean (MDB) 及 MXBean 的消息传递支持
- 用于通过创建第三方 xigemaAS 功能部件扩展运行时环境的选项
- 将应用程序和已配置的服务器作为一个软件包来部署

2.1 概述

xigemaAS 是一个高度可组合、快速启动的动态应用程序服务器运行时环境。

因为 xigemaAS 不包含 Java™ 运行时环境 (JRE)，所以您必须事先安装兼容的 Java™ 实现（JRE 或 SDK）。有关受支持的 Java™ 环境及如何获取这些环境的更多信息，请参阅支持的最低 *Java* 级别（见第 1669 页）。

此服务器支持两个应用程序部署模式：

- 通过xigemaAS 管理中心部署应用程序。
- 通过将应用程序拖放至 apps 目录中，并在服务器配置文件（server.xml）添加相关配置描述来部署应用程序。

xigemaAS 支持下列部件的子集：

- Web 应用程序
- OSGi 应用程序
- Enterprise JavaBeans™ (EJB) 应用程序

事务和安全性等相关联的服务可供这些应用程序类型使用。

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。有关主要 xigemaAS 功能部件的列表，请参阅*xigemaAS* 功能部件（见第 906 页）。

您也可以创建自己的功能部件，如扩展 *xigemaAS*（见第 1232 页）中所述。

您可以直接使用运行时环境，也可以使用管理中心以通过 Web 浏览器管理 xigemaAS 服务器和应用程序及其他资源。请参阅管理 *xigemaAS*（见第 1113 页）。

在分布式平台上，xigemaAS 提供开发环境和运营环境。

2.1.1 xigemaAS 中的 Java EE 7

xigemaAS 现在支持完整 Java™ Platform Enterprise Edition (Java EE) 7。

Java EE 7 的优点

- 通过 Java EE 7 Web 概要文件，更易于为台式机、平板电脑和智能手机提供 HTML5 动态可缩放应用程序。

  Java EE 6 引入了 Web 概要文件以帮助开发者开发动态 Web 应用程序，它提供 Enterprise JavaBeans™ (EJB) Lite、Java Persistence API (JPA) 和 Java Transaction API (JTA) 之类的技术。

  JavaEE 7 Web 概要文件添加了对 HTML5 的支持。WebSocket 和 JSON 这两种新技术提高了数据交换速度，并简化了可移植应用程序的数据解析。对现有技术（JAX-RS 2.0、Java Server Faces (JSF) 2.2 和 Servlet 3.1）的更新增强你对动态 HTML5 应用程序的开发能力。例如，JAX-RS 2.0 通过可缩放的高性能 RESTful 服务提供异步响应处理。
- 提高开发者工作效率

  简化后的应用程序体系结构实现业务逻辑（例如，在 JMS 2.0 中）和 JAX-RS 2.0 客户机 API 时所需的样板代码更少。更稳健的 POJO 开发模型允许更广泛地使用注解，例如，在 JAX-RS 2.0 拦截器和过滤器及 CDI 中。Bean 验证 1.1 提供方法级验证。
- 改进了对企业需求的支持

  Java EE 仍使用 Java EE 连接器体系结构 (JCA)、Java 事务 API (JTA) 和 Java 消息服务 (JMS) 提供对企业需求的支持。Java EE 7 引入了使用 Java 编写批处理应用程序的功能，这些应用程序使用标准 API 并且可在多个运行时之间移植。批处理应用程序允许您更好地利用计算资源，方法是将处理时间移至资源通常空闲的时间段。并行实用程序允许开发者以安全可靠的方式编写与 Java EE 运行时集成的可缩放应用程序。
- 已废弃旧技术

  之前技术是可选的：应用程序部署 (JSR-88)；JAXR，用于与 UDDI 注册中心交互 (JSR-93)；JAX-RPC，用于基于 XML 的 RPC (JSR-101)；以及 EJB 2.x 容器管理的持久性，Java Persistence API (JSR-338) 是其替代项。

Java EE 7 完整平台和 Web 概要文件

Java EE 7 规范定义完整平台企业版。Web 概要文件是合理定义的完整平台子集。

Web 概要文件

Web 概要文件定义合理的完整堆栈，此堆栈的目标为“现代”Web 应用程序。此堆栈是完整平台标准 API 的子集，能够满足大部分 Web 应用程序的需求。

完整平台

完整平台定义对 Java EE 编程模型的完整补充。除 Web 概要文件功能部件外，完整平台具有企业所需的高级企业功能（例如，用于连接、安全性、企业 bean、消息传递和应用程序客户机）的规范。

安装 xigemaAS 及其可用功能部件时，您需要考虑以下定义。安装选项提供 xigemaAS 运行时（内核）以及支持完整平台或 Web 概要文件的功能部件。可选择安装仅带有 Web 概要文件功能部件的 xigemaAS 内核，或将各 Java EE 功能部件安装至现有 xigemaAS 内核安装。

将应用程序部署至 xigemaAS 服务器时，您也需要考虑这些定义。例如，如果 xigemaAS 安装支持 Web 概要文件功能部件，但您的应用程序需要一个支持企业连接 API 的服务器，那么您必须将完整平台功能部件添加至 xigemaAS 安装，或将该应用程序部署至具有完整平台功能部件的 xigemaAS 安装。

要将对 Java EE 的支持快速添加至 xigemaAS 服务器，请在服务器配置的功能部件管理器中使用 `webProfile-7.0` 或 `javaee-7.0` 便利功能部件；例如：

```
<featureManager>
   <feature>javaee-7.0</feature>
</featureManager>
```

第三个便利功能部件 `javaeeClient-7.0` 也可用于快速配置应用程序客户机组件：

```
<featureManager>
   <feature>javaeeClient-7.0</feature>
</featureManager>
```

用于帮助您开始使用 Java EE 7 技术的资源

- 有关 Java EE 7 的体系结构主题
  - *Java EE 7* 编程模型支持（见第 28 页）
  - 受支持的 *Java EE 6* 与 *Java EE 7* 功能部件组合（见第 32 页）
  - *Java EE 7* 行为更改（见第 34 页）
- *xigemaAS* 功能部件（见第 906 页）
- 准备和运行应用程序客户机（见第 1109 页）
- *Java EE 7* 规范
- *Java EE 7* 样本

2.1.2 体系结构

xigemaAS 概要文件是一个高度可组合的动态运行时环境。使用 OSGi 服务来管理组件生命周期，以及管理依赖性和配置的注入过程。服务器进程由单个 JVM、xigemaAS 内核以及任何数目的可选功能部件组成。功能部件代码及大部分内核代码作为 OSGi 捆绑软件（OSGi bundle）在 OSGi 框架中运行。功能部件提供应用程序所需要的编程模型和服务。

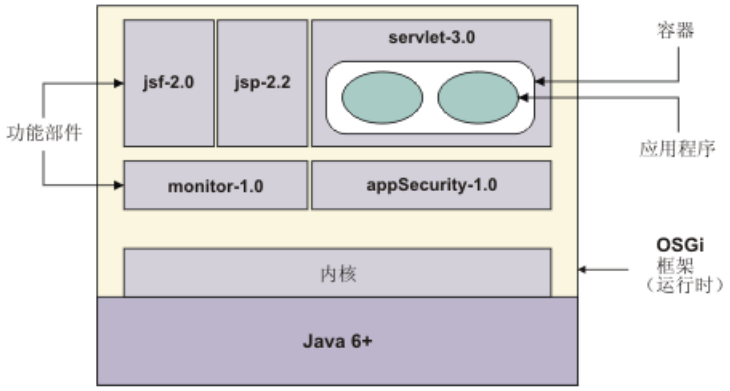


图 10: xigemaAS 体系结构

内核启动程序会引导系统，并启动 OSGi 框架。配置会加以解析，然后所配置的功能部件会由功能部件管理器装入。内核会大量地使用 OSGi 服务来提供高度动态的运行时环境。OSGi 配置管理服务会管理系统配置，而 OSGi 声明式服务组件会管理系统服务的生命周期。文件监视器服务会检测应用程序和配置文件更改，而记录服务会将消息和调试信息写入本地文件系统。

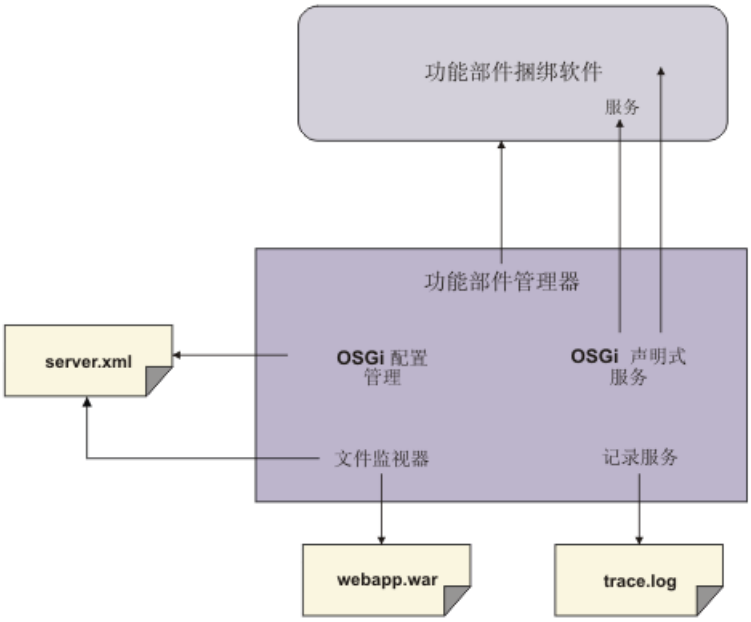


图 11: xigemaAS 概要文件内核

在系统配置文件（server.xml 文件及任何其他随附的文件）中指定功能部件。服务器配置文件会填充 OSGi 配置管理服务，而该服务会将功能部件配置插入功能部件管理器服务。功能部件管理器会将每个功能部件名称映射到提供该功能部件的捆绑软件列表。这些捆绑软件会安装到 OSGi 框架并启动。功能部件管理器会在服务器处于运行状态时通过动态地添加和移除功能部件，对配置更改作出响应。

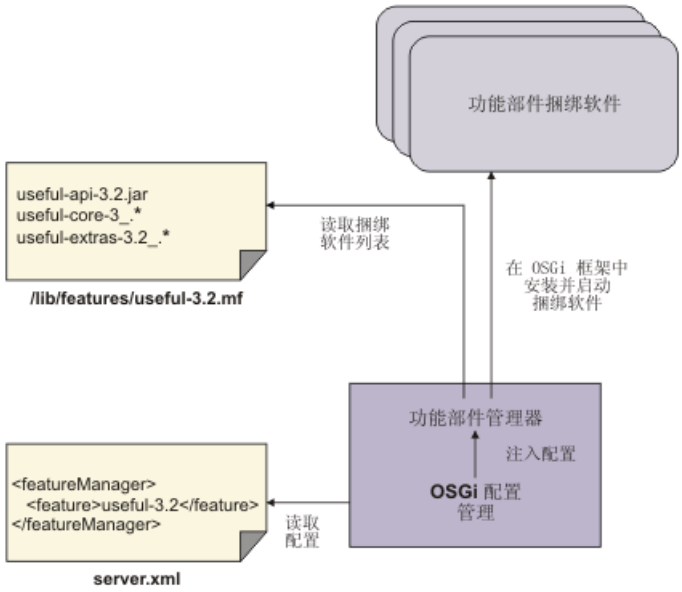


图 12: 功能部件管理

运行时服务会提供配置缺省设置，以便最大程度地减少您需要指定的配置。在 server.xml 文件中，指定您需要的功能部件，以及对系统缺省设置的任何新增项或覆盖项。可选择将您的配置构造成一些单独的文件，然后使用“include”语法将这些文件链接至父 server.xml 文件。服务器启动时，或者用户配置文件发生更改时，内核配置管理会解析您的配置并在系统缺省设置上应用您的配置。每次更新配置时，属于每项服务的配置属性集都会插入服务。

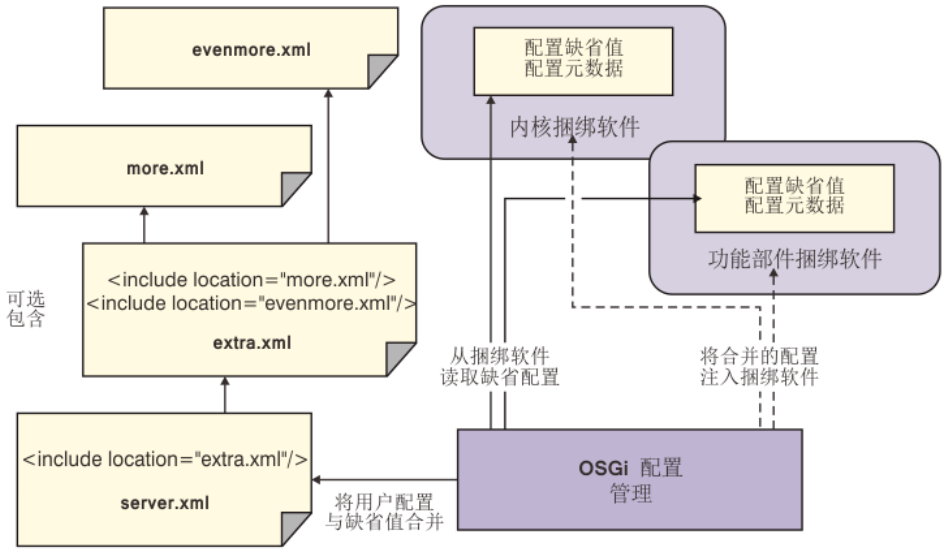


图 13: 配置管理

使用 OSGi 声明式服务组件，以便能将功能分解成独立的服务，这些服务可以按需激活。此行为有助于使运行时环境“迟钝一些”，从而保持较小的资源占用空间，让启动速度变快。会将声明的服务添加到 OSGi 服务注册表，并且可以解析服务之间的依赖性，而不需载入实现类。可以延迟服务激活操作，直至用到服务为

止：当解析服务引用时。每项服务的配置会在服务激活时插入，而且如果稍后修改了配置，那么会重新插入。

Java EE 7 编程模型支持

xigemaAS 遵循 Java™ Platform Enterprise Edition (Java EE) 7。Java EE 7 表和链接显示 xigemaAS 服务器对 Java EE 7 编程模型的支持程度。

Java EE 7 技术

表 2: Java EE 7 （概要文件类型支持）

Java™ EE 技术列表，针对以下各项分为若干部分：Web Service、Web 应用程序、企业应用程序、管理和安全性以及 Java™ SE 中与 Java™ EE 相关的规范。对于每种技术都有规范参考、任何相关 xigemaAS 功能部件以及xigemaAS 是否支持该技术的指示。

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java™ Platform Enterprise Edition 7 (Java EE 7)	*JSR 342*	*javaee-7.0* *javaeeClient-7.0*	✓
Java™ Platform Enterprise Edition 7 Web 概要文件	*JSR 342*	*webProfile-7.0*	✓
Web Service 技术			
Java™ API for RESTful Web Services (JAX-RS) 2.0	*JSR 339*	*jaxrs-2.0*	✓
实现企业 Web Service 1.4	*JSR 109*		✓
Java™ API for XML-Based Web Services (JAX-WS) 2.2	*JSR 224*	*jaxws-2.2*	✓
Web Service 互操作性组织 (WS-I) 基本概要文件	*WS-I 基本概要文件 1.2* *WS-I 基本概要文件 2.0*	*jaxws-2.2*	✓
Java™ XML 绑定体系结构 (JAXB) 2.2	*JSR 222*	*jaxb-2.2*	✓
用于 Java™ 平台的 Web Service 元数据	*JSR 181*		✓
Java™ API for XML-based RPC (JAX-RPC) 1.1（可选）	*JSR 101*		
Java™ API for WSDL (JWSDL)	*JSR 110*		✓
针对 XML 消息传递的 Java™ API 1.3（可选）	*JSR 67*		
带附件的 SOAP Java API (SAAJ) 1.3	*JSR 67*		✓

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
针对 XML 注册表的 Java™ API (JAXR) 1.0（可选）	*JSR 93*		
Web 应用程序技术			
针对 JSON 处理的 Java API (JSON-P) 1.0	*JSR 353*	*jsonp-1.0*	✓
Java™ Servlet 3.1	*JSR 340*	*servlet-3.1*	✓
JavaServer Faces (JSF) 2.2	*JSR 344*	*jsf-2.2*	✓
JavaServer Pages 2.3	*JSR 245*	*jsp-2.3*	✓
Expression Language (JSP/EL) 3.0	*JSR 341*	*el-3.0*	✓
JavaServer Pages 标准标记库 (JSTL) 1.2	*JSR 52*		✓
对其他语言的调试支持 1.0	*JSR 45*		✓
WebSocket 1.1	*JSR 356*	*websocket-1.1*	✓
WebSocket 1.0	*JSR 356*	*websocket-1.0*	✓
企业应用程序技术			
EE 并行实用程序 1.0	*JSR 236*	*concurrent-1.0*	✓
Java™ 上下文和依赖性注入 (Web Beans) 1.2	*JSR 346*	*cdi-1.2*	✓
Java™ 上下文和依赖性注入 (Web Beans) 1.1	*JSR 346*	*cdi-1.2*[1]	✓
Java™ 依赖性注入 1.0	*JSR 330*		✓
Bean 验证 1.1	*JSR 349*	*beanValidation-1.1*	✓
Enterprise JavaBeans™ (EJB) 3.2 完整版	*JSR 345*	*ejb-3.2*[2]	✓
Enterprise JavaBeans (EJB) 3.2 Lite	*JSR 345*	*ejbLite-3.2*	✓
Interceptors 1.2	*JSR 318*		✓
Java™ EE 连接器体系结构 (JCA) 1.7	*JSR 322*	*jca-1.7*	✓
Java™ 持久性 2.1	*JSR 338*	*jpa-2.1*	✓

1 Java EE 7 定义 CDI 1.1。CDI 维护版 CDI 1.2。`cdi-1.2` 功能部件同时支持 CDI 1.1 和 CDI 1.2。
2 *ejb-3.2* 功能部件包含以下 EJB 子功能部件：*ejbLite-3.2*、*ejbHome-3.2*、*ejbPersistentTimer-3.2*、*ejbRemote-3.2* 和 *mdb-3.2*。

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java™ 平台的常用注解 1.2	*JSR 250*		✓
Java™ 消息服务 (JMS) API 2.0	*JSR 343*	*jms-2.0*	✓
Java™ 事务 API (JTA) 1.2	*JSR 907*		✓
JavaMail™ 1.5	*JSR 919*	*javaMail-1.5*	✓
Java 平台的批处理应用程序 1.0	*JSR 352*	*batch-1.0*	✓
管理和安全性技术			
Java™ 容器认证服务提供者接口 (JASPIC) 1.1	*JSR 196*	*jaspic-1.1*	✓
Java™ 容器授权合同 (JACC) 1.5	*JSR 115*	*jacc-1.5*	✓
Java™ EE 应用程序部署 1.2（可选）	*JSR 88*		
J2EE Management 1.1[4]	*JSR 77*	*j2eeManagement-1.1*	✓
Java SE 中与 **Java EE** 相关的规范			
针对 XML 处理的 Java™ API (JAXP) 1.4	*JSR 206*		✓
Java™ 数据库连接 (JDBC) 4.1	*JSR 221*	*jdbc-4.1*	✓
Java™ 管理扩展 (JMX) 2.0	*JSR 255*		✓
JavaBeans™ 激活框架 (JAF) 1.1	*JSR 925*		✓
针对 XML 的流式 API (StAX) 1.0	*JSR 173*		✓

Java EE 6 编程模型支持

Java™ EE 6 表和链接显示 xigemaAS 对 Java™ EE 6 编程模型的支持程度。

Java EE 6 技术

表 3: Java EE 6 支持（按概要文件列示）

Java EE 技术列表，针对以下各项分为若干部分：Web Service、Web 应用程序、企业应用程序、管理和安全性以及 Java SE 中与 Java EE 相关的规范。对于每种技术都有规范参考、任何相关 xigemaAS 功能部件以及 xigemaAS 是否支持该技术的指示。

[3] 常用注解 1.2 添加了 `javax.annotation.Priority` 单一注解类型，上下文和依赖性注入 1.2 使用此类型。有关 CDI 1.2 的信息，请参阅 *Contexts and Dependency Injection 1.2*（见第 943 页）。

[4] 要调用管理 EJB API，服务器配置必须在功能部件管理器中同时具有 `j2eeManagement-1.1` 和 `ejbRemote-3.2` 功能部件。如果这两个功能部件都已包含在服务器配置中，那么您可通过 JNDI 名称查找来调用管理 EJB API。管理 EJB 绑定名称（JNDI 查找名称）为 `ejb/mejb/MEJB`。

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java Platform Enterprise Edition 6 (Java EE 6)	*JSR 316*		
Java Platform Enterprise Edition 6 Web 概要文件	*JSR 316*	*webProfile-6.0*	✓
Web Service 技术			
Java API for RESTful Web Service s (JAX-RS) 1.1	*JSR 311*	*jaxrs-1.1*	✓
实现企业 Web Service 1.4	*JSR 109*		✓
Java API for XML-Based Web Serv ices (JAX-WS) 2.2	*JSR 224*	*jaxws-2.2*	✓
Web Service 互操作性组织 (WS-I) 基本概要文件	*WS-I 基本概要文件 1.2* *WS-I 基本概要文件 2.0*	*jaxws-2.2*	✓
Java XML 绑定体系结构 (JAXB) 2.2	*JSR 222*	*jaxb-2.2*	✓
用于 Java 平台的 Web Service 元数据	*JSR 181*		✓
Java API for XML-based RPC (JA X-RPC) 1.1	*JSR 101*		
Java API for WSDL (JWSDL)	*JSR 110*		✓
针对 XML 消息传递的 Java API 1 .3	*JSR 67*		
带附件的 SOAP Java API (SAAJ) 1.3	JSR 67		✓
Java API for XML Registries (JAX R) 1.0	*JSR 93*		
Web 应用程序技术			
Java Servlet 3.0	*JSR 315*	*servlet-3.0*	✓
JavaServer Faces (JSF) 2.0	*JSR 314*	*jsf-2.0*	✓
JavaServer Pages 2.2/Expression La nguage (JSP/EL) 2.2	*JSR 245*	*jsp-2.2*	✓
JavaServer Pages 标准标记库 (JST L) 1.2	*JSR 52*		✓
对其他语言的调试支持 1.0	*JSR 45*		✓
企业应用程序技术			

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java 上下文和依赖性注入 (Web Beans 1.0)	*JSR 299*	*cdi-1.0*	✓
Java 依赖性注入 1.0	*JSR 330*		✓
Bean 验证 1.0	*JSR 303*	*beanValidation-1.0*	✓
Enterprise JavaBeans™ (EJB) 3.1（包含 Interceptors 1.1）	*JSR 318*	*ejbLite-3.1*	✓[5]
Java EE 连接器体系结构 1.6	*JSR 322*	*jca-1.6*	✓
Java 持久性 2.0	*JSR 317*		✓
Java 平台的常用注解 1.1	*JSR 250*		✓
Java 消息服务 (JMS) API 1.1	*JSR 914*	*jms-1.1*	✓
Java 事务 API (JTA) 1.1	*JSR 907*		✓
JavaMail™ 1.4	*JSR 919*		
管理和安全性技术			
Java 容器认证服务提供者接口 (JASPIC)	*JSR 196*		
Java 容器授权合同 (JACC) 1.3	*JSR 115*		
Java EE 应用程序部署 1.2	*JSR 88*		
J2EE 管理 1.1	*JSR 77*		
Java SE 中与 **Java EE** 相关的规范			
Java API for XML Processing (JAXP) 1.4	*JSR 206*		✓
Java 数据库连接 (JDBC) 4.0	*JSR 221*	*jdbc-4.0*	✓
Java 管理扩展 (JMX) 2.0	*JSR 255*		✓
JavaBeans 激活框架 (JAF) 1.1	*JSR 925*		✓
针对 XML 的流式 API (StAX) 1.0	*JSR 173*		✓

受支持的 Java EE 6 与 Java EE 7 功能部件组合

服务器配置中的一些 Java™ EE 7 与 Java EE 6 xigemaAS 功能部件组合是兼容的。但是，许多组合不兼容，服务器启动时会导致错误。

错误消息类似如下所示：

[5] xigemaAS 仅支持 EJB Lite 子集和消息驱动的 Bean。请参阅 *xigemaAS 功能部件*（见第 906 页） 的“Enterprise JavaBeans (EJB) Lite 子集”一节。

```
CWWKF0033E: 无法同时装入单例功能部件
com.ibm.websphere.appserver.javaeeCompatible-6.0 和
 com.ibm.websphere.appserver.javaeeCompatible-7.0。
所配置功能部件 servlet-3.0 和 ejbLite-3.2 包含导致该冲突的一个或多个功能部件。您的配置不
受支持。
```

下表使用复选标记 (✓) 来标记兼容功能部件组合。确保您的服务器配置未包含不兼容功能部件。

表 4: 受支持的 Java EE 7 与 Java EE 6 xigemaAS 功能部件组合

Java EE 7 功能部件纵向列示。Java EE 6 功能部件横向列示。复选标记 (✓) 指示 Java EE 7 与 Java EE 6 功能部件的组合受支持，并且服务器配置可同时包含这两种功能部件。空单元格（没有 ✓） 指示 Java EE 7 与 Java EE 6 功能部件的组合不受支持。

一个表，此表纵向列示 Java EE 7 功能部件，横向列示 Java EE 6 功能部件。复选标记 (✓) 指示该列中的 Java EE 7 功能部件与该行中的 Java EE 6 功能部件兼容。空白（没有复选标记）指示该 Java EE 7 功能部件与该 Java EE 6 功能部件不兼容。请不要将不兼容的功能部件放置在服务器配置中。

	Java EE 6 功能部件														
	beanValida - tion-1.0	**cdi-1.0**	**ejb Lite -3.1**	**jaxb-2.2**	**jaxrs -1.1**	**jaxws-2.2**	**jca-1.6**	**jdbc-4.0**	**jms-1.1**	**jpa-2.0**	**jsf-2.0**	**jsp-2.2**	**managed-Beans-1.0**	**mdb-3.1**	**servlet-3.0**
Java EE 7 功能部件															
batch-1.0				✓		✓		✓		✓			✓		
beanValidation-1.1				✓		✓		✓		✓			✓		
cdi-1.2				✓		✓		✓		✓			✓		
concurrent-1.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
el-3.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
ejb-3.2				✓		✓				✓			✓		
ejbLite-3.2				✓		✓		✓		✓			✓		
javaMail-1.5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
jacc-1.5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
jaspic1.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
jaxrs-2.0				✓		✓		✓		✓			✓		
jca-1.7				✓		✓		✓		✓			✓		
jdbc-4.1	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
jms-2.0				✓		✓		✓		✓			✓		
jpa-2.1				✓		✓							✓		
jsf-2.2				✓		✓		✓		✓			✓		
jsonp-1.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
jsp-2.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	
mdb-3.2				✓		✓		✓		✓			✓		
servlet-3.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
websocket-1.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
websocket-1.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

该表显示 Java EE 6 的 `servlet-3.0` 功能部件与 Java EE 7 的 `websocket-1.1` 功能部件不兼容。因此，带有以下功能部件的服务器配置会导致错误：

```
<featureManager>
   <feature>servlet-3.0</feature>
   <feature>websocket-1.1</feature>
</featureManager>
```

要解决此错误，请在服务器配置中使用 `servlet-3.1` 而不是 `servlet-3.0`。`servlet-3.1` 功能部件与 `websocket-1.1` 功能部件兼容。

有关功能部件的兼容性或容许性的更多信息，请参阅容许功能部件。

Java EE 7 行为更改

如果先前已在 xigemaAS 概要文件环境中合并了 Java Platform Enterprise Edition (Java EE) 6 功能部件，那么可在移至 Java EE 7 功能部件时遇到行为更改。

对于每个服务器实例，可在 Java EE 6 与 Java EE 7 功能部件实现之间进行选择并考虑行为更改。如果只有 Java EE 7 功能部件包含所需行为，那么必须使用 Java EE 7 功能部件。如果 Java EE 7 功能部件中的行为更改对现有应用程序可能产生负面影响，那么应使用 Java EE 6 功能部件以保留该应用程序的现有行为。必须确保您选择的 Java EE 实现与服务器中的其他 Java EE 功能部件兼容；有关更多信息，请参阅 受支持的 *Java EE 6* 与 *Java EE 7* 功能部件组合（见第 32 页）。

表 5: 具有 Java EE 6 和 Java EE 7 实现的功能部件

包含四列的表，用于列示 Java EE 技术、Java EE 6 功能部件、Java EE 7 功能部件及从 Java EE 6 功能部件迁移至 Java EE 7 功能部件时应用的所有行为更改。

技术	**Java EE 6** 功能部件	**Java EE 7** 功能部件	行为更改
Bean Validation	*beanValidation-1.0*	*beanValidation-1.1*	没有行为更改。
Java 上下文和依赖性注入 (CDI)	*cdi-1.0*	*cdi-1.2*	请参阅上下文和依赖性注入 *1.2* 行为更改（见第 1223 页）。
Enterprise JavaBeans (EJB)	*ejbLite-3.1*	*ejbLite-3.2*	没有行为更改。
Expression Language (EL)	包含在 *jsp-2.2* 中	*el-3.0*	请参阅 *Expression Language (EL) 3.0* 功能部件的功能（见第 1220 页）。
Java API for RESTful Web Services (JAX-RS)	*jaxrs-1.1*	*jaxrs-2.0*	请参阅 *JAX-RS 2.0* 行为更改（见第 1519 页）。
Java EE Connector Architecture (JCA)	*jca-1.6*	*jca-1.7*	没有行为更改。
Java Database Connectivity (JDBC)	*jdbc-4.0*	*jdbc-4.1*	没有行为更改。
Java Message Service (JMS)	*jms-1.1*	*jms-2.0*	没有行为更改。
	wasJmsClient-1.1	*wasJmsClient-2.0*	请参阅 *JMS* 消息传递 *(wasJmsClient-2.0)* 行为更改（见第 1067 页）。
	wmqJmsClient-1.1	*wmqJmsClient-2.0*	请参阅*JMS 1.1 Client for xigemaMQ*（见第 954 页） 以及 *JMS 2.0 Client for xigemaMQ*（见第 956 页）。

技术	**Java EE 6** 功能部件	**Java EE 7** 功能部件	行为更改
Java Persistence API (JPA)	*jpa-2.0*	*jpa-2.1*	请参阅 *Java Persistence API 2.1 行为更改*（见第 1070 页）。
Java Servlet	*servlet-3.0*	*servlet-3.1*	请参阅 *Servlet 3.1 行为更改*（见第 1212 页）。
Java Transaction API (JTA)	transaction-1.1（受保护的功能部件）	transaction-1.2（受保护的功能部件）	没有行为更改。
JavaServer Faces (JSF)	*jsf-2.0*	*jsf-2.2*	请参阅配置 *xigemaAS* 以使用 *JavaServer Faces 2.2*（见第 1220 页）。
JavaServer Pages (JSP)	*jsp-2.2*	*jsp-2.3*	没有行为更改。
Message Driven Beans (MDB)	*mdb-3.1*	*mdb-3.2*	没有行为更改。

企业 OSGi 编程模型支持

企业 OSGi 表和链接显示 xigemaAS 概要文件对企业 OSGi 编程模型的支持程度。

企业 OSGi 技术

表 6: 企业 OSGi 的概要文件支持

企业 OSGi 技术列表，针对 Blueprint、Web 以及其他企业技术划分为若干部分。每种技术都包含规范参考及有关 xigemaAS 是否支持该技术的指示。

此表列示企业 OSGi 技术，这些技术划分为下列各个部分：Blueprint、Web 以及其他企业技术。每种技术都包含规范参考及 xigemaAS 概要文件是否支持该技术的指示。

技术	规范参考	**xigemaAS**
与 **Blueprint** 相关的技术		
Blueprint Container	R4.2 企业章节 121	✓
Blueprint 事务		✓
Blueprint 管理的 JPA		✓
Blueprint 安全性		
Blueprint 资源引用		
定制 Blueprint 命名空间		✓
与 **Web** 相关的资源		
Web 应用程序捆绑软件	R4.2 企业章节 128	✓
WebSocket 应用程序		✓

技术	规范参考	**xigemaAS**
JNDI	R4.2 企业章节 126	✓
JSP		✓
JSTL		✓
JSF		✓
JAX-RS		✓
其他企业技术		
EJB 捆绑软件		
远程服务	R4.2 概要章节 13	
SCA 配置类型规范	R4.2 企业章节 129	
远程捆绑软件存储库		✓
SIP		
本地 OSGi 应用程序集成		✓

👉 注：WebSockets 当前仅在 xigemaAS 上受支持。

xigemaAS 外部支持

xigemaAS 的外部函数和资源可以直接使用，而且在下一个发行版中可以继续沿用。当您应用服务或升级到未来发行版时，概要文件的内部或附带特征可能会改变。

我可以直接使用 xigemaAS 中的哪些内容并在下一个发行版中依赖这些内容?
下列资源可以直接使用，而且在下一个发行版中仍可以使用：

- ${wlp.install.dir}/dev 目录中的 JAR 文件内容定义的应用程序编程接口 (API) 和系统编程接口 (SPI)。
  - 应用程序类装入器可以看见由服务器配置中的功能部件所提供的 API。产品扩展功能部件可以看见服务器配置中的功能部件所提供的所有 API 和 SPI。
  - 针对 ${wlp.install.dir}/dev 目录中的 JAR 文件编译代码。在 ${wlp.install.dir}/dev 目录中提供 JAR 文件只是为了编译应用程序和功能部件，系统不支持将它们用于运行时。不要在应用程序、库或测试中使用这些 JAR 文件。
- 服务器配置，包括具有 *public* 或 *protected* 可视性的功能部件。可在 server.xml 文件和所包含文件中指定公共功能部件和配置元素；受保护功能部件可包括在您自己的功能部件中。
- ${wlp.install.dir}/bin 目录和子目录中的命令、脚本和归档。
- ${wlp.install.dir}/clients 目录和子目录中的客户机实用程序。

我应该避免依赖哪些内容?

不要构建对产品附带方面的依赖性，否则在应用服务或升级到未来发行版时会受到影响。您应该避免依赖的产品内部内容示例包括但不限于下列情况：

- 产品二进制 JAR 文件的名称，例如，${wlp.install.dir}/dev 目录中的那些 JAR 文件的名称。使用工具或 `javac -extdirs` 选项针对这些 JAR 文件来编译代码。

如果您要使用 Apache Ant 编译代码，请使用通配符以避免与特定 JAR 版本产生依赖关系；例如：

```
<fileset dir="${wlp.install.dir}/dev/api/spec" includes="com.ibm.ws.javaee.servlet.3.0_*.jar"/>
```

请参阅 覆盖 *Java SDK* 中的类（见第 1670 页）。

- 直接使用 ${wlp.install.dir}/lib 目录中的产品二进制文件。可以直接调用的 JAR 文件仅位于 ${wlp.install.dir}/bin/tools 目录中。
- 由服务器在运行时输出的消息。消息的文本及插入会随服务和版本的升级而改变。只要实践上可能，产品将在特定操作点输出的消息标识方面保持一致，但这无法保证，因为底层实现可能会改变。
- 产品安装的布局，而不是 ${wlp.install.dir}/bin 和 ${wlp.install.dir}/dev 目录。
- ${wlp.install.dir}/templates 目录中的示例和模板文件。将服务应用到安装时可修改这些文件。
- 未显式提供为 API 的专用或第三方 Java™ 包。这些对于运行时的应用程序类加载器不可见。
- 不要对服务器输出的自动处理使用 console.log 文件。请改用 messages.log 文件访问和处理消息，这允许以易于处理的格式提供更多详细信息。

应用服务或升级时可修改哪些内容?

应用服务或升级时可修改下列目录及其子目录的内容。不要自行修改下列位置中的文件，否则它们可能会为产品维护或升级所覆盖：

- ${wlp.install.dir}/bin
- ${wlp.install.dir}/clients
- ${wlp.install.dir}/dev
- ${wlp.install.dir}/java
- ${wlp.install.dir}/lib
- ${wlp.install.dir}/templates

下列目录的内容不会加以修改。这些是您的文件，应用服务或升级将不会修改这些文件：

- ${wlp.install.dir}/etc（您可能在此目录中添加了 server.env 或 jvm.options 文件）。
- ${wlp.install.dir}/usr（用户配置和应用程序的缺省位置）。
- 通过 `WLP_USER_DIR` 环境变量指定的任何非缺省目录。

第三方 API 可能随时间的推移而更改，但未考虑向后兼容性问题。这些是 Java™ 包，视为开放式源代码社区所开发功能部件实现的一部分，且作为 xigemaAS 的一部分来提供。第三方 API 在缺省情况下对于应用程序不可见；具有显式允许第三方访问的类加载器配置的 Java™ EE 应用程序将可以访问应用程序类加载器上的那些包，并且 OSGi 应用程序必须显式导入那些包。决定使用第三方 API 之前，请考虑不兼容更改的影响。

2.1.3 服务器配置

根据异常来配置 xigemaAS 。运行时环境会从一组内置配置缺省设置进行操作，并且您只需要指定用来覆盖那些缺省设置的配置。如果要这样做，请在运行时编辑 server.xml 文件或者 server.xml 中包括的另一个 XML 文件。

配置具有下列特征：

- 在 XML 文件中描述。
- 符合人类阅读习惯，可在文本编辑器中编辑。
- 较小、易于备份及易于复制到另一个系统。

- 可以在应用程序开发团队之间共享。
- 可组合，因此功能部件能轻松地将其自己的配置添加到系统。
- 可扩展的类型化，因此您不必修改当前配置，即可与更高版本的运行时环境一起使用。
- 动态地响应更新。
- 更宽容，因此会假定缺少的值并忽略无法识别的属性。

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。 它们是使服务器可组合的主机制。您在服务器配置中指定的功能部件列表会提供功能性服务器。 请参阅 *xigemaAS* 功能部件（见第 906 页）。

第一次安装并启动服务器时，有功能部件管理器及缺省服务器配置可供使用：

- 缺省情况下，服务器包含 `jsp-2.2` 功能部件来支持 servlet 和 JSP 应用程序。可以使用功能部件管理器来添加所需要的功能部件。
- 根据异常来配置服务器。指定所需要的功能部件时，那些功能部件的缺省配置会提供一个设计成涵盖最常见需求的丰富环境，因此您只需要指定对缺省配置的更改。

您也可以使用 bootstrap.properties 文件来指定在处理主要配置之前需要的属性，以及定义主要配置中使用的变量。

有关配置文件的完整列表，请参阅 目录位置和属性。

服务作者透视图：配置的运行时管理

xigemaAS 配置服务会解析主 server.xml 文件及其包含的任何文件，以及 configDropins 目录中的配置文件，将内容合并到已安装捆绑软件所提供的缺省配置值中，然后将产生的属性集提供给 OSGi 配置管理服务 (CA)。CA 会将每个属性集插入到拥有该属性集的服务（如果已向 CA 注册该服务）。

这些步骤的排序很灵活。服务可以在建立初始属性集之前或之后向 CA 注册。可以在初始注入之后，也就是在将更新的属性插入到拥有服务之后，在 CA 中更新属性。因此，只要服务处于活动状态，就一定要能接收及相应地响应其配置更新。具体地说，如果服务延迟服务激活，直到服务配置可用，那么服务必须仍能够激活。

涉及一些步骤以使服务能接收配置数据。 请参阅启用服务来接收配置数据（见第 1249 页）。

激活规范 (activationSpec)

定义激活规范配置。

- *authData*

属性名称	数据类型	缺省值	描述
authDataRef	对顶级 authData 元素的引用（字符串）。		激活规范的缺省认证数据。
id	字符串		唯一配置标识。
maxEndpoints	整型 最小值：0	500	要分派至的最大端点数。

authData

激活规范的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

Microsoft Active Directory LDAP 过滤器 (actived LdapFilterProperties)

指定缺省 Microsoft Active Directory LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectcategory=group))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	memberOf:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(sAMAccountName=%v)(objectcategory=user))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	user:sAMAccountName	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

受管对象 (adminObject)

定义受管对象配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。

管理员角色 (administrator-role)

分配有服务器管理员角色的用户和/或组的集合。

- *group*
- *user*

group

分配有角色的组。

false

字符串

user

分配有角色的用户。

false

字符串

API 发现 (apiDiscovery)

用于描述 REST API 的 API 发现功能部件的配置。

- *webModuleDoc*

属性名称	数据类型	缺省值	描述
apiName	字符串		聚集 API 的名称。
id	字符串		唯一配置标识。
maxSubscriptions	整型 最小值：0 最大值：100	20	指定正在侦听更新的并行 REST API 客户机的最大数。

webModuleDoc

提供要展示的 API 文档的每个 Web 模块的配置。

false

属性名称	数据类型	缺省值	描述
contextRoot	字符串		您要为其提供文档的 Web 模块的上下文根。
docURL	字符串		此 Web 模块的文档的 URL。此 URL 可以是上下文根的相对 URL（通过以正斜杠 (/) 开头），也可以是绝对 URL（以 http 或 https 开头）。

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	一个布尔值，用于控制此 Web 模块的文档的处理。
id	字符串		唯一配置标识。

应用程序 (application)

定义应用程序的属性。

- *application-bnd*
  - *security-role*
    - *group*
    - *run-as*
    - *special-subject*
    - *user*
- *classloader*
  - *commonLibrary*
    - *file*
    - *fileset*
    - *folder*
  - *privateLibrary*
    - *file*
    - *fileset*
    - *folder*
- *resourceAdapter*
  - *contextService*
    - *baseContext*
      - *baseContext*
      - *classloaderContext*
      - *jeeMetadataContext*
      - *securityContext*
      - *syncToOSThreadContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *customize*

属性名称	数据类型	缺省值	描述
autoStart	布尔型	true	指示服务器是否自动启动应用程序。
context-root	字符串		应用程序的上下文根。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
location	文件、目录或 URL。		应用程序的位置，表示为绝对路径或相对于服务器级应用程序目录的路径。
name	字符串		应用程序的名称。
suppressUncoveredHttpMethodWarning	布尔型	false	指示在应用程序部署期间阻止未覆盖 HTTP 方法警告消息的选项。
type	字符串		应用程序归档的类型。

application-bnd

将应用程序中包括的常规部署信息绑定到特定资源。

false

属性名称	数据类型	缺省值	描述
version	字符串		应用程序绑定规范的版本。

application-bnd > security-role

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		安全角色的名称。

application-bnd > security-role > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		组访问标识
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

application-bnd > security-role > run-as

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
password	可逆向编码的密码（字符串）		从一个 Bean 访问另一个 Bean 时需要的用户密码。可以采用明文或编码格式存储该值。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
userid	字符串		从一个 Bean 访问另一个 Bean 时需要的用户标识。

application-bnd > security-role > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。 **EVERYONE** Everyone **ALL_AUTHENTICATED_USERS** 所有已认证的用户

application-bnd > security-role > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

classloader

定义应用程序类装入器的设置。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
classProviderRef	顶级 resourceAdapter 元素的引用列表（以逗号分隔的字符串）。		类提供程序引用的列表。搜索类或资源时，此类装入器将在搜索其自己的类路径之后授权给指定的类提供程序。
commonLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。会与其他类装入器共享库类实例。
delegation	• parentFirst • parentLast	parentFirst	控制父类装入器是用在此类装入器之前还是之后。如果选择“父代最先”，那么在搜索类路径之前，授权给直接父代。如果选择“父代最后”，那么在授权给直接父代之前，搜索类路径。 **parentFirst** 父代最先 **parentLast** 父代最后
privateLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。库类实例是此类装入器特有的，与来自其他类

属性名称	数据类型	缺省值	描述
			装入器的类实例无关。
serializablePattern	字符串	NULL	为符合条件的 java bean 增加序列化接口。多个正则表达式使用‘%%’ 分隔，例如 ^com.test.\w+? %% ^com.test.\w+?。
serializableLocation	字符串	NULL	为符合条件的 java bean 增加序列化接口。serializableLocation 的值为文件路径，此文件中的每一行为一个 java 类路径。例如：com.test.Test。

classloader > commonLibrary

库引用的列表。会与其他类装入器共享库类实例。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > commonLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > commonLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条

属性名称	数据类型	缺省值	描述
			目中。例如，1s500ms 相当于 1.5 秒。

classloader > commonLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

classloader > privateLibrary

库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > privateLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > privateLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > privateLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

resourceAdapter

为嵌入在应用程序中的资源适配器指定配置。

false

属性名称	数据类型	缺省值	描述
alias	字符串	${id}	覆盖资源适配器的缺省标识。该标识用于资源适配器的配置属性元素的名称中，该名称转而用于为由资源适配器提供的任何资源确定配置属性元素的名称。资源适配器的配置属性元素名称具有 properties.<APP_NAME>.<ALIAS> 格式，其中 <APP_NAME> 是应用程序的名称，<ALIAS> 是所配置别名。如果未指定，那么别名缺省为资源适配器的模块名称。
autoStart	布尔型		配置资源适配器的启动方式是在其部署时自动启动，还是在注入或查找资源时缓慢启动。
contextServiceRef	对顶级 contextService 元素的引用（字符串）。		配置上下文捕获及传播至线程的方式。
id	字符串		确定此配置适用的嵌入式资源适配器模块的名称。

resourceAdapter > contextService

配置上下文捕获及传播至线程的方式。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

resourceAdapter > contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

resourceAdapter > contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

resourceAdapter > contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

resourceAdapter > contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

resourceAdapter > contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

resourceAdapter > contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

resourceAdapter > contextService > classloaderContext

类装入器上下文传播配置。

false

resourceAdapter > contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

resourceAdapter > contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

resourceAdapter > contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

resourceAdapter > customize

定制具有指定接口和/或实现类的激活规范、受管对象或连接工厂的配置属性元素。

false

属性名称	数据类型	缺省值	描述
implementation	字符串		标准实现类名，应该针对该类名定制配置属性元素。
interface	字符串		标准接口类名，应该针对该类名定制配置属性元素。
suffix	字符串		覆盖配置属性元素的缺省后缀。例如，properties.rarModule1.C

属性名称	数据类型	缺省值	描述
			ustomConnectionFactory 中的“CustomConnectionFactory”。当资源适配器提供了多种类型的连接工厂、受管对象或端点激活时，该后缀对于进行区别很有用。如果配置属性元素定制省略该后缀或将它保留为空白，那么不会使用后缀。

应用程序管理器 (applicationManager)

用于控制应用程序管理器的行为的属性

属性名称	数据类型	缺省值	描述
autoExpand	布尔型	false	允许自动解压 WAR 文件和 EAR 文件
startTimeout	精度为秒的时间段	30s	指定服务器在将应用程序视为缓慢之前等待该应用程序启动的时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

应用程序监视 (applicationMonitor)

定义服务器对应用程序新增项、更新项和删除项作出响应的方式。

属性名称	数据类型	缺省值	描述
dropins	目录路径	dropins	应用程序混入目录的位置，表示为绝对路径或相对于服务器目录的路径。
dropinsEnabled	布尔型	true	监视混入目录中的应用程序新增项、更新项和删除项。
pollingRate	具有毫秒精度的时间段	500ms	服务器检查应用程序新增项、更新项和删除项的频

属性名称	数据类型	缺省值	描述
			率。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
updateTrigger	• mbean • polled • disabled	polled	应用程序更新方法或触发器。 **mbean** 仅当接收到来自外部程序（例如集成开发环境或管理应用程序）所调用的 MBean 的提示时，服务器才会更新应用程序。 **polled** 服务器将按照轮询时间间隔对应用程序进行扫描以检测更改，并更新具有可检测到的更改的任何应用程序。 **disabled** 禁用所有更新监视。在服务器处于运行状态时，应用程序更改将不会加以应用。

认证高速缓存 (authCache)

控制认证高速缓存的操作。

属性名称	数据类型	缺省值	描述
allowBasicAuthLookup	布尔型	true	允许按照用户标识和散列密码进行查找。
initialSize	整型 最小值：1	50	受认证高速缓存支持的初始条目数。

属性名称	数据类型	缺省值	描述
maxSize	整型 最小值：1	25000	受认证高速缓存支持的最大条目数。
timeout	具有毫秒精度的时间段	600s	在其后将移除高速缓存中的条目的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

认证数据 (authData)

企业信息系统 (EIS) 或数据库的连接的认证别名。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

认证过滤器 (authFilter)

指定表示条件的选择规则，HTTP 请求头必须与这些条件匹配才能确定是否对认证选择该 HTTP 请求。

- *host*
- *remoteAddress*
- *requestUrl*
- *userAgent*
- *webApp*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

host

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

remoteAddress

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ip	字符串		指定 IP 地址。
matchType	• lessThan • equals • greaterThan • contains • notContain	contains	指定匹配类型。 **lessThan** 小于 **equals** 等于 **greaterThan** 大于 **contains** 包含 **notContain** 不包含

requestUrl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
urlPattern	字符串		指定 URL 模式。

userAgent

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
agent	字符串		指定用户代理
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含

webApp

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于

属性名称	数据类型	缺省值	描述
			contains 包含 **notContain** 不包含
name	字符串		指定名称。

认证 (authentication)

控制内置认证服务配置。

属性名称	数据类型	缺省值	描述
allowHashtableLoginWithIdOnly	布尔型	false	允许应用程序仅使用散列表属性中的身份进行登录。仅当您具有需要此选项的应用程序并且具有其他方法来验证该身份时，才使用此选项。
cacheEnabled	布尔型	true	启用认证高速缓存。

功能部件授权角色映射 (authorization-roles)

用户、组或特殊主体集的角色的角色名称和映射的集合。

- *security-role*
  - *group*
  - *special-subject*
  - *user*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

security-role

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		角色名称。

security-role > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		以常规格式 group:realmName/groupUniqueId 表示的组访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

security-role > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。 **EVERYONE** 每个请求的所有用户（即使请求未得到认证）。 **ALL_AUTHENTICATED_USERS** 所有已认证的用户。

security-role > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
name	字符串		拥有安全角色的用户的名称。

基本用户注册表 (basicRegistry)

基于 XML 的简单用户注册表。

- *group*
  - *member*
- *user*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ignoreCaseForAuthentication	布尔型	false	允许不区分大小写的用户名认证。
领域	字符串	BasicRegistry	域名表示用户注册表。

group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		基本用户注册表中组的名称。

group > member

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		基本用户注册表组中用户的名称。

user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
name	字符串		基本用户注册表中用户的名称。
password	单向可散列或可逆向编码的密码（字符串）		基本用户注册表中用户的密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。

批处理 JMS 分派器 (batchJmsDispatcher)

配置批处理 JMS 分派器。

- *connectionFactory*
  - *connectionManager*
  - *containerAuthData*
  - *properties.wasJms*
  - *properties.wmqJms*
  - *recoveryAuthData*
- *queue*
  - *properties.wasJms*
  - *properties.wmqJms*

属性名称	数据类型	缺省值	描述
connectionFactoryRef	对顶级 jmsConnectionFactory 元素的引用（字符串）。	batchConnectionFactory	JMS 连接工厂的标识，批处理分派器应使用该连接工厂来获取 JMS 连接。
id	字符串		唯一配置标识。
queueRef	对顶级 jmsQueue 元素的引用（字符串）。	batchJobSubmissionQueue	JMS 队列的标识，批处理 JMS 分派器使用该队列来发送 JMS 消息。

connectionFactory

JMS 连接工厂的标识，批处理分派器应使用该连接工厂来获取 JMS 连接。

false

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。

属性名称	数据类型	缺省值	描述
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
jndiName	字符串		资源的 JNDI 名称。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

connectionFactory > connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。

属性名称	数据类型	缺省值	描述
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。

属性名称	数据类型	缺省值	描述
			EntirePool 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

connectionFactory > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

connectionFactory > properties.wasJms

JMS 连接工厂用于创建与 JMS 目标的相关 JMS 提供程序的连接，以便进行点到点消息传递和发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
clientID	字符串	clientID	所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
durableSubscriptionHome	字符串	defaultME	持久预订本地名称定义 ME 名称，需要与该 ME 名称建立连接。
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent

属性名称	数据类型	缺省值	描述
			ReliablePersistent ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default **AlwaysOn** AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
temporaryQueueNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此队列连接工厂的应用程序来创建的临时队列。
temporaryTopicNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此主题连接工厂的应

属性名称	数据类型	缺省值	描述
			用程序创建的临时主题。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

connectionFactory > properties.wmqJms

xigemaMQ JMS 连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		发送已发布消息的队列（流队列）的名称。
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1

属性名称	数据类型	缺省值	描述
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE **NONDUR** NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行。 **ENABLED** ENABLED

属性名称	数据类型	缺省值	描述
			已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。 **RLE** RLE

属性名称	数据类型	缺省值	描述
			NONE NONE
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省密码。（建议使用容器管理的认证别名，而不配置此属性）
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将

属性名称	数据类型	缺省值	描述
			被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
pubAckInterval	整型 最小值：0	25	xigemaMQ JMS 类请求代理程序的应答之前，发布程序发布的消息数
queueManager	字符串		要连接至的队列管理器的名称
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 5

属性名称	数据类型	缺省值	描述
			00ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序。
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略。
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSS

属性名称	数据类型	缺省值	描述
			EFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类是否存储有关活动预订的持久数据。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息的 JMSReplyTo 头字

属性名称	数据类型	缺省值	描述
			段标识的队列的应答消息才具有 MQRFH 2 头。
tempQPrefix	字符串		用于构成 xigemaMQ 动态队列名称的前缀。
tempTopicPrefix	字符串		创建临时主题时，JMS 会生成一个格式为 TEMP/TEMPTOPIC PREFIX/unique_id 的字符串，或者，如果让此属性保持为缺省值，那么只会生成格式为 TEMP/unique_id 的字符串。指定非空 TEMPTOPICPREFIX 允许定义特定模型队列，以便为在此连接下面创建的临时主题的订户创建受管队列。
temporaryModel	字符串		用来创建 JMS 临时队列的模型队列的名称。JMS 层可以使用 SYSTEM.JMS.TEMPQ.MODEL 来创建可接受持久消息的队列，而缺省值则不能。SYSTEM.DEFAULT.MODEL.QUEUE 只能打开一次。SYSTEM.JMS.TEMPQ.MODEL 可以多次打开。建议不要使用 SYSTEM.DEFAULT.MODEL.QUEUE。
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT

属性名称	数据类型	缺省值	描述
			BINDINGS BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

connectionFactory > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

queue

JMS 队列的标识，批处理 JMS 分派器使用该队列来发送 JMS 消息。

false

属性名称	数据类型	缺省值	描述
jndiName	字符串		资源的 JNDI 名称。

queue > properties.wasJms

此 JMS 队列分配至的队列的名称。

false

属性名称	数据类型	缺省值	描述
deliveryMode	• NonPersistent • 应用程序 • 持久	应用程序	发送至此目标的消息的传递方式。此选项控制此目标上的消息持久性。 **NonPersistent** NonPersistent 应用程序 应用程序 持久 持久
priority	整型 最小值：0 最大值：9		发送至此目标的消息的相对优先级，范围是 0（最低）到 9（最高）。
queueName	字符串	Default.Queue	关联队列的名称
readAhead	• AlwaysOff • AsConnection • AlwaysOn	AsConnection	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **AsConnection** AsConnection **AlwaysOn** AlwaysOn
timeToLive	精度为秒的时间段	0s	缺省时间（毫秒），从系统必须使消息在目标中保持活动的分派时间开始算起。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。

queue > properties.wmqJms

xigemaMQ JMS 队列

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	1208	要用于连接或目标的编码字符集标识
arbitraryProperties	字符串		能够指定其他位置未定义的属性
baseQueueManagerName	字符串		定义了此队列的队列管理器名称
baseQueueName	字符串		队列管理器上队列的名称
encode	字符串	NATIVE	将消息发送至此目标时，如何表示该消息的正文中的数字数据。该属性指定二进制整数、压缩十进制整数和浮点数的表示法。
expiry	字符串	APP	一个时间段，在该时间段后目标上的消息将到期
failIfQuiesce	布尔型	true	队列管理器处于停顿状态时，对某些方法的调用是否会失败。
persistence	• APP • QDEF • HIGH • NON • PERS	APP	发送至目标的消息的持久性 **APP** APP **QDEF** QDEF **HIGH** HIGH **NON** NON **PERS** PERS

属性名称	数据类型	缺省值	描述
priority	• 3 • 2 • 1 • APP • 0 • 7 • 6 • 5 • QDEF • 4 • 9 • 8	APP	发送至目标的消息的优先级 **3** 3 **2** 2 **1** 1 **APP** APP **0** 0 **7** 7 **6** 6 **5** 5 **QDEF** QDEF **4** 4 **9** 9 **8** 8
putAsyncAllowed	• ENABLED • DESTINATION • DISABLED	DESTINATION	是否允许消息生产者使用异步放入来将消息发送至此目标 **ENABLED** ENABLED **DESTINATION** DESTINATION

属性名称	数据类型	缺省值	描述
			已禁用 已禁用
readAheadAllowed	• ENABLED • DESTINATION • DISABLED	DESTINATION	是否允许 MDB 使用预先读取，以便在接收来自此目标的非持久消息之前将这些消息存储到内部缓冲区 **ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadClosePolicy	• CURRENT • ALL	ALL	当管理员停止 MDB 时，内部预先读取缓冲区中的消息发生的情况。 **CURRENT** CURRENT **ALL** ALL
receiveCCSID	整型 最小值：0		用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_QMG，否则会忽略此值
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG

属性名称	数据类型	缺省值	描述
targetClient	• JMS • MQ	JMS	是否使用 xigemaMQ RFH2 格式与目标应用程序交换信息 **JMS** JMS **MQ** MQ

批处理 JMS 事件 (batchJmsEvents)

配置批处理 JMS 事件。

- *connectionFactory*
  - *connectionManager*
  - *containerAuthData*
  - *properties.wasJms*
  - *properties.wmqJms*
  - *recoveryAuthData*

属性名称	数据类型	缺省值	描述
connectionFactoryRef	对顶级 jmsConnectionFactory 元素的引用（字符串）。	batchConnectionFactory	JMS 连接工厂的标识。
id	字符串		唯一配置标识。

connectionFactory

JMS 连接工厂的标识。

false

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
jndiName	字符串		资源的 JNDI 名称。

属性名称	数据类型	缺省值	描述
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

connectionFactory > connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)

属性名称	数据类型	缺省值	描述
			、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。

属性名称	数据类型	缺省值	描述
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

connectionFactory > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

connectionFactory > properties.wasJms

JMS 连接工厂用于创建与 JMS 目标的相关 JMS 提供程序的连接，以便进行点到点消息传递和发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
clientID	字符串	clientID	所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消

属性名称	数据类型	缺省值	描述
			息传递，那么需要此标识。
durableSubscriptionHome	字符串	defaultME	持久预订本地名称定义 ME 名称，需要与该 ME 名称建立连接。
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent **ReliablePersistent** ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。

属性名称	数据类型	缺省值	描述
			AlwaysOff AlwaysOff **Default** Default **AlwaysOn** AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
temporaryQueueNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此队列连接工厂的应用程序来创建的临时队列。
temporaryTopicNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此主题连接工厂的应用程序创建的临时主题。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

connectionFactory > properties.wmqJms

xigemaMQ JMS 连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		发送已发布消息的队列（流队列）的名称。
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒

属性名称	数据类型	缺省值	描述
			）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE **NONDUR** NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行。 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示

属性名称	数据类型	缺省值	描述
			对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。 **RLE** RLE **NONE** NONE
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省

属性名称	数据类型	缺省值	描述
			密码。（建议使用容器管理的认证别名，而不配置此属性）
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6

属性名称	数据类型	缺省值	描述
			unspecified unspecified
pubAckInterval	整型 最小值：0	25	xigemaMQ JMS 类请求代理程序的应答之前，发布程序发布的消息数
queueManager	字符串		要连接至的队列管理器的名称
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据

属性名称	数据类型	缺省值	描述
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序。
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略。
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。

属性名称	数据类型	缺省值	描述
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类是否存储有关活动预订的持久数据。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息的 JMSReplyTo 头字段标识的队列的应答消息才具有 MQRFH2 头。
tempQPrefix	字符串		用于构成 xigemaMQ 动态队列名称的前缀。
tempTopicPrefix	字符串		创建临时主题时，JMS 会生成一个格式为 TEMP/TEMPTOPICPREFIX/unique_id 的

属性名称	数据类型	缺省值	描述
			字符串，或者，如果让此属性保持为缺省值，那么只会生成格式为 TEMP/unique_id 的字符串。指定非空 TEMPTOPICPREFIX 允许定义特定模型队列，以便为在此连接下面创建的临时主题的订户创建受管队列。
temporaryModel	字符串		用来创建 JMS 临时队列的模型队列的名称。JMS 层可以使用 SYSTEM.JMS.TEMPQ.MODEL 来创建可接受持久消息的队列，而缺省值则不能。SYSTEM.DEFAULT.MODEL.QUEUE 只能打开一次。SYSTEM.JMS.TEMPQ.MODEL 可以多次打开。建议不要使用 SYSTEM.DEFAULT.MODEL.QUEUE。
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR

属性名称	数据类型	缺省值	描述
			TOPIC TOPIC

connectionFactory > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

批处理 JMS 执行程序 (batchJmsExecutor)

配置批处理 JMS 执行程序。

- *activationSpec*
  - *authData*
  - *properties.wasJms*
  - *properties.wmqJms*
- *queue*
  - *properties.wasJms*
  - *properties.wmqJms*
- *replyConnectionFactory*
  - *connectionManager*
  - *containerAuthData*
  - *properties.wasJms*
  - *properties.wmqJms*
  - *recoveryAuthData*

属性名称	数据类型	缺省值	描述
activationSpecRef	对顶级 jmsActivationSpec 元素的引用（字符串）。	batchActivationSpec	JMS 激活规范的标识，批处理执行程序使用该激活规范来创建 JMS 侦听器。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
queueRef	对顶级 jmsQueue 元素的引用（字符串）。	batchJobSubmissionQueue	JMS 队列的标识，批处理激活规范使用该队列来侦听批处理 JMS 消息。
replyConnectionFactoryRef	对顶级 jmsConnectionFactory 元素的引用（字符串）。	batchConnectionFactory	JMS 连接工厂的标识，批处理执行程序应使用该连接工厂来获取 JMS 连接。

activationSpec

JMS 激活规范的标识，批处理执行程序使用该激活规范来创建 JMS 侦听器。

false

属性名称	数据类型	缺省值	描述
authDataRef	对顶级 authData 元素的引用（字符串）。		激活规范的缺省认证数据。
maxEndpoints	整型 最小值：0	500	要分派至的最大端点数。

activationSpec > authData

激活规范的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

activationSpec > properties.wasJms

JMS 激活规范与一个或多个消息驱动的 bean 相关联，并为它们提供接收消息所必需的配置。

false

属性名称	数据类型	缺省值	描述
acknowledgeMode	• Dups-ok-acknowledge • Auto-acknowledge	Auto-acknowledge	确认方式表明应该如何确认消息驱动的 bean 接收的消息。 **Dups-ok-acknowledge** Dups-ok-acknowledge **Auto-acknowledge** Auto-acknowledge
clientId	字符串		所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
connectionFactoryLookup	字符串		此属性可用于指定通过管理方式定义的 javax.jms.ConnectionFactory、javax.jms.QueueConnectionFactory 或 javax.jms.TopicConnectionFactory 对象的查找名称，这些对象用于连接到 JMS 提供者，而端点（消息驱动的 bean）可从该提供者接收消息。
destinationLookup	字符串		此属性可用于指定以管理方式定义的 javax.jms.Queue 或 javax.jms.Topic 对象的查找名称，这些对象定义 JMS 队列或主题，端点（消息驱动的 bean）可从该队列或主题接收消息。

属性名称	数据类型	缺省值	描述
destinationRef	对顶级 adminObject 元素的引用（字符串）。		对 JMS 目标的引用
destinationType	• javax.jms.Topic • javax.jms.Queue	javax.jms.Queue	目标的类型：javax.jms.Queue 或 javax.jms.Topic。 **javax.jms.Topic** javax.jms.Topic **javax.jms.Queue** javax.jms.Queue
maxBatchSize	整型 最小值：1 最大值：2147483647		在一个消息批次中能够从消息传递引擎接收的最大消息数。
maxConcurrency	整型 最小值：1 最大值：2147483647	5	最大端点数，会将消息并行传递给这些端点。增大该数字会提高性能，但是它也会增大在指定时间使用的线程数。如果必须为所有失败的传递维持消息顺序，请将“最大并行端点数”值设置为 1。
messageSelector	字符串		用于确定消息驱动的 bean 所接收消息的 JMS 消息选择器。值是用于选择一小部分可用消息的字符串。
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default

属性名称	数据类型	缺省值	描述
			AlwaysOn AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
retryInterval	精度为秒的时间段	30s	尝试连接至消息传递引擎操作之间的延迟（以秒计），适合初始连接以及建立更好连接的任何后续尝试。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
subscriptionDurability	• DurableShared • Durable • NonDurable • NonDurableShared	NonDurable	MS 主题预订的类型。此值可为下列任一值：Durable DurableShared NonDurable NonDurableShared **DurableShared** DurableShared

属性名称	数据类型	缺省值	描述
			Durable Durable **NonDurable** NonDurable **NonDurableShared** NonDurableShared
subscriptionName	字符串		持久（及共享非持久）所需的预订名。使用持久（及共享非持久）主题预订时的必需字段。此预订名在给定客户机标识内必须唯一。

activationSpec > properties.wmqJms

xigemaMQ JMS 激活规范

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
brokerCCDurSubQueue	字符串		为 ConnectionConsumer 检索非持续预订消息所使用的队列名称
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		已发布消息发送到其中的队列（流队列）的名称

属性名称	数据类型	缺省值	描述
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串	SYSTEM.DEF.SVRCONN	要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE

属性名称	数据类型	缺省值	描述
			NONDUR NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
destinationRef	对顶级 adminObject 元素的引用（字符串）。		目标
destinationType	• javax.jms.Topic • javax.jms.Queue	javax.jms.Queue	目标的类型 - javax.jms.Queue 或 javax.jms.Topic **javax.jms.Topic** javax.jms.Topic **javax.jms.Queue** javax.jms.Queue
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。

属性名称	数据类型	缺省值	描述
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串	localhost	队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
maxMessages	整型 最小值：0	1	一次可分配给服务器会话的最大消息数。如果激活规范正在 XA 事务中向 MDB 传送消息，那么会使用值 1 而不理会此属性的设置。
maxPoolDepth	整型 最小值：0	10	激活规范上的 maxPoolDepth 属性定义可用 MDB（消息驱动的 Bean）实例的数目。减小此属性的值会减少当前可交付的消息数。
messageBatchSize	整型 最小值：0		要在一个批处理中处理的最大消息数。
messageCompression	• RLE • NONE	NONE	可用于压缩连接上的消息数据的方法列表

属性名称	数据类型	缺省值	描述
			RLE RLE **NONE** NONE
messageRetention	布尔型		连接使用者是否保留输入队列中不想要的消息
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
messageSelector	字符串		确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。如果 brokerVersion 值为 1，那么不支持使用代理程序来选择消息
pollingInterval	具有毫秒精度的时间段	5s	此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，

属性名称	数据类型	缺省值	描述
			1s500ms 相当于 1.5 秒。
poolTimeout	具有毫秒精度的时间段	5m	未使用的服务器会话在由于不活动而被关闭前，在服务器会话池中保持打开状态的时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
queueManager	字符串		要连接至的队列管理器的名称
receiveCCSID	整型 最小值：0	0	用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_Q

属性名称	数据类型	缺省值	描述
			MG，否则会忽略此值
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDelivery Sequence 属性所确定的顺序在该队列中搜索合适的消息。xigemaMQ JMS 类找到合适的消息并将其传递给使用者后，xigemaMQ JMS 类将从队列中其当前位置继续搜索下一条合适的消息。xigemaMQ JMS 类将以此方式继续搜索队列，直到它到达队列末尾或者达到此属性值所确定的时间间隔（毫秒）为止。在这两种情况下，xigemaMQ JMS 类将返回到队列开头并

属性名称	数据类型	缺省值	描述
			继续进行搜索，并且新的时间间隔开始。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器
sslCipherSuite	字符串		要用于 SSL 连接的密码套件
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIP

属性名称	数据类型	缺省值	描述
			S 提供者 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数
startTimeout	具有毫秒精度的时间段	10s	配置一个持续时间（毫秒），执行必须在此持续时间内开始。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

属性名称	数据类型	缺省值	描述
subscriptionDurability	• Durable • NonDurable		是使用持久预订还是非持久预订来将消息传递至预订该主题的 MDB **Durable** Durable **NonDurable** NonDurable
subscriptionName	字符串		持久预订的名称
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类用于存储有关活动预订的持久数据的位置。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

queue

JMS 队列的标识，批处理激活规范使用该队列来侦听批处理 JMS 消息。

false

属性名称	数据类型	缺省值	描述
jndiName	字符串		资源的 JNDI 名称。

queue > properties.wasJms

此 JMS 队列分配至的队列的名称。

false

属性名称	数据类型	缺省值	描述
deliveryMode	• NonPersistent • Application • Persistent	Application	发送至此目标的消息的传递方式。此选项控制此目标上的消息持久性。 **NonPersistent** NonPersistent **Application** Application **Persistent** Persistent
priority	整型 最小值：0 最大值：9		发送至此目标的消息的相对优先级，范围是 0（最低）到 9（最高）。
queueName	字符串	Default.Queue	关联队列的名称
readAhead	• AlwaysOff • AsConnection • AlwaysOn	AsConnection	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **AsConnection** AsConnection **AlwaysOn** AlwaysOn
timeToLive	精度为秒的时间段	0s	缺省时间（毫秒），从系统必须使消息在目标中保持活动的分派时间开始算起。指定后跟时间单位的正整数，时间单位可以

属性名称	数据类型	缺省值	描述
			是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。

queue > properties.wmqJms

xigemaMQ JMS 队列

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	1208	要用于连接或目标的编码字符集标识
arbitraryProperties	字符串		能够指定其他位置未定义的属性
baseQueueManagerName	字符串		定义了此队列的队列管理器名称
baseQueueName	字符串		队列管理器上队列的名称
encoding	字符串	NATIVE	将消息发送至此目标时，如何表示该消息的正文中的数字数据。该属性指定二进制整数、压缩十进制整数和浮点数的表示法。
expiry	字符串	APP	一个时间段，在该时间段后目标上的消息将到期
failIfQuiesce	布尔型	true	队列管理器处于停顿状态时，对某些方法的调用是否会失败。
persistence	• APP • QDEF • HIGH • NON • PERS	APP	发送至目标的消息的持久性 **APP** APP **QDEF** QDEF

属性名称	数据类型	缺省值	描述
			HIGH HIGH **NON** NON **PERS** PERS
priority	• 3 • 2 • 1 • APP • 0 • 7 • 6 • 5 • QDEF • 4 • 9 • 8	APP	发送至目标的消息的优先级 **3** 3 **2** 2 **1** 1 **APP** APP **0** 0 **7** 7 **6** 6 **5** 5 **QDEF** QDEF **4** 4 **9** 9 **8** 8
putAsyncAllowed	• ENABLED • DESTINATION	DESTINATION	是否允许消息生产者使用异步放入来将消息发送至此目标

属性名称	数据类型	缺省值	描述
	• 已禁用		**ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadAllowed	• ENABLED • DESTINATION • 已禁用	DESTINATION	是否允许 MDB 使用预先读取，以便在接收来自此目标的非持久消息之前将这些消息存储到内部缓冲区 **ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadClosePolicy	• CURRENT • ALL	ALL	当管理员停止 MDB 时，内部预先读取缓冲区中的消息发生的情况。 **CURRENT** CURRENT **ALL** ALL
receiveCCSID	整型 最小值：0		用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_QMG，否则会忽略此值

属性名称	数据类型	缺省值	描述
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG
targetClient	• JMS • MQ	JMS	是否使用 xigemaMQ RFH2 格式与目标应用程序交换信息 **JMS** JMS **MQ** MQ

replyConnectionFactory

JMS 连接工厂的标识，批处理执行程序应使用该连接工厂来获取 JMS 连接。

false

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
jndiName	字符串		资源的 JNDI 名称。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

replyConnectionFactory > connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (

属性名称	数据类型	缺省值	描述
			m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

replyConnectionFactory > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

replyConnectionFactory > properties.wasJms

JMS 连接工厂用于创建与 JMS 目标的相关联 JMS 提供者的连接，以便进行点到点消息传递和发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
clientID	字符串	clientID	所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
durableSubscriptionHome	字符串	defaultME	持久预订主地址定义需要与其建立连接的 ME 名称。

属性名称	数据类型	缺省值	描述
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent **ReliablePersistent** ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default **AlwaysOn** AlwaysOn

属性名称	数据类型	缺省值	描述
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
temporaryQueueNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此队列连接工厂的应用程序来创建的临时队列。
temporaryTopicNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此主题连接工厂的应用程序创建的临时主题。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

replyConnectionFactory > properties.wmqJms

xigemaMQ JMS 连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。

属性名称	数据类型	缺省值	描述
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		发送已发布消息的队列（流队列）的名称。
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 5

属性名称	数据类型	缺省值	描述
			00 毫秒指定为 500ms 。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE **NONDUR** NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行。 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表

属性名称	数据类型	缺省值	描述
			SYSTEM SYSTEM **NONE** NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。 **RLE** RLE **NONE** NONE
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省密码。（建议使用容器管理的认证别名，而不配置此属性）

属性名称	数据类型	缺省值	描述
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
pubAckInterval	整型 最小值：0	25	xigemaMQ JMS 类请求代理程序的应答之

属性名称	数据类型	缺省值	描述
			前，发布程序发布的消息数
queueManager	字符串		要连接至的队列管理器的名称
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续

属性名称	数据类型	缺省值	描述
			运行的一系列发送出口程序。
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略。
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供者 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅

属性名称	数据类型	缺省值	描述
			当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类是否存储有关活动预订的持久数据。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息的 JMSReplyTo 头字段标识的队列的应答消息才具有 MQRFH2 头。
tempQPrefix	字符串		用于构成 xigemaMQ 动态队列名称的前缀。
tempTopicPrefix	字符串		创建临时主题时，JMS 会生成一个格式为 TEMP/TEMPTOPICPREFIX/unique_id 的字符串，或者，如果让此属性保持为缺省值，那么只会生成格式为 TEMP/unique_id 的字符串。指定非空 TEMPTOPICPREF

属性名称	数据类型	缺省值	描述
			IX 允许定义特定模型队列，以便为在此连接下面创建的临时主题的订户创建受管队列。
temporaryModel	字符串		用来创建 JMS 临时队列的模型队列的名称。JMS 层可以使用 SYSTEM.JMS.TEMPQ.MODEL 来创建可接受持久消息的队列，而缺省值则不能。SYSTEM.DEFAULT.MODEL.QUEUE 只能打开一次。SYSTEM.JMS.TEMPQ.MODEL 可以多次打开。建议不要使用 SYSTEM.DEFAULT.MODEL.QUEUE。
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

replyConnectionFactory > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

批处理作业日志记录 (batchJobLogging)

配置批处理作业日志记录。

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	启用或禁用批处理作业日志记录。
maxRecords	整型 最小值：0 最大值：2147483647	1000	每个作业日志文件在滚动至下一个文件之前的最大日志记录数。

批处理持久性 (batchPersistence)

配置批处理持久性存储。

- *jobStore*
  - *authData*
  - *dataSource*
    - *connectionManager*
    - *containerAuthData*
    - *jaasLoginContextEntry*
    - *jdbcDriver*
      - *library*
        - *file*
        - *fileset*
        - *folder*
    - *properties*
    - *properties.datadirect.sqlserver*
    - *properties.db2.i.native*

- *properties.db2.i.toolbox*
- *properties.db2.jcc*
- *properties.derby.client*
- *properties.derby.embedded*
- *properties.informix*
- *properties.informix.jcc*
- *properties.microsoft.sqlserver*
- *properties.oracle*
- *properties.sybase*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
jobStoreRef	对顶级 databaseStore 元素的引用（字符串）。	defaultDatabaseStore	批数据的持久性存储。

jobStore

批数据的持久性存储。

false

属性名称	数据类型	缺省值	描述
authDataRef	对顶级 authData 元素的引用（字符串）。		任务安排、查询和执行的认证数据。
createTables	布尔型	true	设置为 true 时创建数据库表。
dataSourceRef	对顶级 dataSource 元素的引用（字符串）。	DefaultDataSource	连接至持久性存储的数据源。
keyGenerationStrategy	• TABLE • SEQUENCE • IDENTITY • AUTO	AUTO	用于生成唯一主键的首选策略。如果数据库不支持所选策略，那么可使用另一策略。 **TABLE** 使用数据库表来生成唯一主键。 **SEQUENCE** 使用数据库序列来生成唯一主键。

属性名称	数据类型	缺省值	描述
			IDENTITY 使用数据库标识列来生成唯一主键。 **AUTO** 自动选择用于生成唯一主键的策略。
模式	字符串		带有对数据库表的读写访问权的模式名称。
tablePrefix	字符串	WLP	表、序列和其他数据库工件的名称前缀。

jobStore > authData

任务安排、查询和执行的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

jobStore > dataSource

连接至持久性存储的数据源。

false

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。

属性名称	数据类型	缺省值	描述
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。 **MatchOriginalRequest** 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
isolationLevel	• TRANSACTION_REPEATABLE_READ		缺省事务隔离级别。

属性名称	数据类型	缺省值	描述
	• TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		**TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。 **TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻象读取受到阻止。 **TRANSACTION_READ_UNCOMMITTED** 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。

属性名称	数据类型	缺省值	描述
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此缺省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作

属性名称	数据类型	缺省值	描述
			SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource **javax.sql.XADataSource** javax.sql.XADataSource **javax.sql.ConnectionPoolDataSource** javax.sql.ConnectionPoolDataSource

jobStore > dataSource > connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小

属性名称	数据类型	缺省值	描述
			时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭

属性名称	数据类型	缺省值	描述
			发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

jobStore > dataSource > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请

属性名称	数据类型	缺省值	描述
			将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

jobStore > dataSource > jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

jobStore > dataSource > jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

jobStore > dataSource > jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

jobStore > dataSource > jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

jobStore > dataSource > jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

jobStore > dataSource > jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

jobStore > dataSource > properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1 • 0	0	JDBC 驱动程序属性：JDBCBehavior。值为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。

属性名称	数据类型	缺省值	描述
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm **userIdPassword** userIdPassword **kerberos** kerberos **auto** auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDel

属性名称	数据类型	缺省值	描述
			ay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。

属性名称	数据类型	缺省值	描述
			describeIfString describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeOutputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL

属性名称	数据类型	缺省值	描述
			requestSSL requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck **atomicWithRepositioning** atomicWithRepositioning **nonAtomic** nonAtomic 原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select **extended** extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。

属性名称	数据类型	缺省值	描述
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m

属性名称	数据类型	缺省值	描述
			) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar

属性名称	数据类型	缺省值	描述
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

jobStore > dataSource > properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all

属性名称	数据类型	缺省值	描述
			read call read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCR

属性名称	数据类型	缺省值	描述
			OLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			b 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个

属性名称	数据类型	缺省值	描述
			条目中。例如，1m30 s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 63
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal

属性名称	数据类型	缺省值	描述
			。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• \, • b • : • .		JDBC 驱动程序属性：timeSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。

属性名称	数据类型	缺省值	描述
			• 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only

属性名称	数据类型	缺省值	描述
			all all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128

属性名称	数据类型	缺省值	描述
			0 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso		JDBC 驱动程序属性：dateFormat。

属性名称	数据类型	缺省值	描述
	• eur • ymd • julian • jis • usa • mdy		**dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• • \, • . • / • -		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。 \, 逗号字符 (,)。 . 句点字符 (.)。 / 正斜杠字符 (/)。 - 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 \, 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			• 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。

属性名称	数据类型	缺省值	描述
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql

属性名称	数据类型	缺省值	描述
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none • warning	warning	JDBC 驱动程序属性：packageError。 **exception** exception **none** none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。

属性名称	数据类型	缺省值	描述
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。 **system** system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。

属性名称	数据类型	缺省值	描述
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex **table** table **language** language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared

属性名称	数据类型	缺省值	描述
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。 \, 逗号字符 (,)。 : 冒号字符 (:)。 . 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion

属性名称	数据类型	缺省值	描述
	• warning		**error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all **jdbc** jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

jobStore > dataSource > properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConnectionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.

属性名称	数据类型	缺省值	描述
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_WITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **2** TYPE_SCROLL_ASENSITIVE **1** TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。

属性名称	数据类型	缺省值	描述
			4 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。 **2** NO **1** YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES

属性名称	数据类型	缺省值	描述
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。 **2** NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OP

属性名称	数据类型	缺省值	描述
			TION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL **0** INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。

属性名称	数据类型	缺省值	描述
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES

属性名称	数据类型	缺省值	描述
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT

属性名称	数据类型	缺省值	描述
			1 HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURI

属性名称	数据类型	缺省值	描述
	• 13 • 11 • 12		TY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY

属性名称	数据类型	缺省值	描述
			16 ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY **12** ENCRYPTED_USER_AND_DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO

属性名称	数据类型	缺省值	描述
			1 YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL **1** QUERYTIMEOUT_STATEMENT_LEVEL **-1** QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURAT

属性名称	数据类型	缺省值	描述
			ION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。

属性名称	数据类型	缺省值	描述
			2 TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。 **2** NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

jobStore > dataSource > properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7	3	JDBC 驱动程序属性：securityMechanism

属性名称	数据类型	缺省值	描述
	• 4 • 9 • 8		。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。

属性名称	数据类型	缺省值	描述
			shutdown 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off **peerAuthentication** peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。

属性名称	数据类型	缺省值	描述
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。

属性名称	数据类型	缺省值	描述
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。

属性名称	数据类型	缺省值	描述
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。

属性名称	数据类型	缺省值	描述
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。

属性名称	数据类型	缺省值	描述
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。

属性名称	数据类型	缺省值	描述
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。 **ON** ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。

属性名称	数据类型	缺省值	描述
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2 **1** 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。

属性名称	数据类型	缺省值	描述
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。

属性名称	数据类型	缺省值	描述
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512

属性名称	数据类型	缺省值	描述
			、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication		JDBC 驱动程序属性：authenticationScheme。

属性名称	数据类型	缺省值	描述
	• JavaKerberos		**NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可

属性名称	数据类型	缺省值	描述
			以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。 **full** full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

jobStore > dataSource > properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。

属性名称	数据类型	缺省值	描述
			oci oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

jobStore > dataSource > properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。

属性名称	数据类型	缺省值	描述
			false false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL **socket** socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
version	整型		JDBC 驱动程序属性：version。

jobStore > dataSource > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

BELL (bell)

此功能部件允许配置使用 xigemaAS 库的基本扩展 (BELL)。BELL 允许使用共享库来扩展服务器运行时。

- *library*
  - *file*
  - *fileset*
  - *folder*
- *service*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
libraryRef	对顶级库元素的引用（字符串）。		要用于 BELL 的库。

library

要用于 BELL 的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔

属性名称	数据类型	缺省值	描述
			列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。

属性名称	数据类型	缺省值	描述
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

service

系统将在 /META-INF/services 文件夹中查找的服务的名称。如果未指定，那么系统会发现位于 META-INF/services 文件夹中的所有服务。

false

字符串

OSGi 应用程序捆绑软件存储库 (bundleRepository)

一个内部捆绑软件存储库，您可以在其中存储 OSGi 应用程序的捆绑软件。

- *fileset*

属性名称	数据类型	缺省值	描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		文件集引用的空格分隔列表
id	字符串		唯一配置标识。
location	文件、目录或 URL。		远程存储库的位置表示为绝对 URL 或者表示为相对于服务器主目录的相对 URL。

fileset

文件集引用的空格分隔列表

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是

属性名称	数据类型	缺省值	描述
			小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

上下文和依赖性注入 (CDI) V1.2 (cdi12)

定义上下文和依赖性注入 (CDI) V1.2 的行为。

属性名称	数据类型	缺省值	描述
enableImplicitBeanArchives	布尔型	true	将扫描隐式 Bean 归档以发现任何 Bean。

CDI 容器 (cdiContainer)

定义上下文和依赖性注入 (CDI) 容器的行为。

通道框架 (channelfw)

定义通道和链管理设置。

属性名称	数据类型	缺省值	描述
chainQuiesceTimeout	具有毫秒精度的时间段	30s	停顿链时要等待的缺省时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
chainStartRetryAttempts	整型 最小值：0	60	要为每个链进行重试尝试的次数。
chainStartRetryInterval	具有毫秒精度的时间段	5s	两次启动重试之间的时间间隔。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单

属性名称	数据类型	缺省值	描述
			个条目中。例如，1s500 ms 相当于 1.5 秒。
warningWaitTime	具有毫秒精度的时间段	10s	通知缺少出厂配置之前要等待的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

类装入 (classloading)

全局类装入

属性名称	数据类型	缺省值	描述
useJarUrls	布尔型	false	是否使用 jar: 或 wsjar: URL 来引用归档中的文件。

受限代理 (constrainedDelegation)

控制受限代理的操作。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
s4U2selfEnabled	布尔型	false	通过 true 或 false 指示是否启用 s4U2self。

线程上下文传播 (contextService)

配置将上下文传播至线程的方式

- *baseContext*
  - *baseContext*
  - *classloaderContext*
  - *jeeMetadataContext*
  - *securityContext*
  - *syncToOSThreadContext*
- *classloaderContext*
- *jeeMetadataContext*
- *securityContext*
- *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

baseContext > classloaderContext

类装入器上下文传播配置。

false

baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

classloaderContext

类装入器上下文传播配置。

false

jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

跨源资源共享 (cors)

指基于 JavaScript 的客户机的跨域安全性过程。

属性名称	数据类型	缺省值	描述
allowCredentials	布尔型		布尔值，指示请求中是否可以包含用户凭证。
allowedHeaders	字符串		对配置域进行请求时允许源域使用的 HTTP 头的逗号分隔列表。
allowedMethods	字符串		对配置域进行请求时允许源域使用的 HTTP 方法的逗号分隔列表。
allowedOrigins	字符串	null	允许访问配置域的源的逗号分隔列表。
domain	字符串		域名，它表示使用这些 CORS 设置来设定的 URL 。

属性名称	数据类型	缺省值	描述
exposeHeaders	字符串		可安全公开给调用 API 的 HTTP 头的逗号分隔列表。
id	字符串		唯一配置标识。
maxAge	长整型		长整型，指示预检请求的响应可以在浏览器中高速缓存的秒数。

CouchDB (couchdb)

CouchDB 连接器的配置。

- *library*
  - *file*
  - *fileset*
  - *folder*

属性名称	数据类型	缺省值	描述
高速缓存	布尔型	true	允许从 HTTP 客户机的高速缓存装入文档（如果此高速缓存存在并且修订版自上次访问后未更改）。
cleanupIdleConnections	布尔型	true	自动关闭视为空闲的连接。
connectionTimeout	具有毫秒精度的时间段	30s	建立连接的超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
enableSSL	布尔型	false	连接至 CouchDB 时启用 SSL。
host	字符串		CouchDB 服务器的主机名。
id	字符串		唯一配置标识。
jndiName	字符串		CouchDB 实例的 JNDI 名称。

属性名称	数据类型	缺省值	描述
libraryRef	对顶级库元素的引用（字符串）。		指定包含 CouchDB 客户机库及其依赖项的库。
maxCacheEntries	整型 最小值：0	1000	HTTP 客户机的高速缓存条目的最大数目。
maxConnections	整型	20	与主机的并行连接的最大数目。
maxObjectSizeBytes	整型 最小值：0	8192	设置已存储文档的最大大小。
password	可逆向编码的密码（字符串）		对应 Couchdb 用户标识的密码。
port	整型	5984	CouchDB 服务器的端口号。
relaxedSSLSettings	布尔型	false	启用松弛型 SSL 设置，这允许信任管理器接受任何主机和证书。
socketTimeout	具有毫秒精度的时间段	1m	等待响应的持续时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
url	字符串		CouchDB 服务器的 URL（包括主机和端口）。
useExpectContinue	布尔型	true	对 CouchDB 发出请求时在请求中使用 expect continue 头。
username	字符串		用于登录和访问数据库的 CouchDB 用户标识。

library

指定包含 CouchDB 客户机库及其依赖项的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

定制 LDAP 过滤器 (customLdapFilterProperties)

指定缺省定制 LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。

属性名称	数据类型	缺省值	描述
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

数据源 (dataSource)

定义数据源配置。

- *connectionManager*
- *containerAuthData*
- *jaasLoginContextEntry*
- *jdbcDriver*
  - *library*
    - *file*
    - *fileset*
    - *folder*
- *properties*
- *properties.datadirect.sqlserver*
- *properties.db2.i.native*
- *properties.db2.i.toolbox*
- *properties.db2.jcc*
- *properties.derby.client*
- *properties.derby.embedded*
- *properties.informix*
- *properties.informix.jcc*
- *properties.microsoft.sqlserver*
- *properties.oracle*
- *properties.sybase*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。 **MatchOriginalRequest** 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
id	字符串		唯一配置标识。
isolationLevel	• TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED		缺省事务隔离级别。

属性名称	数据类型	缺省值	描述
	• TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		**TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。 **TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻象读取受到阻止。 **TRANSACTION_READ_UNCOMMITTED** 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此缺

属性名称	数据类型	缺省值	描述
			省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作 SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource **javax.sql.XADataSource** javax.sql.XADataSource

属性名称	数据类型	缺省值	描述
			javax.sql.ConnectionPoolDataSource javax.sql.ConnectionPoolDataSource

connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间

属性名称	数据类型	缺省值	描述
			单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使

属性名称	数据类型	缺省值	描述
			用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。

属性名称	数据类型	缺省值	描述
name	字符串		JAAS 配置条目的名称。

jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条

属性名称	数据类型	缺省值	描述
			目中。例如，1s500ms 相当于 1.5 秒。

jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1	0	JDBC 驱动程序属性：JDBCBehavior。值

属性名称	数据类型	缺省值	描述
	• 0		为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm **userIdPassword** userIdPassword **kerberos** kerberos

属性名称	数据类型	缺省值	描述
			auto auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDelay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime

属性名称	数据类型	缺省值	描述
			dateTimeOffset dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeOutputParameters。 **describeIfString** describeIfString

属性名称	数据类型	缺省值	描述
			noDescribe noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL **requestSSL** requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck **atomicWithRepositioning** atomicWithRepositioning **nonAtomic** nonAtomic

属性名称	数据类型	缺省值	描述
			原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select **extended** extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。

属性名称	数据类型	缺省值	描述
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128

属性名称	数据类型	缺省值	描述
			0 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso

属性名称	数据类型	缺省值	描述
	• mdy		**eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。

属性名称	数据类型	缺省值	描述
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 63
maximumScale	整型 最小值：0	31	JDBC 驱动程序属性：maximumScale。

属性名称	数据类型	缺省值	描述
	最大值：63		
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur

属性名称	数据类型	缺省值	描述
			jis jis **usa** usa **hms** hms
timeSeparator	• \, • b • : • .		JDBC 驱动程序属性：timeSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。 **.** 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary

属性名称	数据类型	缺省值	描述
			character character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时

属性名称	数据类型	缺省值	描述
			进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TY

属性名称	数据类型	缺省值	描述
			PE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• • \, • . • /		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。

属性名称	数据类型	缺省值	描述
	• -		\, 逗号字符 (,)。 . 句点字符 (.)。 / 正斜杠字符 (/)。 - 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 \, 逗号字符 (,)。 . 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。

属性名称	数据类型	缺省值	描述
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。

属性名称	数据类型	缺省值	描述
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none	warning	JDBC 驱动程序属性：packageError。

属性名称	数据类型	缺省值	描述
	• warning		**exception** exception **none** none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。 **system** system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。

属性名称	数据类型	缺省值	描述
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex

属性名称	数据类型	缺省值	描述
			table table **language** language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。

属性名称	数据类型	缺省值	描述
			\, 逗号字符 (,)。 : 冒号字符 (:)。 . 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion **error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all **jdbc** jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。

属性名称	数据类型	缺省值	描述
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConn

属性名称	数据类型	缺省值	描述
			ectionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_WITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION

属性名称	数据类型	缺省值	描述
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。

属性名称	数据类型	缺省值	描述
			2 TYPE_SCROLL_ASENSITIVE **1** TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。 **4** 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。

属性名称	数据类型	缺省值	描述
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。 **2** NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OPTION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET

属性名称	数据类型	缺省值	描述
			1 INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL **0** INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。

属性名称	数据类型	缺省值	描述
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STA

属性名称	数据类型	缺省值	描述
			TEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT

属性名称	数据类型	缺省值	描述
			1 HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **16** ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY **12** ENCRYPTED_USER_AND_

属性名称	数据类型	缺省值	描述
			DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO **1** YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL **1** QUERYTIMEOUT_STATE

属性名称	数据类型	缺省值	描述
			MENT_LEVEL **-1** QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRAC

属性名称	数据类型	缺省值	描述
			E_TRACEPOINTS=262144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。 **2** TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。

属性名称	数据类型	缺省值	描述
			create 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7 • 4 • 9 • 8	3	JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off

属性名称	数据类型	缺省值	描述
			peerAuthentication peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 3

属性名称	数据类型	缺省值	描述
			0s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。

属性名称	数据类型	缺省值	描述
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。

属性名称	数据类型	缺省值	描述
			例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。

属性名称	数据类型	缺省值	描述
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。

属性名称	数据类型	缺省值	描述
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将

属性名称	数据类型	缺省值	描述
			30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。

属性名称	数据类型	缺省值	描述
			ON ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2

属性名称	数据类型	缺省值	描述
			1 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例

属性名称	数据类型	缺省值	描述
			如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication • JavaKerberos		JDBC 驱动程序属性：authenticationScheme。 **NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。

属性名称	数据类型	缺省值	描述
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。

属性名称	数据类型	缺省值	描述
			full full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。 **oci** oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。 **false** false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL

属性名称	数据类型	缺省值	描述
			socket socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
version	整型		JDBC 驱动程序属性：version。

recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

数据库存储 (databaseStore)

指定关系数据库作为服务器功能部件的持久性存储。

- *authData*
- *dataSource*

- *connectionManager*
- *containerAuthData*
- *jaasLoginContextEntry*
- *jdbcDriver*
  - *library*
    - *file*
    - *fileset*
    - *folder*
- *properties*
- *properties.datadirect.sqlserver*
- *properties.db2.i.native*
- *properties.db2.i.toolbox*
- *properties.db2.jcc*
- *properties.derby.client*
- *properties.derby.embedded*
- *properties.informix*
- *properties.informix.jcc*
- *properties.microsoft.sqlserver*
- *properties.oracle*
- *properties.sybase*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
authDataRef	对顶级 authData 元素的引用（字符串）。		任务安排、查询和执行的认证数据。
createTables	布尔型	true	设置为 true 时创建数据库表。
dataSourceRef	对顶级 dataSource 元素的引用（字符串）。	DefaultDataSource	连接至持久性存储的数据源。
id	字符串		唯一配置标识。
keyGenerationStrategy	• TABLE • SEQUENCE • IDENTITY • AUTO	AUTO	用于生成唯一主键的首选策略。如果数据库不支持所选策略，那么可使用另一策略。 **TABLE** 使用数据库表来生成唯一主键。 **SEQUENCE** 使用数据库序列来生成唯一主键。

属性名称	数据类型	缺省值	描述
			IDENTITY 使用数据库标识列来生成唯一主键。 **AUTO** 自动选择用于生成唯一主键的策略。
schema	字符串		带有对数据库表的读写访问权的模式名称。
tablePrefix	字符串	WLP	表、序列和其他数据库工件的名称前缀。

authData

任务安排、查询和执行的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

dataSource

连接至持久性存储的数据源。

false

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的

属性名称	数据类型	缺省值	描述
			连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。 **MatchOriginalRequest** 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
isolationLevel	• TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE		缺省事务隔离级别。 **TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。

属性名称	数据类型	缺省值	描述
	• TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		**TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻象读取受到阻止。 **TRANSACTION_READ_UNCOMMITTED** 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTime

属性名称	数据类型	缺省值	描述
			out 可以覆盖此缺省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作 SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource

属性名称	数据类型	缺省值	描述
			javax.sql.XADataSource javax.sql.XADataSource **javax.sql.ConnectionPoolDataSource** javax.sql.ConnectionPoolDataSource

dataSource > connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。

属性名称	数据类型	缺省值	描述
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。

属性名称	数据类型	缺省值	描述
			EntirePool 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

dataSource > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

dataSource > jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

dataSource > jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

dataSource > jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识

属性名称	数据类型	缺省值	描述
name	字符串		管理员的共享库的名称

dataSource > jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

dataSource > jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定

属性名称	数据类型	缺省值	描述
			后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

dataSource > jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

dataSource > properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1 • 0	0	JDBC 驱动程序属性：JDBCBehavior。值为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm

属性名称	数据类型	缺省值	描述
			userIdPassword userIdPassword **kerberos** kerberos **auto** auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDelay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。

属性名称	数据类型	缺省值	描述
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe		JDBC 驱动程序属性：describeOutputParameters。

属性名称	数据类型	缺省值	描述
	• describeIfDateTime • describeAll		**describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL **requestSSL** requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck

属性名称	数据类型	缺省值	描述
			atomicWithRepositioning atomicWithRepositioning **nonAtomic** nonAtomic 原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select **extended** extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以

属性名称	数据类型	缺省值	描述
			是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。

属性名称	数据类型	缺省值	描述
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

dataSource > properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128	32	JDBC 驱动程序属性：blockSize。

属性名称	数据类型	缺省值	描述
	• 0 • 32 • 64 • 16 • 8 • 256		**512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy		JDBC 驱动程序属性：dateFormat。

属性名称	数据类型	缺省值	描述
	• iso • eur • ymd • julian • jis • usa • mdy		**dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			• 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31

属性名称	数据类型	缺省值	描述
			63 63
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。

属性名称	数据类型	缺省值	描述
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• \, • b • : • .		JDBC 驱动程序属性：timeSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。 **.** 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。

属性名称	数据类型	缺省值	描述
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。

属性名称	数据类型	缺省值	描述
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa

属性名称	数据类型	缺省值	描述
			mdy mdy
dateSeparator	• • \, • . • / • -		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic

属性名称	数据类型	缺省值	描述
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。

属性名称	数据类型	缺省值	描述
			31 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。

属性名称	数据类型	缺省值	描述
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none • warning	warning	JDBC 驱动程序属性：packageError。 **exception** exception **none** none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。

属性名称	数据类型	缺省值	描述
			system system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多

属性名称	数据类型	缺省值	描述
			个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex **table** table **language** language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa

属性名称	数据类型	缺省值	描述
			hms hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。 **\,** 逗号字符 (,)。 **:** 冒号字符 (:)。 **.** 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion **error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all

属性名称	数据类型	缺省值	描述
			jdbc jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

dataSource > properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。

属性名称	数据类型	缺省值	描述
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConnectionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_W

属性名称	数据类型	缺省值	描述
			ITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCR

属性名称	数据类型	缺省值	描述
			OLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **2** TYPE_SCROLL_ASENSITIVE **1** TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。 **4** 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO

属性名称	数据类型	缺省值	描述
			1 YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。 **2** NO **1** YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。

属性名称	数据类型	缺省值	描述
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。 **2** NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OPTION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。

属性名称	数据类型	缺省值	描述
			2 INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL **0** INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET

属性名称	数据类型	缺省值	描述
			1 INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER

属性名称	数据类型	缺省值	描述
			_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **16** ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY

属性名称	数据类型	缺省值	描述
			12 ENCRYPTED_USER_AND_DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO **1** YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL

属性名称	数据类型	缺省值	描述
			1 QUERYTIMEOUT_STATEMENT_LEVEL **-1** QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRAC

属性名称	数据类型	缺省值	描述
			E_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。 **2** TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。

属性名称	数据类型	缺省值	描述
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。 **2** NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

dataSource > properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。

属性名称	数据类型	缺省值	描述
			create 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7 • 4 • 9 • 8	3	JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off

属性名称	数据类型	缺省值	描述
			peerAuthentication peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 3

属性名称	数据类型	缺省值	描述
			0s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。

属性名称	数据类型	缺省值	描述
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。

属性名称	数据类型	缺省值	描述
			例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。

属性名称	数据类型	缺省值	描述
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。

属性名称	数据类型	缺省值	描述
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将

属性名称	数据类型	缺省值	描述
			30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。

属性名称	数据类型	缺省值	描述
			ON ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2

属性名称	数据类型	缺省值	描述
			1 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例

属性名称	数据类型	缺省值	描述
			如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication • JavaKerberos		JDBC 驱动程序属性：authenticationScheme。 **NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。

属性名称	数据类型	缺省值	描述
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。

属性名称	数据类型	缺省值	描述
			full full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

dataSource > properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。 **oci** oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。 **false** false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL

属性名称	数据类型	缺省值	描述
			socket socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
version	整型		JDBC 驱动程序属性：version。

dataSource > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

分布式映射 (distributedMap)

本地高速缓存的分布式映射配置。

- *cacheGroup*
  - *member*

- *adapterBeanName*
- *diskCache*
- *library*
  - *file*
  - *fileset*
  - *folder*

属性名称	数据类型	缺省值	描述
cacheProviderName	字符串	缺省值	指定备用高速缓存提供程序的名称。
highThreshold	整型 最小值：-1 最大值：100	-1	指定内存高速缓存逐出策略的启动时间。阈值以内存高速缓存大小（以兆字节 (MB) 计）的百分比表示。
id	字符串		唯一配置标识。
jndiName	字符串	${id}	高速缓存实例的 JNDI 名称。
libraryRef	对顶级库元素的引用（字符串）。		指定对共享库的引用。
lowThreshold	整型 最小值：-1 最大值：100	-1	指定内存高速缓存逐出策略的结束时间。阈值以内存高速缓存大小（以兆字节 (MB) 计）的百分比表示。
memorySizeInEntries	整型 最小值：0	2000	指定一个正整数以定义高速缓存可以保存的最大条目数。值通常以千计。最小值为 100，而未设置最大值。缺省值为 2000。
memorySizeInMB	整型 最小值：-1	-1	指定最大内存高速缓存大小的值（以兆字节 (MB) 计）。

cacheGroup

指定由 IBM WebSphere(R) Edge Server 和 IBM(R) HTTP Server 等服务器上的 xigemaAS 所控制的外部高速缓存集。

false

属性名称	数据类型	缺省值	描述
name	字符串		指定外部高速缓存组的唯一名称。外部高

属性名称	数据类型	缺省值	描述
			速缓存组名必须与 servlet 中或 Java(TM) Server Pages (JSP) cachespec.xml 文件中所定义的 ExternalCache 属性匹配。

cacheGroup > member

xigemaAS 控制的外部高速缓存组的成员。

false

属性名称	数据类型	缺省值	描述
host	字符串		标准主机名
port	整型 最小值：0		端口。

cacheGroup > member > adapterBeanName

指定 xigemaAS 和此外部高速缓存之间的适配器的类名（位于 xigemaAS 类路径上）。

false

字符串

diskCache

启用磁盘卸载，以指定高速缓存变满时从高速缓存中移除高速缓存条目并将它们保存到磁盘。此位置是磁盘卸载功能使用的标准目录位置。“停止时清空到磁盘”选项指定在服务器停止时，将内存高速缓存的内容移至磁盘。

false

属性名称	数据类型	缺省值	描述
evictionPolicy	• RANDOM • SIZE	RANDOM	指定磁盘高速缓存用来逐出条目的逐出算法和阈值。当磁盘大小达到阈值上限时，磁盘高速缓存垃圾收集器会启动，并逐出磁盘上随机选择（随机）的条目或最大（大小）条目，直到磁盘大小达到阈值下限。 **RANDOM** 随机

属性名称	数据类型	缺省值	描述
			SIZE 大小
flushToDiskOnStopEnabled	布尔型	false	将此值设为 true，以在服务器停止时将内存中高速缓存的对象保存到磁盘。如果将“启用磁盘卸载”设为 false，那么会忽略此值。
highThreshold	整型 最小值：0 最大值：100	80	指定逐出策略的启动时间。
location	目录路径		指定要用于磁盘卸载的目录。
lowThreshold	整型 最小值：0 最大值：100	70	指定逐出策略的结束时间。
sizeInEntries	整型 最小值：0	100000	指定最大磁盘高速缓存大小的值（以条目数计）。
sizeInGB	整型 最小值：3	3	指定最大磁盘高速缓存大小的值（以千兆字节 (GB) 计）。

library

指定对共享库的引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述

属性名称	数据类型	缺省值	描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示

属性名称	数据类型	缺省值	描述
			秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

SIP 域解析器 (domainResolver)

SIP 域解析器的配置

- *dnsServers*

属性名称	数据类型	缺省值	描述
addTtl	布尔型	false	将 IBMTTL 参数添加到 SIP URI。
dnsAllowedFailures	整型	5	对于 DNS 故障转移机制，这是指允许 DNS 查找发生故障的次数。
dnsAutoResolve	布尔型	false	自动解析域名。
dnsEdns	• N • Y	Y	定义用于进行 DNS 查找以解析 RFC 3263 SIP URI 的传输协议。如果设置为 Y，那么将使用 UDP。如果设置为 N，那么将使用 TCP。

属性名称	数据类型	缺省值	描述
			N N **Y** Y
dnsRequestCacheTimeoutMin	具有分钟精度的时间段	10m	已高速缓存的 DNS 查询超时之前要经过的最短时间（以分钟计）。指定后跟时间单位的正整数，时间单位可以是小时 (h) 或分钟 (m)。例如，将 30 分钟指定为 30m。可以将多个值包括在单个条目中。例如，1h30m 相当于 90 分钟。
dnsSingleQueryTimeoutSec	精度为秒的时间段	5s	对于 DNS 故障转移机制，这是指单个查询发生超时之前要经过的秒数。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
dnsUdpPayloadSize	短整型 最小值：512 最大值：32767	1280	DNS 解析器服务的 UDP 有效内容大小（按字节计）。
dnsWindowSizeInterval	整型	10	对于 DNS 故障转移机制，这是解析器服务时间段间隔。
dnsWindowSizeMin	整型	600	对于 DNS 故障转移机制，这是最短时间段。

dnsServers

允许通过 DNS 将 SIP URI 解析为 IP 地址、端口和传输协议。值是一个字符串，其中只包含一个或两个地址与端口组成的元组，两个元组之间用空格分隔。

false

字符串

IBM Lotus Domino LDAP 过滤器 (domino50LdapFilterProperties)

指定缺省 IBM Lotus Domino LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=dominoGroup))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	dominoGroup:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(uid=%v)(objectclass=Person))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	person:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

Novell eDirectory LDAP 过滤器 (edirectoryLdapFilterProperties)

指定 Novell eDirectory LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=groupOfNames))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	groupOfNames:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(cn=%v)(objectclass=Person))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	person:cn	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

EJB 应用程序 (ejbApplication)

定义 EJB 应用程序的属性。

- *application-bnd*
  - *security-role*
    - *group*
    - *run-as*
    - *special-subject*
    - *user*
- *classloader*
  - *commonLibrary*
    - *file*
    - *fileset*
    - *folder*
  - *privateLibrary*
    - *file*
    - *fileset*
    - *folder*

属性名称	数据类型	缺省值	描述
autoStart	布尔型	true	指示服务器是否自动启动应用程序。
context-root	字符串		应用程序的上下文根。
id	字符串		唯一配置标识。
location	文件、目录或 URL。		应用程序的位置，表示为绝对路径或相对于服务器级应用程序目录的路径。
name	字符串		应用程序的名称。
suppressUncoveredHttpMethodWarning	布尔型	false	指示在应用程序部署期间阻止未覆盖 HTTP 方法警告消息的选项。

application-bnd

将应用程序中包括的常规部署信息绑定到特定资源。

false

属性名称	数据类型	缺省值	描述
version	字符串		应用程序绑定规范的版本。

application-bnd > security-role

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		安全角色的名称。

application-bnd > security-role > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		组访问标识
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

application-bnd > security-role > run-as

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
password	可逆向编码的密码（字符串）		从一个 Bean 访问另一个 Bean 时需要的用户密码。可以采用明文或编码格式存储该值。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
userid	字符串		从一个 Bean 访问另一个 Bean 时需要的用户标识。

application-bnd > security-role > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE		下列其中一种特殊主体集类型：ALL_AU

属性名称	数据类型	缺省值	描述
	• ALL_AUTHENTICATED_USERS		THENTICATED_USERS 和 EVERYONE。 **EVERYONE** Everyone **ALL_AUTHENTICATED_USERS** 所有已认证的用户

application-bnd > security-role > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

classloader

定义应用程序类装入器的设置。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
classProviderRef	顶级 resourceAdapter 元素的引用列表（以逗号分隔的字符串）。		类提供程序引用的列表。搜索类或资源时，此类装入器将在搜索其自己的类路径之后授权给指定的类提供程序。

属性名称	数据类型	缺省值	描述
commonLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。会与其他类装入器共享库类实例。
delegation	• parentFirst • parentLast	parentFirst	控制父类装入器是用在此类装入器之前还是之后。如果选择“父代最先”，那么在搜索类路径之前，授权给直接父代。如果选择“父代最后”，那么在授权给直接父代之前，搜索类路径。 **parentFirst** 父代最先 **parentLast** 父代最后
privateLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。
serializablePattern	字符串	NULL	为符合条件的 java bean 增加序列化接口。多个正则表达式使用‘%%’ 分隔，例如 ^com.test.\w+? %% ^com.test.\w+?。
serializableLocation	字符串	NULL	为符合条件的 java bean 增加序列化接口。serializableLocation 的值为文件路径，此文件中的每一行为一个 java 类路径。例如：com.test.Test。

classloader > commonLibrary

库引用的列表。会与其他类装入器共享库类实例。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > commonLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > commonLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > commonLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

classloader > privateLibrary

库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔

属性名称	数据类型	缺省值	描述
			列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > privateLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > privateLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号

属性名称	数据类型	缺省值	描述
			或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > privateLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

EJB 容器 (ejbContainer)

定义 EJB 容器的行为。

- *asynchronous*
  - *contextService*
    - *baseContext*
      - *baseContext*
      - *classloaderContext*
      - *jeeMetadataContext*
      - *securityContext*
      - *syncToOSThreadContext*
    - *classloaderContext*

- *jeeMetadataContext*
- *securityContext*
- *syncToOSThreadContext*

- *timerService*
  - *persistentExecutor*
    - *contextService*
      - *baseContext*
        - *baseContext*
        - *classloaderContext*
        - *jeeMetadataContext*
        - *securityContext*
        - *syncToOSThreadContext*
      - *classloaderContext*
      - *jeeMetadataContext*
      - *securityContext*
      - *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
cacheCleanupInterval	精度为秒的时间段	3s	超过大小时两次清除未使用的有状态会话 Bean 实例之间的时间间隔。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
cacheSize	整型 最小值：1	2053	应该在内存中高速缓存的有状态会话 Bean 实例数。
poolCleanupInterval	精度为秒的时间段	30s	两次移除未使用的 Bean 实例之间的时间间隔。此设置仅适用于无状态会话 Bean 和消息驱动的 Bean。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
startEJBsAtAppStart	布尔型		指定将对 EJB 类型进行初始化的时间。如果此属

属性名称	数据类型	缺省值	描述
			性设置为 true，那么将在应用程序首次启动时对 EJB 类型进行初始化。如果此属性设置为 false，那么将在应用程序首次使用 EJB 类型时对 EJB 类型进行初始化。如果未设置此属性，那么将根据 ibm-ejb-jar-ext.xml 文件中的 start-at-app-start 属性逐个 Bean 来确定行为。此设置不适用于消息驱动的 Bean 或者启动独立 Bean。消息驱动的 Bean 和启动独立 Bean 始终将在应用程序启动时进行初始化。
unmanagedThreadCheck	布尔型	true	当通过 JNDI 名称查找 EJB 资源时，如果此时执行查找的线程为非受管线程，容器会抛出异常：CWWKN0100E。如果此属性设置为 false，容器将不检查此线程是否为受管线程。此设置仅对 java:global 生效。

asynchronous

定义 EJB 异步方法的行为。

false

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。		用于管理上下文传播至异步 EJB 方法线程的过程的上下文服务。
maxUnclaimedRemoteResults	整型 最小值：1	1000	在所有将返回 Future 对象的远程异步方法调用中，服务器保留的未声明结果的最大数目。如果超过了最大数目，那么服务器将清除最早完成的方

属性名称	数据类型	缺省值	描述
			法调用的结果，以防止发生内存泄漏。
unclaimedRemoteResultTimeout	精度为秒的时间段	24h	服务器为每个将返回 Future 对象的远程异步方法调用保留结果的时间。如果应用程序在所指定的时间段内未声明结果，那么服务器将清除该方法调用的结果，以防止发生内存泄漏。 指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

asynchronous > contextService

用于管理上下文传播至异步 EJB 方法线程的过程的上下文服务。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将

属性名称	数据类型	缺省值	描述
			不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

asynchronous > contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何

属性名称	数据类型	缺省值	描述
			警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

asynchronous > contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

asynchronous > contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

asynchronous > contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

asynchronous > contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

asynchronous > contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

asynchronous > contextService > classloaderContext

类装入器上下文传播配置。

false

asynchronous > contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

asynchronous > contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

asynchronous > contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

timerService

定义 EJB 计时器服务的行为。

false

属性名称	数据类型	缺省值	描述
lateTimerThreshold	具有分钟精度的时间段	5m	一个分钟数，在计时器的已安排到期时间过了该分钟数之后，计时器启动将视为延迟。当计时器延迟启动时，将记录一条警告消息，指示计时器的启动时间晚于安排的时间。缺省阈值为 5 分钟，值为 0 分钟时关闭警告消息功能部件。 指定后跟时间单位的正整数，时间单位可以是小时 (h) 或分钟 (m)。例如，将 30 分钟指定为 30 m。可以将多个值包括在单个条目中。例如，1h30m 相当于 90 分钟。
nonPersistentMaxRetries	整型 最小值：-1	-1	当非持久计时器到期时，将调用超时回调方法。此设置控制 EJB 容器重试该计时器的次数。如果此回调方法的事务失败或已回滚，那么 EJB 容器必须至少重试该计时器一次。缺省值为 -1，这表示 EJB 容器将无限制地进行重试，直到该计时器成功

属性名称	数据类型	缺省值	描述
			为止。如果该值设置为 0，那么 EJB 容器不会重试该计时器，然而，这会导致出现不符合 EJB 规范的行为。
nonPersistentRetryInterval	精度为秒的时间段	300s	当非持久计时器到期时，将调用超时回调方法。如果此回调方法的事务失败或已回滚，那么容器必须重试该计时器。第一次重试将立即进行，后续重试将延迟指定的秒数。如果将此值设置为 0，那么将立即进行所有重试。如果您未指定值，那么缺省时间间隔为 300 秒。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
persistentExecutorRef	对顶级 persistentExecutor 元素的引用（字符串）。		安排并运行 EJB 持久计时器任务。

timerService > persistentExecutor

安排并运行 EJB 持久计时器任务。

false

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。	DefaultContextService	配置上下文捕获及传播至线程的方式。
enableTaskExecution	布尔型	true	确定此实例能否运行任务。

属性名称	数据类型	缺省值	描述
initialPollDelay	具有毫秒精度的时间段	0	在此实例可轮询持久性存储以查找要运行的任务之前等待的持续时间。如果值为 -1，那么将延迟轮询，直到通过程序启动轮询为止。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
pollInterval	具有毫秒精度的时间段	-1	轮询要运行的任务之间的时间间隔。值 -1 将禁用初始轮询之后的所有轮询。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
pollSize	整型 最小值：1		轮询持久性存储以获取要运行的任务时要查找的最大任务条目数。如果未指定，那么表示没有限制。
retryInterval	具有毫秒精度的时间段	1m	在第二次重试失败任务与后续连续重试失败任务之间必须经过的时间。无论此属性的值如何，都将立即进行第一次重试。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m

属性名称	数据类型	缺省值	描述
			)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
retryLimit	短整型 最小值：-1 最大值：10000	10	对已失败或者已回滚的任务连续重试的次数限制；达到此限制之后，就会认为该任务永久失败，并且不再进一步重试。如果值为 -1，那么将允许无限制地重试。
taskStoreRef	对顶级 databaseStore 元素的引用（字符串）。	defaultDatabaseStore	计划任务的持久性存储。

timerService > persistentExecutor > contextService

配置上下文捕获及传播至线程的方式。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何

属性名称	数据类型	缺省值	描述
			警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

timerService > persistentExecutor > contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何

属性名称	数据类型	缺省值	描述
			警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

timerService > persistentExecutor > contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

timerService > persistentExecutor > contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

timerService > persistentExecutor > contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

timerService > persistentExecutor > contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

timerService > persistentExecutor > contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

timerService > persistentExecutor > contextService > classloaderContext

类装入器上下文传播配置。

false

timerService > persistentExecutor > contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

timerService > persistentExecutor > contextService > securityContext
指定了此属性时，会将工作发起方的安全上下文传播至工作单元。
false

timerService > persistentExecutor > contextService > syncToOSThreadContext
指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。
false

企业应用程序 (enterpriseApplication)

定义企业应用程序的属性。

- *application-bnd*
  - *security-role*
    - *group*
    - *run-as*
    - *special-subject*
    - *user*
- *classloader*
  - *commonLibrary*
    - *file*
    - *fileset*
    - *folder*
  - *privateLibrary*
    - *file*
    - *fileset*
    - *folder*
- *resourceAdapter*
  - *contextService*
    - *baseContext*
      - *baseContext*
      - *classloaderContext*
      - *jeeMetadataContext*
      - *securityContext*
      - *syncToOSThreadContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *customize*

属性名称	数据类型	缺省值	描述
autoStart	布尔型	true	指示服务器是否自动启动应用程序。

属性名称	数据类型	缺省值	描述
defaultClientModule	字符串		企业应用程序的缺省客户机模块。
id	字符串		唯一配置标识。
location	文件、目录或 URL。		应用程序的位置，表示为绝对路径或相对于服务器级应用程序目录的路径。
name	字符串		应用程序的名称。
suppressUncoveredHttpMethodWarning	布尔型	false	指示在应用程序部署期间阻止未覆盖 HTTP 方法警告消息的选项。

application-bnd

将应用程序中包括的常规部署信息绑定到特定资源。

false

属性名称	数据类型	缺省值	描述
version	字符串		应用程序绑定规范的版本。

application-bnd > security-role

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		安全角色的名称。

application-bnd > security-role > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		组访问标识
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

application-bnd > security-role > run-as

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
password	可逆向编码的密码（字符串）		从一个 Bean 访问另一个 Bean 时需要的用户密码。可以采用明文或编码格式存储该值。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
userid	字符串		从一个 Bean 访问另一个 Bean 时需要的用户标识。

application-bnd > security-role > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。 **EVERYONE** Everyone **ALL_AUTHENTICATED_USERS** 所有已认证的用户

application-bnd > security-role > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

classloader

定义应用程序类装入器的设置。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
classProviderRef	顶级 resourceAdapter 元素的引用列表（以逗号分隔的字符串）。		类提供程序引用的列表。搜索类或资源时，此类装入器将在搜索其自己的类路径之后授权给指定的类提供程序。
commonLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。会与其他类装入器共享库类实例。
delegation	• parentFirst • parentLast	parentFirst	控制父类装入器是用在此类装入器之前还是之后。如果选择“父代最先”，那么在搜索类路径之前，授权给直接父代。如果选择“父代最后”，那么在授权给直接父代之前，搜索类路径。 **parentFirst** 父代最先 **parentLast** 父代最后
privateLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。库类实例是此类装入器特有的，与来自其他类

属性名称	数据类型	缺省值	描述
			装入器的类实例无关。
serializablePattern	字符串	NULL	为符合条件的 java bean 增加序列化接口。多个正则表达式使用‘%%’分隔，例如 ^com.test.\w+? %% ^com.test.\w+?。
serializableLocation	字符串	NULL	为符合条件的 java bean 增加序列化接口。serializableLocation 的值为文件路径，此文件中的每一行为一个 java 类路径。例如：com.test.Test。

classloader > commonLibrary

库引用的列表。会与其他类装入器共享库类实例。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > commonLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > commonLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条

属性名称	数据类型	缺省值	描述
			目中。例如，1s500ms 相当于 1.5 秒。

classloader > commonLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

classloader > privateLibrary

库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > privateLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > privateLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > privateLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

resourceAdapter

为嵌入在应用程序中的资源适配器指定配置。

false

属性名称	数据类型	缺省值	描述
alias	字符串	${id}	覆盖资源适配器的缺省标识。该标识用于资源适配器的配置属性元素的名称中，该名称转而用于为由资源适配器提供的任何资源确定配置属性元素的名称。资源适配器的配置属性元素名称具有 properties.<APP_NAME>.<ALIAS> 格式，其中 <APP_NAME> 是应用程序的名称，<ALIAS> 是所配置别名。如果未指定，那么别名缺省为资源适配器的模块名称。
autoStart	布尔型		配置资源适配器的启动方式是在其部署时自动启动，还是在注入或查找资源时缓慢启动。
contextServiceRef	对顶级 contextService 元素的引用（字符串）。		配置上下文捕获及传播至线程的方式。
id	字符串		确定此配置适用的嵌入式资源适配器模块的名称。

resourceAdapter > contextService

配置上下文捕获及传播至线程的方式。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

resourceAdapter > contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

resourceAdapter > contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

resourceAdapter > contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

resourceAdapter > contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

resourceAdapter > contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

resourceAdapter > contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

resourceAdapter > contextService > classloaderContext

类装入器上下文传播配置。

false

resourceAdapter > contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

resourceAdapter > contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

resourceAdapter > contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

resourceAdapter > customize

定制具有指定接口和/或实现类的激活规范、受管对象或连接工厂的配置属性元素。

false

属性名称	数据类型	缺省值	描述
实现	字符串		标准实现类名，应该针对该类名定制配置属性元素。
interface	字符串		标准接口类名，应该针对该类名定制配置属性元素。
suffix	字符串		覆盖配置属性元素的缺省后缀。例如，properties.rarModule1.C

属性名称	数据类型	缺省值	描述
			ustomConnectionFactory 中的“CustomConnectionFactory”。当资源适配器提供了多种类型的连接工厂、受管对象或端点激活时，该后缀对于进行区别很有用。如果配置属性元素定制省略该后缀或将它保留为空白，那么不会使用后缀。

事件日志记录 (eventLogging)

记录事件（例如，JDBC 请求和 servlet 请求）及它们的持续时间。

属性名称	数据类型	缺省值	描述
eventTypes	字符串	所有	需要记录的事件类型的逗号分隔列表。使用 all 以记录所有事件类型。
includeContextInfo	布尔型	true	指示上下文详细信息是否包含在日志输出中。
logMode	• entry • entryExit • exit	exit	控制是在进入事件时或从事件退出时进行事件日志记录还是在这两个时间都进行事件日志记录。 **entry** 在进入事件时记录 **entryExit** 在进入和退出事件时记录 **exit** 退出事件时记录
minDuration	具有毫秒精度的时间段	1s	对于时间超过 minDuration 的事件，将记录退出条目。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个

属性名称	数据类型	缺省值	描述
			条目中。例如，1s500ms 相当于 1.5 秒。
sampleRate	整型 最小值：1	1	要从每 n 个请求中抽取一个请求作为样本，请将 sampleRate 设置为 n。要抽取所有请求作为样本，请将 sampleRate 设置为 1。

执行程序管理 (executor)

为 xigemaAS 内核缺省执行程序定义设置。请注意，始终有一个且正好一个缺省执行程序，仅供 xigemaAS 运行时使用且应用程序不可直接访问。需要配置并使用执行程序的应用程序应改用受管执行程序。

属性名称	数据类型	缺省值	描述
coreThreads	整型	-1	要与执行程序相关联的稳定状态线程数或核心线程数。与执行程序相关联的线程数将快速地增长到此数目。如果此值小于 0，那么会使用缺省值。根据系统上的硬件线程数计算此缺省值。
keepAlive	具有毫秒精度的时间段	60s	在允许空闲线程终止之前，要在池中保留该线程的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
maxThreads	整型	-1	可以与执行程序相关联的最大线程数。如果大于 0，那么此值必须大于或等于 coreThreads 的值。如果 maxThreads 的值小于或等于 0，那么最大线程数为无限制。请注意，与此执行程序相关联的实际线程数由 xigemaAS 内核动态确定，所以使最大线程数不受限制并不意味着运行时将主动创建大量线

属性名称	数据类型	缺省值	描述
			程；这只是让 xigemaAS 内核在没有已定义上限的情况下决定要与此执行程序相关联的线程数。
name	字符串	缺省执行程序	xigemaAS 内核缺省执行程序的名称。
rejectedWorkPolicy	• CALLER_RUNS • ABORT	ABORT	当执行程序无法为要执行的工作设置阶段时要使用的策略。 **CALLER_RUNS** 在调用者的线程上立即执行工作。 **ABORT** 产生异常。
stealPolicy	• STRICT • NEVER • LOCAL	LOCAL	要使用的工作偷取策略。此策略的选项确定对工作进行排队的方式以及线程获取已排队的工作的方式。 **STRICT** 生成工作的所有线程都拥有一个本地工作堆。当本地工作堆耗尽时，与执行程序相关联的线程会从其他线程接受工作。 **NEVER** 使用全局工作队列为与执行程序相关联的线程提供工作。将不会出现偷取问题。 **LOCAL** 将全局工作队列用于与执行程序无关联的线程所生成的工作。与执行程序相关联的线程所生成的工作会放置在

属性名称	数据类型	缺省值	描述
			本地工作堆上。此工作堆由生成线程所拥有，除非其他线程偷取此工作堆。如果本地工作堆为空并且全局工作队列中没有工作，那么与执行程序相关联的线程将接受与其他线程相关联的工作。

功能部件管理器 (featureManager)

定义服务器装入功能部件的方式。

- *feature*

属性名称	数据类型	缺省值	描述
onError	• IGNORE • FAIL • WARN	WARN	装入功能部件失败后要执行的操作。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

feature

指定在服务器运行时要使用的功能部件。

false

字符串

用户注册表联合 (federatedRepository)

用户注册表联合的配置。

- *extendedProperty*
- *primaryRealm*
  - *defaultParents*
  - *groupDisplayNameMapping*
  - *groupSecurityNameMapping*
  - *participatingBaseEntry*
  - *uniqueGroupIdMapping*
  - *uniqueUserIdMapping*
  - *userDisplayNameMapping*
  - *userSecurityNameMapping*
- *realm*
  - *defaultParents*
  - *groupDisplayNameMapping*
  - *groupSecurityNameMapping*
  - *participatingBaseEntry*
  - *uniqueGroupIdMapping*
  - *uniqueUserIdMapping*
  - *userDisplayNameMapping*
  - *userSecurityNameMapping*
- *supportedEntityType*
  - *defaultParent*
  - *name*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
maxSearchResults	整型	4500	在搜索中可以返回的最大条目数。
pageCacheSize	整型	1000	定义可存储在高速缓存中的分页请求数。需要根据系统上执行的分页请求数及可用硬件系统资源来配置分页高速缓存大小。
pageCacheTimeout	具有毫秒精度的时间段	30000ms	定义已添加至页面高速缓存的条目可用的最长时间。经过指定的时间之后，将会清除页面高速缓存中的条目。需要根据系统上执行的分页搜索请求之间的时间间隔及可用硬件系统资源来配置此项。 指定后跟时间单位的正整数

属性名称	数据类型	缺省值	描述
			，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
searchTimeout	具有毫秒精度的时间段	10m	处理搜索的最长时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

extendedProperty

Person 和 Group 的扩展属性。

false

属性名称	数据类型	缺省值	描述
dataType	• Double • Long • Date • String • BigDecimal • BigInteger • Boolean • Integer		定义 Person 和 Group 的扩展属性的数据类型。基本 Java 数据类型受支持。 **Double** 双精度 **Long** 长整型 **Date** 日期 **String** 字符串 **BigDecimal** BigDecimal **BigInteger** BigInteger

属性名称	数据类型	缺省值	描述
			Boolean 布尔型 **Integer** 整数
defaultValue	字符串		定义写操作期间的属性的缺省值（如果未设置缺省值）。
entityType	• PersonAccount • Group		要映射的实体的名称。实体的名称可为 PersonAccount 或 Group。 **PersonAccount** 个人 **Group** 组
id	字符串		唯一配置标识。
multiValued	布尔型	false	定义 Person 和 Group 的扩展属性是否支持多个值。
name	字符串		定义 Person 和 Group 的扩展属性的名称。

primaryRealm

主域配置。

false

属性名称	数据类型	缺省值	描述
allowOpIfRepoDown	布尔型	false	指定存储库关闭时是否允许执行操作。缺省值为 false。
delimiter	字符串	/	用于限定应在其下执行操作的领域的定界符。例如，userid=test1/myrealm，其中 / 是定界符，myrealm 是域名。
name	字符串		域的名称。

primaryRealm > defaultParents

域的缺省父映射。

false

属性名称	数据类型	缺省值	描述
name	字符串		要映射的实体的名称。实体的名称可为 PersonAccount 或 Group。
parentUniqueName	字符串		存储库中的基本专有名称 (DN) 下的专有名称，将在其下创建所配置类型的所有实体。

primaryRealm > groupDisplayNameMapping

用户注册表操作中的组显示名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	cn	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

primaryRealm > groupSecurityNameMapping

用户注册表操作中的组安全性名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有

属性名称	数据类型	缺省值	描述
			效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	cn	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

primaryRealm > participatingBaseEntry

属于此领域的基本条目。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		基本条目的名称。

primaryRealm > uniqueGroupIdMapping

用户注册表操作中的唯一组标识的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccou

属性名称	数据类型	缺省值	描述
			nt 和组实体类型的属性。

primaryRealm > uniqueUserIdMapping

用户注册表操作中使用的唯一用户标识的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	uniqueName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

primaryRealm > userDisplayNameMapping

用户注册表操作中的用户显示名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	principalName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	principalName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName

属性名称	数据类型	缺省值	描述
			，以及 PersonAccount 和组实体类型的属性。

primaryRealm > userSecurityNameMapping

用户注册表操作中的用户安全性名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	principalName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

领域

对域的引用。

false

属性名称	数据类型	缺省值	描述
allowOpIfRepoDown	布尔型	false	指定存储库关闭时是否允许执行操作。缺省值为 false。
delimiter	字符串	/	用于限定应在其下执行操作的领域的定界符。例如，userid=test1/myrealm，其中 / 是定界符，myrealm 是域名。
id	字符串		唯一配置标识。
name	字符串		域的名称。

realm > defaultParents

域的缺省父映射。

false

属性名称	数据类型	缺省值	描述
name	字符串		要映射的实体的名称。实体的名称可为 PersonAccount 或 Group。
parentUniqueName	字符串		存储库中的基本专有名称 (DN) 下的专有名称，将在其下创建所配置类型的所有实体。

realm > groupDisplayNameMapping

用户注册表操作中的组显示名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	cn	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

realm > groupSecurityNameMapping

用户注册表操作中的组安全性名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有

属性名称	数据类型	缺省值	描述
			效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	cn	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

realm > participatingBaseEntry

属于此领域的基本条目。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		基本条目的名称。

realm > uniqueGroupIdMapping

用户注册表操作中的唯一组标识的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	cn	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccou

属性名称	数据类型	缺省值	描述
			nt 和组实体类型的属性。

realm > uniqueUserIdMapping

用户注册表操作中使用的唯一用户标识的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	uniqueName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

realm > userDisplayNameMapping

用户注册表操作中的用户显示名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	principalName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	principalName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName

属性名称	数据类型	缺省值	描述
			，以及 PersonAccount 和组实体类型的属性。

realm > userSecurityNameMapping

用户注册表操作中的用户安全性名称的输入和输出属性映射。

false

属性名称	数据类型	缺省值	描述
inputProperty	字符串	principalName	映射到输入的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。
outputProperty	字符串	uniqueName	映射至输出的用户注册表属性的特性。有效值为：uniqueId、uniqueName、externalId 和 externalName，以及 PersonAccount 和组实体类型的属性。

supportedEntityType

实体类型映射的缺省父代。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

supportedEntityType > defaultParent

存储库中的基本专有名称 (DN) 下的专有名称，将在其下创建所配置类型的所有实体。

false

字符串

supportedEntityType > name

要映射的实体的名称。实体的名称可为 PersonAccount 或 Group。

false

字符串

文件集 (fileset)

指定从基本目录开始，并且与一组模式相匹配的一组文件。

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500 ms 相当于 1.5 秒。

主机单例 (hostSingleton)

主机单例选择器配置

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串	*	单例的名称。“*”通配符是缺省值，指示此配置适用于此服务器中的所有单例。

属性名称	数据类型	缺省值	描述
port	整型	0	要用于主机单例引导者选择的端口。值 0 为缺省值，意味着不会进行任何选择。在此情况下，每个成员中的单例将是它自己的引导者。

HTTP 访问日志记录 (httpAccessLogging)

HTTP 访问日志包含所有入站 HTTP 客户端请求的记录。

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	启用访问日志记录。
filePath	文件路径	${server.output.dir}/logs/http_access.log	访问日志文件的目录路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.output.dir}。
id	字符串		唯一配置标识。
logFormat	字符串	%h %u %{t}W "%r" %s %b	指定在日志记录客户机访问信息时所使用的日志格式。
maxFileSize	整型 最小值：0	20	回滚之前日志文件的最大大小，以兆字节计；值为 0 时意味着无限制。
maxFiles	整型 最小值：0	2	移除最旧的日志文件之前将保留的最大日志文件数；值为 0 时意味着无限制。

HTTP 分派器 (httpDispatcher)

HTTP 分派器配置。

- *trustedHeaderOrigin*

属性名称	数据类型	缺省值	描述
appOrContextRootMissingMessage	字符串		找不到所请求 URI 中的应用程序时，要返回给客户机的消息。
enableWelcomePage	布尔型	true	当没有应用程序绑定至上下文根“/”时，启用缺省

属性名称	数据类型	缺省值	描述
			xigemaAS 概要文件欢迎页面。缺省值为 true。

trustedHeaderOrigin

专用头供 Web 服务器插件用来提供有关原始请求的信息。这些头优先于 HTTP 主机头，并且用于选择虚拟主机以服务请求。缺省值为“*”，表示将信任任何源中的传入专用头。指定“无”以禁用专用头并且仅依赖于 HTTP 主机头，或者指定 IP 地址列表以将专用头处理限制为特定信任源。

false

字符串

HTTP 传输编码 (httpEncoding)

HTTP 传输编码设置

属性名称	数据类型	缺省值	描述
converter.Big5	字符串	Cp950	Big5 中文转换器
converter.EUC-JP	字符串	Cp33722C	EUC 日语转换器 (EUC-JP)
converter.EUC-KR	字符串	Cp970	EUC 韩国语转换器 (EUC-KR)
converter.EUC-TW	字符串	Cp964	EUC 中文（台湾）转换器 (EUC-TW)
converter.EUC_KR	字符串	Cp970	EUC 韩国语转换器 (EUC_KR)
converter.GB2312	字符串	EUC_CN	GB2312 中文转换器
converter.ISO-2022-KR	字符串	ISO2022KR	ISO-2022 韩国语转换器 (ISO-2022-KR)
converter.Shift_JIS	字符串	Cp943C	Shift_JIS 日语转换器
encoding.ar	字符串	ISO-8859-6	阿拉伯语语言编码 (ar)
encoding.be	字符串	ISO-8859-5	白俄罗斯语语言编码 (be)
encoding.bg	字符串	ISO-8859-5	保加利亚语语言编码 (bg)
encoding.bn	字符串	UTF-8	孟加拉语语言编码 (bn)
encoding.ca	字符串	ISO-8859-1	加泰隆语语言编码 (ca)
encoding.cs	字符串	ISO-8859-2	捷克语语言编码 (cs)
encoding.da	字符串	ISO-8859-1	丹麦语语言编码 (da)
encoding.de	字符串	ISO-8859-1	德语语言编码 (de)

属性名称	数据类型	缺省值	描述
encoding.el	字符串	ISO-8859-7	希腊语语言编码 (el)
encoding.en	字符串	ISO-8859-1	英语语言编码 (en)
encoding.es	字符串	ISO-8859-1	西班牙语语言编码 (es)
encoding.et	字符串	ISO-8859-4	爱沙尼亚语语言编码 (et)
encoding.eu	字符串	ISO-8859-1	巴士克语语言编码 (eu)
encoding.fa	字符串	ISO-8859-6	波斯语语言编码 (fa)
encoding.fi	字符串	ISO-8859-1	芬兰语语言编码 (fi)
encoding.fo	字符串	ISO-8859-2	法罗语语言编码 (fo)
encoding.fr	字符串	ISO-8859-1	法语语言编码 (fr)
encoding.he	字符串	ISO-8859-8	希伯来语语言编码 (he)
encoding.hi	字符串	UTF-8	印地语语言编码 (hi)
encoding.hr	字符串	ISO-8859-2	克罗地亚语语言编码 (hr)
encoding.hu	字符串	ISO-8859-2	匈牙利语语言编码 (hu)
encoding.hy	字符串	UTF-8	亚美尼亚语语言编码 (hy)
encoding.is	字符串	ISO-8859-1	冰岛语语言编码 (is)
encoding.it	字符串	ISO-8859-1	意大利语语言编码 (it)
encoding.iw	字符串	ISO-8859-8	希伯来语语言编码 (iw)
encoding.ja	字符串	Shift_JIS	日语语言编码 (ja)
encoding.ji	字符串	ISO-8859-8	依地语语言编码 (ji)
encoding.ka	字符串	UTF-8	格鲁吉亚语语言编码 (ka)
encoding.ko	字符串	EUC-KR	韩国语语言编码 (ko)
encoding.lt	字符串	ISO-8859-2	立陶宛语语言编码 (lt)
encoding.lv	字符串	ISO-8859-4	拉脱维亚语语言编码 (lv)
encoding.mk	字符串	ISO-8859-5	马其顿语语言编码 (mk)
encoding.mr	字符串	UTF-8	马拉提语语言编码 (mr)
encoding.ms	字符串	ISO-8859-6	马来语语言编码 (ms)
encoding.mt	字符串	ISO-8859-3	马耳他语语言编码 (mt)
encoding.nl	字符串	ISO-8859-1	荷兰语语言编码 (nl)
encoding.no	字符串	ISO-8859-1	挪威语语言编码 (no)
encoding.pl	字符串	ISO-8859-2	波兰语语言编码 (pl)

属性名称	数据类型	缺省值	描述
encoding.pt	字符串	ISO-8859-1	葡萄牙语语言编码 (pt)
encoding.ro	字符串	ISO-8859-2	罗马尼亚语语言编码 (ro)
encoding.ru	字符串	ISO-8859-5	俄语语言编码 (ru)
encoding.sa	字符串	UTF-8	梵语语言编码 (sa)
encoding.sh	字符串	ISO-8859-2	塞尔维亚 - 克罗地亚语语言编码 (sh)
encoding.sk	字符串	ISO-8859-2	斯洛伐克语语言编码 (sk)
encoding.sl	字符串	ISO-8859-2	斯洛文尼亚语语言编码 (sl)
encoding.sq	字符串	ISO-8859-2	阿尔巴尼亚语语言编码 (sq)
encoding.sr	字符串	ISO-8859-5	塞尔维亚语语言编码 (sr)
encoding.sv	字符串	ISO-8859-1	瑞典语语言编码 (sv)
encoding.ta	字符串	UTF-8	泰米尔语语言编码 (ta)
encoding.th	字符串	windows-874	泰国语语言编码 (th)
encoding.tr	字符串	ISO-8859-9	土耳其语语言编码 (tr)
encoding.uk	字符串	ISO-8859-5	乌克兰语语言编码 (uk)
encoding.vi	字符串	windows-1258	越南语语言编码 (vi)
encoding.yi	字符串	ISO-8859-8	依地语语言编码 (yi)
encoding.zh	字符串	GB2312	中文语言编码 (zh)
encoding.zh_TW	字符串	Big5	中文语言编码 (zh_TW)

HTTP 端点 (httpEndpoint)

HTTP 端点的配置属性。

- *accessLogging*
- *httpOptions*
- *sslOptions*
- *tcpOptions*

属性名称	数据类型	缺省值	描述
accessLoggingRef	对顶级 httpAccessLogging 元素的引用（字符串）。		端点的 HTTP 访问日志记录配置。

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	切换端点的可用性。值为 true 时，分派器将激活此端点以处理 HTTP 请求。
host	字符串	localhost	客户机用于请求资源的 IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或仅 DNS 主机名。可使用“*”来表示所有可用网络接口。
httpOptionsRef	对顶级 httpOptions 元素的引用（字符串）。	defaultHttpOptions	端点的 HTTP 协议选项。
httpPort	整型 最小值：-1 最大值：65535		用于客户机 HTTP 请求的端口。使用 -1 来禁用此端口。
httpsPort	整型 最小值：-1 最大值：65535		用于通过 SSL (https) 来保护的客户机 HTTP 请求的端口。使用 -1 来禁用此端口。
id	字符串		唯一配置标识。
onError	• IGNORE • FAIL • WARN	WARN	启动端点失败后要执行的操作。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		端点的 SSL 协议选项。

属性名称	数据类型	缺省值	描述
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	端点的 TCP 协议选项。

accessLogging

端点的 HTTP 访问日志记录配置。

false

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	启用访问日志记录。
filePath	文件路径	${server.output.dir}/logs/http_access.log	访问日志文件的目录路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.output.dir}。
logFormat	字符串	%h %u %{t}W "%r" %s %b	指定在日志记录客户机访问信息时所使用的日志格式。
maxFileSize	整型 最小值：0	20	回滚之前日志文件的最大大小，以兆字节计；值为 0 时意味着无限制。
maxFiles	整型 最小值：0	2	移除最旧的日志文件之前将保留的最大日志文件数；值为 0 时意味着无限制。

httpOptions

端点的 HTTP 协议选项。

false

属性名称	数据类型	缺省值	描述
keepAliveEnabled	布尔型	true	启用持续连接（HTTP 保持活动）。如果为 true，那么连接将保持活动状态，以供多个顺序请求和响应复用。如果为 false，那么发送响应之后会关闭连接。
maxKeepAliveRequests	整型 最小值：-1	100	启用持续连接时，单个 HTTP 连接上可接

属性名称	数据类型	缺省值	描述
			受的最大持续请求数。值为 -1 时意味着不受限制。
persistTimeout	精度为秒的时间段	30s	将允许套接字在两个请求之间保持空闲的时间量。仅当启用持续连接时，此设置才适用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
readTimeout	精度为秒的时间段	60s	发生第一次读取之后，用于等待读请求在套接字上完成的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
removeServerHeader	布尔型	false	从 HTTP 头中移除服务器实现信息，并且还要禁用缺省 xigemaAS 概要文件欢迎页面。
writeTimeout	精度为秒的时间段	60s	响应数据的每个部分在套接字上等待传输的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目

属性名称	数据类型	缺省值	描述
			中。例如，1m30s 相当于 90 秒。

sslOptions

端点的 SSL 协议选项。

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

tcpOptions

端点的 TCP 协议选项。

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)

属性名称	数据类型	缺省值	描述
			。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

HTTP 选项 (httpOptions)

HTTP 协议配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
keepAliveEnabled	布尔型	true	启用持续连接（HTTP 保持活动）。如果为 true，那么连接将保持活动状态，以供多个顺序请求和响应复用。如果为 false，那么发送响应之后会关闭连接。
maxKeepAliveRequests	整型 最小值：-1	100	启用持续连接时，单个 HTTP 连接上可接受的最大持续请求数。值为 -1 时意味着不受限制。
persistTimeout	精度为秒的时间段	30s	将允许套接字在两个请求之间保持空闲的时间量。仅当启用持续连接时，此设置才适用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
readTimeout	精度为秒的时间段	60s	发生第一次读取之后，用于等待读请求在套接字上完成的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，

属性名称	数据类型	缺省值	描述
			将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
removeServerHeader	布尔型	false	从 HTTP 头中移除服务器实现信息，并且还要禁用缺省 xigemaAS 概要文件欢迎页面。
writeTimeout	精度为秒的时间段	60s	响应数据的每个部分在套接字上等待传输的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

HTTP 代理重定向 (httpProxyRedirect)

配置端口重定向。将 HTTP 请求从非安全端口（例如，80）重定向至已启用 SSL 的受保护端口（例如，443）时，使用了 HTTP 代理重定向。

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	此属性确定服务器是否应重定向此配置元素中指定的端口。缺省值为 true。
host	字符串	*	用于此代理重定向的主机名。仅当入局请求指定与此值匹配的主机名时，服务器才会重定向 HTTP 请求。缺省值为 *（所有主机）。
httpPort	整型 最小值：1 最大值：65535		要从其重定向的（非安全）端口。此端口上的入局 HTTP 请求将重定向至指定 HTTPS 端口。
httpsPort	整型 最小值：1 最大值：65535		要重定向至的（安全）端口。使用 HTTP 端口的入局 HTTP 请求将重定向至此端口。
id	字符串		唯一配置标识。

HTTP 会话 (httpSession)

HTTP 会话管理的配置。

属性名称	数据类型	缺省值	描述
allowOverflow	布尔型	true	允许内存中的会话数超过“内存内最大会话数”属性的值。
alwaysEncodeUrl	布尔型	false	Servlet 2.5 规范指定只有在必要时才对 response.encodeURL 调用上的 URL 进行编码。要在启用 URL 编码的情况下支持向后兼容性，请将此属性设为 true 以调用 encodeURL 方法。即使浏览器支持 cookie，也总是对 URL 进行编码。
cloneSeparator	字符串	:	在会话 cookie 中用来将会话标识与克隆标识相分离的单个字符。通常应该使用缺省值。在一些无线应用协议 (WAP) 设备上，不允许使用冒号 (:)，因此应该改用加号 (+)。很少使用不同值。在使用此属性来更改克隆分隔符之前，应该了解系统上运行的其他产品的克隆字符要求。可以将任何字符指定为此属性的值这一事实并不意味着您指定的字符将正常工作。这也不表示 Vsettan 有责任修正使用备用字符时可能产生的任何问题。
cookieDomain	字符串		会话跟踪 cookie 的域字段。
cookieHttpOnly	布尔型	true	指定会话 cookie 包含 HttpOnly 字段。支持 HttpOnly 字段的浏览器不支持客户机端脚本访问 cookie。使用 HttpOnly 字段将有助于防止跨站点脚本编制攻击。

属性名称	数据类型	缺省值	描述
cookieMaxAge	精度为秒的时间段	-1	cookie 可在客户机浏览器上驻留的最大时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
cookieName	字符串	JSESSIONID	会话管理 cookie 的唯一名称。
cookiePath	字符串	/	会将 cookie 发送到路径中指定的 URL。
cookieSecure	布尔型	false	指定会话 cookie 要包含安全字段。
cookiesEnabled	布尔型	true	指定会话跟踪使用 cookie 传送会话标识。
debugCrossover	布尔型	false	启用此选项以执行附加检查来验证，是否仅访问或引用与请求相关联的会话，并在检测到任何差异的情况下记录消息。禁用此选项以跳过附加检查。
forceInvalidationMultiple	整型	3	如果您的请求通常不受限于响应时间限制，请指定 0 来指示在尝试使会话失效之前，会话管理器应该无限期地等待直到请求完成。否则，请将此属性设为正整数，以延迟使活动会话失效的操作。到第一个失效时间间隔过去为止，将不使已超时的活动会话失效，但是到基于此值的时间间隔过去为止，将使其失效。例如，值为 2 时，将在会话超时到期之后第二个失效时间间隔过去时使活动会话失效。
idLength	整型	23	会话标识的长度。

属性名称	数据类型	缺省值	描述
idReuse	布尔型	false	在没有针对会话持久性进行配置的多 JVM 环境中，如果将此属性设置为“true”，那么会使会话管理器能够将同一会话信息用于用户的所有请求（即使处理这些请求的 Web 应用程序由不同 JVM 控制）。此属性的缺省值为 false。如果您想要使会话管理器能够使用从浏览器发送的会话标识，跨 Web 应用程序（在没有针对会话持久性配置的环境中运行）保存会话数据，请将此属性设为 true。
invalidateOnUnauthorizedSessionRequestException	布尔型	false	如果您想要会话管理器使会话失效（而不是发出 UnauthorizedSessionRequestException），以对未经授权的请求作出响应，请将此属性设为 true。如果会话失效，那么请求者可以创建新会话，但无权访问先前保存的任何会话数据。这允许单用户在注销后继续处理对其他应用程序的请求，但仍然保护会话数据。
invalidationTimeout	精度为秒的时间段	30m	在会话不再有效之前可处于未使用状态的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxInMemorySessionCount	整型	1000	在内存中为每个 Web 模块保留的最大会话数。
noAdditionalInfo	布尔型	false	强制移除会话标识中不需要的信息。

属性名称	数据类型	缺省值	描述
protocolSwitchRewritingEnabled	布尔型	false	当 URL 需要从 HTTP 切换至 HTTPS 或从 HTTPS 切换至 HTTP 时，将会话标识添加到该 URL。
reaperPollInterval	精度为秒的时间段	-1	进程（用于移除无效会话）的唤醒时间间隔，以秒计。最小值为 30 秒。如果输入了小于最小值的值，那么会自动确定并使用相应的值。此值会覆盖介于 30 秒和 360 秒之间的缺省安装值（基于会话超时值）。因为缺省会话超时为 30 分钟，所以收割程序时间间隔通常介于 2 分钟和 3 分钟之间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
rewriteId	字符串	jsessionid	使用此属性来更改配合 URL 重写功能使用的密钥。
securityIntegrationEnabled	布尔型	true	启用安全性集成，这会促使会话管理工具将用户的身份与他们的 HTTP 会话相关联。
securityUserIgnoreCase	布尔型	false	指示应该将会话安全身份和客户机安全身份视为匹配（即使它们的大小写不同）。例如，当此属性设置为 true，那么会话安全性身份 USER1 与客户机安全性身份 User1 和 user1 相匹配。
sslTrackingEnabled	布尔型	false	指定会话跟踪使用安全套接字层 (SSL) 信息作为会话标识。

属性名称	数据类型	缺省值	描述
urlRewritingEnabled	布尔型	false	指定会话管理工具使用重写的 URL 来传送会话标识。
useContextRootAsCookiePath	布尔型	false	指定 cookie 路径相当于 Web 模块的上下文根（而不是 /）

HTTP 会话数据库 (httpSessionDatabase)

控制将 HTTP 会话持久存储到数据库的方式。

属性名称	数据类型	缺省值	描述
dataSourceRef	字符串		会话管理器应该用来持久存储 HTTP 会话数据的数据源的标识。
db2RowSize	• 32KB • 4KB • 8KB • 16KB	4KB	为会话表配置的表空间页大小（如果正在使用 DB 2 数据库）。增大此值可以改进某些环境中的数据库性能。 **32KB** 使用表空间页大小 32 KB。必须额外创建 DB2 缓冲池和表空间，并且指定 32KB 作为这两者的页大小。还必须指定您已创建的表空间的名称。 **4KB** 使用缺省表空间页大小 4 KB。不需要创建 DB2 缓冲池或表空间，并且不需要指定表空间名称。 **8KB** 使用表空间页大小 8 KB。必须额外创建 DB2 缓冲池和表空间，并且指定 8KB 作为这两

属性名称	数据类型	缺省值	描述
			者的页大小。还必须指定您已创建的表空间的名称。 **16KB** 使用表空间页大小 16 KB。必须额外创建 DB2 缓冲池和表空间，并且指定 16KB 作为这两者的页大小。还必须指定您已创建的表空间的名称。
onlyCheckInCacheDuringPreInvoke	布尔型	false	如果值为 true，那么指示仅当请求获取会话时，才应该更新该会话的上次访问时间。如果值为 false，那么指示每次请求时，都应该更新会话的上次访问时间。更改此值可以改进某些环境中的性能。
scheduleInvalidation	布尔型	false	启用此选项，以减少使 HTTP 会话保持活动状态所需的数据库更新数。请指定一天内应用程序服务器中的活动最少的那两个小时。禁用此选项时，失效器进程会每隔几分钟运行一次，以移除失效的 HTTP 会话。
scheduleInvalidationFirstHour	整型	0	指示其间会从持久性存储中清除失效会话的第一个小时。请将此值指定为 0 与 23 之间的任意整数。仅当启用了调度失效时，此值才有效。
scheduleInvalidationSecondHour	整型	0	指示其间会从持久性存储中清除失效会话的第二个小时。请将此值指定为 0 与 23 之间的任意整数。仅当启用了调度失效时，此值才有效。

属性名称	数据类型	缺省值	描述
skipIndexCreation	布尔型	false	将此属性设置为“true”以禁止在服务器启动时创建索引。仅当您要手动创建自己的数据库索引以实现会话持久性时，才应该使用此定制属性。但是，建议您让会话管理器创建数据库索引。启用此属性之前，请确保会话数据库上确实存在正确的索引。
tableName	字符串	会话	数据库表名。
tableSpaceName	字符串		要用于会话表的表空间。仅当 DB2 行大小超过 4KB 时，才需要此值。
useInvalidatedId	布尔型	true	将此属性设置为“true”，以复用入局标识（如果最近已使具有该标识的会话失效）。因为这会阻止检查持久性存储，所以这是对性能的优化。
useMultiRowSchema	布尔型	false	启用时，会将每个会话数据属性置于数据库中的单独一行上，从而允许为每个会话存储更大量的数据。如果会话属性非常大，而且基本不需要更改，那么此配置可以产生较好的性能。禁用时，对于每个会话，会将所有会话数据属性都置于同一行中。
useOracleBlob	布尔型	false	将此属性设置为“true”，以使用介质列的二进制大对象 (BLOB) 数据类型创建数据库表。使用 Oracle 数据库时，此值会提高持久会话的性能。由于存在 Oracle 限制，如果数据超过 4000 字节，那么 BLOB 支持将要求使用 Oracle 调用接口 (OCI) 数据库驱动程序。您也必须确保通过执行以下操作重新启动

属性名称	数据类型	缺省值	描述
			服务器前已创建了一个新的会话表：删除旧的会话表或将数据源定义更改为引用不包含会话表的数据库。
usingCustomSchemaName	布尔型	false	如果您要将 DB2 用于会话持久性并且在数据源中设置 currentSchema 属性，请将此属性设置为“true”。
writeContents	• ALL_SESSION_ATTRIBUTES • ONLY_UPDATED_ATTRIBUTES	ONLY_UPDATED_ATTRIBUTES	指定应该写入持久性存储的会话数据量。缺省情况下，仅写入已更新的属性，但是可改为写入全部属性（无论它们是否已更改）。 **ALL_SESSION_ATTRIBUTES** 所有属性都写入持久性存储。 **ONLY_UPDATED_ATTRIBUTES** 仅将更新的属性写入持久性存储。
writeFrequency	• TIME_BASED_WRITE • END_OF_SERVLET_SERVICE • MANUAL_UPDATE	END_OF_SERVLET_SERVICE	指定将会话数据写入持久性存储的时间。缺省情况下，会在 servlet 完成执行之后将会话数据写入持久性存储。更改此值可以改进某些环境中的性能。 **TIME_BASED_WRITE** 根据指定的写时间间隔值，将会话数据写入持久性存储。 **END_OF_SERVLET_SERVICE** 在 servlet 完成执行之后，将会话数

属性名称	数据类型	缺省值	描述
			据写入持久性存储。 **MANUAL_UPDATE** 需要在 IBMSession 对象上进行程序化同步，以将会话数据写入持久性存储。
writeInterval	精度为秒的时间段	2m	将会话数据写入持久性存储之前应该经过的秒数。缺省值为 120 秒。仅当启用基于时间的写频率时，才会使用此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

HTTP 会话 Redis 数据库 (httpSessionRedis)

控制将 HTTP 会话持久保存到 redis 数据库的方式。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
sessionDBRef	字符串		对 Redis 数据源的引用。

HTTP Whiteboard (httpWhiteboard)

HTTP Whiteboard 提供了用于托管由 OSGi 服务提供的 servlet 和资源的运行时环境。

属性名称	数据类型	缺省值	描述
contextPath	字符串	/osgi/http	要用于 HTTP Whiteboard 运行时环境的上下文路径。

IBM Tivoli Directory Server LDAP 过滤器 (idsLdapFilterProperties)

指定缺省 IBM Tivoli Directory Server LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

IIOP 端点 (iiopEndpoint)

IIOP 端点配置

- *iiopsOptions*
- *tcpOptions*

属性名称	数据类型	缺省值	描述
host	字符串	localhost	IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或者只是 DNS 主机名。可使用“*”来表示所有可用网络接口。
id	字符串		唯一配置标识。
iiopPort	整型		由此 IIOP 端点打开的没有安全保障的服务器套接字的端口
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	IIOP 端点的 TCP 协议选项

iiopsOptions

由此 IIOP 端点打开的安全服务器套接字的规范

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
iiopsPort	整型		指定要使用 SSL 选项进行配置的端口。
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

tcpOptions

IIOP 端点的 TCP 协议选项

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒

属性名称	数据类型	缺省值	描述
			指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

IIOP 服务器策略 (iiopServerPolicies)

IIOP 服务器策略的配置

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

包含 (include)

指定要包含在服务器配置中的配置资源。

属性名称	数据类型	缺省值	描述
location	文件、目录或 URL。		指定资源位置。它可能是远程资源的文件路径或 URI。
onConflict	• IGNORE • REPLACE • MERGE	MERGE	指定发现冲突时用于合并元素的行为。 **IGNORE** 被包含文件中的冲突元素将被忽略。 **REPLACE** 如果元素有冲突，那么被包含文件中的元素将替换冲突元素。 **MERGE** 冲突元素将合并到一起。
optional	布尔型	false	允许跳过找不到的所包括资源。

Sun Java System Directory Server LDAP 过滤器 (iplanetLdapFilterProperties)

指定缺省 Sun Java System Directory Server LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=ldapsubentry))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	nsRole:nsRole	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(uid=%v)(objectclass=inetOrgPerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	inetOrgPerson:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

JAAS 登录上下文条目 (jaasLoginContextEntry)

JAAS 登录上下文条目配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

JAAS 登录模块 (jaasLoginModule)

JAAS 配置中的登录模块。

- *library*
  - *file*
  - *fileset*
  - *folder*
- *options*

属性名称	数据类型	缺省值	描述
className	字符串		JAAS 登录模块类的标准程序包名。
controlFlag	• SUFFICIENT • REQUISITE • REQUIRED • OPTIONAL	REQUIRED	登录模块的控制标志。有效值包括 REQUIRED、REQUISITE、SUFFICIENT 和 OPTIONAL。 **SUFFICIENT** 按照 JAAS 规范，此登录模块是足够的 (SUFFICIENT)。此登录模块并非成功操作所必需的。如果认证成功，那么将不会调用任何其他登录模块，并且会将控制权返回给调用者。 **REQUISITE** 此登录模块是每个 JAAS 规范的必需项。此登录模块是成功操作所必需的。如果认证失败，那么将不会调用任何其他登录模块，并且会将控制权返回给调用者。 **REQUIRED** 按照 JAAS 规范，此登录模块是必需的 (REQUIRED)。此登录模块是成功操作所必需的。 **OPTIONAL** 按照 JAAS 规范，此登录模块是可选的 (OPTIONAL)。此登录模块并非成功操作所必需的。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
libraryRef	对顶级库元素的引用（字符串）。		对共享库配置的标识的引用。

library

对共享库配置的标识的引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。

属性名称	数据类型	缺省值	描述
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

options

JAAS 登录模块选项的集合

false

Java 2 安全性 (javaPermission)

Java 2 安全性的许可权配置。

属性名称	数据类型	缺省值	描述
actions	字符串		授予的许可权允许对目标名称执行的操作。例如，如果该许可权为 java.io.FilePermission，那么允许对目标名称执行的操作为“读取”。
className	字符串		类的名称，该类用于实现要授予的许可权。例如，java.io.FilePermission。
codebase	字符串		要被授予该许可权的代码库。
id	字符串		唯一配置标识。
name	字符串		该许可权应用于的目标。例如，如果该许可权为 java.io.FilePermission，那么目标为“所有文件”。
principalName	字符串		对其授予该许可权的主体。
principalType	字符串		应针对给定主体名称进行匹配的类名。
restriction	布尔型	false	声明是限制还是授予该许可权。如果限制设置为“true”，那么将拒绝而不是授予此许可权。

JDBC 驱动程序 (jdbcDriver)

标识 JDBC 驱动程序。

- *library*
  - *file*
  - *fileset*
  - *folder*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

JMS 激活规范 (jmsActivationSpec)

定义 JMS 激活规范配置。

- *authData*
- *properties.wasJms*
- *properties.wmqJms*

属性名称	数据类型	缺省值	描述
authDataRef	对顶级 authData 元素的引用（字符串）。		激活规范的缺省认证数据。
id	字符串		唯一配置标识。
maxEndpoints	整型 最小值：0	500	要分派至的最大端点数。

authData

激活规范的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

properties.wasJms

JMS 激活规范与一个或多个消息驱动的 bean 相关联，并为它们提供接收消息所必需的配置。

false

属性名称	数据类型	缺省值	描述
acknowledgeMode	• Dups-ok-acknowledge • Auto-acknowledge	Auto-acknowledge	确认方式表明应该如何确认消息驱动的 bean 接收的消息。 **Dups-ok-acknowledge** Dups-ok-acknowledge **Auto-acknowledge** Auto-acknowledge
clientId	字符串		所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
connectionFactoryLookup	字符串		此属性可用于指定通过管理方式定义的 javax.jms.ConnectionFactory、javax.jms.QueueConnectionFactory 或 javax.jms.TopicConnectionFactory 对象的查找名称，这些对象用于连接到 JMS 提供者，而端点（消息驱动的 bean）可从该提供者接收消息。
destinationLookup	字符串		此属性可用于指定以管理方式定义的 javax.jms.Queue 或 javax.jms.Topic 对象的查找名称，这些对象定义 JMS 队列或主题，端点（消息驱动的 bean）可从该队列或主题接收消息。

属性名称	数据类型	缺省值	描述
destinationRef	对顶级 adminObject 元素的引用（字符串）。		对 JMS 目标的引用
destinationType	• javax.jms.Topic • javax.jms.Queue	javax.jms.Queue	目标的类型：javax.jms.Queue 或 javax.jms.Topic。 **javax.jms.Topic** javax.jms.Topic **javax.jms.Queue** javax.jms.Queue
maxBatchSize	整型 最小值：1 最大值：2147483647		在一个消息批次中能够从消息传递引擎接收的最大消息数。
maxConcurrency	整型 最小值：1 最大值：2147483647	5	最大端点数，会将消息并行传递给这些端点。增大该数字会提高性能，但是它也会增大在指定时间使用的线程数。如果必须为所有失败的传递维持消息顺序，请将“最大并行端点数”值设置为 1。
messageSelector	字符串		用于确定消息驱动的 bean 所接收消息的 JMS 消息选择器。值是用于选择一小部分可用消息的字符串。
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default

属性名称	数据类型	缺省值	描述
			AlwaysOn AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
retryInterval	精度为秒的时间段	30s	尝试连接至消息传递引擎操作之间的延迟（以秒计），适合初始连接以及建立更好连接的任何后续尝试。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
subscriptionDurability	• DurableShared • Durable • NonDurable • NonDurableShared	NonDurable	MS 主题预订的类型。此值可为下列任一值：Durable DurableShared NonDurable NonDurableShared **DurableShared** DurableShared

属性名称	数据类型	缺省值	描述
			Durable Durable **NonDurable** NonDurable **NonDurableShared** NonDurableShared
subscriptionName	字符串		持久（及共享非持久）所需的预订名。使用持久（及共享非持久）主题预订时的必需字段。此预订名在给定客户机标识内必须唯一。

properties.wmqJms

xigemaMQ JMS 激活规范

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
brokerCCDurSubQueue	字符串		为 ConnectionConsumer 检索非持续预订消息所使用的队列名称
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		已发布消息发送到其中的队列（流队列）的名称

属性名称	数据类型	缺省值	描述
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串	SYSTEM.DEF.SVRCONN	要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE

属性名称	数据类型	缺省值	描述
			NONDUR NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
destinationRef	对顶级 adminObject 元素的引用（字符串）。		目标
destinationType	• javax.jms.Topic • javax.jms.Queue	javax.jms.Queue	目标的类型 - javax.jms.Queue 或 javax.jms.Topic **javax.jms.Topic** javax.jms.Topic **javax.jms.Queue** javax.jms.Queue
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。

属性名称	数据类型	缺省值	描述
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串	localhost	队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
maxMessages	整型 最小值：0	1	一次可分配给服务器会话的最大消息数。如果激活规范正在 XA 事务中向 MDB 传送消息，那么会使用值 1 而不理会此属性的设置。
maxPoolDepth	整型 最小值：0	10	激活规范上的 maxPoolDepth 属性定义可用 MDB（消息驱动的 Bean）实例的数目。减小此属性的值会减少当前可交付的消息数。
messageBatchSize	整型 最小值：0		要在一个批处理中处理的最大消息数。
messageCompression	• RLE • NONE	NONE	可用于压缩连接上的消息数据的方法列表

属性名称	数据类型	缺省值	描述
			RLE RLE **NONE** NONE
messageRetention	布尔型		连接使用者是否保留输入队列中不想要的消息
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
messageSelector	字符串		确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。如果 brokerVersion 值为 1，那么不支持使用代理程序来选择消息
pollingInterval	具有毫秒精度的时间段	5s	此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，

属性名称	数据类型	缺省值	描述
			1s500ms 相当于 1.5 秒。
poolTimeout	具有毫秒精度的时间段	5m	未使用的服务器会话在由于不活动而被关闭前，在服务器会话池中保持打开状态的时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
queueManager	字符串		要连接至的队列管理器的名称
receiveCCSID	整型 最小值：0	0	用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_Q

属性名称	数据类型	缺省值	描述
			MG，否则会忽略此值
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDelivery Sequence 属性所确定的顺序在该队列中搜索合适的消息。xigemaMQ JMS 类找到合适的消息并将其传递给使用者后，xigemaMQ JMS 类将从队列中其当前位置继续搜索下一条合适的消息。xigemaMQ JMS 类将以此方式继续搜索队列，直到它到达队列末尾或者达到此属性值所确定的时间间隔（毫秒）为止。在这两种情况下，xigemaMQ JMS 类将返回到队列开头并

属性名称	数据类型	缺省值	描述
			继续进行搜索，并且新的时间间隔开始。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器
sslCipherSuite	字符串		要用于 SSL 连接的密码套件
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIP

属性名称	数据类型	缺省值	描述
			S 提供者 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数
startTimeout	具有毫秒精度的时间段	10s	配置一个持续时间（毫秒），执行必须在此持续时间内开始。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

属性名称	数据类型	缺省值	描述
subscriptionDurability	• Durable • NonDurable		是使用持久预订还是非持久预订来将消息传递至预订该主题的 MDB **Durable** Durable **NonDurable** NonDurable
subscriptionName	字符串		持久预订的名称
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类用于存储有关活动预订的持久数据的位置。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

JMS 连接工厂 (jmsConnectionFactory)

定义 JMS 连接工厂配置。

- *connectionManager*

- *containerAuthData*
- *properties.wasJms*
- *properties.wmqJms*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将

属性名称	数据类型	缺省值	描述
			多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。

属性名称	数据类型	缺省值	描述
			FailingConnectionOnly 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具

属性名称	数据类型	缺省值	描述
			与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

properties.wasJms

JMS 连接工厂用于创建与 JMS 目标的相关 JMS 提供程序的连接，以便进行点到点消息传递和发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
clientID	字符串	clientID	所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
durableSubscriptionHome	字符串	defaultME	持久预订本地名称定义 ME 名称，需要与该 ME 名称建立连接。
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent **ReliablePersistent** ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default **AlwaysOn** AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。

属性名称	数据类型	缺省值	描述
temporaryQueueNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此队列连接工厂的应用程序来创建的临时队列。
temporaryTopicNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此主题连接工厂的应用程序创建的临时主题。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.wmqJms

xigemaMQ JMS 连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		发送已发布消息的队列（流队列）的名称。
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称

属性名称	数据类型	缺省值	描述
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE **NONDUR** NONDUR **NONE** NONE **STRONG** STRONG

属性名称	数据类型	缺省值	描述
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行。 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口

属性名称	数据类型	缺省值	描述
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。 **RLE** RLE **NONE** NONE
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省密码。（建议使用容器管理的认证别名，而不配置此属性）
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，

属性名称	数据类型	缺省值	描述
			1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
pubAckInterval	整型 最小值：0	25	xigemaMQ JMS 类请求代理程序的应答之前，发布程序发布的消息数
queueManager	字符串		要连接至的队列管理器的名称
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合

属性名称	数据类型	缺省值	描述
			适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序。
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略。
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。

属性名称	数据类型	缺省值	描述
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类是否存储有关活动预订的持久数据。 **MIGRATE** MIGRATE **BROKER** BROKER

属性名称	数据类型	缺省值	描述
			QUEUE QUEUE
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息的 JMSReplyTo 头字段标识的队列的应答消息才具有 MQRFH 2 头。
tempQPrefix	字符串		用于构成 动态队列名称的前缀。
tempTopicPrefix	字符串		创建临时主题时，JMS 会生成一个格式为 TEMP/TEMPTOPIC PREFIX/unique_id 的字符串，或者，如果让此属性保持为缺省值，那么只会生成格式为 TEMP/unique_id 的字符串。指定非空 TEMPTOPICPREFIX 允许定义特定模型队列，以便为在此连接下面创建的临时主题的订户创建受管队列。
temporaryModel	字符串		用来创建 JMS 临时队列的模型队列的名称。JMS 层可以使用 SYSTEM.JMS.TEMPQ.MODEL 来创建可接受持久消息的队列，而缺省值则不能。SYSTEM.DEFAULT.MODEL.QUEUE 只能打开一次。SYSTEM.JMS.TEMPQ.MODEL 可以多次打开。建议不要使用 SYSTEM.DEFAULT.MODEL.QUEUE。

属性名称	数据类型	缺省值	描述
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

JMS 目标 (jmsDestination)

定义 JMS 目标配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。

JMS 队列 (jmsQueue)

定义 JMS 队列配置。

- *properties.wasJms*
- *properties.wmqJms*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。

properties.wasJms

此 JMS 队列分配至的队列的名称。

false

属性名称	数据类型	缺省值	描述
deliveryMode	• NonPersistent • 应用程序 • 持久	应用程序	发送至此目标的消息的传递方式。此选项控制此目标上的消息持久性。 **NonPersistent** NonPersistent 应用程序 应用程序 持久 持久
priority	整型 最小值：0 最大值：9		发送至此目标的消息的相对优先级，范围是 0（最低）到 9（最高）。
queueName	字符串	Default.Queue	关联队列的名称
readAhead	• AlwaysOff • AsConnection • AlwaysOn	AsConnection	预读是一种优化措施，即抢先将消息指定

属性名称	数据类型	缺省值	描述
			给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **AsConnection** AsConnection **AlwaysOn** AlwaysOn
timeToLive	精度为秒的时间段	0s	缺省时间（毫秒），从系统必须使消息在目标中保持活动的分派时间开始算起。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。

properties.wmqJms

xigemaMQ JMS 队列

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	1208	要用于连接或目标的编码字符集标识
arbitraryProperties	字符串		能够指定其他位置未定义的属性
baseQueueManagerName	字符串		定义了此队列的队列管理器名称
baseQueueName	字符串		队列管理器上队列的名称
编码	字符串	NATIVE	将消息发送至此目标时，如何表示该消息的正文中的数字数据。该属性指定二进制整数、压缩十进制整

属性名称	数据类型	缺省值	描述
			数和浮点数的表示法。
expiry	字符串	APP	一个时间段，在该时间段后目标上的消息将到期
failIfQuiesce	布尔型	true	队列管理器处于停顿状态时，对某些方法的调用是否会失败。
persistence	• APP • QDEF • HIGH • NON • PERS	APP	发送至目标的消息的持久性 **APP** APP **QDEF** QDEF **HIGH** HIGH **NON** NON **PERS** PERS
priority	• 3 • 2 • 1 • APP • 0 • 7 • 6 • 5 • QDEF • 4 • 9 • 8	APP	发送至目标的消息的优先级 **3** 3 **2** 2 **1** 1 **APP** APP **0** 0 **7** 7

属性名称	数据类型	缺省值	描述
			6 6 **5** 5 **QDEF** QDEF **4** 4 **9** 9 **8** 8
putAsyncAllowed	• ENABLED • DESTINATION • 已禁用	DESTINATION	是否允许消息生产者使用异步放入来将消息发送至此目标 **ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadAllowed	• ENABLED • DESTINATION • 已禁用	DESTINATION	是否允许 MDB 使用预先读取，以便在接收来自此目标的非持久消息之前将这些消息存储到内部缓冲区 **ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用

属性名称	数据类型	缺省值	描述
readAheadClosePolicy	• CURRENT • ALL	ALL	当管理员停止 MDB 时，内部预先读取缓冲区中的消息发生的情况。 **CURRENT** CURRENT **ALL** ALL
receiveCCSID	整型 最小值：0		用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_QMG，否则会忽略此值
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG
targetClient	• JMS • MQ	JMS	是否使用 xigemaMQ RFH2 格式与目标应用程序交换信息 **JMS** JMS **MQ** MQ

JMS 队列连接工厂 (jmsQueueConnectionFactory)

定义 JMS 队列连接工厂配置。

- *connectionManager*
- *containerAuthData*
- *properties.wasJms*
- *properties.wmqJms*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅

属性名称	数据类型	缺省值	描述
			关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

properties.wasJms

JMS 队列连接工厂用于创建与 JMS 队列的关联 JMS 提供程序的连接，以便进行点到点消息传递。

false

属性名称	数据类型	缺省值	描述
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent **ReliablePersistent** ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default

属性名称	数据类型	缺省值	描述
			AlwaysOn AlwaysOn
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
temporaryQueueNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此队列连接工厂的应用程序来创建的临时队列。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.wmqJms

xigemaMQ JMS 队列连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CC

属性名称	数据类型	缺省值	描述
			DT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
clientId	字符串		连接的客户机标识。
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM **NONE** NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。

属性名称	数据类型	缺省值	描述
			RLE RLE **NONE** NONE
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省密码。（建议使用容器管理的认证别名，而不配置此属性）
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。

属性名称	数据类型	缺省值	描述
			7 7 **6** 6 **unspecified** unspecified
queueManager	字符串		要连接至的队列管理器的名称
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据

属性名称	数据类型	缺省值	描述
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序。
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999		在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息

属性名称	数据类型	缺省值	描述
			的 JMSReplyTo 头字段标识的队列的应答消息才具有 MQRFH 2 头。
tempQPrefix	字符串		用于构成 xigemaMQ 动态队列名称的前缀。
temporaryModel	字符串		用来创建 JMS 临时队列的模型队列的名称。JMS 层可以使用 SYSTEM.JMS.TEMPQ.MODEL 来创建可接受持久消息的队列，而缺省值则不能。SYSTEM.DEFAULT.MODEL.QUEUE 只能打开一次。SYSTEM.JMS.TEMPQ.MODEL 可以多次打开。建议不要使用 SYSTEM.DEFAULT.MODEL.QUEUE。
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）

recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采

属性名称	数据类型	缺省值	描述
			用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

JMS 主题 (jmsTopic)

定义 JMS 主题配置。

- *properties.wasJms*
- *properties.wmqJms*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。

properties.wasJms

在“主题空间”属性定义的主题空间中对此 JMS 主题指定的主题的名称。

false

属性名称	数据类型	缺省值	描述
deliveryMode	• NonPersistent • 应用程序 • 持久	应用程序	发送至此目标的消息的传递方式。此选项控制此目标上的消息持久性。 **NonPersistent** NonPersistent 应用程序 应用程序 持久 持久
priority	整型 最小值：0 最大值：9		发送至此目标的消息的相对优先级，范围是 0（最低）到 9（最高）。

属性名称	数据类型	缺省值	描述
readAhead	• AlwaysOff • AsConnection • AlwaysOn	AsConnection	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **AsConnection** AsConnection **AlwaysOn** AlwaysOn
timeToLive	精度为秒的时间段	0s	缺省时间（毫秒），从系统必须使消息在目标中保持活动的分派时间开始算起。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
topicName	字符串	Default.Topic	在“主题空间”属性定义的主题空间中对此 JMS 主题指定的主题的名称。
topicSpace	字符串	Default.Topic.Space	主题空间是用于进行发布/预订消息传递的位置。

properties.wmqJms

xigemaMQ JMS 主题

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	1208	要用于连接或目标的编码字符集标识
arbitraryProperties	字符串		能够指定其他位置未定义的属性

属性名称	数据类型	缺省值	描述
baseTopicName	字符串		要打开的主题字符串
brokerCCDurSubQueue	字符串	SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE	连接使用者从中检索非持久预订消息的队列的名称
brokerDurSubQueue	字符串	SYSTEM.JMS.D.SUBSCRIBER.QUEUE	检索持续预订消息所使用的队列名称
brokerPubQueue	字符串		已发布消息发送到其中的队列（流队列）的名称
brokerPubQueueManager	字符串		队列管理器的名称，此队列管理器拥有有关该主题的消息发送到的队列
brokerVersion	• 2 • 1	1	要使用的代理程序的版本 **2** 2 **1** 1
编码	字符串	NATIVE	将消息发送至此目标时，如何表示该消息的正文中的数字数据。该属性指定二进制整数、压缩十进制整数和浮点数的表示法。
expiry	字符串	APP	一个时间段，在该时间段后目标上的消息将到期
failIfQuiesce	布尔型	true	队列管理器处于停顿状态时，对某些方法的调用是否会失败。
persistence	• APP • QDEF • HIGH • NON • PERS	APP	发送至目标的消息的持久性 **APP** APP **QDEF** QDEF

属性名称	数据类型	缺省值	描述
			HIGH HIGH **NON** NON **PERS** PERS
priority	• 3 • 2 • 1 • APP • 0 • 7 • 6 • 5 • QDEF • 4 • 9 • 8	APP	发送至目标的消息的优先级 **3** 3 **2** 2 **1** 1 **APP** APP **0** 0 **7** 7 **6** 6 **5** 5 **QDEF** QDEF **4** 4 **9** 9 **8** 8
putAsyncAllowed	• ENABLED • DESTINATION	DESTINATION	是否允许消息生产者使用异步放入来将消息发送至此目标

属性名称	数据类型	缺省值	描述
	• 已禁用		**ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadAllowed	• ENABLED • DESTINATION • 已禁用	DESTINATION	是否允许 MDB 使用预先读取，以便在接收来自此目标的非持久消息之前将这些消息存储到内部缓冲区 **ENABLED** ENABLED **DESTINATION** DESTINATION 已禁用 已禁用
readAheadClosePolicy	• CURRENT • ALL	ALL	当管理员停止 MDB 时，内部预先读取缓冲区中的消息发生的情况。 **CURRENT** CURRENT **ALL** ALL
receiveCCSID	整型 最小值：0		用于对队列管理器消息转换设置目标编码字符集标识的目标属性。除非接收转换设置为 WMQ_RECEIVE_CONVERSION_QMG，否则会忽略此值

属性名称	数据类型	缺省值	描述
receiveConversion	• QMGR • CLIENT_MSG	CLIENT_MSG	用于确定是否将由队列管理器执行数据转换的目标属性。 **QMGR** QMGR **CLIENT_MSG** CLIENT_MSG
targetClient	• JMS • MQ	JMS	是否使用 xigemaMQ RFH2 格式与目标应用程序交换信息 **JMS** JMS **MQ** MQ

JMS 主题连接工厂 (jmsTopicConnectionFactory)

定义 JMS 主题连接工厂配置。

- *connectionManager*
- *containerAuthData*
- *properties.wasJms*
- *properties.wmqJms*
- *recoveryAuthData*

属性名称	数据类型	缺省值	描述
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		连接工厂的连接管理器。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
id	字符串		唯一配置标识。
jndiName	字符串		资源的 JNDI 名称。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

connectionManager

连接工厂的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (

属性名称	数据类型	缺省值	描述
			m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

properties.wasJms

JMS 主题连接工厂用于创建与 JMS 目标的关联 JMS 提供程序的连接，以便进行发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
clientID	字符串	clientID	所有连接上持久（及共享非持久）主题预订所需的 JMS 客户机标识。如果应用程序要执行持久（及共享非持久）发布/预订消息传递，那么需要此标识。
durableSubscriptionHome	字符串	defaultME	持久预订本地名称定义 ME 名称，需要与该 ME 名称建立连接。

属性名称	数据类型	缺省值	描述
nonPersistentMapping	• BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent	ExpressNonPersistent	对使用此连接工厂发送的非持久性 JMS 消息应用的可靠性。 **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
persistentMapping	• AssuredPersistent • ReliablePersistent	ReliablePersistent	对使用此连接工厂发送的持久 JMS 消息应用的可靠性。 **AssuredPersistent** AssuredPersistent **ReliablePersistent** ReliablePersistent
readAhead	• AlwaysOff • Default • AlwaysOn	Default	预读是一种优化措施，即抢先将消息指定给使用者。这会更快地处理使用者请求。 **AlwaysOff** AlwaysOff **Default** Default **AlwaysOn** AlwaysOn

属性名称	数据类型	缺省值	描述
remoteServerAddress	字符串		具有用来连接至引导服务器的三元组（用逗号分隔，语法为 hostName:portNumber:chainName）的远程服务器地址。例如，Merlin:7276:BootstrapBasicMessaging。如果未指定主机名，那么缺省值为 localhost。如果未指定端口号，那么缺省值为 7276。如果未指定链名，那么缺省值为 BootstrapBasicMessaging。有关更多信息，请参阅信息中心。
shareDurableSubscription	字符串		控制持久预订是否可在连接之间共享。
temporaryTopicNamePrefix	字符串	temp	该前缀最多为十二个字符，用于表示使用此主题连接工厂的应用程序创建的临时主题。
userName	字符串		建议使用容器管理的认证别名，而不配置此属性。

properties.wmqJms

xigemaMQ JMS 主题连接工厂

false

属性名称	数据类型	缺省值	描述
CCSID	整型 最小值：1	819	连接的编码字符集标识。
applicationName	字符串		应用程序用于向队列管理器注册的名称。
arbitraryProperties	字符串		能够指定其他位置未定义的属性

属性名称	数据类型	缺省值	描述
brokerCCSubQueue	字符串		连接使用者从中接收非持续预订消息的队列的名称
brokerControlQueue	字符串		代理程序控制队列的名称
brokerPubQueue	字符串		发送已发布消息的队列（流队列）的名称。
brokerQueueManager	字符串		队列管理器的名称，代理程序正在此队列管理器上运行
brokerSubQueue	字符串		非持续消息使用者从中接收消息的队列的名称
brokerVersion	• 2 • 1		要使用的代理程序的版本 **2** 2 **1** 1
ccdtURL	字符串		一个 URL，它标识客户机通道定义表 (CCDT) 所属文件的名称和位置，并指定可以如何访问此文件。
channel	字符串		要使用的 MQI 通道的名称。
cleanupInterval	具有毫秒精度的时间段		在后台两次运行发布/预订清除实用程序之间的时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如

属性名称	数据类型	缺省值	描述
			，1s500ms 相当于 1.5 秒。
cleanupLevel	• SAFE • FORCE • NONDUR • NONE • STRONG	SAFE	基于代理程序的预订存储的清除级别。 **SAFE** SAFE **FORCE** FORCE **NONDUR** NONDUR **NONE** NONE **STRONG** STRONG
clientId	字符串		连接的客户机标识
cloneSupport	• ENABLED • 已禁用	已禁用	同一持久主题订户的两个或更多实例是否可以同时运行。 **ENABLED** ENABLED 已禁用 已禁用
connectionNameList	字符串		用于通信的 TCP/IP 连接名称（主机名（端口））的列表。ConnectionNameList 将取代主机名和端口属性。
failIfQuiesce	布尔型	true	如果队列管理器处于停顿状态，那么指示对某些方法的调用是否会失败。
headerCompression	• SYSTEM • NONE	NONE	可用于在连接时压缩头数据的方法列表 **SYSTEM** SYSTEM

属性名称	数据类型	缺省值	描述
			NONE NONE
hostName	字符串		队列管理器所在系统的主机名或 IP 地址。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
localAddress	字符串		要连接至队列管理器，此属性指定下列一项或两项：(1) 要使用的本地网络接口；(2) 要使用的本地端口或者某个范围的本地端口
messageCompression	• RLE • NONE	NONE	可用于在连接时压缩消息数据的方法列表。 **RLE** RLE **NONE** NONE
messageSelection	• CLIENT • BROKER	CLIENT	确定消息选择是由 xigemaMQ JMS 类还是代理程序完成。 **CLIENT** CLIENT **BROKER** BROKER
password	可逆向编码的密码（字符串）		创建与队列管理器的连接时要使用的缺省密码。（建议使用容器管理的认证别名，而不配置此属性）
pollingInterval	具有毫秒精度的时间段		此值是以毫秒计的最大时间间隔，如果会话中的每个消息侦听器在其队列中都没有

属性名称	数据类型	缺省值	描述
			合适的消息，那么在此时间过后，每个消息侦听器都将再次尝试从其队列中获取消息。如果在一个会话中频繁地发生任何消息侦听器都没有适当的消息可用，那么应考虑增大此属性的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
port	整型 最小值：1	1414	队列管理器侦听的端口。指定了 ConnectionNameList 属性时，主机名和端口属性将被 ConnectionNameList 属性取代。
providerVersion	• 7 • 6 • unspecified	unspecified	应用程序打算连接的队列管理器的版本、发行版、修改级别和修订包。 **7** 7 **6** 6 **unspecified** unspecified
pubAckInterval	整型 最小值：0	25	xigemaMQ JMS 类请求代理程序的应答之前，发布程序发布的消息数
queueManager	字符串		要连接至的队列管理器的名称

属性名称	数据类型	缺省值	描述
receiveExit	字符串		标识一个通道接收出口程序或一系列要连续运行的接收出口程序
receiveExitInit	字符串		调用通道接收出口程序时，传递到这些程序的用户数据
rescanInterval	具有毫秒精度的时间段	5s	当点到点域中的消息使用者使用消息选择器来选择所要接收的消息时，xigemaMQ JMS 类将按 xigemaMQ 队列的 MsgDeliverySequence 属性所确定的顺序在该队列中搜索合适的消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securityExit	字符串		标识通道安全性出口程序
securityExitInit	字符串		调用通道安全性出口程序时，传递到该程序的用户数据
sendCheckCount	整型 最小值：0		在单个非事务 JMS 会话中，允许在两次检查异步放置错误之间发送的调用数。
sendExit	字符串		标识一个通道发送出口程序，或者要连续运行的一系列发送出口程序。

属性名称	数据类型	缺省值	描述
sendExitInit	字符串		调用通道发送出口程序时，传递至这些程序的用户数据。
shareConvAllowed	布尔型	true	如果通道定义匹配，客户机连接是否可以与从同一个流程至同一个队列管理器的其他顶级 JMS 连接共享其套接字
sparseSubscriptions	布尔型	false	控制 TopicSubscriber 对象的消息检索策略。
sslCertStores	字符串		拥有要在 SSL 连接时使用的证书撤销列表 (CRL) 的轻量级目录访问协议 (LDAP) 服务器。
sslCipherSuite	字符串		要用于 SSL 连接的密码套件。
sslFipsRequired	布尔型		SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的密码套件。
sslPeerName	字符串		对于 SSL 连接，这是用来检查由队列管理器提供的数字证书中的专有名称的模板。
sslResetCount	整型 最小值：0 最大值：999999999	0	在重新协商 SSL 所使用的密钥之前，SSL 连接所发送和接收的字节总数。
statusRefreshInterval	具有毫秒精度的时间段	1m	这是以毫秒计的时间间隔，用于检测订户与队列管理器之间的连接是否中断的长时间运行事务将按此时间间隔执行刷新。仅当预订存储器的值为 QUEUE 时，此属性才起作用。指定后跟

属性名称	数据类型	缺省值	描述
			时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
subscriptionStore	• MIGRATE • BROKER • QUEUE	BROKER	确定 xigemaMQ JMS 类是否存储有关活动预订的持久数据。 **MIGRATE** MIGRATE **BROKER** BROKER **QUEUE** QUEUE
targetClientMatching	布尔型	true	是否仅当入局消息具有 MQRFH2 头时，发送至由该入局消息的 JMSReplyTo 头字段标识的队列的应答消息才具有 MQRFH2 头。
tempTopicPrefix	字符串		创建临时主题时，JMS 会生成一个格式为 TEMP/TEMPTOPICPREFIX/unique_id 的字符串，或者，如果让此属性保持为缺省值，那么只会生成格式为 TEMP/unique_id 的字符串。指定非空 TEMPTOPICPREFIX 允许定义特定模型队列，以便为在此连接下面创建的临时主题的订户创建受管队列。

属性名称	数据类型	缺省值	描述
transportType	• CLIENT • BINDINGS	CLIENT	与队列管理器的连接是使用客户机方式还是使用绑定方式。 **CLIENT** CLIENT **BINDINGS** BINDINGS
userName	字符串		创建与队列管理器的连接时要使用的缺省用户名。（建议使用容器管理的认证别名，而不配置此属性）
wildcardFormat	• CHAR • TOPIC	TOPIC	要使用哪个版本的通配符语法。 **CHAR** CHAR **TOPIC** TOPIC

recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

JNDI 条目 (jndiEntry)

JNDI 缺省名称空间中的单个条目。

属性名称	数据类型	缺省值	描述
decode	布尔型	false	如果在查找时需要对值进行解码，那么为 true。
id	字符串		唯一配置标识。
jndiName	字符串		要用于此条目的 JNDI 名称。
value	字符串		要与名称相关联的 JNDI 值。

JNDI 对象工厂 (jndiObjectFactory)

要由 JNDI 引用条目使用的 ObjectFactory。

- *library*
  - *file*
  - *fileset*
  - *folder*

属性名称	数据类型	缺省值	描述
className	字符串		ObjectFactory 实现类名。
id	字符串		唯一配置标识。
libraryRef	对顶级库元素的引用（字符串）。		包含工厂实现类的库。
objectClassName	字符串	java.lang.Object	从工厂返回的对象类型。

library

包含工厂实现类的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述

属性名称	数据类型	缺省值	描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示

属性名称	数据类型	缺省值	描述
			秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

JNDI 引用条目 (jndiReferenceEntry)

JNDI 缺省名称空间中的引用条目。

- *factory*
  - *library*
    - *file*
    - *fileset*
    - *folder*
- *properties*

属性名称	数据类型	缺省值	描述
factoryRef	对顶级 jndiObjectFactory 元素的引用（字符串）。		引用条目的对象工厂。
id	字符串		唯一配置标识。
jndiName	字符串		引用条目的 JNDI 名称。

factory

引用条目的对象工厂。

false

属性名称	数据类型	缺省值	描述
className	字符串		ObjectFactory 实现类名。
libraryRef	对顶级库元素的引用（字符串）。		包含工厂实现类的库。
objectClassName	字符串	java.lang.Object	从工厂返回的对象类型。

factory > library

包含工厂实现类的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

factory > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

factory > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

factory > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

properties

引用条目的属性。

false

JNDI URL 条目 (jndiURLEntry)

JNDI 缺省名称空间中的单个条目，用于绑定 java.net.URL 条目。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
jndiName	字符串		要用于此条目的 JNDI 名称。
value	字符串		要与该名称相关联的 JNDI URL 值。

JPA 容器 (jpa)

Java Persistence API 容器的配置属性。

- *excludedApplication*

属性名称	数据类型	缺省值	描述
defaultJtaDataSourceJndiName	字符串		要供此服务器中运行的应用程序使用的缺省 Java™ Transaction API (JTA) 数据源 JNDI 名称。缺省情况下，数据源为 JTA。此字段只接受事务性的数据源。
defaultNonJtaDataSourceJndiName	字符串		要供此服务器中运行的应用程序使用的缺省非事务性数据源 JNDI 名称。此字段只接受已标记为非事务性的数据源。
defaultPersistenceProvider	字符串		缺省持久性提供程序类名。如果未指定此属性，那

属性名称	数据类型	缺省值	描述
			么缺省提供程序依赖于所启用的 JPA 功能部件。
entityManagerPoolCapacity	整型	-1	每个 PersistenceContext 引用的 EntityManager 池容量。最小值为 0，最大值为 500。
ignoreDataSourceErrors	布尔型		如果为 true，那么尝试查找 persistence.xml 文件中的 <jta-data-source> 或 <non-jta-data-source> 元素指定的数据源时发生的错误将被报告并被忽略，这允许持久性提供程序确定缺省数据源。如果为 false，那么系统会将错误传播至持久性提供程序，以便系统可更轻松地诊断错误，但错误配置的应用程序可能不工作。缺省情况下，如果已启用 JPA 2.0，那么此属性为 true，否则为 false。

excludedApplication

要从 JPA 处理中排除的应用程序。

false

字符串

JSP 引擎 (jspEngine)

JSP 2.2 配置

属性名称	数据类型	缺省值	描述
disableJspRuntimeCompilation	布尔型	false	在运行时禁止 JSP 的编译。
disableResourceInjection	布尔型	false	禁止将资源注入 JSP 中。
extendedDocumentRoot	字符串		JSP 引擎将在其中搜索要使用的其他 JSP 文件的目录。
jdkSourceLevel	• 17 • 18 • 15	15	JSP 引擎编译的 JSP 的缺省 Java 源级别。

属性名称	数据类型	缺省值	描述
	• 16 • 13 • 14		**17** 17 **18** 18 **15** 15 **16** 16 **13** 13 **14** 14
keepGenerated	布尔型	false	保留为 JSP 生成的 Java 源文件。
prepareJSPs	整型		如果存在此属性，那么大于该值的所有 JSP（以千字节计）会在应用程序服务器启动时编译。将此值设为 0 以编译所有 JSP。
recompileJspOnRestart	布尔型	false	在重新启动应用程序之后，重新编译 JSP。JSP 在第一次访问时重新编译。
useImplicitTagLibs	布尔型	true	允许 JSP 使用 jsx 和 tsx 标记库。
useInMemory	布尔型	false	在内存中生成 Java 源和类（不写入磁盘）。

密钥库 (keyStore)

用于 SSL 加密的安全证书的存储库。

- *keyEntry*

属性名称	数据类型	缺省值	描述
fileBased	布尔型	true	如果密钥库基于文件，请指定 true；如果密钥库的类型是 SAF 密钥环或硬件密钥库，请指定 false。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
location	文件路径	${server.output.dir}/resources/security/key.jks	密钥库文件的绝对或相对路径。如果提供了相对路径，那么服务器将尝试在 ${server.config.dir}/resources/security 目录中查找该文件。将密钥库文件用于基于文件的密钥库，将密钥环名称用于 SAF 密钥环，或者将设备配置文件用于硬件加密设备。在 SSL 最低配置中，会假定该文件的位置为 ${server.config.dir}/resources/security/key.jks。
password	可逆向编码的密码（字符串）		用于装入密钥库文件的密码。可以采用明文或编码格式存储该值。使用 securityUtility 工具来对该密码进行编码。
pollingRate	具有毫秒精度的时间段	500ms	服务器检查密钥库文件更新的速率。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
readOnly	布尔型	false	如果密钥库要由服务器用于读取，请指定 true；如果服务器将对密钥库执行写操作，请指定 false。
type	字符串	jks	受目标 SDK 支持的密钥库类型。
updateTrigger	• mbean • polled • disabled	mbean	密钥库文件更新方法或触发器。 **mbean** 仅当 FileNotificationMbean 提示进行更新时，服务器才会更新密钥库。FileNotificationMbea

属性名称	数据类型	缺省值	描述
			n 通常由外部程序（例如，集成开发环境或管理应用程序）调用。 **polled** 服务器将按照轮询时间间隔扫描密钥库文件更改，并在密钥库文件具有可检测更改时进行更新。 **disabled** 禁用所有更新监视。在服务器处于运行状态时，将不会应用对密钥库文件的更改。

keyEntry

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
keyPassword	可逆向编码的密码（字符串）		密钥库中的专用密钥条目的密码。
name	字符串		密钥库中的专用密钥条目的名称。

LDAP 用户注册表 (ldapRegistry)

LDAP 用户注册表的配置属性。

- *activedFilters*
- *attributeConfiguration*
  - *attribute*
  - *externalIdAttribute*
- *contextPool*
- *customFilters*
- *domino50Filters*
- *edirectoryFilters*
- *failoverServers*

属性名称	数据类型	缺省值	描述
actived FiltersRef	对顶级 actived LdapFilterProperties 元素的引用（字符串）。		指定缺省 Microsoft Active Directory LDAP 过滤器的列表。
baseDN	字符串		目录服务的基本专有名称 (DN)，用于指示目录服务中 LDAP 搜索的起始点。
bindDN	字符串		应用程序服务器的专有名称 (DN)，用于绑定到目录服务。
bindPassword	可逆向编码的密码（字符串）		绑定 DN 的密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
certificateFilter	字符串		为 LDAP 过滤器指定过滤器证书映射属性。使用过滤器将客户机证书中的属性映射至 LDAP 注册表中的条目。例如，可以将过滤器指定为：uid=${SubjectCN}。
certificateMapMode	• EXACT_DN • CERTIFICATE_FILTER		指定是使用 EXACT_DN 还是 CERTIFICATE_FILTER 将 X.509 证书映射到 LDAP 目录。指定 CERTIFICATE_FILTER 以使用指定的证书过滤器进行映射。

属性名称	数据类型	缺省值	描述
			EXACT_DN exactDN **CERTIFICATE_FILTER** certFilter
connectTimeout	具有毫秒精度的时间段	1m	用于建立至 LDAP 服务器的连接的最长时间。如果指定的时间到期，那么将记录错误消息。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
customFiltersRef	对顶级 customLdapFilterProperties 元素的引用（字符串）。		指定缺省定制 LDAP 过滤器的列表。
domino50FiltersRef	对顶级 domino50LdapFilterProperties 元素的引用（字符串）。		指定缺省 IBM Lotus Domino LDAP 过滤器的列表。
edirectoryFiltersRef	对顶级 edirectoryLdapFilterProperties 元素的引用（字符串）。		指定 Novell eDirectory LDAP 过滤器的列表。
host	字符串		LDAP 服务器的地址，采用 IP 地址或域名服务 (DNS) 名称的形式。
id	字符串		唯一配置标识。
idsFiltersRef	对顶级 idsLdapFilterProperties 元素的引用（字符串）。		指定缺省 IBM Tivoli Directory Server LDAP 过滤器的列表。
ignoreCase	布尔型	true	执行不区分大小写的认证检查。
iplanetFiltersRef	对顶级 iplanetLdapFilterProperties 元素的引用（字符串）。		指定缺省 Sun Java System Directory Server LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
ldapType	• Sun Java System Directory Server • Netscape Directory Server • Microsoft Active Directory • IBM Tivoli Directory Server • IBM Lotus Domino • 定制 • IBM SecureWay Directory Server • Novell eDirectory		LDAP 服务器的类型，将建立至该服务器的连接。 **Sun Java System Directory Server** iplanet **Netscape Directory Server** netscape **Microsoft Active Directory** actived **IBM Tivoli Directory Server** ibm_dir_server **IBM Lotus Domino** domino50 定制 定制 **IBM SecureWay Directory Server** secureway **Novell eDirectory** edirectory
netscapeFiltersRef	对顶级 netscapeLdapFilterProperties 元素的引用（字符串）。		指定缺省 Netscape Directory Server LDAP 过滤器的列表。
port	整型		LDAP 服务器的端口号。
领域	字符串	LdapRegistry	表示用户注册表的领域名。
recursiveSearch	布尔型	false	执行嵌套组搜索。仅当 LDAP 服务器不支持递归服务器端搜索时，才应该选择此选项。
returnToPrimaryServer	布尔型	true	一个布尔值，用于指示是否应对主服务器执行搜索。

属性名称	数据类型	缺省值	描述
reuseConnection	布尔型	true	请求应用程序服务器复用 LDAP 服务器连接。
searchTimeout	具有毫秒精度的时间段	1m	在取消请求之前供 LDAP 服务器作出响应的最长时间。这相当于建立连接之后的读取超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
securewayFiltersRef	对顶级 securewayLdapFilterProperties 元素的引用（字符串）。		指定缺省 IBM SecureWay Directory Server LDAP 过滤器的列表。
sslEnabled	布尔型	false	指示是否应该建立至 LDAP 服务器的 SSL 连接。
sslRef	字符串		要用于连接至启用 SSL 的 LDAP 服务器的 SSL 配置的标识。

activedFilters

指定缺省 Microsoft Active Directory LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectcategory=group))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	memberOf:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(sAMAccountName=%v)(objectcategory=user))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。

属性名称	数据类型	缺省值	描述
userIdMap	字符串	user:sAMAccountName	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

attributeConfiguration

此配置映射带有用户注册表模式（例如，Person、PersonAccount 或 Group）字段名的 LDAP 属性。

false

attributeConfiguration > attribute

定义要映射至 LDAP 属性的用户注册表模式字段名。

false

属性名称	数据类型	缺省值	描述
defaultValue	字符串		属性的缺省值。
entityType	字符串		属性的实体类型。
id	字符串		唯一配置标识。
name	字符串		LDAP 属性的名称。
propertyName	字符串		需要使用 LDAP 属性映射的用户注册表模式字段名。
syntax	字符串		属性语法。

attributeConfiguration > externalIdAttribute

定义 LDAP 属性的名称及其特性，此属性需要映射至用户注册表 externalId 属性。

false

属性名称	数据类型	缺省值	描述
autoGenerate	布尔型	false	启用了此项时，externalId 属性值由用户注册表自动生成，而不是使用 LDAP 中存储的值。缺省情况下禁用该项。
entityType	字符串		属性的实体类型。
id	字符串		唯一配置标识。
name	字符串		要用于用户注册表 externalId 属性的 LDAP 属性的名称。

属性名称	数据类型	缺省值	描述
syntax	字符串		属性语法。

contextPool

上下文池的属性。

false

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	一个布尔值，用于确定是否启用了上下文池。禁用它可能导致性能下降。
initialSize	整型	1	一个整数值，用于确定上下文池的初始大小。根据存储库上的负载来进行此设置。
maxSize	整型	0	用于定义上下文池的最大大小的整数值。请根据存储库上的最大负载来设置此值。
preferredSize	整型	3	上下文池的首选大小。根据存储库上的负载来进行此设置。
timeout	具有毫秒精度的时间段	0s	上下文池超时之前的持续时间。一个整数，用于表示空闲上下文实例可在池中保留而不会被关闭并从池中移除的时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
waitTime	具有毫秒精度的时间段	3s	上下文池超时之前的持续时间。一个时间间隔，上下文实例数达到最大池大小时，请求会等待此时间间

属性名称	数据类型	缺省值	描述
			隔直到上下文池再次检查池中是否有空闲上下文实例。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

customFilters

指定缺省定制 LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

domino50Filters

指定缺省 IBM Lotus Domino LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=dominoGroup))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	dominoGroup:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=Person))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	person:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

edirectoryFilters

指定 Novell eDirectory LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=groupOfNames))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	groupOfNames:member	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(cn=%v)(objectclass=Person))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	person:cn	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

failoverServers

LDAP 故障转移服务器的列表。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		LDAP 故障转移服务器的配置属性。将其指定为主 LDAP 服务器的备份服务器。例如，<failoverServers name="failoverLdapServers"><server host="myfullyqualifiedhostname1" port="389"/><server host="myfullyqualifiedhostname2" port="389"/></failoverServers>。

failoverServers > server

LDAP 故障转移服务器的配置属性。

false

属性名称	数据类型	缺省值	描述
host	字符串		LDAP 服务器主机名，它可以使用 IP 地址或域名服务 (DNS) 名称。
id	字符串		唯一配置标识。
port	整型		LDAP 故障转移服务器端口。

idsFilters

指定缺省 IBM Tivoli Directory Server LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。

属性名称	数据类型	缺省值	描述
groupMemberIdMap	字符串	ibm-allGroups:member;ibm-allGroups:uniqueMember;groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

iplanetFilters

指定缺省 Sun Java System Directory Server LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(objectclass=ldapsubentry))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	nsRole:nsRole	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=inetOrgPerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	inetOrgPerson:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

ldapCache

配置高速缓存的属性。

false

ldapCache > attributesCache

属性高速缓存特性配置。

false

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	用于指示是否已启用此属性的布尔值。
serverTTLAttribute	字符串		一个时间，高速缓存条目在此时间后到期。系统将直接从服务器中访存针对此条目的后续调用，并再次将其放在高速缓存中。
size	整型	2000	定义可存储在高速缓存中的实体数目。您可以根据需要存储在高速缓存中的实体数目，增大高速缓存的大小。
sizeLimit	整型	2000	高速缓存的大小限制。
timeout	具有毫秒精度的时间段	1200ms	定义 LDAP 属性高速缓存的内容可用的最长时间。经过指定的时间之后，将会清除 LDAP 属性高速缓存。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

ldapCache > searchResultsCache

搜索结果高速缓存的配置。

false

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	用于指示是否已启用此属性的布尔值。
resultsSizeLimit	整型	2000	在搜索中可以返回的最大结果数。

属性名称	数据类型	缺省值	描述
size	整型	2000	高速缓存的大小。高速缓存中存储的搜索结果数。需要根据系统上执行的搜索查询数及可用硬件系统资源来配置此大小。
timeout	具有毫秒精度的时间段	1200ms	定义搜索结果高速缓存的内容可用的最长时间。经过指定的时间之后，将会清除搜索结果高速缓存。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

ldapEntityType

为个人、组和组织单位配置 LDAP 对象类、搜索过滤器、搜索条件和 LDAP 相对专有名称 (RDN)。例如，Group 实体类型可具有搜索过滤器（例如，(&(ObjectCategory=Groupofnames)(ObjectClass=Groupofnames))）及对象类（例如，Groupofnames），搜索条件为 ou=iGroups,o=vsettan,c=cn。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		LDAP 实体类型的名称。
searchFilter	字符串		搜索实体类型时使用的定制 LDAP 搜索表达式。例如，searchFilter="(|(ObjectCategory=User)(ObjectClass=User))"。

ldapEntityType > objectClass

为 LDAP 服务器中的给定 LDAP 实体类型定义的对象类。例如，组 LDAP 实体类型的对象类可以是 Groupofnames。

false

字符串

ldapEntityType > searchBase

为给定实体类型指定搜索调用的 LDAP 服务器的子树，该子树将覆盖搜索操作中的基本 DN 。例如，如果基本 DN 为 o=vsettan,c=cn，并且个人帐户实体类型的搜索条件已定义为 ou=iUsers,o=vsettan,c=cn，那么将在子树 ou=iUsers,o=vsettan,c=cn 下执行对个人帐户的所有搜索调用。可以为同一实体类型配置多个搜索条件。

false

字符串

netscapeFilters

指定缺省 Netscape Directory Server LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=inetOrgPerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	inetOrgPerson:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

securewayFilters

指定缺省 IBM SecureWay Directory Server LDAP 过滤器的列表。

false

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。

属性名称	数据类型	缺省值	描述
groupMemberIdMap	字符串	groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

共享库 (library)

共享库

- *file*
- *fileset*
- *folder*

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

日志记录 (logging)

控制日志和跟踪消息的捕获及输出。

属性名称	数据类型	缺省值	描述
consoleLogLevel	• ERROR • WARNING • AUDIT • OFF • INFO	AUDIT	用于对写入系统流的消息进行过滤的记录级别。缺省值为 audit。 **ERROR** 会将错误消息写入系统错误流。 **WARNING** 会将警告消息写入系统输出流。会将错误消息写入系统错误流。 **AUDIT** 会将审计消息和警告消息写入系统输出流。会将错误消息写入系统错误流。 **OFF** 不会将服务器输出写入系统流。仅将 JVM 输出写入系统流。 **INFO** 会将参考消息、审计消息和警告消息写入系统输出流。会将错误消息写入系统错误流。
copySystemStreams	布尔型	true	如果值为 true，那么会将 System.out 写入系统输出

属性名称	数据类型	缺省值	描述
			流，会将 System.err 写入系统错误流。如果值为 false，那么会将 System.out 和 System.err 写入所配置的日志（例如，messages.log 或 trace.log），但是不会写入系统流。缺省值为 true。
hideMessage	字符串		配置为在 console.log 和 message.log 文件中隐藏的消息的逗号分隔列表。如果这些消息配置为隐藏，那么它们会重定向至 trace.log 文件。
logDirectory	目录路径	${server.output.dir}/logs	日志文件的目录位置。缺省值为 ${server.output.dir}/logs。
maxFileSize	整型 最小值：0	20	回滚之前日志文件的最大大小，以兆字节计；值为 0 时意味着无限制。
maxFiles	整型 最小值：0	2	移除最旧的日志文件之前将保留的最大日志文件数；值为 0 时意味着无限制。
messageFileName	字符串	messages.log	会将消息输出写入的文件的名称（相对于所配置的日志目录）。缺省值为 messages.log。
suppressSensitiveTrace	布尔型	false	跟踪隐式类型数据（例如通过网络连接收到的字节）时，服务器跟踪可能会暴露敏感数据。如果设为 true，那么可防止潜在的敏感信息暴露在日志和跟踪文件中。缺省值为 false。
traceFileName	字符串	trace.log	会将跟踪输出写入的文件的名称（相对于所配置的日志目录）。缺省值为 trace.log。
traceFormat	• ENHANCED	ENHANCED	此格式用于跟踪日志。

属性名称	数据类型	缺省值	描述
	• BASIC • ADVANCED		**ENHANCED** 使用增强型基本跟踪格式。 **BASIC** 使用基本跟踪格式。 **ADVANCED** 使用高级跟踪格式。
traceSpecification	字符串	*=info	符合跟踪规范语法并且指定各种跟踪组件的初始状态的跟踪规范。接受空值，并将空值视为“禁止所有跟踪”。未指定的任何组件都将初始化为缺省状态 *=info。

Logstash 收集器 (logstashCollector)

Logstash 收集器从各种源收集数据，并使用 Lumberjack 协议将这些数据转发到 Logstash 服务器。

- 源

属性名称	数据类型	缺省值	描述
hostName	字符串		Logstash 服务器的主机名。
port	整型 最小值：1 最大值：65535		Logstash 服务器的端口号。
sslRef	字符串		指定用于连接至 Logstash 服务器的 SSL 指令表的标识。

source

指定要由 Logstash 收集器使用的源。

false

字符串

LTPA 令牌 (ltpa)

轻量级第三方认证 (LTPA) 令牌配置。

属性名称	数据类型	缺省值	描述
expiration	具有分钟精度的时间段	120m	令牌到期之前的时间量，以分钟计。指定后跟时间单位的正整数，时间单位可以是小时 (h) 或分钟 (m)。例如，将 30 分钟指定为 30m。可以将多个值包括在单个条目中。例如，1h30m 相当于 90 分钟。
keysFileName	文件路径	${server.output.dir}/resources/security/ltpa.keys	包含令牌密钥的文件的路径。
keysPassword	可逆向编码的密码（字符串）	{xor}CDo9Hgw=	令牌密钥的密码。可以采用明文或编码格式存储该值。建议将 securityUtility 工具与编码选项配合使用以对该密码进行编码。
monitorInterval	具有毫秒精度的时间段	0ms	服务器检查 LTPA 令牌密钥文件更新的频率。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

邮件会话对象 (mailSession)

邮件会话实例的配置。

- *property*

属性名称	数据类型	缺省值	描述
description	字符串		邮件会话的描述
from	字符串		用于通过邮件会话实例发送邮件的电子邮件地址。
host	字符串		邮件会话的主机
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
jndiName	字符串		用于 JNDI 查找的邮件会话引用的名称
mailSessionID	字符串		特定邮件会话实例的标识
password	可逆向编码的密码（字符串）		用户的密码，通常需要此密码以便连接至主机。
storeProtocol	字符串	imap	由邮件会话实例使用的存储协议。缺省存储协议为 IMAP。
transportProtocol	字符串	smtp	由邮件会话实例使用的传输协议。缺省传输协议为 SMTP。
user	字符串		主机上使用的用户电子邮件地址。

property

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		额外属性的名称。
value	字符串		与名称匹配的属性的值。

受管执行程序 (managedExecutorService)

受管执行程序服务

- *contextService*
  - *baseContext*
    - *baseContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *classloaderContext*
  - *jeeMetadataContext*
  - *securityContext*
  - *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。	DefaultContextService	配置将上下文传播至线程的方式
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称

contextService

配置将上下文传播至线程的方式

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。

属性名称	数据类型	缺省值	描述
			WARN 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。

属性名称	数据类型	缺省值	描述
			WARN 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

contextService > classloaderContext

类装入器上下文传播配置。

false

contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

受管计划执行程序 (managedScheduledExecutorService)

受管计划执行程序服务

- *contextService*
  - *baseContext*
    - *baseContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *classloaderContext*
  - *jeeMetadataContext*
  - *securityContext*
  - *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。	DefaultContextService	配置将上下文传播至线程的方式
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称

contextService

配置将上下文传播至线程的方式

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。

属性名称	数据类型	缺省值	描述
			IGNORE 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。

属性名称	数据类型	缺省值	描述
			IGNORE 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

contextService > classloaderContext

类装入器上下文传播配置。

false

contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

受管线程工厂 (managedThreadFactory)

受管线程工厂

- *contextService*
  - *baseContext*
    - *baseContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *classloaderContext*
  - *jeeMetadataContext*
  - *securityContext*
  - *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。	DefaultContextService	配置将上下文传播至线程的方式
createDaemonThreads	布尔型	false	配置由受管线程工厂创建的线程是否应该为守护程序线程。
defaultPriority	整型 最小值：1 最大值：10		由受管线程工厂创建的线程的缺省优先级。如果未指定，那么会使用创建线程的优先级。优先级不能超过受管线程工厂的最大优先级，在此情况下，会改为使用最大优先级。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
jndiName	字符串		JNDI 名称
maxPriority	整型 最小值：1 最大值：10		由受管线程工厂创建的线程的最大优先级。

contextService

配置将上下文传播至线程的方式

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。

属性名称	数据类型	缺省值	描述
			WARN 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。

属性名称	数据类型	缺省值	描述
			WARN 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

contextService > classloaderContext

类装入器上下文传播配置。

false

contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

消息传递引擎 (messagingEngine)

消息传递引擎是在服务器内部运行的组件，用于管理消息传递资源。应用程序发送和接收消息时会连接至消息传递引擎。

- *alias*
- *fileStore*
- *messagingSecurity*
  - *role*
    - *group*
    - *queuePermission*
      - *action*
    - *tempDestinationPermission*
      - *action*
    - *topicPermission*
      - *action*
    - *user*
- *queue*
- *topicSpace*

alias

别名目的地将映射总线目标的备用名称。可以将别名目标用于点到点消息传递或发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
forceReliability	• AssuredPersistent • BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent • ReliablePersistent	AssuredPersistent	当生产者未设置显式可靠性时要对面向此目标生成的消息指定的可靠性。 **AssuredPersistent** AssuredPersistent **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent

属性名称	数据类型	缺省值	描述
			ExpressNonPersistent ExpressNonPersistent **ReliablePersistent** ReliablePersistent
id	字符串		别名队列或别名主题空间的名称。
sendAllowed	• false • true	true	生产者可以将消息发送至此目标。 **false** false **true** true
targetDestination	字符串	Default.Queue	目标目的地参数标识一个目标，该目标可能与别名目的地在同一总线内。缺省情况下，如果未设置属性，那么它将指向 Default.Queue。

fileStore

消息传递文件存储库。

false

属性名称	数据类型	缺省值	描述
fileStoreSize	长整型 最小值：20	400	永久和临时存储器的组合大小（以兆字节计）。文件存储器大小在永久存储器与临时存储器之间平均分配。例如，如果指定文件存储器大小为 400 MB，那么 200 MB 用于永久存储器，200 MB 用于临时存储器。

属性名称	数据类型	缺省值	描述
logFileSize	长整型 最小值：10	10	日志文件的大小，以兆字节计。日志文件大小最多只能为文件存储器大小的一半。建议的日志文件大小为文件存储器大小的 25%。例如，如果文件存储器大小设置为 400 MB，那么日志文件大小不能超过 200 MB。在此示例中，建议的日志文件大小值将为 100 MB。
path	字符串	${server.output.dir}/messaging/messageStore	文件存储的路径。

messagingSecurity

wasJmsServer-1.0 功能部件的安全性。

false

messagingSecurity > role

映射至用户和组的一组许可权

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		角色的名称

messagingSecurity > role > group

分配给该角色的组

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		定义为用户注册表的一部分的组

messagingSecurity > role > queuePermission

为一组用户和组定义的队列许可权

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
queueRef	字符串		对消息传递引擎中已定义的队列的引用

messagingSecurity > role > queuePermission > action

目标中允许的操作

false

messagingSecurity > role > tempDestinationPermission

为一组用户和组定义的临时目标许可权

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
prefix	字符串		为临时目标定义的前缀

messagingSecurity > role > tempDestinationPermission > action

目标中允许的操作

false

messagingSecurity > role > topicPermission

为一组用户和组定义的主题许可权

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
topicName	字符串		主题空间内主题的名称
topicSpaceRef	字符串	Default.Topic.Space	对消息传递引擎中定义的主题空间的引用

messagingSecurity > role > topicPermission > action

目标中允许的操作

false

messagingSecurity > role > user

分配给特定角色的用户

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		定义为注册表的一部分的用户

queue

队列目标表示消息队列，并用于点到点消息传递。

false

属性名称	数据类型	缺省值	描述
exceptionDestination	字符串	_SYSTEM.Exception.Destination	消息无法传递至此目标时，系统会将其转发至的目标。
failedDeliveryPolicy	• KEEP_TRYING • SEND_TO_EXCEPTION_DESTINATION • DISCARD	SEND_TO_EXCEPTION_DESTINATION	列示当达到消息的 maxredeliverycount 时，消息传递引擎必须执行的操作。 **KEEP_TRYING** KEEP_TRYING **SEND_TO_EXCEPTION_DESTINATION** SEND_TO_EXCEPTION_DESTINATION **DISCARD** DISCARD
forceReliability	• AssuredPersistent • BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent • ReliablePersistent	AssuredPersistent	当生产者未设置显式可靠性时要对面向此目标生成的消息指定的可靠性。 **AssuredPersistent** AssuredPersistent **BestEffortNonPersistent** BestEffortNonPersistent

属性名称	数据类型	缺省值	描述
			ReliableNonPersistent ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent **ReliablePersistent** ReliablePersistent
id	字符串		队列的名称。
maintainStrictOrder	布尔型	false	维持生产者向目的地发送消息的顺序。
maxMessageDepth	长整型 最小值：1	50000	消息传递引擎可在其消息点上放置的最大消息数。
maxRedeliveryCount	整型	5	消息的处理尝试最大失败次数。在此失败尝试次数之后，如果配置了异常目标，那么将把消息从预期目标转发至它的异常目标。如果未配置异常目标，那么将应用重试之间的时间间隔。
receiveAllowed	布尔型	true	清除此选项（设置为false）以使使用者无法从此目标接收消息。
redeliveryInterval	长整型	5000	未配置异常目标时，这是达到此目标的最大失败传递次数限制后，将在重试之间应用的时间间隔。
sendAllowed	布尔型	true	生产者可以将消息发送至此目标。

topicSpace

主题空间目标表示“发布和预订”主题集，并用于发布/预订消息传递。

false

属性名称	数据类型	缺省值	描述
exceptionDestination	字符串	_SYSTEM.Exception.Destination	消息无法传递至此目标时，系统会将其转发至的目标。
failedDeliveryPolicy	• KEEP_TRYING • SEND_TO_EXCEPTION_DESTINATION • DISCARD	SEND_TO_EXCEPTION_DESTINATION	列示当达到消息的 maxredeliverycount 时，消息传递引擎必须执行的操作。 **KEEP_TRYING** KEEP_TRYING **SEND_TO_EXCEPTION_DESTINATION** SEND_TO_EXCEPTION_DESTINATION **DISCARD** DISCARD
forceReliability	• AssuredPersistent • BestEffortNonPersistent • ReliableNonPersistent • ExpressNonPersistent • ReliablePersistent	AssuredPersistent	当生产者未设置显式可靠性时要对面向此目标生成的消息指定的可靠性。 **AssuredPersistent** AssuredPersistent **BestEffortNonPersistent** BestEffortNonPersistent **ReliableNonPersistent** ReliableNonPersistent **ExpressNonPersistent** ExpressNonPersistent

属性名称	数据类型	缺省值	描述
			ReliablePersistent ReliablePersistent
id	字符串		主题空间的名称。
maintainStrictOrder	布尔型	false	维持生产者向目的地发送消息的顺序。
maxMessageDepth	长整型 最小值：1	50000	消息传递引擎可在其消息点上放置的最大消息数。
maxRedeliveryCount	整型	5	消息的处理尝试最大失败次数。在此失败尝试次数之后，如果配置了异常目标，那么将把消息从预期目标转发至它的异常目标。如果未配置异常目标，那么将应用重试之间的时间间隔。
receiveAllowed	布尔型	true	清除此选项（设置为false）以使使用者无法从此目标接收消息。
redeliveryInterval	长整型	5000	未配置异常目标时，这是达到此目标的最大失败传递次数限制后，将在重试之间应用的时间间隔。
sendAllowed	布尔型	true	生产者可以将消息发送至此目标。

缺省 MIME 类型 (mimeTypes)

所有 HTTP 虚拟主机共享的 MIME 类型的定义

type

MIME 类型的定义，以“标识=值”表示。使用扩展名作为标识，并使用关联类型作为值。

false

字符串

Mongo (mongo)

Mongo 实例的配置。

- *hostNames*
- *library*
  - *file*
  - *fileset*
  - *folder*
- *ports*

属性名称	数据类型	缺省值	描述
autoConnectRetry	布尔型		如果无法打开套接字，请按最长为 maxAutoConnectRetryTime 的时间间隔重试与服务器的连接。
connectTimeout	具有毫秒精度的时间段		新连接的连接超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
connectionsPerHost	整型 最小值：0		限制与每个主机的打开连接数。未使用的连接会加以汇聚。
cursorFinalizerEnabled	布尔型		尝试清除未关闭的数据库游标。
description	字符串		Mongo 实例的描述。
id	字符串		唯一配置标识。
libraryRef	对顶级库元素的引用（字符串）。		指定包含 MongoDB Java 驱动程序的库。
maxAutoConnectRetryTime	具有毫秒精度的时间段		在其间可以重试打开与服务器的连接的时间间隔。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

属性名称	数据类型	缺省值	描述
maxWaitTime	具有毫秒精度的时间段		等待可用连接的最长时间。如果为负值，那么连接请求永不超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。
password	可逆向编码的密码（字符串）		数据库用户的密码。
readPreference	• primary • secondaryPreferred • secondary • primaryPreferred • nearest		配置读取首选项。 **primary** primary **secondaryPreferred** secondaryPreferred **secondary** secondary **primaryPreferred** primaryPreferred

属性名称	数据类型	缺省值	描述
			nearest nearest
socketKeepAlive	布尔型		配置是否使套接字保持活动。
socketTimeout	具有毫秒精度的时间段		套接字超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
threadsAllowedToBlockForConnectionMultiplier	整型 最小值：0		此值与 connectionsPerHost 相乘以确定允许等待可用连接的线程数上限。
user	字符串		数据库用户名。
writeConcern	• ERRORS_IGNORED • ACKNOWLEDGED • SAFE • JOURNALED • NORMAL • REPLICA_ACKNOWLEDGED • FSYNC_SAFE • MAJORITY • FSYNCED • JOURNAL_SAFE • REPLICAS_SAFE • NONE • UNACKNOWLEDGED		针对 Mongo 服务器的写操作的可靠性。 **ERRORS_IGNORED** ERRORS_IGNORED **ACKNOWLEDGED** ACKNOWLEDGED **SAFE** SAFE **JOURNALED** JOURNALED **NORMAL** NORMAL **REPLICA_ACKNOWLEDGED** REPLICA_ACKNOWLEDGED **FSYNC_SAFE** FSYNC_SAFE

属性名称	数据类型	缺省值	描述
			MAJORITY MAJORITY **FSYNCED** FSYNCED **JOURNAL_SAFE** JOURNAL_SAFE **REPLICAS_SAFE** REPLICAS_SAFE **NONE** NONE **UNACKNOWLEDGED** UNACKNOWLEDGED

hostNames

主机名列表。此列表的排序必须与端口列表的排序一致，以使主机名列表中的第一个元素对应于端口列表中的第一个元素，以此类推。

false

字符串

library

指定包含 MongoDB Java 驱动程序的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指

属性名称	数据类型	缺省值	描述
			定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

端口

端口号列表。此列表的排序必须与主机名列表的排序一致，以使主机名列表中的第一个元素对应于端口列表中的第一个元素，以此类推。

false

MongoDB 数据库 (mongoDB)

MongoDB 数据库实例的配置。

- *mongo*
  - *hostNames*
  - *library*
    - *file*
    - *fileset*
    - *folder*
  - *ports*

属性名称	数据类型	缺省值	描述
databaseName	字符串		数据库的名称。
id	字符串		唯一配置标识。
jndiName	字符串		MongoDB 数据库实例的 JNDI 名称
mongoRef	对顶级 mongo 元素的引用（字符串）。		指定此数据库实例所属的 Mongo 实例。

mongo

指定此数据库实例所属的 Mongo 实例。

false

属性名称	数据类型	缺省值	描述
autoConnectRetry	布尔型		如果无法打开套接字，请按最长为 maxAutoConnectRetryTime 的时间间隔重试与服务器的连接。
connectTimeout	具有毫秒精度的时间段		新连接的连接超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
connectionsPerHost	整型 最小值：0		限制与每个主机的打开连接数。未使用的连接会加以汇聚。
cursorFinalizerEnabled	布尔型		尝试清除未关闭的数据库游标。
description	字符串		Mongo 实例的描述。
libraryRef	对顶级库元素的引用（字符串）。		指定包含 MongoDB Java 驱动程序的库。
maxAutoConnectRetryTime	具有毫秒精度的时间段		在其间可以重试打开与服务器的连接的时间间隔。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
maxWaitTime	具有毫秒精度的时间段		等待可用连接的最长时间。如果为负值，那么连接请求永不超时。指定后跟时间单

属性名称	数据类型	缺省值	描述
			位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。
password	可逆向编码的密码（字符串）		数据库用户的密码。
readPreference	• primary • secondaryPreferred • secondary • primaryPreferred • nearest		配置读取首选项。 **primary** primary **secondaryPreferred** secondaryPreferred **secondary** secondary

属性名称	数据类型	缺省值	描述
			primaryPreferred primaryPreferred **nearest** nearest
socketKeepAlive	布尔型		配置是否使套接字保持活动。
socketTimeout	具有毫秒精度的时间段		套接字超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
threadsAllowedToBlockForConnectionMultiplier	整型 最小值：0		此值与 connectionsPerHost 相乘以确定允许等待可用连接的线程数上限。
user	字符串		数据库用户名。
writeConcern	• ERRORS_IGNORED • ACKNOWLEDGED • SAFE • JOURNALED • NORMAL • REPLICA_ACKNOWLEDGED • FSYNC_SAFE • MAJORITY • FSYNCED • JOURNAL_SAFE • REPLICAS_SAFE • NONE • UNACKNOWLEDGED		针对 Mongo 服务器的写操作的可靠性。 **ERRORS_IGNORED** ERRORS_IGNORED **ACKNOWLEDGED** ACKNOWLEDGED **SAFE** SAFE **JOURNALED** JOURNALED **NORMAL** NORMAL

属性名称	数据类型	缺省值	描述
			REPLICA_ACKNOWLEDGED REPLICA_ACKNOWLEDGED **FSYNC_SAFE** FSYNC_SAFE **MAJORITY** MAJORITY **FSYNCED** FSYNCED **JOURNAL_SAFE** JOURNAL_SAFE **REPLICAS_SAFE** REPLICAS_SAFE **NONE** NONE **UNACKNOWLEDGED** UNACKNOWLEDGED

mongo > hostNames

主机名列表。此列表的排序必须与端口列表的排序一致，以使主机名列表中的第一个元素对应于端口列表中的第一个元素，以此类推。

false

字符串

mongo > library

指定包含 MongoDB Java 驱动程序的库。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的

属性名称	数据类型	缺省值	描述
			任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

mongo > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

mongo > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。

属性名称	数据类型	缺省值	描述
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

mongo > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

mongo > ports

端口号列表。此列表的排序必须与主机名列表的排序一致，以使主机名列表中的第一个元素对应于端口列表中的第一个元素，以此类推。

false

监视器 (monitor)

包括已启用的传统 PMI、FineGrained 以及将来的任何配置更新的监视功能部件的配置。

属性名称	数据类型	缺省值	描述
enableTraditionalPMI	布尔型	false	用于启用或禁用传统 PMI 报告方式的属性。
filter	字符串		允许用户根据组名（例如 WebContainer、JVM、T

属性名称	数据类型	缺省值	描述
			hreadPool、Session 和 Connection Pool 等）启用/禁用监视器。

Netscape Directory Server LDAP 过滤器 (netscapeLdapFilterProperties)

指定缺省 Netscape Directory Server LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。
userFilter	字符串	(&(uid=%v)(objectclass=inetOrgPerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	inetOrgPerson:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

OAuth 角色映射 (oauth-roles)

OAuth Web 应用程序安全角色映射。

- *authenticated*
  - *group*
  - *special-subject*
  - *user*
- *clientManager*
  - *group*
  - *special-subject*
  - *user*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

authenticated

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

authenticated > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		以常规格式 group:realmName/groupUniqueId 表示的组访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

authenticated > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。 **EVERYONE** 每个请求的所有用户（即使请求未得到认证）。 **ALL_AUTHENTICATED_USERS** 所有已认证的用户。

authenticated > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

clientManager

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。

clientManager > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		以常规格式 group:realmName/groupUniqueId 表示的组访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

clientManager > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。

属性名称	数据类型	缺省值	描述
			EVERYONE 每个请求的所有用户（即使请求未得到认证）。 **ALL_AUTHENTICATED_USERS** 所有已认证的用户。

clientManager > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

OAuth 提供者定义 (oauthProvider)

OAuth 提供者定义。

- *autoAuthorizeClient*
- *databaseStore*
  - *dataSource*
    - *connectionManager*
    - *containerAuthData*
    - *jaasLoginContextEntry*
    - *jdbcDriver*
      - *library*
        - *file*
        - *fileset*
        - *folder*
    - *properties*
    - *properties.datadirect.sqlserver*

    - *properties.db2.i.native*
    - *properties.db2.i.toolbox*
    - *properties.db2.jcc*
    - *properties.derby.client*
    - *properties.derby.embedded*
    - *properties.informix*
    - *properties.informix.jcc*
    - *properties.microsoft.sqlserver*
    - *properties.oracle*
    - *properties.sybase*
    - *recoveryAuthData*
- *grantType*
- *jwtGrantType*
- *library*
    - *file*
    - *fileset*
    - *folder*
- *localStore*
    - *client*
        - *functionalUserGroupIds*
        - *grantTypes*
        - *postLogoutRedirectUris*
        - *redirect*
        - *responseTypes*
- *mediatorClassname*

属性名称	数据类型	缺省值	描述
accessTokenLength	长整型	40	所生成 OAuth 访问令牌的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.access.token.length。
accessTokenLifetime	精度为秒的时间段	7200	访问令牌的有效时间（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.token.lifetime.seconds。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
allowPublicClients	布尔型	false	如果值为 false，那么将禁止访问公用客户机（有关详细信息，请参阅 OAuth 规范）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.allow.public.clients。
authorizationCodeLength	长整型	30	所生成授权代码的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.code.length。
authorizationCodeLifetime	精度为秒的时间段	60	授权代码生存期（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.code.lifetime.seconds。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
authorizationErrorTemplate	字符串		定制授权错误页面模板的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.error.template。
authorizationFormTemplate	字符串	template.html	定制授权页面模板的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.form.template。
authorizationGrantLifetime	精度为秒的时间段	604800	权限授权生存期（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.max.authorization.grant.lifetime.seconds。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s

属性名称	数据类型	缺省值	描述
			。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
autoAuthorize	布尔型	false	要使用自动授权，请在请求后面附加值为 true 的 autoAuthorize 参数。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.param。
autoAuthorizeParam	字符串	autoauthz	要使用自动授权，请在请求后面附加值为 true 的 autoAuthorize 参数。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.param。
certAuthentication	布尔型	false	在 HTTPS 请求中启用客户机证书的认证。
characterEncoding	字符串		将请求字符编码设置为此值。完整应用程序服务器概要文件中的等价提供者参数为 characterEncoding。
clientTokenCacheSize	长整型		客户机令牌高速缓存中的最大条目数。
clientURISubstitutions	字符串		可选值，用于替换动态主机名的客户机 URI 字符串。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.client.uri.substitutions。
consentCacheEntryLifetime	精度为秒的时间段	1800	准许的高速缓存中条目有效的时间（秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
consentCacheSize	长整型 最小值：0	1000	准许的高速缓存中允许的最大条目数。
coverageMapSessionMaxAge	精度为秒的时间段	600	指示覆盖范围映射服务的 cache-control 头的 max-age 值（秒数）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
customLoginURL	字符串	login.jsp	定制登录页面的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.loginURL。
filter	字符串		URI 过滤器会选择要由此提供者授权的请求。完整应用程序服务器概要文件中的等价提供者参数为 Filter。
httpsRequired	布尔型	true	需要 OAuth 客户机与提供者之间的 SSL 通信。
id	字符串		唯一配置标识。
includeTokenInSubject	布尔型	true	如果值为 true，那么添加 com.ibm.wsspi.security.oauth20.token.WSOAuth20Token 作为专用凭证。完整应用程序服务器概要文件中的等价提供者参数为 includeToken。
issueRefreshToken	布尔型	true	如果值为 false，那么将禁止生成和使用刷新令牌。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.issue.refresh.token。
libraryRef	对顶级库元素的引用（字符串）。		对包含介体插件类的共享库的引用。

属性名称	数据类型	缺省值	描述
oauthOnly	布尔型	true	如果值为 true，那么与过滤器相匹配的请求必须具有访问令牌，否则它们将失败。如果为 false，那么没有访问令牌时系统将检查匹配请求中是否存在其他认证数据。完整应用程序服务器概要文件中的等价提供者参数为 oauthOnly。
refreshTokenLength	长整型	50	所生成刷新令牌的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.refresh.token.length。
skipResourceOwnerValidation	布尔型	false	如果值为 true，那么会跳过资源所有者验证。
userClientTokenLimit	长整型		每个用户和客户机组合的令牌限制。

autoAuthorizeClient

可使用自动授权的客户机的名称。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.clients。

false

字符串

databaseStore

定义了客户机，并且令牌高速缓存在数据库中。

false

属性名称	数据类型	缺省值	描述
cleanupExpiredTokenInterval	精度为秒的时间段	3600	到期令牌清除时间间隔（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauthjdbc.CleanupInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目

属性名称	数据类型	缺省值	描述
			中。例如，1m30s 相当于 90 秒。
dataSourceRef	对顶级 dataSource 元素的引用（字符串）。		对用于存储器的数据源的引用。
password	可逆向编码的密码（字符串）		用于访问数据库的密码。
模式	字符串	OAuthDBSchema	模式
user	字符串		用户

databaseStore > dataSource

对用于存储器的数据源的引用。

false

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。

属性名称	数据类型	缺省值	描述
			MatchOriginalRequest 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
isolationLevel	• TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		缺省事务隔离级别。 **TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。 **TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻象读取受到阻止。

属性名称	数据类型	缺省值	描述
			TRANSACTION_READ_UNCOMMITTED 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此缺省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。

属性名称	数据类型	缺省值	描述
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作 SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource **javax.sql.XADataSource** javax.sql.XADataSource **javax.sql.ConnectionPoolDataSource** javax.sql.ConnectionPoolDataSource

databaseStore > dataSource > connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (

属性名称	数据类型	缺省值	描述
			m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

databaseStore > dataSource > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

databaseStore > dataSource > jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

databaseStore > dataSource > jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

databaseStore > dataSource > jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

databaseStore > dataSource > jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

databaseStore > dataSource > jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

databaseStore > dataSource > jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

databaseStore > dataSource > properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1 • 0	0	JDBC 驱动程序属性：JDBCBehavior。值为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0

属性名称	数据类型	缺省值	描述
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm **userIdPassword** userIdPassword **kerberos** kerberos **auto** auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。

属性名称	数据类型	缺省值	描述
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDelay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。

属性名称	数据类型	缺省值	描述
			dateTime dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeOutputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll

属性名称	数据类型	缺省值	描述
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL **requestSSL** requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck **atomicWithRepositioning** atomicWithRepositioning **nonAtomic** nonAtomic 原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select

属性名称	数据类型	缺省值	描述
			extended extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar

属性名称	数据类型	缺省值	描述
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

databaseStore > dataSource > properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8

属性名称	数据类型	缺省值	描述
			256 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa

属性名称	数据类型	缺省值	描述
			mdy mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。

属性名称	数据类型	缺省值	描述
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 63
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• \, • b • :		JDBC 驱动程序属性：timeSeparator。

属性名称	数据类型	缺省值	描述
	• .		\, 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。 **.** 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128	32	JDBC 驱动程序属性：blockSize。

属性名称	数据类型	缺省值	描述
	• 0 • 32 • 64 • 16 • 8 • 256		**512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。

属性名称	数据类型	缺省值	描述
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• • \, • . • / • -		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。

属性名称	数据类型	缺省值	描述
			- 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 \, 逗号字符 (,)。 . 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。

属性名称	数据类型	缺省值	描述
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。

属性名称	数据类型	缺省值	描述
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none • warning	warning	JDBC 驱动程序属性：packageError。 **exception** exception

属性名称	数据类型	缺省值	描述
			none none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。 **system** system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。

属性名称	数据类型	缺省值	描述
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex **table** table

属性名称	数据类型	缺省值	描述
			language language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。 \, 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			: 冒号字符 (:)。 **.** 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion **error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all **jdbc** jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。

属性名称	数据类型	缺省值	描述
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

databaseStore > dataSource > properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConnectionTimeout。指定后跟时间单位的正整数，时间单位可以是

属性名称	数据类型	缺省值	描述
			小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_WITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。

属性名称	数据类型	缺省值	描述
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **2** TYPE_SCROLL_ASENSITIVE

属性名称	数据类型	缺省值	描述
			1 TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。 **4** 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。 **2** NO

属性名称	数据类型	缺省值	描述
			1 YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OPTION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL

属性名称	数据类型	缺省值	描述
			0 INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。

属性名称	数据类型	缺省值	描述
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFr

属性名称	数据类型	缺省值	描述
			omServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY

属性名称	数据类型	缺省值	描述
			7 ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **16** ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY **12** ENCRYPTED_USER_AND_DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO **1** YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL **1** QUERYTIMEOUT_STATEMENT_LEVEL

属性名称	数据类型	缺省值	描述
			-1 QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=2

属性名称	数据类型	缺省值	描述
			62144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。 **2** TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

databaseStore > dataSource > properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。

属性名称	数据类型	缺省值	描述
			create 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7 • 4 • 9 • 8	3	JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off

属性名称	数据类型	缺省值	描述
			peerAuthentication peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 3

属性名称	数据类型	缺省值	描述
			0s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。

属性名称	数据类型	缺省值	描述
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。

属性名称	数据类型	缺省值	描述
			例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。

属性名称	数据类型	缺省值	描述
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。

属性名称	数据类型	缺省值	描述
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将

属性名称	数据类型	缺省值	描述
			30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。

属性名称	数据类型	缺省值	描述
			ON ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2

属性名称	数据类型	缺省值	描述
			1 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例

属性名称	数据类型	缺省值	描述
			如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication • JavaKerberos		JDBC 驱动程序属性：authenticationScheme。 **NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。

属性名称	数据类型	缺省值	描述
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。

属性名称	数据类型	缺省值	描述
			full full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

databaseStore > dataSource > properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。 **oci** oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

databaseStore > dataSource > properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。 **false** false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL

属性名称	数据类型	缺省值	描述
			socket socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
version	整型		JDBC 驱动程序属性：version。

databaseStore > dataSource > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

grantType

提供者所接受的访问令牌授权类型（如 OAuth 规范中详细描述）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.grant.types.allowed。

false

字符串

jwtGrantType

JWT 令牌处理程序的授予类型

false

属性名称	数据类型	缺省值	描述
clockSkew	精度为秒的时间段	300s	OpenID Connect 客户机系统与 OpenID Connect 提供者系统不同步时它们之间允许的时差。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
iatRequired	布尔型	false	JWT 令牌中的 iat 声明是必需的。
maxJtiCacheSize	长整型 最小值：1	10000	最大高速缓存大小，高速缓存保存 JWT 令牌的 jti 数据以避免复用 jti。
tokenMaxLifetime	精度为秒的时间段	7200s	此时间指示生效 JWT 令牌自发出时间后的最长生存期。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

library

对包含介体插件类的共享库的引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的

属性名称	数据类型	缺省值	描述
			任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。

属性名称	数据类型	缺省值	描述
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

localStore

在 server.xml 中定义了客户机，并且令牌高速缓存在服务器中。

false

属性名称	数据类型	缺省值	描述
tokenStoreSize	长整型	2000	令牌存储器大小

localStore > client

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
applicationType	• native	web	最能描述客户机的应用程序类型。

属性名称	数据类型	缺省值	描述
	• web		**native** native **web** web
displayname	字符串		客户机的显示名称。
enabled	布尔型	true	如果为 true，那么启用客户机；如果为 false，那么禁用客户机。
functionalUserId	字符串		要与此客户机使用客户机凭证授予类型获取的访问令牌关联的用户标识。如果指定了此客户机参数，那么来自内省端点的 functional_user_id 响应参数中会返回此值。
id	字符串		唯一配置标识。
introspectTokens	布尔型	false	布尔值，指定是否允许客户机访问内省端点以内省授权服务器颁发的令牌。
name	字符串		客户机的名称（有时称为标识）。
preAuthorizedScope	字符串		请求被认为资源所有者已预先批准并因此不需要资源所有者同意的访问令牌时，客户机可使用的作用域值的空格分隔列表。
scope	字符串		指定客户机的作用域列表，各项之间用空格分隔。
secret	可逆向编码的密码（字符串）		客户机的密钥。
sessionManaged	布尔型	false	布尔值，指示客户机是否参与 OpenID 会话管理。

属性名称	数据类型	缺省值	描述
subjectType	• public		所请求的主体集类型，用于响应此客户机。 **public** public
tokenEndpointAuthMethod	• client_secret_post • none • client_secret_basic	client_secret_basic	针对客户机的令牌端点请求的认证方法。 **client_secret_post** client_secret_post **none** none **client_secret_basic** client_secret_basic

localStore > client > functionalUserGroupIds

要与此客户机使用客户机凭证授予类型获取的访问令牌关联的组标识列表。如果指定了此客户机参数，那么来自内省端点的 functional_user_groupIds 响应参数中会返回此值。

false

字符串

localStore > client > grantTypes

客户机可使用的授予类型。

false

localStore > client > postLogoutRedirectUris

RP 提供的 URL 数组，RP 可能在执行注销后使用 post_logout_redirect_uri 参数请求最终用户的用户代理重定向至这些 URL。

false

字符串

localStore > client > redirect

要在基于重定向的流（例如，客户机的授权代码和隐式授予类型）中使用的重定向 URI 的数组。如果请求中未指定，那么第一个重定向 URI 用作缺省值。

false

字符串

localStore > client > responseTypes

客户机可使用的响应类型。

false

mediatorClassname

介体插件类名。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.mediator.classnames。

false

字符串

OpenId 认证 (openId)

OpenId 认证。

- *authFilter*
  - *host*
  - *remoteAddress*
  - *requestUrl*
  - *userAgent*
  - *webApp*
- *userInfo*

属性名称	数据类型	缺省值	描述
authFilterRef	对顶级 authFilter 元素的引用（字符串）。		指定认证过滤器引用。
authenticationMode	• checkid_immediate • checkid_setup	checkid_setup	指定 OpenID 提供程序认证方式 checkid_immediate 或 checkid_setup。checkid_setup 是缺省认证方式。 **checkid_immediate** checkid_immediate 不允许浏览器与用户交互。 **checkid_setup** checkid_setup 使 openID 提供程序能够与用户交互，以请求认证或自注册，然后将结果返回给 openId 依赖方。
hashAlgorithm	• SHA256 • SHA1	SHA256	指定用于对 OpenID 提供程序响应参数进行签名和加密的散列算法。

属性名称	数据类型	缺省值	描述
			SHA256 安全散列算法 SHA256 **SHA1** 安全散列算法 SHA1
hostNameVerificationEnabled	布尔型	true	指定是否启用主机名验证。
httpsRequired	布尔型	true	在 OpenID 依赖方与提供程序服务之间需要 SSL 通信。
mapIdentityToRegistryUser	布尔型	false	指定是否将身份映射至注册表用户。用户注册表不会用来创建用户主体集。
providerIdentifier	字符串		指定用户在其中获取 OpenID 的缺省 OpenId 提供程序 URL。
realmIdentifier	字符串		对 OpenID 提供程序名称指定属性。
sslRef	字符串		指定用来连接至 OpenID 提供程序的 SSL 配置的标识。
useClientIdentity	布尔型	false	指定是否使用客户 OpenID 身份创建用户主体集。如果设置为 true，那么仅会使用 OpenID 客户机身份。如果设置为 false 并且已找到 userInfoRef 的第一个元素，那么我们使用它创建用户主体集。否则，我们使用 OpenID 身份创建用户主体集。
userInfoRef	顶级 userInfo 元素的引用列表（以逗号分隔的字符串）。	email	指定供 OpenID 提供程序包括在响应中的用户信息引用的列表（以逗号分隔这些引用）。

authFilter

指定认证过滤器引用。

false

authFilter > host

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

authFilter > remoteAddress

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ip	字符串		指定 IP 地址。
matchType	• lessThan • equals • greaterThan • contains • notContain	contains	指定匹配类型。 **lessThan** 小于 **equals** 等于 **greaterThan** 大于 **contains** 包含 **notContain** 不包含

authFilter > requestUrl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
urlPattern	字符串		指定 URL 模式。

authFilter > userAgent

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
agent	字符串		指定用户代理
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含

authFilter > webApp

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于

属性名称	数据类型	缺省值	描述
			contains 包含 **notContain** 不包含
name	字符串		指定名称。

userInfo

指定供 OpenID 提供程序包括在响应中的用户信息引用的列表（以逗号分隔这些引用）。

false

属性名称	数据类型	缺省值	描述
alias	字符串	email	指定别名。
count	整型 最小值：1	1	指定包括在 openID 提供程序的响应中的用户信息量。
id	字符串		唯一配置标识。
required	布尔型	true	指定是否需要用户信息。
uriType	字符串	http://axschema.org/contact/email	指定 URI 类型。

OpenID Connect 客户机 (openidConnectClient)

OpenID Connect 客户机。

- *authFilter*
  - *host*
  - *remoteAddress*
  - *requestUrl*
  - *userAgent*
  - *webApp*

属性名称	数据类型	缺省值	描述
authFilterRef	对顶级 authFilter 元素的引用（字符串）。		指定认证过滤器引用。
authorizationEndpointUrl	字符串		指定授权端点 URL。
clientId	字符串		客户的身份。
clientSecret	可逆向编码的密码（字符串）		客户机的密钥。

属性名称	数据类型	缺省值	描述
createSession	布尔型	true	指定当前 HttpSession 不存在时是否创建 HttpSession。
hostNameVerificationEnabled	布尔型	false	指定是否启用主机名验证。
httpsRequired	布尔型	true	在 OpenID 依赖方与提供者服务之间需要 SSL 通信。
id	字符串		唯一配置标识。
includeIdTokenInSubject	布尔型	true	指定是否在客户机主体集中包含标识令牌。
initialStateCacheCapacity	整型 最小值：0	3000	指定状态高速缓存的起始容量。此容量在本身需要时会增长。
isClientSideRedirectSupported	布尔型	true	指定客户机是否支持在客户机端重定向。
issuerIdentifier	字符串		颁发者标识是使用 HTTPS 方案且区分大小写的 URL，其中包含方案、主机及（可选的）端口号和路径部分。
mapIdentityToRegistryUser	布尔型	false	指定是否将身份映射至注册表用户。用户注册表不会用来创建用户主体集。
nonceEnabled	布尔型	false	在授权代码流程中启用现时标志参数。
redirectToRPHostAndPort	字符串		指定重定向 OpenID 依赖方主机和端口号。
scope	tokenType	OpenID 概要文件	OpenID Connect 提供者允许的 OpenID Connect 作用域（OpenID Connect 规范中已详述）。
signatureAlgorithm	• HS256 • none • RS256	HS256	指定将用于验证标识令牌的签名的签名算法。 **HS256** %tokenSignAlgorithm.HS256

属性名称	数据类型	缺省值	描述
			none %tokenSignAlgorithm.NONE **RS256** %tokenSignAlgorithm.RS256
sslRef	字符串		指定用于连接至 OpenID Connect 提供者的 SSL 配置的标识。
tokenEndpointUrl	字符串		指定令牌端点 URL。
trustAliasName	字符串		用于查找使用非对称算法验证签名时所需的公用密钥的密钥别名名称。
trustStoreRef	字符串		密钥库，包含在验证标识令牌的签名时必需的公用密钥。
userIdentityToCreateSubject	字符串	sub	在标识令牌中指定用于创建用户主体集的用户身份。

authFilter

指定认证过滤器引用。

false

authFilter > host

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

authFilter > remoteAddress

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ip	字符串		指定 IP 地址。
matchType	• lessThan • equals • greaterThan • contains • notContain	contains	指定匹配类型。 **lessThan** 小于 **equals** 等于 **greaterThan** 大于 **contains** 包含 **notContain** 不包含

authFilter > requestUrl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
urlPattern	字符串		指定 URL 模式。

authFilter > userAgent

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
agent	字符串		指定用户代理
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含

authFilter > webApp

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

OpenID Connect 服务器提供者 (openidConnectProvider)

OpenID Connect 服务器提供者

- *claimToUserRegistryMap*
  - *property*
- *customClaims*
- *discovery*
  - *claimsSupported*
  - *grantTypesSupported*
  - *idTokenSigningAlgValuesSupported*
  - *responseModesSupported*
  - *responseTypesSupported*

- *scopesSupported*
- *tokenEndpointAuthMethodsSupported*

- *oauthProvider*
  - *autoAuthorizeClient*
  - *databaseStore*
    - *dataSource*
      - *connectionManager*
      - *containerAuthData*
      - *jaasLoginContextEntry*
      - *jdbcDriver*
        - *library*
          - *file*
          - *fileset*
          - *folder*
      - *properties*
      - *properties.datadirect.sqlserver*
      - *properties.db2.i.native*
      - *properties.db2.i.toolbox*
      - *properties.db2.jcc*
      - *properties.derby.client*
      - *properties.derby.embedded*
      - *properties.informix*
      - *properties.informix.jcc*
      - *properties.microsoft.sqlserver*
      - *properties.oracle*
      - *properties.sybase*
      - *recoveryAuthData*
  - *grantType*
  - *jwtGrantType*
  - *library*
    - *file*
    - *fileset*
    - *folder*
  - *localStore*
    - *client*
      - *functionalUserGroupIds*
      - *grantTypes*
      - *postLogoutRedirectUris*
      - *redirect*
      - *responseTypes*
  - *mediatorClassname*
- *scopeToClaimMap*
  - *property*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
idTokenLifetime	精度为秒的时间段	2h	标识令牌有效的时间（以秒计）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
issuerIdentifier	字符串		为响应的发出者指定发出者标识。
keyAliasName	字符串		用于查找使用非对称算法进行签名所需的专用密钥的密钥别名名称。
keyStoreRef	字符串	opKeyStore	密钥库，包含使用非对称算法进行签名所必需的专用密钥。
oauthProviderRef	对顶级 oauthProvider 元素的引用（字符串）。		对 OAuth 提供者的标识的引用。
sessionManaged	布尔型	false	通过 true 或 false 指示是否支持会话管理。缺省值为 false。
signatureAlgorithm	• HS256 • none • RS256	HS256	指定将用于对标识令牌进行签名的签名算法。 **HS256** 使用 SHA-256 散列的 HMAC **none** 没有签名 **RS256** 使用 SHA-256 散列的 RSASSA-PKCS-v1_5
trustStoreRef	字符串		密钥库，包含在验证 JWT 令牌的签名时必需的公用密钥。

claimToUserRegistryMap

指定声明的用户注册表键。

false

属性名称	数据类型	缺省值	描述
address	字符串	postalAddress	指定将对地址声明进行检索的用户注册表键。
email	字符串	mail	指定将对电子邮件声明进行检索的用户注册表键。
given_name	字符串	givenName	指定将对指定名称声明进行检索的用户注册表键。
name	字符串	displayName	指定将对名称声明进行检索的用户注册表键。
phone_number	字符串	telephoneNumber	指定将对电话号码声明进行检索的用户注册表键。
picture	字符串	photoURL	指定将对照片声明进行检索的用户注册表键。

claimToUserRegistryMap > property

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		指定属性名
value	字符串		指定属性的值

customClaims

除缺省 realmName、uniqueSecurityName 和 groupIds 声明以外，要放入标识令牌有效内容中的额外声明。

false

字符串

discovery

发现基于 OpenID Connect 和 Jazz Authorization Server 概要文件。

false

属性名称	数据类型	缺省值	描述
claimsParameterSupported	布尔型	false	通过 true 或 false 指示是否支持声明参数。
requestParameterSupported	布尔型	false	通过 true 或 false 指示是否支持请求参数。
requestUriParameterSupported	布尔型	false	通过 true 或 false 指示是否支持请求 URI 参数。
requireRequestUriRegistration	布尔型	false	通过 true 或 false 指示是否支持需要请求 URI 注册。

discovery > claimsSupported

指定将支持的声明的逗号分隔列表。

false

字符串

discovery > grantTypesSupported

指定将使用的授予类型的逗号分隔列表。

false

discovery > idTokenSigningAlgValuesSupported

指定将用于对标识令牌进行签名的签名算法。

false

discovery > responseModesSupported

指定将使用的响应方式的逗号分隔列表。

false

discovery > responseTypesSupported

指定 OP 将支持的响应类型的逗号分隔列表。

false

discovery > scopesSupported

指定将支持的作用域的逗号分隔列表。

false

字符串

discovery > tokenEndpointAuthMethodsSupported

指定将使用的令牌端点认证方法的逗号分隔列表。

false

oauthProvider

对 OAuth 提供者的标识的引用。

false

属性名称	数据类型	缺省值	描述
accessTokenLength	长整型	40	所生成 OAuth 访问令牌的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.access.token.length。
accessTokenLifetime	精度为秒的时间段	7200	访问令牌的有效时间（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.token.lifetime.seconds。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
allowPublicClients	布尔型	false	如果值为 false，那么将禁止访问公用客户机（有关详细信息，请参阅 OAuth 规范）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.allow.public.clients。
authorizationCodeLength	长整型	30	所生成授权代码的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.code.length。
authorizationCodeLifetime	精度为秒的时间段	60	授权代码生存期（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.code.lifetime.seconds。指定后跟时间

属性名称	数据类型	缺省值	描述
			单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
authorizationErrorTemplate	字符串		定制授权错误页面模板的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.error.template。
authorizationFormTemplate	字符串	template.html	定制授权页面模板的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.form.template。
authorizationGrantLifetime	精度为秒的时间段	604800	权限授权生存期（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.max.authorization.grant.lifetime.seconds。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
autoAuthorize	布尔型	false	要使用自动授权，请在请求后面附加值为 true 的 autoAuthorize 参数。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.param。

属性名称	数据类型	缺省值	描述
autoAuthorizeParam	字符串	autoauthz	要使用自动授权，请在请求后面附加值为 true 的 autoAuthorize 参数。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.param。
certAuthentication	布尔型	false	在 HTTPS 请求中启用客户机证书的认证。
characterEncoding	字符串		将请求字符编码设置为此值。完整应用程序服务器概要文件中的等价提供者参数为 characterEncoding。
clientTokenCacheSize	长整型		客户机令牌高速缓存中的最大条目数。
clientURISubstitutions	字符串		可选值，用于替换动态主机名的客户机 URI 字符串。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.client.uri.substitutions。
consentCacheEntryLifetime	精度为秒的时间段	1800	准许的高速缓存中条目有效的时间（秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
consentCacheSize	长整型 最小值：0	1000	准许的高速缓存中允许的最大条目数。
coverageMapSessionMaxAge	精度为秒的时间段	600	指示覆盖范围映射服务的 cache-control 头的 max-age 值（秒数

属性名称	数据类型	缺省值	描述
			）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
customLoginURL	字符串	login.jsp	定制登录页面的 URL。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.authorization.loginURL。
filter	字符串		URI 过滤器会选择要由此提供者授权的请求。完整应用程序服务器概要文件中的等价提供者参数为 Filter。
httpsRequired	布尔型	true	需要 OAuth 客户机与提供者之间的 SSL 通信。
includeTokenInSubject	布尔型	true	如果值为 true，那么添加 com.ibm.wsspi.security.oauth20.token.WSOAuth20Token 作为专用凭证。完整应用程序服务器概要文件中的等价提供者参数为 includeToken。
issueRefreshToken	布尔型	true	如果值为 false，那么将禁止生成和使用刷新令牌。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.issue.refresh.token。
libraryRef	对顶级库元素的引用（字符串）。		对包含介体插件类的共享库的引用。

属性名称	数据类型	缺省值	描述
oauthOnly	布尔型	true	如果值为 true，那么与过滤器相匹配的请求必须具有访问令牌，否则它们将失败。如果为 false，那么没有访问令牌时系统将检查匹配请求中是否存在其他认证数据。完整应用程序服务器概要文件中的等价提供者参数为 oauthOnly。
refreshTokenLength	长整型	50	所生成刷新令牌的长度。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.refresh.token.length。
skipResourceOwnerValidation	布尔型	false	如果值为 true，那么会跳过资源所有者验证。
userClientTokenLimit	长整型		每个用户和客户机组合的令牌限制。

oauthProvider > autoAuthorizeClient

可使用自动授权的客户机的名称。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.autoauthorize.clients。

false

字符串

oauthProvider > databaseStore

定义了客户机，并且令牌高速缓存在数据库中。

false

属性名称	数据类型	缺省值	描述
cleanupExpiredTokenInterval	精度为秒的时间段	3600	到期令牌清除时间间隔（秒）。完整应用程序服务器概要文件中的等价提供者参数为 oauthjdbc.CleanupInterval。指定后跟时间单位的正整数，时间单位可以是小时 (

属性名称	数据类型	缺省值	描述
			h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
dataSourceRef	对顶级 dataSource 元素的引用（字符串）。		对用于存储器的数据源的引用。
password	可逆向编码的密码（字符串）		用于访问数据库的密码。
模式	字符串	OAuthDBSchema	模式
user	字符串		用户

oauthProvider > databaseStore > dataSource

对用于存储器的数据源的引用。

false

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。

属性名称	数据类型	缺省值	描述
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。 **MatchOriginalRequest** 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
isolationLevel	• TRANSACTION_REPEATABLE_READ • TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		缺省事务隔离级别。 **TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。 **TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻

属性名称	数据类型	缺省值	描述
			象读取受到阻止。 **TRANSACTION_READ_UNCOMMITTED** 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此缺省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作 SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource **javax.sql.XADataSource** javax.sql.XADataSource **javax.sql.ConnectionPoolDataSource** javax.sql.ConnectionPoolDataSource

oauthProvider > databaseStore > dataSource > connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中

属性名称	数据类型	缺省值	描述
			。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的

属性名称	数据类型	缺省值	描述
			正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

oauthProvider > databaseStore > dataSource > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

oauthProvider > databaseStore > dataSource > jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

oauthProvider > databaseStore > dataSource > jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

oauthProvider > databaseStore > dataSource > jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

oauthProvider > databaseStore > dataSource > jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

oauthProvider > databaseStore > dataSource > jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

oauthProvider > databaseStore > dataSource > jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

oauthProvider > databaseStore > dataSource > properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1 • 0	0	JDBC 驱动程序属性：JDBCBehavior。值为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0

属性名称	数据类型	缺省值	描述
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm **userIdPassword** userIdPassword **kerberos** kerberos **auto** auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。

属性名称	数据类型	缺省值	描述
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDelay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。

属性名称	数据类型	缺省值	描述
			dateTime dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeOutputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll

属性名称	数据类型	缺省值	描述
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL **requestSSL** requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck **atomicWithRepositioning** atomicWithRepositioning **nonAtomic** nonAtomic 原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select

属性名称	数据类型	缺省值	描述
			extended extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar

属性名称	数据类型	缺省值	描述
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

oauthProvider > databaseStore > dataSource > properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8

属性名称	数据类型	缺省值	描述
			256 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa

属性名称	数据类型	缺省值	描述
			mdy mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。

属性名称	数据类型	缺省值	描述
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 63
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• \, • b • :		JDBC 驱动程序属性：timeSeparator。

属性名称	数据类型	缺省值	描述
	• .		\, 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。 **.** 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128	32	JDBC 驱动程序属性：blockSize。

属性名称	数据类型	缺省值	描述
	• 0 • 32 • 64 • 16 • 8 • 256		**512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。

属性名称	数据类型	缺省值	描述
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• • \, • . • / • -		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。 \, 逗号字符 (,)。 . 句点字符 (.)。 / 正斜杠字符 (/)。

属性名称	数据类型	缺省值	描述
			- 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 \, 逗号字符 (,)。 . 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。

属性名称	数据类型	缺省值	描述
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。

属性名称	数据类型	缺省值	描述
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none • warning	warning	JDBC 驱动程序属性：packageError。 **exception** exception

属性名称	数据类型	缺省值	描述
			none none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。 **system** system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。

属性名称	数据类型	缺省值	描述
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex **table** table

属性名称	数据类型	缺省值	描述
			language language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。 \, 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			: 冒号字符 (:)。 **.** 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc • warning		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion **error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all **jdbc** jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。

属性名称	数据类型	缺省值	描述
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

oauthProvider > databaseStore > dataSource > properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConnectionTimeout。指定后跟时间单位的正整数，时间单位可以是

属性名称	数据类型	缺省值	描述
			小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_WITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。

属性名称	数据类型	缺省值	描述
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **2** TYPE_SCROLL_ASENSITIVE

属性名称	数据类型	缺省值	描述
			1 TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。 **4** 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。 **2** NO

属性名称	数据类型	缺省值	描述
			1 YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OPTION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL

属性名称	数据类型	缺省值	描述
			0 INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。

属性名称	数据类型	缺省值	描述
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFr

属性名称	数据类型	缺省值	描述
			omServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16 • 13 • 11 • 12		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURITY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY

属性名称	数据类型	缺省值	描述
			7 ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **16** ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY **12** ENCRYPTED_USER_AND_DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO **1** YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL **1** QUERYTIMEOUT_STATEMENT_LEVEL

属性名称	数据类型	缺省值	描述
			-1 QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=2

属性名称	数据类型	缺省值	描述
			62144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。 **2** TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。

属性名称	数据类型	缺省值	描述
			2 NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

oauthProvider > databaseStore > dataSource > properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。

属性名称	数据类型	缺省值	描述
			create 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7 • 4 • 9 • 8	3	JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off

属性名称	数据类型	缺省值	描述
			peerAuthentication peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 3

属性名称	数据类型	缺省值	描述
			0s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。

属性名称	数据类型	缺省值	描述
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。

属性名称	数据类型	缺省值	描述
			例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。

属性名称	数据类型	缺省值	描述
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。

属性名称	数据类型	缺省值	描述
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将

属性名称	数据类型	缺省值	描述
			30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。

属性名称	数据类型	缺省值	描述
			ON ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2

属性名称	数据类型	缺省值	描述
			1 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例

属性名称	数据类型	缺省值	描述
			如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication • JavaKerberos		JDBC 驱动程序属性：authenticationScheme。 **NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。

属性名称	数据类型	缺省值	描述
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。

属性名称	数据类型	缺省值	描述
			full full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

oauthProvider > databaseStore > dataSource > properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。 **oci** oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。

属性名称	数据类型	缺省值	描述
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

oauthProvider > databaseStore > dataSource > properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。 **false** false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL

属性名称	数据类型	缺省值	描述
			socket socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
version	整型		JDBC 驱动程序属性：version。

oauthProvider > databaseStore > dataSource > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

oauthProvider > grantType

提供者所接受的访问令牌授权类型（如 OAuth 规范中详细描述）。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.grant.types.allowed。

false

字符串

oauthProvider > jwtGrantType

JWT 令牌处理程序的授予类型

false

属性名称	数据类型	缺省值	描述
clockSkew	精度为秒的时间段	300s	OpenID Connect 客户机系统与 OpenID Connect 提供者系统不同步时它们之间允许的时差。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
iatRequired	布尔型	false	JWT 令牌中的 iat 声明是必需的。
maxJtiCacheSize	长整型 最小值：1	10000	最大高速缓存大小，高速缓存保存 JWT 令牌的 jti 数据以避免复用 jti。
tokenMaxLifetime	精度为秒的时间段	7200s	此时间指示生效 JWT 令牌自发出时间后的最长生存期。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

oauthProvider > library

对包含介体插件类的共享库的引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的

属性名称	数据类型	缺省值	描述
			任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

oauthProvider > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

oauthProvider > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。

属性名称	数据类型	缺省值	描述
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

oauthProvider > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

oauthProvider > localStore

在 server.xml 中定义了客户机，并且令牌高速缓存在服务器中。

false

属性名称	数据类型	缺省值	描述
tokenStoreSize	长整型	2000	令牌存储器大小

oauthProvider > localStore > client

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
applicationType	• native	web	最能描述客户机的应用程序类型。

属性名称	数据类型	缺省值	描述
	• web		**native** native **web** web
displayname	字符串		客户机的显示名称。
enabled	布尔型	true	如果为 true，那么启用客户机；如果为 false，那么禁用客户机。
functionalUserId	字符串		要与此客户机使用客户机凭证授予类型获取的访问令牌关联的用户标识。如果指定了此客户机参数，那么来自内省端点的 functional_user_id 响应参数中会返回此值。
id	字符串		唯一配置标识。
introspectTokens	布尔型	false	布尔值，指定是否允许客户机访问内省端点以内省授权服务器颁发的令牌。
name	字符串		客户机的名称（有时称为标识）。
preAuthorizedScope	字符串		请求被认为资源所有者已预先批准并因此不需要资源所有者同意的访问令牌时，客户机可使用的作用域值的空格分隔列表。
scope	字符串		指定客户机的作用域列表，各项之间用空格分隔。
secret	可逆向编码的密码（字符串）		客户机的密钥。
sessionManaged	布尔型	false	布尔值，指示客户机是否参与 OpenID 会话管理。

属性名称	数据类型	缺省值	描述
subjectType	• public		所请求的主体集类型，用于响应此客户机。 **public** public
tokenEndpointAuthMethod	• client_secret_post • none • client_secret_basic	client_secret_basic	针对客户机的令牌端点请求的认证方法。 **client_secret_post** client_secret_post **none** none **client_secret_basic** client_secret_basic

oauthProvider > localStore > client > functionalUserGroupIds

要与此客户机使用客户机凭证授予类型获取的访问令牌关联的组标识列表。如果指定了此客户机参数，那么来自内省端点的 functional_user_groupIds 响应参数中会返回此值。

false

字符串

oauthProvider > localStore > client > grantTypes

客户机可使用的授予类型。

false

oauthProvider > localStore > client > postLogoutRedirectUris

RP 提供的 URL 数组，RP 可能在执行注销后使用 post_logout_redirect_uri 参数请求最终用户的用户代理重定向至这些 URL。

false

字符串

oauthProvider > localStore > client > redirect

要在基于重定向的流（例如，客户机的授权代码和隐式授予类型）中使用的重定向 URI 的数组。如果请求中未指定，那么第一个重定向 URI 用作缺省值。

false

字符串

oauthProvider > localStore > client > responseTypes

客户机可使用的响应类型。

false

oauthProvider > mediatorClassname

介体插件类名。完整应用程序服务器概要文件中的等价提供者参数为 oauth20.mediator.classnames。

false

字符串

scopeToClaimMap

对作用域指定声明。

false

属性名称	数据类型	缺省值	描述
address	字符串	address	指定与地址作用域相关联的声明的逗号分隔列表。
email	字符串	email, email_verified	指定与电子邮件作用域相关联的声明的逗号分隔列表。
phone	字符串	phone_number, phone_number_verified	指定与电话作用域相关联的声明的逗号分隔列表。
profile	字符串	name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, updated_at	指定与概要文件作用域相关联的声明的逗号分隔列表。

scopeToClaimMap > property

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		指定属性名
value	字符串		指定属性的值

对象请求代理程序 (ORB) (orb)

服务器或客户机 ORB 的配置。指定客户机 ORB 的 nameService 属性，或者指定服务器 ORB 的一个或多个 iiopEndpoint 引用。

- *clientPolicy.clientContainerCsiv2*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
iiopEndpointRef	顶级 iiopEndpoint 元素的引用列表（以逗号分隔的字符串）。	defaultIiopEndpoint	可选 IIOP 端点，用于描述为此 ORB 打开的端口
nameService	字符串	corbaname::localhost:2809	远程名称服务的可选 URL，例如，corbaname::localhost:2809

clientPolicy.clientContainerCsiv2

用于传出互联网 ORB 间协议 (IIOP) 请求的公共安全互操作性 V2 (CSIv2)。

false

clientPolicy.clientContainerCsiv2 > layers

指定 CSIv2 层，例如，传输、认证和属性。

false

clientPolicy.clientContainerCsiv2 > layers > authenticationLayer

确定客户机要对传出 CSIv2 请求执行的认证机制和关联选项。

false

属性名称	数据类型	缺省值	描述
establishTrustInClient	• Required • Never • Supported	Supported	指定此关联选项对于此层是受支持的、必需的还是从不使用的。它指示认证层的认证要求。 **Required** 需要关联选项 **Never** 不得使用该关联选项 **Supported** 该关联选项受支持
password	可逆向编码的密码（字符串）		与用户名配合使用的用户密码。
user	字符串		用来登录远程服务器的用户名。

clientPolicy.clientContainerCsiv2 > layers > authenticationLayer > mechanisms

以逗号分隔列表的形式指定认证机制。例如：GSSUP

false

字符串

clientPolicy.clientContainerCsiv2 > layers > transportLayer

配置如何信任客户机。

false

属性名称	数据类型	缺省值	描述
sslEnabled	布尔型	true	通过 true 或 false 指示是否为 CSIv2 请求启用 SSL。缺省值为 true 且为建议值。如果此属性设置为 false，那么在使用 IIOP 时，会通过不受保护的通道发送诸如密码和令牌的敏感信息。
sslRef	字符串		指定建立安全连接时所需的 SSL 配置。

clientPolicy.csiv2

用于传出互联网 ORB 间协议 (IIOP) 请求的公共安全互操作性 V2 (CSIv2)。

false

clientPolicy.csiv2 > layers

指定 CSIv2 层，例如，传输、认证和属性。

false

clientPolicy.csiv2 > layers > attributeLayer

确定客户机要对传出 CSIv2 请求执行的属性层选项。

false

属性名称	数据类型	缺省值	描述
identityAssertionEnabled	布尔型	false	通过 true 或 false 指示是否启用身份断言。缺省值为 false。
trustedIdentity	字符串		用于对远程服务器断言实体的可信身份。
trustedPassword	可逆向编码的密码（字符串）		指定与可信身份配合使用的密码。

clientPolicy.csiv2 > layers > attributeLayer > identityAssertionTypes

对身份断言指定受支持的身份令牌类型。

false

clientPolicy.csiv2 > layers > authenticationLayer

确定客户机要对传出 CSIv2 请求执行的认证机制和关联选项。

false

属性名称	数据类型	缺省值	描述
establishTrustInClient	• Required • Never • Supported	Supported	指定此关联选项对于此层是受支持的、必需的还是从不使用的。它指示认证层的认证要求。 **Required** 需要关联选项 **Never** 不得使用该关联选项 **Supported** 该关联选项受支持

clientPolicy.csiv2 > layers > authenticationLayer > mechanisms

以逗号分隔列表的形式指定认证机制。例如：GSSUP, LTPA

false

字符串

clientPolicy.csiv2 > layers > transportLayer

配置如何信任客户机。

false

属性名称	数据类型	缺省值	描述
sslEnabled	布尔型	true	通过 true 或 false 指示是否为 CSIv2 请求启用 SSL。缺省值为 true 且为建议值。如果此属性设置为 false，那么在使用 IIOP 时，会通过不受保护的通道发送诸如密码和令牌的敏感信息。
sslRef	字符串		指定建立安全连接时所需的 SSL 配置。

iiopEndpoint

可选 IIOP 端点，用于描述为此 ORB 打开的端口

false

属性名称	数据类型	缺省值	描述
host	字符串	localhost	IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或者只是 DNS 主机名。可使用“*”来表示所有可用网络接口。
id	字符串		唯一配置标识。
iiopPort	整型		由此 IIOP 端点打开的没有安全保障的服务器套接字的端口
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	IIOP 端点的 TCP 协议选项

iiopEndpoint > iiopsOptions

由此 IIOP 端点打开的安全服务器套接字的规范

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
iiopsPort	整型		指定要使用 SSL 选项进行配置的端口。
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

iiopEndpoint > tcpOptions

IIOP 端点的 TCP 协议选项

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个

属性名称	数据类型	缺省值	描述
			条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

serverPolicy.csiv2

用于传入互联网 ORB 间协议 (IIOP) 请求的公共安全互操作性 V2 (CSIv2)。

false

serverPolicy.csiv2 > layers

指定 CSIv2 层，例如，传输、认证和属性。

false

serverPolicy.csiv2 > layers > attributeLayer

确定服务器针对传入 CSIv2 请求声明的属性层选项。

false

属性名称	数据类型	缺省值	描述
identityAssertionEnabled	布尔型	false	通过 true 或 false 指示是否启用身份断言。缺省值为 false。
trustedIdentities	字符串		指定服务器身份的竖线 (|) 分隔列表，系统信任这些服务器身份，允许其对此服务器执行身份断言。系统也接受值“*”以指示隐式可信（信任所有对象）。

serverPolicy.csiv2 > layers > attributeLayer > identityAssertionTypes

对身份断言指定受支持的身份令牌类型。

false

serverPolicy.csiv2 > layers > authenticationLayer

确定服务器针对传入 CSIv2 请求声明的认证机制和关联选项。

false

属性名称	数据类型	缺省值	描述
establishTrustInClient	• Required • Never • Supported	Required	指定此关联选项对于此层是受支持的、必需的还是从不使用的

属性名称	数据类型	缺省值	描述
			。它指示认证层的认证要求。 **Required** 需要关联选项 **Never** 不得使用该关联选项 **Supported** 该关联选项受支持

serverPolicy.csiv2 > layers > authenticationLayer > mechanisms

以逗号分隔列表的形式指定认证机制。例如：GSSUP, LTPA

false

字符串

serverPolicy.csiv2 > layers > transportLayer

配置如何信任客户机。

false

属性名称	数据类型	缺省值	描述
sslEnabled	布尔型	true	通过 true 或 false 指示是否为 CSIv2 请求启用 SSL。缺省值为 true 且为建议值。如果此属性设置为 false，那么在使用 IIOP 时，会通过不受保护的通道发送诸如密码和令牌的敏感信息。
sslRef	字符串		指定建立安全连接时所需的 SSL 配置。

OSGi 应用程序 (osgiApplication)

定义 OSGi 应用程序的属性。

- *application-bnd*
  - *security-role*
    - *group*
    - *run-as*
    - *special-subject*

- *user*

属性名称	数据类型	缺省值	描述
autoStart	布尔型	true	指示服务器是否自动启动应用程序。
id	字符串		唯一配置标识。
location	文件、目录或 URL。		应用程序的位置，表示为绝对路径或相对于服务器级应用程序目录的路径。
name	字符串		应用程序的名称。
suppressUncoveredHttpMethodWarning	布尔型	false	指示在应用程序部署期间阻止未覆盖 HTTP 方法警告消息的选项。

application-bnd

将应用程序中包括的常规部署信息绑定到特定资源。

false

属性名称	数据类型	缺省值	描述
version	字符串		应用程序绑定规范的版本。

application-bnd > security-role

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	字符串		安全角色的名称。

application-bnd > security-role > group

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		组访问标识
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的组的名称。

application-bnd > security-role > run-as

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
password	可逆向编码的密码（字符串）		从一个 Bean 访问另一个 Bean 时需要的用户密码。可以采用明文或编码格式存储该值。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
userid	字符串		从一个 Bean 访问另一个 Bean 时需要的用户标识。

application-bnd > security-role > special-subject

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
type	• EVERYONE • ALL_AUTHENTICATED_USERS		下列其中一种特殊主体集类型：ALL_AUTHENTICATED_USERS 和 EVERYONE。 **EVERYONE** Everyone **ALL_AUTHENTICATED_USERS** 所有已认证的用户

application-bnd > security-role > user

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
access-id	字符串		常规格式为 user:realmName/userUniqueId 的用户访问标识。如

属性名称	数据类型	缺省值	描述
			果未指定值，那么将生成值。
id	字符串		唯一配置标识。
name	字符串		拥有安全角色的用户的名称。

OSGi 应用程序 (osgiApplications)

所有 OSGi 应用程序的设置

属性名称	数据类型	缺省值	描述
enable.debug	布尔型	false	此选项设置为 true 时，未能启动的 OSGi 应用程序会保持已安装状态以允许调试应用程序。

OSGi 库 (osgiLibrary)

启用 OSGi 应用程序以使用共享库提供的软件包。

- *library*
  - *file*
  - *fileset*
  - *folder*
- *package*

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
libraryRef	对顶级库元素的引用（字符串）。		要用于 OSGi 库的共享库引用。

library

要用于 OSGi 库的共享库引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。

属性名称	数据类型	缺省值	描述
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位

属性名称	数据类型	缺省值	描述
			后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

package

系统可供 OSGi 应用程序使用的软件包的包导出规则。

false

字符串

持久性计划执行程序 (persistentExecutor)

安排并运行持久任务。

- *contextService*
  - *baseContext*
    - *baseContext*
    - *classloaderContext*
    - *jeeMetadataContext*
    - *securityContext*
    - *syncToOSThreadContext*
  - *classloaderContext*
  - *jeeMetadataContext*

- *securityContext*
- *syncToOSThreadContext*

属性名称	数据类型	缺省值	描述
contextServiceRef	对顶级 contextService 元素的引用（字符串）。	DefaultContextService	配置上下文捕获及传播至线程的方式。
enableTaskExecution	布尔型	true	确定此实例能否运行任务。
id	字符串		唯一配置标识。
initialPollDelay	具有毫秒精度的时间段	0	在此实例可轮询持久性存储以查找要运行的任务之前等待的持续时间。如果值为 -1，那么将延迟轮询，直到通过程序启动轮询为止。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500 ms 相当于 1.5 秒。
pollInterval	具有毫秒精度的时间段	-1	轮询要运行的任务之间的时间间隔。值 -1 将禁用初始轮询之后的所有轮询。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
pollSize	整型 最小值：1		轮询持久性存储以获取要运行的任务时要查找的最大任务条目数。如果未指定，那么表示没有限制。
retryInterval	具有毫秒精度的时间段	1m	在第二次重试失败任务与后续连续重试失败任务之间必须经过的时间。无论此属性的值如何，都将立即进行第一次重试。 指定后跟时间单位的正整数

属性名称	数据类型	缺省值	描述
			，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
retryLimit	短整型 最小值：-1 最大值：10000	10	对已失败或者已回滚的任务连续重试的次数限制；达到此限制之后，就会认为该任务永久失败，并且不再进一步重试。如果值为 -1，那么将允许无限制地重试。
taskStoreRef	对顶级 databaseStore 元素的引用（字符串）。	defaultDatabaseStore	计划任务的持久性存储。

contextService

配置上下文捕获及传播至线程的方式。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何

属性名称	数据类型	缺省值	描述
			警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

属性名称	数据类型	缺省值	描述
baseContextRef	对顶级 contextService 元素的引用（字符串）。		指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。
id	字符串		唯一配置标识。
jndiName	字符串		JNDI 名称
onError	• IGNORE • FAIL • WARN	WARN	确定用于对配置错误作出响应的操作。例如，如果为此 contextService 配置了 securityContext，但未启用安全性功能，那么 onError 会确定是使错误配置部分失效、针对其发出警告还是将其忽略。 **IGNORE** 服务器在引发配置错误时将不会发出任何

属性名称	数据类型	缺省值	描述
			警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。

contextService > baseContext > baseContext

指定从其继承上下文的基本上下文服务（尚未在此上下文服务上定义此上下文）。

false

com.ibm.ws.context.service-factory

contextService > baseContext > classloaderContext

类装入器上下文传播配置。

false

contextService > baseContext > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > baseContext > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > baseContext > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

contextService > classloaderContext

类装入器上下文传播配置。

false

contextService > jeeMetadataContext

使提交上下文任务的应用程序组件的名称空间可用于该任务。

false

contextService > securityContext

指定了此属性时，会将工作发起方的安全上下文传播至工作单元。

false

contextService > syncToOSThreadContext

指定了此属性时，工作单元的 runAs 主体集的身份会与操作系统身份同步。

false

生成插件 (pluginConfiguration)

生成插件配置

- *httpEndpoint*
  - *accessLogging*
  - *httpOptions*
  - *sslOptions*
  - *tcpOptions*

属性名称	数据类型	缺省值	描述
connectTimeout	精度为秒的时间段	5s	标识应用程序服务器应保持与 Web 服务器的连接的最长时间。 指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
extendedHandshake	布尔型	false	如果为 true，那么 Web 服务器插件使用扩展握手来确定应用程序服务器是否正在运行。
ipv6Preferred	布尔型	false	首选 IPv6
logDirLocation	目录路径		标识 http_plugin.log 文件所在的目录。
pluginInstallRoot	字符串	.	文件系统中的 Web 容器插件安装位置
serverIOTimeout	精度为秒的时间段	900s	标识 Web 服务器插件等待发送请求或从应用程序服务器接收响应的最长时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒

属性名称	数据类型	缺省值	描述
			(s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
sslCertlabel	字符串	xigemaASCert	SSL 证书标签
sslKeyringLocation	字符串	keyring.kdb	SSL 密钥环的位置
sslStashfileLocation	字符串	keyring.sth	SSL 隐藏文件的位置
waitForContinue	布尔型	false	如果为 false（缺省值），那么 Web 服务器插件使用具有消息体的 HTTP 请求来发送“Expect: 100-continue”头。如果设置为 true，那么 Web 服务器插件使用每个 HTTP 请求发送“Expect: 100-continue”头。如果 Web 服务器与应用程序服务器之间有防火墙，并且您对没有请求主体的请求重试很敏感，请考虑将此值设置为 true。
webserverPort	整型 最小值：-1 最大值：65535	80	Web 服务器 HTTP 端口
webserverSecurePort	整型 最小值：-1 最大值：65535	443	Web 服务器 HTTPS 端口
wsServerIOTimeout	精度为秒的时间段		确定 Web 服务器插件等待发送请求或从应用程序服务器接收 websocket 响应的最长时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
wsServerIdleTimeout	精度为秒的时间段		确定 Web 服务器插件等待终止空闲 websocket 连

属性名称	数据类型	缺省值	描述
			接的最长时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

httpEndpoint

指定要包括在已生成 plugin-cfg.xml 文件中的 HTTP 端点的标识。缺省值为“defaultHttpEndpoint”。

false

属性名称	数据类型	缺省值	描述
accessLoggingRef	对顶级 httpAccessLogging 元素的引用（字符串）。		端点的 HTTP 访问日志记录配置。
已启用	布尔型	true	切换端点的可用性。值为 true 时，分派器将激活此端点以处理 HTTP 请求。
host	字符串	localhost	客户机用于请求资源的 IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或仅 DNS 主机名。可使用“*”来表示所有可用网络接口。
httpOptionsRef	对顶级 httpOptions 元素的引用（字符串）。	defaultHttpOptions	端点的 HTTP 协议选项。
httpPort	整型 最小值：-1 最大值：65535		用于客户机 HTTP 请求的端口。使用 -1 来禁用此端口。
httpsPort	整型 最小值：-1 最大值：65535		用于通过 SSL (https) 来保护的客户机 HTTP 请求的端口。使用 -1 来禁用此端口。

属性名称	数据类型	缺省值	描述
onError	• IGNORE • FAIL • WARN	WARN	启动端点失败后要执行的操作。 **IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		端点的 SSL 协议选项。
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	端点的 TCP 协议选项。

httpEndpoint > accessLogging

端点的 HTTP 访问日志记录配置。

false

属性名称	数据类型	缺省值	描述
已启用	布尔型	true	启用访问日志记录。
filePath	文件路径	${server.output.dir}/logs/http_access.log	访问日志文件的目录路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.output.dir}。

属性名称	数据类型	缺省值	描述
logFormat	字符串	%h %u %{t}W "%r" %s %b	指定在日志记录客户机访问信息时所使用的日志格式。
maxFileSize	整型 最小值：0	20	回滚之前日志文件的最大大小，以兆字节计；值为 0 时意味着无限制。
maxFiles	整型 最小值：0	2	移除最旧的日志文件之前将保留的最大日志文件数；值为 0 时意味着无限制。

httpEndpoint > httpOptions

端点的 HTTP 协议选项。

false

属性名称	数据类型	缺省值	描述
keepAliveEnabled	布尔型	true	启用持续连接（HTTP 保持活动）。如果为 true，那么连接将保持活动状态，以供多个顺序请求和响应复用。如果为 false，那么发送响应之后会关闭连接。
maxKeepAliveRequests	整型 最小值：-1	100	启用持续连接时，单个 HTTP 连接上可接受的最大持续请求数。值为 -1 时意味着不受限制。
persistTimeout	精度为秒的时间段	30s	将允许套接字在两个请求之间保持空闲的时间量。仅当启用持续连接时，此设置才适用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中

属性名称	数据类型	缺省值	描述
			。例如，1m30s 相当于 90 秒。
readTimeout	精度为秒的时间段	60s	发生第一次读取之后，用于等待读请求在套接字上完成的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
removeServerHeader	布尔型	false	从 HTTP 头中移除服务器实现信息，并且还要禁用缺省 xigemaAS 概要文件欢迎页面。
writeTimeout	精度为秒的时间段	60s	响应数据的每个部分在套接字上等待传输的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

httpEndpoint > sslOptions

端点的 SSL 协议选项。

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m

属性名称	数据类型	缺省值	描述
			) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

httpEndpoint > tcpOptions

端点的 TCP 协议选项。

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

快速启动安全性 (quickStartSecurity)

简单管理安全性配置。

属性名称	数据类型	缺省值	描述
userName	字符串		作为快速启动安全性配置的一部分进行定义的单用户。此用户被授予管理员角色。
userPassword	可逆向编码的密码（字符串）		作为快速启动安全性配置的一部分进行定义的单用户的密码。建议您对此密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。

Redis DB (redisDB)

Redis 数据库配置。

- *node*（见第 765 页）

属性名称	数据类型	缺省值	描述
jndiName	字符串		唯一的 JNDI 名称。
redisPoolRef	字符串		要引用的 redis 数据库连接池。
type	字符串	standAlone	Redis 服务器的类型。默认为单节点类型。可以为： • standAlone：单机 • cluster：集群
password	密码		单节点配置的 redis 数据库授权密码。仅支持单机配置。
database	整型	0	指定的数据库名称。
maxRedirections	整型	5	最大跳转数。
timeout	整型	2000	通信超时时间。默认为2000毫秒。
soTimeout	整型	2000	Socket 通信超时时间。默认为2000毫秒。
clientName	字符串		客户端名称。

node

Redis 服务器的节点信息。

必填项。

属性名称	数据类型	缺省值	描述
host	字符串		Redis 主机名称或 IP 地址。
port	整型	6379	Redis 服务端口号。

Redis 数据源连接池 (redisPool)

配置 redis 数据源连接池信息。

属性名称	数据类型	缺省值	描述
blockWhenExhausted	布尔值	true	指定连接耗尽时是否阻塞。默认为 true。该值可以为： • true：阻塞直到超时； • false：抛出异常。
evictionPolicyClassName	字符串		逐出策略类名(当连接超过最大空闲时间或连接数超过最大空闲连接数)。
jmxEnabled	布尔值	true	指定是否启用连接池的 JMX 管理功能。默认为true。
jmxNamePrefix	字符串		JMX 名称前缀。
lifo	布尔值	true	是否启用后进先出。默认为 true。
maxIdle	整型	8	最大空闲数。默认值为8。
maxTotal	整型	8	最大连接数。默认值为8。负数时表示没有限制大小。
maxWaitMillis	整型	-1	获取连接时的最大等待时间。 如果该值为负数，则会导致阻塞。 默认值为 -1。当设置了 blockWhenExhausted 为 true 时, 超时将会抛出异常。
minEvictableIdleTimeMillis	长整型	60000	逐出连接的最小空闲时间。默认为60000毫秒。

属性名称	数据类型	缺省值	描述
minIdle	整型		最小的空闲数。默认值为0。
numTestsPerEvictionRun	整型		每次逐出检查时,逐出的最大数目, 默认为-1。
softMinEvictableIdleTimeMillis	长整型	1000*60*30	指定在空闲多长时间后逐出。当空闲时间大于该值，并且空闲连接大于最大空闲数时直接逐出，不再根据 minEvictableIdleTimeMillis 判断。默认值为1000 * 60 * 30毫秒。
testOnBorrow	布尔值	false	获取连接时检查有效性。默认 false。
testOnReturn	布尔值	false	归还连接时检查有效性。默认 false。
testWhileIdle	布尔值	false	在空闲时检查有效性。默认 false。
timeBetweenEvictionRunsMillis	长整型	30000	逐出扫描的时间间隔。默认为30000毫秒

远程文件访问 (remoteFileAccess)

此元素包含用于控制对远程连接公开的文件访问级别的工件。

- *readDir*
- *writeDir*

readDir

允许远程客户机读取的目录。可有多个 readDir 元素，每个元素表示一个可引用变量或绝对路径的目录。缺省值为 ${wlp.install.dir}、${wlp.user.dir} 和 ${server.output.dir}

false

目录路径

writeDir

允许远程客户机读取和写入的目录。可有多个 writeDir 元素，每个元素表示一个可引用变量或绝对路径的目录。缺省值为一组空目录。

false

目录路径

请求计时 (requestTiming)

对于缓慢请求或已挂起的请求，提供警告和诊断信息。

属性名称	数据类型	缺省值	描述
hungRequestThreshold	具有毫秒精度的时间段	10m	请求在被视为已挂起之前可运行的持续时间。 指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
includeContextInfo	布尔型	true	指示上下文详细信息是否包含在日志输出中。
sampleRate	整型 最小值：1	1	为跟踪缓慢请求应该采用的采样比率。要每 n 个请求抽取一个请求作为样本，请将 sampleRate 设置为 n。要抽取所有请求作为样本，请将 sampleRate 设置为 1。
slowRequestThreshold	具有毫秒精度的时间段	10s	请求在被视为响应时间很长之前可运行的持续时间。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

资源适配器 (resourceAdapter)

定义资源适配器安装。

- *classloader*
  - *commonLibrary*
    - *file*
    - *fileset*
    - *folder*
  - *privateLibrary*

- *file*
- *fileset*
- *folder*

- *customize*

属性名称	数据类型	缺省值	描述
autoStart	布尔型		配置资源适配器的启动方式是在其部署时自动启动，还是在注入或查找资源时缓慢启动。
id	字符串		唯一配置标识。
location	文件、目录或 URL。		定义要安装的 RAR 文件的路径。
suppressUncoveredHttpMethodWarning	布尔型	false	指示在应用程序部署期间阻止未覆盖 HTTP 方法警告消息的选项。

classloader

定义应用程序类装入器的设置。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
classProviderRef	顶级 resourceAdapter 元素的引用列表（以逗号分隔的字符串）。		类提供程序引用的列表。搜索类或资源时，此类装入器将在搜索其自己的类路径之后授权给指定的类提供程序。
commonLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。会与其他类装入器共享库类实例。
delegation	• parentFirst • parentLast	parentFirst	控制父类装入器是用在此类装入器之前还是之后。如果选择“父代最先”，那么在搜索类路径之前，授权给直接父代。如果

属性名称	数据类型	缺省值	描述
			选择“父代最后”，那么在授权给直接父代之前，搜索类路径。 **parentFirst** 父代最先 **parentLast** 父代最后
privateLibraryRef	顶级库元素的引用列表（以逗号分隔的字符串）。		库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。
serializablePattern	字符串	NULL	为符合条件的 java bean 增加序列化接口。多个正则表达式使用‘%%’ 分隔，例如 ^com.test.\w+? %% ^com.test.\w+?。
serializableLocation	字符串	NULL	为符合条件的 java bean 增加序列化接口。serializableLocation 的值为文件路径，此文件中的每一行为一个 java 类路径。例如：com.test.Test。

classloader > commonLibrary

库引用的列表。会与其他类装入器共享库类实例。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述

属性名称	数据类型	缺省值	描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。
name	字符串		管理员的共享库的名称

classloader > commonLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > commonLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，

属性名称	数据类型	缺省值	描述
			m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > commonLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

classloader > privateLibrary

库引用的列表。库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
name	字符串		管理员的共享库的名称

classloader > privateLibrary > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

classloader > privateLibrary > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定

属性名称	数据类型	缺省值	描述
			后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

classloader > privateLibrary > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

customize

定制具有指定接口和/或实现类的激活规范、受管对象或连接工厂的配置属性元素。

false

属性名称	数据类型	缺省值	描述
实现	字符串		标准实现类名，应该针对该类名定制配置属性元素。
interface	字符串		标准接口类名，应该针对该类名定制配置属性元素。
suffix	字符串		覆盖配置属性元素的缺省后缀。例如，properties.rarModule1.CustomConnectionFactory 中的“CustomConnectionFactory”。当资源适配器提供了多种类型的连接工厂、受管对象或端点激活时，该后缀对于进行区别很有用。如果配置属性元素定制省略该

属性名称	数据类型	缺省值	描述
			后缀或将它保留为空白，那么不会使用后缀。

Real-Time Communications (rtcomm)

定义与 Rtcomm 功能部件及其所有相关组件相关的配置。与 Rtcomm 功能部件相关联的所有实时功能都在此单一实体下配置。

- *callQueue*
- *gateway*
- *iceServerURL*

属性名称	数据类型	缺省值	描述
alternateEndpointRoutingEnabled	布尔型	false	启用备用端点路由选项。那些要控制端点路由的管理员必须启用此选项。
messageServerHost	字符串	localhost	MQTT 代理程序的主机。指定的主机可以是 IP 地址或域名服务器 (DNS) 主机名。
messageServerPort	整型	1883	MQTT 代理程序侦听的端口。端口为任意有效端口号。
rtcommTopicPath	字符串	/rtcomm/	与此功能部件相关联的 MQTT 主题路径。在“Rtcomm 选项”内配置的所有 Rtcomm 主题名称都以此路径为前缀。要在消息代理程序内为此 Rtcomm 实例创建唯一名称空间，请修改此路径以使其唯一。
sharedSubscriptionPath	字符串		指定用作主题路径前缀的共享预订路径。使用共享预订时，消息代理程序需要此路径。
sslEnabled	布尔型	false	在 Rtcomm 功能部件和 MQTT 代理程序之间使用 SSL。
sslRef	字符串		要用于连接至启用 SSL 的 MQTT 代理的 SSL 配置的标识。

callQueue

定义调用队列的单个实例。此实例完全封装与单个队列相关的所有配置。

false

属性名称	数据类型	缺省值	描述
callQueueID	字符串	callQueueID	与调用队列主题相关联的名称。这是调用者用于调用特定队列的目标端点标识。
description	字符串		指定此调用队列实例的描述。此描述在服务查询响应中返回，可用于让客户机更好地了解此队列的相关信息。
id	字符串		唯一配置标识。
timeout	精度为秒的时间段	600s	终止在此队列中等待的调用之前等待的秒数。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

gateway

WebRTC 网关，提供从 Rtcomm 至 SIP 网络的连接

false

属性名称	数据类型	缺省值	描述
allowFromSipEndpointRef	顶级 sipEndpoint 元素的引用列表（以逗号分隔的字符串）。	defaultSipUAEndpoint	此连接器的 SIP 端点的列表。
externalPR	字符串		SIP 代理/注册器的 host:port 地址。如果指定了地址，那么所有出局 SIP 消息都将转发到此地址。

属性名称	数据类型	缺省值	描述
sipContainer	布尔型	false	出局 SIP 消息路由到在本地安装的 SIP 应用程序。

iceServerURL

指定可供客户机使用的交互连接建立 (ICE) 服务器 URL。客户机可请求要与 WebRTC 配合使用（通过使用 Rtcomm 服务查询）的 ICE 服务器 URL 的列表。以下是 ICE 服务器 URL 的示例格式：stun:hostname:port。

false

字符串

SAML Web SSO 2.0 认证 (samlWebSso20)

控制安全性断言标记语言 Web SSO 2.0 机制的操作。

- *authFilter*
  - *host*
  - *remoteAddress*
  - *requestUrl*
  - *userAgent*
  - *webApp*
- *authnContextClassRef*
- *pkixTrustEngine*
  - *crl*
  - *trustedIssuers*
  - *x509Certificate*

属性名称	数据类型	缺省值	描述
allowCreate	布尔型		允许 IdP 在请求用户不具有帐户的情况下创建新帐户。
allowCustomCacheKey	布尔型	true	允许生成定制高速缓存密钥以访问认证高速缓存和获取主体集。
authFilterRef	对顶级 authFilter 元素的引用（字符串）。		指定认证过滤器引用。
authnContextComparisonType	• minimum • better • maximum • exact	exact	如果指定了 authnContextClassRef，那么可设置 authnContextComparisonType。 **minimum** 最小值。认证语句中的认证上下文必

属性名称	数据类型	缺省值	描述
			须至少达到所指定认证上下文的其中一项的强度。 **better** 更强。认证语句中的认证上下文必须比所指定认证上下文中的任何一项更强。 **maximum** 最大值。认证语句中的认证上下文必须尽可能强，但不超过所指定认证上下文的至少其中一项的强度。 **exact** 完全匹配。认证语句中的认证上下文必须与所指定认证上下文中的至少一项的强度完全匹配。
authnRequestTime	具有毫秒精度的时间段	10m	指定从服务提供者生成并发送到 IdP 以请求 SAML 令牌的 authnReuqest 的生存时间段。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
authnRequestsSigned	布尔型	true	指示是否签署此服务提供者发送的 <samlp:AuthnRequest> 消息。
clockSkew	具有毫秒精度的时间段	5m	此属性用来指定验证 SAML 令牌时允许的时钟偏差（分钟）。指定后跟时间单位的正整数，时间单

属性名称	数据类型	缺省值	描述
			位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
createSession	布尔型	true	指定当前 HttpSession 不存在时是否创建 HttpSession。
customizeNameIDFormat	字符串		指定与 SAML 核心规范中未定义的名称标识格式对应的定制 URI 引用。
disableLtpaCookie	布尔型	true	处理 SAML 断言期间不创建 LTPA 令牌。改为创建特定服务提供者的 cookie。
enabled	布尔型	true	如果为 true，那么启用服务提供者；如果为 false，那么禁用服务提供者。
errorPageURL	字符串		指定在 SAML 验证失败的情况下将显示的错误页面。如果未指定此属性，那么所接收 SAML 无效，用户将重定向回 SAML IdP 以重新启动 SSO。
forceAuthn	布尔型	false	指示 IdP 是否应强制该用户重新认证。
groupIdentifier	字符串		指定 SAML 属性，该属性将用作已认证主体所属的组的名称。没有缺省值。
httpsRequired	布尔型	true	访问 SAML WebSSO 服务提供者端点（例如，acs 或元数据）时强制使用 SSL 通信。
id	字符串		唯一配置标识。
idpMetadata	字符串	${server.config.dir}/resources/security/idpMetadata.xml	指定 IdP 元数据文件。

属性名称	数据类型	缺省值	描述
includeTokenInSubject	布尔型	true	指定是否在主体集中包含 SAML 断言。
isPassive	布尔型	false	指示 IdP 不得控制最终用户界面。
keyAlias	字符串		用于查找签名和解密所需的专用密钥的密钥别名。如果密钥库正好具有一个密钥条目或如果具有一个别名为“samlsp”的密钥，那么这为可选。
keyStoreRef	字符串		密钥库，它包含用于签署 AuthnRequest 和解密 EncryptedAssertion 元素的专用密钥。缺省值为服务器的缺省密钥库。
loginPageURL	字符串		指定未认证请求重定向至的 SAML IdP 登录应用程序 URL。此属性触发 IdP 发起的 SSO，仅 IdP 发起的 SSO 需要此属性。
mapToUserRegistry	• User • No • Group	No	指定如何将身份映射至注册表用户。选项为 No、User 和 Group。缺省值为 No，不会使用用户注册表来创建用户主体集。 **User** 将 SAML 身份映射至注册表中定义的用户 **No** 不会将 SAML 身份映射至注册表中的用户或组 **Group** 将 SAML 身份映射至用户注册表中定义的组
nameIDFormat	• encrypted • customize	email	指定与 SAML 核心规范中定义的名称标识格式对应的 URI 引用。

属性名称	数据类型	缺省值	描述
	• persistent • x509SubjectName • email • transient • entity • unspecified • kerberos • windowsDomainQualifiedName		**encrypted** urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted **customize** 已定制名称标识格式。 **persistent** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent **x509SubjectName** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName **email** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress **transient** urn:oasis:names:tc:SAML:2.0:nameid-format:transient **entity** urn:oasis:names:tc:SAML:2.0:nameid-format:entity **unspecified** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified **kerberos** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos **windowsDomainQualifiedName** urn:oasis:names:tc:SAML:1.1:nameid

属性名称	数据类型	缺省值	描述
			-format:WindowsDomainQualifiedName
realmIdentifier	字符串		指定将用作领域名的 SAML 属性。缺省值为“issuer”。
realmName	字符串		mapToUserRegistry 设置为 No 或 Group 时，指定领域名。
sessionNotOnOrAfter	具有毫秒精度的时间段	120m	指示 SAML 会话持续时间的上限，之后 xigemaAS SP 应要求用户向 IdP 重新认证。如果从 IdP 返回的 SAML 令牌未包含 sessionNotOnOrAfter 断言，那么将使用此属性指定的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
signatureMethodAlgorithm	• SHA256 • SHA1	SHA256	指示此服务提供者所需的算法。 **SHA256** SHA-256 签名算法 **SHA1** SHA-1 签名算法
spHostAndPort	字符串		指定 SAML 服务提供者主机名和端口号。
tokenReplayTimeout	具有毫秒精度的时间段	30m	此属性用于指定 xigemaAS SP 应阻止令牌重放的时间长度。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多

属性名称	数据类型	缺省值	描述
			个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
userIdentifier	字符串		指定将用作主体集中的用户主体名称的 SAML 属性。缺省值为 NameID 断言。
userUniqueIdentifier	字符串		指定 SAML 属性，此属性将在应用于主体集中的 WSCredential 时用作唯一用户名。缺省值与 userId entifier 属性值相同。
wantAssertionsSigned	布尔型	true	指示签署此服务提供者接收的 <saml:Assertion> 元素的要求。

authFilter

指定认证过滤器引用。

false

authFilter > host

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

authFilter > remoteAddress

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ip	字符串		指定 IP 地址。
matchType	• lessThan • equals • greaterThan • contains • notContain	contains	指定匹配类型。 **lessThan** 小于 **equals** 等于 **greaterThan** 大于 **contains** 包含 **notContain** 不包含

authFilter > requestUrl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
urlPattern	字符串		指定 URL 模式。

authFilter > userAgent

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
agent	字符串		指定用户代理

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含

authFilter > webApp

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

authnContextClassRef

标识描述了认证上下文声明的认证上下文类的 URI 引用。缺省值为空。

false

字符串

pkixTrustEngine

指定用于评估 SAML 响应中 XML 签名的可信度和有效性的 PKIX 信任信息。请勿在 samlWebSso20 中指定多个 pkixTrustEngine。

false

属性名称	数据类型	缺省值	描述
trustAnchor	字符串		密钥库，包含在验证 SAMLResponse 和断

属性名称	数据类型	缺省值	描述
			言的签名时必需的公用密钥。

pkixTrustEngine > crl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
path	字符串		指定 CRL 的路径。

pkixTrustEngine > trustedIssuers

指定受信任 IdP 签发者的身份。如果值为“ALL_ISSUERS”，那么将信任所有 IdP 身份。

false

字符串

pkixTrustEngine > x509Certificate

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
path	字符串		指定 x509 证书的路径。

IBM SecureWay Directory Server LDAP 过滤器 (securewayLdapFilterProperties)

指定缺省 IBM SecureWay Directory Server LDAP 过滤器的列表。

属性名称	数据类型	缺省值	描述
groupFilter	字符串	(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))	用于在用户注册表中搜索组的 LDAP 过滤器子句。
groupIdMap	字符串	*:cn	用于将组的名称映射到 LDAP 条目的 LDAP 过滤器。
groupMemberIdMap	字符串	groupOfNames:member;groupOfUniqueNames:uniqueMember	用于确定用户是否具有组成员资格的 LDAP 过滤器。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
userFilter	字符串	(&(uid=%v)(objectclass=ePerson))	用于在用户注册表中搜索用户的 LDAP 过滤器子句。
userIdMap	字符串	*:uid	用于将用户的名称映射到 LDAP 条目的 LDAP 过滤器。

sessionPolicy

会话持久性策略配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
writeFrequency	• END_OF_SERVLET_SERVICE • MANUAL_UPDATE • TIME_BASED_WRITE	END_OF_SERVLET_SERVICE	Session 数据写入数据库中的时间点。默认值为 END_OF_SERVLET_SERVICE，即在 Servlet 执行完成后将会话数据写入持久存储。
writeContents	• ONLY_UPDATED_ATTRIBUTES • ALL_SESSION_ATTRIBUTES	ONLY_UPDATED_ATTRIBUTES	指定应写入持久存储的会话数据量。默认值为 ONLY_UPDATED_ATTRIBUTES，即仅会写入已更新的属性。

SIP 应用程序路由器 (sipApplicationRouter)

SIP 应用程序路由器的配置

属性名称	数据类型	缺省值	描述
carProvider	字符串	*	定制应用程序路由器提供程序的标准域名 (FQDN)。设置为星号 (*) 以使用任意可用提供程序。
enable.car	布尔型	true	将使用可用定制应用程序路由器路由应用程序；否则，将使用缺省应用程序路由器。
sipDarConfiguration	文件路径		缺省应用程序路由器 (DAR) 属性文件所在的位置。此值映射至 JSR 289 javax.servlet.sip.ar.dar.configuration。必须按 JSR 28

属性名称	数据类型	缺省值	描述
			9 中所指定那样配置 DAR。
sipNoRouteErrorCode	整型	403	没有活动 Servlet 可以映射至入局初始请求时，由 SIP 容器发送的错误响应代码。

SIP 容器 (sipContainer)

SIP Servlet 容器的配置

- *sipIntrospect*
- *sipTasksDispatcher*

属性名称	数据类型	缺省值	描述
dispatcherMessageQueueSize	整型	1000	SIP 容器队列在声明超负荷状态之前可以包含的最大任务数。
invalidateSessionOnShutdown	布尔型	false	SIP 容器关闭时，会话不再有效。缺省情况下，会话将保持活动状态，直到它们到期为止。
markInternalResponse	布尔型	false	通过对响应设置 com.ibm.websphere.sip.container.internal.message 属性来标记内部生成的响应。
maxAppSessions	整型	120000	允许一次进行的最大 SIP 应用程序会话数。
maxMessageRate	整型	5000	每个平均时间段允许的最大消息数。
maxResponseTime	具有毫秒精度的时间段	0ms	允许的最长响应时间（毫秒）。当设置为 0 时，响应时间不受限制。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

属性名称	数据类型	缺省值	描述
messageQueueBurstFactor	整型	10	消息队列大小的突发因子。消息队列大小设置为 (分派器消息队列大小) * (消息队列突发因子)。设置突发因子，以在废弃新消息之前通过在队列中提供更多空间来处理流量中的突发因子。
msgArrivalTimeAttr	布尔型	false	将消息到达时间另存为消息的属性。

sipIntrospect

SIP 自省的配置

false

属性名称	数据类型	缺省值	描述
method	• VERBOSE • SUCCINCT	SUCCINCT	生成服务器转储时，要包括的 SIP 状态详细信息的级别。 **VERBOSE** 包含服务器转储中 SIP 应用程序会话和 SIP 会话的详细状态。 **SUCCINCT** 服务器转储仅包括 SIP 应用程序会话和 SIP 会话标识

sipTasksDispatcher

并行 SIP 任务的执行程序的配置

false

属性名称	数据类型	缺省值	描述
concurrentContainerTasks	整型	15	可以分派给执行程序的最大并行 SIP 任务数。

SIP 端点 (sipEndpoint)

SIP 端点的配置

- *sslOptions*
- *tcpOptions*

属性名称	数据类型	缺省值	描述
bindRetries	整型	60	在端口绑定不成功时尝试的重试次数。
bindRetryDelay	具有毫秒精度的时间段	5000ms	两次重试之间的延迟（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500 ms 相当于 1.5 秒。
host	字符串	localhost	端点主机的 IP 地址
id	字符串		唯一配置标识。
sipTCPPort	整型	5060	TCP 端口号
sipTLSPort	整型	5061	TLS 端口号
sipUDPPort	整型	5060	UDP 端口号
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		定义 SSL 协议设置
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	定义 TCP 协议设置

sslOptions

定义 SSL 协议设置

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可

属性名称	数据类型	缺省值	描述
			以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

tcpOptions

定义 TCP 协议设置

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

SIP 堆栈 (sipStack)

SIP 堆栈的配置。

- *commaSeparatedHeaders*
- *hideMessageHeaders*
- *sipQuotedParameters*

属性名称	数据类型	缺省值	描述
acceptNonUtf8Bytes	布尔型	false	接受未采用 UTF-8 编码的字节序列。
auto100OnInvite	布尔型	true	SIP 容器在接收 INVITE 请求时会自动发送响应代码 100。
auto482OnMergedRequests	布尔型	false	SIP 容器在接收合并请求时会自动发送响应代码 482。在 SIP RFC 3261 的 8.2.2.2 一节中定义了此行为。
compactHeaders	• API • Never • Always • MtuExceeds	MtuExceeds	定义在 SIP 堆栈对消息编码时何时使用压缩头。 **API** 根据 JSR289 javax.servlet.sip.SipServletMessage.setHeaderForm(javax.servlet.sip.SipServletMessage.HeaderForm) 发送头 **Never** 从不以紧凑格式发送头 **Always** 始终以紧凑格式发送头 **MtuExceeds** 仅当超过 MTU 时才以紧凑格式发送头
forceConnectionReuse	布尔型	true	即使 via 头中存在别名参数，也会在后续请求中复用连接。
hideMessageBody	布尔型	false	在 SIP 容器日志中隐藏消息内容。
hideMessageReqUri	布尔型	false	在 SIP 容器日志中隐藏消息请求 URI。

属性名称	数据类型	缺省值	描述
networkAddressCacheTtl	字符串		SIP 容器保留已高速缓存的 InetAddress 项的时间，并且不会再次解析它。
pathMtu	整型	1500	按 RFC 3261-18.1.1 中定义的出站 UDP 请求的最大传输单元 (MTU)。
timerA	具有毫秒精度的时间段	500ms	按 RFC 3261 中定义的初始 INVITE 请求重新传输时间间隔（仅适用于 UDP）（毫秒）。缺省值等于 T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerB	具有毫秒精度的时间段	32000ms	INVITE 客户机事务超时计时器（毫秒），在 RFC 3261 中进行了定义。缺省值等于 64*T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerD	具有毫秒精度的时间段	32000ms	按 RFC 3261 中定义的 INVITE 响应重新传输的等待时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

属性名称	数据类型	缺省值	描述
timerE	具有毫秒精度的时间段	500ms	初始非 INVITE 请求重新传输时间间隔（仅适用于 UDP）（毫秒），在 RFC 3261 中进行了定义。缺省值等于 T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerF	具有毫秒精度的时间段	32000ms	按 RFC 3261 中定义的非 INVITE 事务超时计时器（毫秒）。缺省值等于 64*T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerG	具有毫秒精度的时间段	500ms	按 RFC 3261 中定义的初始 INVITE 响应重新传输时间间隔（毫秒）。缺省值等于 T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerH	具有毫秒精度的时间段	32000ms	按 RFC 3261 中定义的 ACK 回执的等待时间（毫秒）。缺省值等于 64*T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。

属性名称	数据类型	缺省值	描述
			可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerI	具有毫秒精度的时间段	5000ms	按 RFC 3261 中定义的 ACK 重新传输的等待时间（毫秒）。缺省值等于 T4。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerJ	具有毫秒精度的时间段	32000ms	按 RFC 3261 中定义的非 INVITE 请求重新传输的等待时间（毫秒）。缺省值等于 64*T1。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerK	具有毫秒精度的时间段	5000ms	按 RFC 3261 中定义的非 INVITE 响应重新传输的等待时间（毫秒）。缺省值等于 T4。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerT1	具有毫秒精度的时间段	500ms	按 RFC 3261 中定义的估算双向传递时间 (RTT)（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如

属性名称	数据类型	缺省值	描述
			，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
timerT2	具有毫秒精度的时间段	4000ms	按 RFC 3261 中定义的非 INVITE 请求和 INVITE 响应的最大重新传输时间间隔（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
timerT4	具有毫秒精度的时间段	5000ms	按 RFC 3261 中定义的消息在网络中的最长保留时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

commaSeparatedHeaders

应当以逗号分隔的头字段的列表。如果同一个头有多个值，这些头不会重复，值将在同一个头中用逗号分隔。

false

字符串

hideMessageHeaders

未输出到 SIP 容器日志的头的逗号分隔列表。

false

字符串

sipQuotedParameters

值带有引号的头参数的列表。

false

字符串

SPNEGO 认证 (spnego)

控制简单且受保护的 GSS-API 协商机制的操作。

- *authFilter*
  - *host*
  - *remoteAddress*
  - *requestUrl*
  - *userAgent*
  - *webApp*

属性名称	数据类型	缺省值	描述
authFilterRef	对顶级 authFilter 元素的引用（字符串）。		指定认证过滤器引用。
canonicalHostName	布尔型	true	控制是否要使用规范主机名。
disableFailOverToAppAuthType	布尔型	true	指定先使用 SPNEGO 登录 xigemaAS。但是，如果登录失败，那么会使用应用程序认证机制登录 xigemaAS。
includeClientGSSCredentialInSubject	布尔型	true	指定是否应将客户机授权凭证存储在客户机主体集中。
krb5Config	字符串		指定标准 Kerberos 配置路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.config.dir}。
krb5Keytab	字符串		指定标准 Kerberos 密钥表路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.config.dir}。Kerberos 密钥表文件包含类似用户密码的密钥的列表。主机通过将其 Kerberos 密钥表文件存储在本地磁盘上来保护这些文件，这一点至关重要。
ntlmTokenReceivedErrorPageURL	字符串		指定资源的 URL，此资源包含 SPNEGO 在 HTTP 响应中包含的内容，浏览器客户端应用程序会显示此 HTTP 响应。

属性名称	数据类型	缺省值	描述
servicePrincipalNames	字符串		指定 Kerberos 服务主体名称的逗号分隔列表。
spnegoNotSupportedErrorPageURL	字符串		指定资源的 URL，此资源包含 SPNEGO 在 HTTP 响应中包含的内容，浏览器客户端应用程序不支持 SPNEGO 认证时会显示此 HTTP 响应。
trimKerberosRealmNameFromPrincipal	布尔型	true	指定 SPNEGO 是否移除 Kerberos 主体用户名的后缀（以 Kerberos 域名之前的 @ 开头）。如果此属性设置为 true，那么将移除主体用户名的后缀。如果此属性设置为 false，那么将保留主体名的后缀。

authFilter

指定认证过滤器引用。

false

authFilter > host

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

authFilter > remoteAddress

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
ip	字符串		指定 IP 地址。
matchType	• lessThan • equals • greaterThan • contains • notContain	contains	指定匹配类型。 **lessThan** 小于 **equals** 等于 **greaterThan** 大于 **contains** 包含 **notContain** 不包含

authFilter > requestUrl

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
urlPattern	字符串		指定 URL 模式。

authFilter > userAgent

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
agent	字符串		指定用户代理

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含

authFilter > webApp

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
matchType	• equals • contains • notContain	contains	指定匹配类型。 **equals** 等于 **contains** 包含 **notContain** 不包含
name	字符串		指定名称。

SSL 指令表 (ssl)

具有标识、已定义的密钥库以及可选信任库的 SSL 指令表。

属性名称	数据类型	缺省值	描述
clientAuthentication	布尔型	false	指定是否启用客户机认证。如果设置为 true，那么需要进行客户机认证，并且客户机必须提供证书以获得服务器信任。
clientAuthenticationSupported	布尔型	false	指定客户机认证是否受支持。如果设置为 true，那么客户机认证支持意味着当客户机提供证书时，服

属性名称	数据类型	缺省值	描述
			务器将检查该客户机是否受信任。
clientKeyAlias	字符串		指定密钥库中用作密钥的证书的别名，该密钥将发送至已启用客户机认证的服务器。仅当密钥库具有多个密钥条目时，才需要此属性。
enabledCiphers	字符串		指定定制的密码列表。请使用空格来分隔列表中的每个密码。受支持的密码将取决于所使用的底层 JRE。请检查 JRE 以获取有效密码。
id	字符串		唯一配置标识。
keyStoreRef	字符串		包含 SSL 指令表的密钥条目的密钥库。此属性是必需的。
securityLevel	• MEDIUM • CUSTOM • HIGH • LOW	HIGH	指定 SSL 握手所使用的密码套件组。HIGH 表示 3DES 和 128 位及更多位的密码，MEDIUM 表示 DES 和 40 位密码，LOW 表示不加密的密码。如果使用了 enabledCiphers 属性，那么将忽略安全级别列表。 **MEDIUM** %repertoire.MEDIUM **CUSTOM** %repertoire.CUSTOM **HIGH** 密码套件 3DES 和 128 位及更多位 **LOW** %repertoire.LOW
serverKeyAlias	字符串		指定密钥库中用作服务器密钥的证书的别名。仅当

属性名称	数据类型	缺省值	描述
			密钥库具有多个密钥条目时，才需要此属性。
sslProtocol	字符串		SSL 握手协议。可以在底层 JRE 的 Java 安全套接字扩展 (JSSE) 提供程序的文档中找到协议值。使用 IBM JRE 时，缺省值为 SSL_TLS；使用 Oracle 时，缺省值为 SSL。
trustStoreRef	字符串	${keyStoreRef}	包含由 SSL 指令表用于签名验证的可信证书条目的密钥库。此属性是可选的。如果未指定，那么会将同一密钥库用于密钥条目和可信证书条目。

SSL 缺省指令表 (sslDefault)

SSL 服务的缺省指令表。

属性名称	数据类型	缺省值	描述
sslRef	字符串	defaultSSLConfig	SSL 指令表的名称，该指令表将用作缺省 SSL 指令表。如果未指定缺省 SSL 指令表，那么将使用名为 defaultSSLConfig 的缺省 SSL 指令表。

SSL 选项 (sslOptions)

用于传输的 SSL 协议配置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包

属性名称	数据类型	缺省值	描述
			括在单个条目中。例如，1m30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

TCP 选项 (tcpOptions)

定义 TCP 协议设置。

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

定时操作 (timedOperation)

定时操作帮助 xigemaAS 管理员了解其应用程序服务器中某些操作的运行速度比预期更慢的时间。

属性名称	数据类型	缺省值	描述
enableReport	布尔型	true	启用将报告定期生成到日志，该报告详细说明十个最长的定时操作，这些操作按类型分组，并且在每个组中按期望持续时间进行排序

属性名称	数据类型	缺省值	描述
maxNumberTimedOperations	整型	10000	定时操作总数达到此值时记录警告。
reportFrequency	具有小时精度的时间段		将报告生成到日志的频率，该报告详细说明十个最长的定时操作，这些操作按类型分组，并且在每个组中按实际持续时间进行排序。指定后跟时间单位的正整数，时间单位可以是小时 (h)。例如，将 12 小时指定为 12h。

事务管理器 (transaction)

事务管理器服务的配置属性

- *dataSource*
  - *connectionManager*
  - *containerAuthData*
  - *jaasLoginContextEntry*
  - *jdbcDriver*
    - *library*
      - *file*
      - *fileset*
      - *folder*
  - *properties*
  - *properties.datadirect.sqlserver*
  - *properties.db2.i.native*
  - *properties.db2.i.toolbox*
  - *properties.db2.jcc*
  - *properties.derby.client*
  - *properties.derby.embedded*
  - *properties.informix*
  - *properties.informix.jcc*
  - *properties.microsoft.sqlserver*
  - *properties.oracle*
  - *properties.sybase*
  - *recoveryAuthData*

属性名称	数据类型	缺省值	描述
acceptHeuristicHazard	布尔型	true	指定此服务器上的所有应用程序是否接受在包含一阶段资源的两阶段事务中

属性名称	数据类型	缺省值	描述
			产生启发式风险的可能性。
clientInactivityTimeout	精度为秒的时间段	60s	来自远程客户机的事务性请求之间的最大持续时间。任何超出此超时值的客户机不活动时间段都将导致在此应用程序服务器中回滚事务。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
dataSourceRef	对顶级 dataSource 元素的引用（字符串）。		这是可选属性。缺省情况下，事务服务会将其恢复日志存储在文件中。或者，可将日志存储在 RDBMS 中。此操作是通过设置用于定义非事务性数据源（事务日志存储在其中）的属性来实现。
defaultMaxShutdownDelay	精度为秒的时间段	2s	缺省最大关闭延迟。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
enableLoggingForHeuristicReporting	布尔型	false	指定应用程序服务器是否记录既涉及一阶段落实资源又涉及两阶段落实资源的事务的“准备落实一阶段资源”事件。
heuristicRetryInterval	精度为秒的时间段	60s	在资源管理器或远程伙伴发生瞬态异常之后，应用程序服务器在重试完成信号（例如落实或回滚）之前等待的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、

属性名称	数据类型	缺省值	描述
			分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
heuristicRetryWait	整型	5	应用程序服务器重试完成信号（例如落实或回滚）的次数。重试在资源管理器或远程伙伴发生瞬态异常之后进行。
lpsHeuristicCompletion	• COMMIT • MANUAL • ROLLBACK	ROLLBACK	指定用来完成具有启发式结果的事务的方向；此方向或者是应用程序服务器落实或回滚该事务，或者是依靠管理员手动完成该事务。允许值为：COMMIT、ROLLBACK 和 MANUAL **COMMIT** 落实 **MANUAL** 手动 **ROLLBACK** 回滚
propogatedOrBMTTranLifetimeTimeout	精度为秒的时间段	0	此服务器中所运行事务的事务超时的上限。此值应大于或等于为“事务超时总时间”所指定的值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoverOnStartup	布尔型	false	指定服务器是否应在服务器启动时开始事务恢复。
recoveryGroup	字符串		此服务器所属的恢复组的名称。恢复组成员可恢复

属性名称	数据类型	缺省值	描述
			组中其他服务器的事务日志。
recoveryIdentity	字符串		用于事务对等恢复的这个服务器的唯一身份。
timeoutGracePeriodEnabled	布尔型	false	指定事务超时与正在运行事务的服务方区域异常结束之间是否存在延迟。
totalTranLifetimeTimeout	精度为秒的时间段	120s	允许此服务器上启动的事务完成的缺省最长时间。会回滚在发生此超时之前未完成的任何此类事务。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
transactionLogDBTableSuffix	字符串		如果恢复日志存储在 RDBMS 表中，那么此属性允许在表名后加上字符串以使其对此服务器唯一。
transactionLogDirectory	字符串	${server.output.dir}/tranlog/	此服务器的目录，事务服务将该目录用于存储日志文件以进行恢复。
transactionLogSize	整型	1024	指定事务日志文件的大小，以千字节计。
waitForRecovery	布尔型	false	指定服务器是否应在接受新事务性工作之前等待事务恢复完成。

dataSource

这是可选属性。缺省情况下，事务服务会将其恢复日志存储在文件中。或者，可将日志存储在 RDBMS 中。此操作是通过设置用于定义非事务性数据源（事务日志存储在其中）的属性来实现。

false

属性名称	数据类型	缺省值	描述
beginTranForResultSetScrollingAPIs	布尔型	true	使用结果集滚动接口时尝试事务登记。

属性名称	数据类型	缺省值	描述
beginTranForVendorAPIs	布尔型	true	使用供应商接口时尝试事务登记。
commitOrRollbackOnCleanup	• commit • rollback		确定当关闭数据库工作单元 (AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。 **commit** 通过落实来清除连接。 **rollback** 通过回滚来清除连接。
connectionManagerRef	对顶级 connectionManager 元素的引用（字符串）。		数据源的连接管理器。
connectionSharing	• MatchOriginalRequest • MatchCurrentState	MatchOriginalRequest	指定共享连接的匹配方式。 **MatchOriginalRequest** 共享连接时，根据原始连接请求进行匹配。 **MatchCurrentState** 共享连接时，根据连接的当前状态进行匹配。
containerAuthDataRef	对顶级 authData 元素的引用（字符串）。		当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。
isolationLevel	• TRANSACTION_REPEATABLE_READ		缺省事务隔离级别。

属性名称	数据类型	缺省值	描述
	• TRANSACTION_READ_COMMITTED • TRANSACTION_SERIALIZABLE • TRANSACTION_READ_UNCOMMITTED • TRANSACTION_SNAPSHOT		**TRANSACTION_REPEATABLE_READ** 脏读取和不可重复读取受到阻止；可以进行幻象读取。 **TRANSACTION_READ_COMMITTED** 脏读取受到阻止；可以进行不可重复读取和幻象读取。 **TRANSACTION_SERIALIZABLE** 脏读取、不可重复读取和幻象读取受到阻止。 **TRANSACTION_READ_UNCOMMITTED** 可以进行脏读取、不可重复读取和幻象读取。 **TRANSACTION_SNAPSHOT** Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。
jaasLoginContextEntryRef	对顶级 jaasLoginContextEntry 元素的引用（字符串）。		用于认证的 JAAS 登录上下文条目。

属性名称	数据类型	缺省值	描述
jdbcDriverRef	对顶级 jdbcDriver 元素的引用（字符串）。		数据源的 JDBC 驱动程序。
jndiName	字符串		数据源的 JNDI 名称。
queryTimeout	精度为秒的时间段		SQL 语句的缺省查询超时。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此缺省值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
recoveryAuthDataRef	对顶级 authData 元素的引用（字符串）。		用于事务恢复的认证数据。
statementCacheSize	整型 最小值：0	10	每个连接的最大高速缓存语句数。
supplementalJDBCTrace	布尔型		补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。JDBC 驱动程序跟踪规范包括：com.ibm.ws.database.logwriter、com.ibm.ws.db2.logwriter、com.ibm.ws.derby.logwriter、com.ibm.ws.informix.logwriter、com.ibm.ws.oracle.logwriter、com.ibm.ws.sqlserver.logwriter 和 com.ibm.ws.sybase.logwriter。
syncQueryTimeoutWithTransactionTimeout	布尔型	false	将 JTA 事务中的剩余时间（如果有）用作

属性名称	数据类型	缺省值	描述
			SQL 语句的缺省查询超时。
transactional	布尔型	true	支持参与由应用程序服务器管理的事务。
type	• javax.sql.DataSource • javax.sql.XADataSource • javax.sql.ConnectionPoolDataSource		数据源的类型。 **javax.sql.DataSource** javax.sql.DataSource **javax.sql.XADataSource** javax.sql.XADataSource **javax.sql.ConnectionPoolDataSource** javax.sql.ConnectionPoolDataSource

dataSource > connectionManager

数据源的连接管理器。

false

属性名称	数据类型	缺省值	描述
agedTimeout	精度为秒的时间段	-1	池维护可以废弃物理连接之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
connectionTimeout	精度为秒的时间段	30s	连接请求超时之前的时间量。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小

属性名称	数据类型	缺省值	描述
			时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxConnectionsPerThread	整型 最小值：0		限制每个线程上打开的连接数。
maxIdleTime	精度为秒的时间段	30m	池维护期间可废弃未使用或空闲的连接之前的时间量（如果这样做不会使池大小减小到小于最小大小）。值为 -1 时会禁用此超时。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maxPoolSize	整型 最小值：0	50	池的最大物理连接数。值为 0 时意味着不受限制。
minPoolSize	整型 最小值：0		池中要保留的最小物理连接数。池不会进行预填充。时效超时可以覆盖此最小值。
numConnectionsPerThreadLocal	整型 最小值：0		为每个线程高速缓存所指定数目的连接。
purgePolicy	• ValidateAllConnections • FailingConnectionOnly • EntirePool	EntirePool	指定在池中检测到旧连接时要销毁哪些连接。 **ValidateAllConnections** 当检测到失效连接时，会测试连接并关闭

属性名称	数据类型	缺省值	描述
			发现存在错误的那些连接。 **FailingConnectionOnly** 当检测到失效连接时，会仅关闭发现存在错误的连接。 **EntirePool** 当检测到失效连接时，会将池中的所有连接都标记为失效，而且当这些连接不再使用时，会予以关闭。
reapTime	精度为秒的时间段	3m	两次运行池维护线程之间的时间量。值为 -1 时会禁用池维护。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

dataSource > containerAuthData

当绑定未使用 res-auth=CONTAINER 来指定资源引用的 authentication-alias 时应用的容器管理的认证的缺省认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请

属性名称	数据类型	缺省值	描述
			将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

dataSource > jaasLoginContextEntry

用于认证的 JAAS 登录上下文条目。

false

属性名称	数据类型	缺省值	描述
loginModuleRef	顶级 jaasLoginModule 元素的引用列表（以逗号分隔的字符串）。	hashtable,userNameAndPassword,certificate,token	对 JAAS 登录模块的标识的引用。
name	字符串		JAAS 配置条目的名称。

dataSource > jdbcDriver

数据源的 JDBC 驱动程序。

false

属性名称	数据类型	缺省值	描述
javax.sql.ConnectionPoolDataSource	字符串		javax.sql.ConnectionPoolDataSource 的 JDBC 驱动程序实现。
javax.sql.DataSource	字符串		javax.sql.DataSource 的 JDBC 驱动程序实现。
javax.sql.XADataSource	字符串		javax.sql.XADataSource 的 JDBC 驱动程序实现。
libraryRef	对顶级库元素的引用（字符串）。		标识 JDBC 驱动程序 JAR 和本机文件。

dataSource > jdbcDriver > library

标识 JDBC 驱动程序 JAR 和本机文件。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

dataSource > jdbcDriver > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

dataSource > jdbcDriver > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。

属性名称	数据类型	缺省值	描述
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

dataSource > jdbcDriver > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

dataSource > properties

数据源的 JDBC 供应商属性的列表。例如，databaseName="dbname" serverName="localhost" portNumber="50000"。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.datadirect.sqlserver

用于 Microsoft SQL Server 的 DataDirect Connect for JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
JDBCBehavior	• 1 • 0	0	JDBC 驱动程序属性：JDBCBehavior。值为：0 (JDBC 4.0) 或 1 (JDBC 3.0)。 **1** JDBC 3.0 **0** JDBC 4.0
XATransactionGroup	字符串		JDBC 驱动程序属性：XATransactionGroup。
XMLDescribeType	• longvarbinary • longvarchar		JDBC 驱动程序属性：XMLDescribeType。 **longvarbinary** longvarbinary **longvarchar** longvarchar
accountingInfo	字符串		JDBC 驱动程序属性：accountingInfo。

属性名称	数据类型	缺省值	描述
alternateServers	字符串		JDBC 驱动程序属性：alternateServers。
alwaysReportTriggerResults	布尔型		JDBC 驱动程序属性：alwaysReportTriggerResults。
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationMethod	• ntlm • userIdPassword • kerberos • auto		JDBC 驱动程序属性：authenticationMethod。 **ntlm** ntlm **userIdPassword** userIdPassword **kerberos** kerberos **auto** auto
bulkLoadBatchSize	长整型		JDBC 驱动程序属性：bulkLoadBatchSize。
bulkLoadOptions	长整型		JDBC 驱动程序属性：bulkLoadOptions。
clientHostName	字符串		JDBC 驱动程序属性：clientHostName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
codePageOverride	字符串		JDBC 驱动程序属性：codePageOverride。
connectionRetryCount	整型		JDBC 驱动程序属性：connectionRetryCount。
connectionRetryDelay	精度为秒的时间段		JDBC 驱动程序属性：connectionRetryDel

属性名称	数据类型	缺省值	描述
			ay。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
convertNull	整型		JDBC 驱动程序属性：convertNull。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateTimeInputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeInputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
dateTimeOutputParameterType	• dateTime • dateTimeOffset • auto		JDBC 驱动程序属性：dateTimeOutputParameterType。 **dateTime** dateTime **dateTimeOffset** dateTimeOffset **auto** auto
describeInputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeInputParameters。

属性名称	数据类型	缺省值	描述
			describeIfString describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
describeOutputParameters	• describeIfString • noDescribe • describeIfDateTime • describeAll		JDBC 驱动程序属性：describeOutputParameters。 **describeIfString** describeIfString **noDescribe** noDescribe **describeIfDateTime** describeIfDateTime **describeAll** describeAll
enableBulkLoad	布尔型		JDBC 驱动程序属性：enableBulkLoad。
enableCancelTimeout	布尔型		JDBC 驱动程序属性：enableCancelTimeout。
encryptionMethod	• loginSSL • requestSSL • SSL • noEncryption		JDBC 驱动程序属性：encryptionMethod。 **loginSSL** loginSSL

属性名称	数据类型	缺省值	描述
			requestSSL requestSSL **SSL** SSL **noEncryption** noEncryption
failoverGranularity	• disableIntegrityCheck • atomicWithRepositioning • nonAtomic • 原子		JDBC 驱动程序属性：failoverGranularity。 **disableIntegrityCheck** disableIntegrityCheck **atomicWithRepositioning** atomicWithRepositioning **nonAtomic** nonAtomic 原子 原子
failoverMode	• connect • select • extended		JDBC 驱动程序属性：failoverMode。 **connect** connect **select** select **extended** extended
failoverPreconnect	布尔型		JDBC 驱动程序属性：failoverPreconnect。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。

属性名称	数据类型	缺省值	描述
initializationString	字符串		JDBC 驱动程序属性：initializationString。
insensitiveResultSetBufferSize	整型		JDBC 驱动程序属性：insensitiveResultSetBufferSize。
javaDoubleToString	布尔型		JDBC 驱动程序属性：javaDoubleToString。
loadBalancing	布尔型		JDBC 驱动程序属性：loadBalancing。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
longDataCacheSize	整型 最小值：-1		JDBC 驱动程序属性：longDataCacheSize。
netAddress	字符串		JDBC 驱动程序属性：netAddress。
packetSize	整型 最小值：-1 最大值：128		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
queryTimeout	精度为秒的时间段		JDBC 驱动程序属性：queryTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m

属性名称	数据类型	缺省值	描述
			) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
resultsetMetaDataOptions	整型		JDBC 驱动程序属性：resultsetMetaDataOptions。
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
serverName	字符串	localhost	数据库正在其中运行的服务器。
snapshotSerializable	布尔型		JDBC 驱动程序属性：snapshotSerializable。
spyAttributes	字符串		JDBC 驱动程序属性：spyAttributes。
stringInputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringInputParameterType。 **varchar** varchar **nvarchar** nvarchar
stringOutputParameterType	• varchar • nvarchar	varchar	JDBC 驱动程序属性：stringOutputParameterType。 **varchar** varchar **nvarchar** nvarchar

属性名称	数据类型	缺省值	描述
suppressConnectionWarnings	布尔型		JDBC 驱动程序属性：suppressConnectionWarnings。
transactionMode	• explicit • implicit		JDBC 驱动程序属性：transactionMode。 **explicit** explicit **implicit** implicit
truncateFractionalSeconds	布尔型		JDBC 驱动程序属性：truncateFractionalSeconds。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
useServerSideUpdatableCursors	布尔型		JDBC 驱动程序属性：useServerSideUpdatableCursors。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
validateServerCertificate	布尔型		JDBC 驱动程序属性：validateServerCertificate。

dataSource > properties.db2.i.native

IBM DB2 for i Native JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only **all** all

属性名称	数据类型	缺省值	描述
			read call read call
autoCommit	布尔型	true	JDBC 驱动程序属性：autoCommit。
batchStyle	• 2.1 • 2.0	2.0	JDBC 驱动程序属性：batchStyle。 **2.1** 2.1 **2.0** 2.0
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128 **0** 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCR

属性名称	数据类型	缺省值	描述
			OLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive
dataTruncation	字符串	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串	*LOCAL	JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso • eur • ymd • julian • jis • usa • mdy		JDBC 驱动程序属性：dateFormat。 **dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• \, • b • . • / • -		JDBC 驱动程序属性：dateSeparator。 **\,** 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			b 字符 b **.** 句点字符 (.)。 **/** 正斜杠字符 (/)。 **-** 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 **\,** 逗号字符 (,)。 **.** 句点字符 (.)。
directMap	布尔型	true	JDBC 驱动程序属性：directMap。
doEscapeProcessing	布尔型	true	JDBC 驱动程序属性：doEscapeProcessing。
fullErrors	布尔型		JDBC 驱动程序属性：fullErrors。
libraries	字符串		JDBC 驱动程序属性：libraries。
lobThreshold	整型 最大值：500000	0	JDBC 驱动程序属性：lobThreshold。
lockTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个

属性名称	数据类型	缺省值	描述
			条目中。例如，1m30 s 相当于 90 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 63
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
networkProtocol	整型		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。
queryOptimizeGoal	• 2 • 1	2	JDBC 驱动程序属性：queryOptimizeGoal

属性名称	数据类型	缺省值	描述
			。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。 **2** *ALLIO **1** *FIRSTIO
reuseObjects	布尔型	true	JDBC 驱动程序属性：reuseObjects。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。
systemNaming	布尔型	false	JDBC 驱动程序属性：systemNaming。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• \, • b • : • .		JDBC 驱动程序属性：timeSeparator。 **\,** 逗号字符 (,)。 **b** 字符 b **:** 冒号字符 (:)。

属性名称	数据类型	缺省值	描述
			• 句点字符 (.)。
trace	布尔型		JDBC 驱动程序属性：trace。
transactionTimeout	精度为秒的时间段	0	JDBC 驱动程序属性：transactionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
useBlockInsert	布尔型	false	JDBC 驱动程序属性：useBlockInsert。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.db2.i.toolbox

IBM DB2 for i Toolbox JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
access	• read only • all • read call	all	JDBC 驱动程序属性：access。 **read only** read only

属性名称	数据类型	缺省值	描述
			all all **read call** read call
behaviorOverride	整型		JDBC 驱动程序属性：behaviorOverride。
bidiImplicitReordering	布尔型	true	JDBC 驱动程序属性：bidiImplicitReordering。
bidiNumericOrdering	布尔型	false	JDBC 驱动程序属性：bidiNumericOrdering。
bidiStringType	整型		JDBC 驱动程序属性：bidiStringType。
bigDecimal	布尔型	true	JDBC 驱动程序属性：bigDecimal。
blockCriteria	• 2 • 1 • 0	2	JDBC 驱动程序属性：blockCriteria。值为：0（不进行任何记录分块）、1（指定 FOR FETCH ONLY 时进行分块）或 2（指定 FOR UPDATE 时进行分块）。 **2** 2 **1** 1 **0** 0
blockSize	• 512 • 128 • 0 • 32 • 64 • 16 • 8 • 256	32	JDBC 驱动程序属性：blockSize。 **512** 512 **128** 128

属性名称	数据类型	缺省值	描述
			0 0 **32** 32 **64** 64 **16** 16 **8** 8 **256** 256
cursorHold	布尔型	false	JDBC 驱动程序属性：cursorHold。
cursorSensitivity	• asensitive • sensitive • insensitive	asensitive	JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **asensitive** asensitive **sensitive** sensitive **insensitive** insensitive
dataCompression	布尔型	true	JDBC 驱动程序属性：dataCompression。
dataTruncation	布尔型	true	JDBC 驱动程序属性：dataTruncation。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
dateFormat	• dmy • iso		JDBC 驱动程序属性：dateFormat。

属性名称	数据类型	缺省值	描述
	• eur • ymd • julian • jis • usa • mdy		**dmy** dmy **iso** iso **eur** eur **ymd** ymd **julian** julian **jis** jis **usa** usa **mdy** mdy
dateSeparator	• • \, • . • / • -		JDBC 驱动程序属性：dateSeparator。 空格字符 ( )。 \, 逗号字符 (,)。 . 句点字符 (.)。 / 正斜杠字符 (/)。 - 破折号字符 (-)。
decimalSeparator	• \, • .		JDBC 驱动程序属性：decimalSeparator。 \, 逗号字符 (,)。

属性名称	数据类型	缺省值	描述
			• 句点字符 (.)。
driver	• toolbox • native	toolbox	JDBC 驱动程序属性：driver。 **toolbox** toolbox **native** native
errors	• full • basic	basic	JDBC 驱动程序属性：errors。 **full** full **basic** basic
extendedDynamic	布尔型	false	JDBC 驱动程序属性：extendedDynamic。
extendedMetaData	布尔型	false	JDBC 驱动程序属性：extendedMetaData。
fullOpen	布尔型	false	JDBC 驱动程序属性：fullOpen。
holdInputLocators	布尔型	true	JDBC 驱动程序属性：holdInputLocators。
holdStatements	布尔型	false	JDBC 驱动程序属性：holdStatements。
isolationLevelSwitchingSupport	布尔型	false	JDBC 驱动程序属性：isolationLevelSwitchingSupport。
keepAlive	布尔型		JDBC 驱动程序属性：keepAlive。
lazyClose	布尔型	false	JDBC 驱动程序属性：lazyClose。
libraries	字符串		JDBC 驱动程序属性：libraries。

属性名称	数据类型	缺省值	描述
lobThreshold	整型 最小值：0 最大值：16777216	0	JDBC 驱动程序属性：lobThreshold。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maximumPrecision	• 31 • 63	31	JDBC 驱动程序属性：maximumPrecision。 **31** 31 **63** 64
maximumScale	整型 最小值：0 最大值：63	31	JDBC 驱动程序属性：maximumScale。
metaDataSource	整型 最小值：0 最大值：1	1	JDBC 驱动程序属性：metaDataSource。
minimumDivideScale	整型 最小值：0 最大值：9	0	JDBC 驱动程序属性：minimumDivideScale。
naming	• system • sql	sql	JDBC 驱动程序属性：naming。 **system** system **sql** sql

属性名称	数据类型	缺省值	描述
package	字符串		JDBC 驱动程序属性：package。
packageAdd	布尔型	true	JDBC 驱动程序属性：packageAdd。
packageCCSID	• 13488 • 1200	13488	JDBC 驱动程序属性：packageCCSID。值为：1200 (UCS-2) 或 13488 (UTF-16)。 **13488** 13488 (UTF-16) **1200** 1200 (UCS-2)
packageCache	布尔型	false	JDBC 驱动程序属性：packageCache。
packageCriteria	• default • select	default	JDBC 驱动程序属性：packageCriteria。 **default** default **select** select
packageError	• exception • none • warning	warning	JDBC 驱动程序属性：packageError。 **exception** exception **none** none **warning** warning
packageLibrary	字符串	QGPL	JDBC 驱动程序属性：packageLibrary。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
prefetch	布尔型	true	JDBC 驱动程序属性：prefetch。

属性名称	数据类型	缺省值	描述
prompt	布尔型	false	JDBC 驱动程序属性：prompt。
proxyServer	字符串		JDBC 驱动程序属性：proxyServer。
qaqqiniLibrary	字符串		JDBC 驱动程序属性：qaqqiniLibrary。
queryOptimizeGoal	整型 最小值：0 最大值：2	0	JDBC 驱动程序属性：queryOptimizeGoal。值为：1 (*FIRSTIO) 或 2 (*ALLIO)。
receiveBufferSize	整型 最小值：1		JDBC 驱动程序属性：receiveBufferSize。
remarks	• system • sql	system	JDBC 驱动程序属性：remarks。 **system** system **sql** sql
rollbackCursorHold	布尔型	false	JDBC 驱动程序属性：rollbackCursorHold。
savePasswordWhenSerialized	布尔型	false	JDBC 驱动程序属性：savePasswordWhenSerialized。
secondaryUrl	字符串		JDBC 驱动程序属性：secondaryUrl。
secure	布尔型	false	JDBC 驱动程序属性：secure。
sendBufferSize	整型 最小值：1		JDBC 驱动程序属性：sendBufferSize。
serverName	字符串		数据库正在其中运行的服务器。
serverTraceCategories	整型	0	JDBC 驱动程序属性：serverTraceCategories。

属性名称	数据类型	缺省值	描述
soLinger	精度为秒的时间段		JDBC 驱动程序属性：soLinger。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
soTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：soTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
sort	• hex • table • language	hex	JDBC 驱动程序属性：sort。 **hex** hex **table** table **language** language
sortLanguage	字符串		JDBC 驱动程序属性：sortLanguage。
sortTable	字符串		JDBC 驱动程序属性：sortTable。
sortWeight	• unique • shared		JDBC 驱动程序属性：sortWeight。 **unique** unique **shared** shared

属性名称	数据类型	缺省值	描述
tcpNoDelay	布尔型		JDBC 驱动程序属性：tcpNoDelay。
threadUsed	布尔型	true	JDBC 驱动程序属性：threadUsed。
timeFormat	• iso • eur • jis • usa • hms		JDBC 驱动程序属性：timeFormat。 **iso** iso **eur** eur **jis** jis **usa** usa **hms** hms
timeSeparator	• • \, • : • .		JDBC 驱动程序属性：timeSeparator。 空格字符 ( )。 \, 逗号字符 (,)。 : 冒号字符 (:)。 . 句点字符 (.)。
toolboxTrace	• diagnostic • information • conversion • error • thread • proxy • none • datastream • pcml • all • jdbc		JDBC 驱动程序属性：toolboxTrace。 **diagnostic** diagnostic **information** information **conversion** conversion

属性名称	数据类型	缺省值	描述
	• warning		**error** error **thread** thread **proxy** proxy **none** none **datastream** datastream **pcml** pcml **all** all **jdbc** jdbc **warning** warning
trace	布尔型		JDBC 驱动程序属性：trace。
translateBinary	布尔型	false	JDBC 驱动程序属性：translateBinary。
translateBoolean	布尔型	true	JDBC 驱动程序属性：translateBoolean。
translateHex	• binary • character	character	JDBC 驱动程序属性：translateHex。 **binary** binary **character** character
trueAutoCommit	布尔型	false	JDBC 驱动程序属性：trueAutoCommit。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
xaLooselyCoupledSupport	整型 最小值：0 最大值：1	0	JDBC 驱动程序属性：xaLooselyCoupledSupport。

dataSource > properties.db2.jcc

用于 DB2 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
activateDatabase	整型		JDBC 驱动程序属性：activateDatabase。
alternateGroupDatabaseName	字符串		JDBC 驱动程序属性：alternateGroupDatabaseName。
alternateGroupPortNumber	字符串		JDBC 驱动程序属性：alternateGroupPortNumber。
alternateGroupServerName	字符串		JDBC 驱动程序属性：alternateGroupServerName。
blockingReadConnectionTimeout	精度为秒的时间段		JDBC 驱动程序属性：blockingReadConnectionTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
clientAccountingInformation	字符串		JDBC 驱动程序属性：clientAccountingInformation。
clientApplicationInformation	字符串		JDBC 驱动程序属性：clientApplicationInformation。
clientRerouteAlternatePortNumber	字符串		JDBC 驱动程序属性：clientRerouteAlternatePortNumber.

属性名称	数据类型	缺省值	描述
clientRerouteAlternateServerName	字符串		JDBC 驱动程序属性：clientRerouteAlternateServerName。
clientUser	字符串		JDBC 驱动程序属性：clientUser。
clientWorkstation	字符串		JDBC 驱动程序属性：clientWorkstation。
connectionCloseWithInFlightTransaction	• 2 • 1		JDBC 驱动程序属性：connectionCloseWithInFlightTransaction。 **2** CONNECTION_CLOSE_WITH_ROLLBACK **1** CONNECTION_CLOSE_WITH_EXCEPTION
currentAlternateGroupEntry	整型		JDBC 驱动程序属性：currentAlternateGroupEntry。
currentFunctionPath	字符串		JDBC 驱动程序属性：currentFunctionPath。
currentLocaleLcCtype	字符串		JDBC 驱动程序属性：currentLocaleLcCtype。
currentLockTimeout	精度为秒的时间段		JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
currentPackagePath	字符串		JDBC 驱动程序属性：currentPackagePath。
currentPackageSet	字符串		JDBC 驱动程序属性：currentPackageSet。
currentSQLID	字符串		JDBC 驱动程序属性：currentSQLID。
currentSchema	字符串		JDBC 驱动程序属性：currentSchema。
cursorSensitivity	• 2 • 1 • 0		JDBC 驱动程序属性：cursorSensitivity。值为：0 (TYPE_SCROLL_SENSITIVE_STATIC)、1 (TYPE_SCROLL_SENSITIVE_DYNAMIC) 和 2 (TYPE_SCROLL_ASENSITIVE)。 **2** TYPE_SCROLL_ASENSITIVE **1** TYPE_SCROLL_SENSITIVE_DYNAMIC **0** TYPE_SCROLL_SENSITIVE_STATIC
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型	true	JDBC 驱动程序属性：deferPrepares。
driverType	• 2 • 4	4	JDBC 驱动程序属性：driverType。 **2** 第 2 类 JDBC 驱动程序。

属性名称	数据类型	缺省值	描述
			4 第 4 类 JDBC 驱动程序。
enableAlternateGroupSeamlessACR	布尔型		JDBC 驱动程序属性：enableAlternateGroupSeamlessACR。
enableClientAffinitiesList	• 2 • 1		JDBC 驱动程序属性：enableClientAffinitiesList。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableExtendedDescribe	• 2 • 1		JDBC 驱动程序属性：enableExtendedDescribe。 **2** NO **1** YES
enableExtendedIndicators	• 2 • 1		JDBC 驱动程序属性：enableExtendedIndicators。 **2** NO **1** YES
enableNamedParameterMarkers	• 2 • 1		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES

属性名称	数据类型	缺省值	描述
enableSeamlessFailover	• 2 • 1		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。
fullyMaterializeInputStreams	布尔型		JDBC 驱动程序属性：fullyMaterializeInputStreams。
fullyMaterializeInputStreamsOnBatchExecution	• 2 • 1		JDBC 驱动程序属性：fullyMaterializeInputStreamsOnBatchExecution。 **2** NO **1** YES
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
implicitRollbackOption	• 2 • 1 • 0		JDBC 驱动程序属性：implicitRollbackOption。 **2** IMPLICIT_ROLLBACK_OPTION_CLOSE_CONNECTION **1** IMPLICIT_ROLLBACK_OP

属性名称	数据类型	缺省值	描述
			TION_NOT_CLOSE_CONNECTION **0** IMPLICIT_ROLLBACK_OPTION_NOT_SET
interruptProcessingMode	• 2 • 1 • 0		JDBC 驱动程序属性：interruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL **0** INTERRUPT_PROCESSING_MODE_DISABLED
keepAliveTimeOut	精度为秒的时间段		JDBC 驱动程序属性：keepAliveTimeOut。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。

属性名称	数据类型	缺省值	描述
kerberosServerPrincipal	字符串		JDBC 驱动程序属性：kerberosServerPrincipal。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
maxConnCachedParamBufferSize	整型		JDBC 驱动程序属性：maxConnCachedParamBufferSize。
maxRetriesForClientReroute	整型		JDBC 驱动程序属性：maxRetriesForClientReroute。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	50000	在其中获取数据库连接的端口。
profileName	字符串		JDBC 驱动程序属性：profileName。
queryCloseImplicit	• 2 • 1		JDBC 驱动程序属性：queryCloseImplicit。值为：1 (QUERY_CLOSE_IMPLICIT_YES) 或 2 (QUERY_CLOSE_IMPLICIT_NO)。 **2** QUERY_CLOSE_IMPLICIT_NO **1** QUERY_CLOSE_IMPLICIT_YES

属性名称	数据类型	缺省值	描述
queryDataSize	整型 最小值：4096 最大值：65535		JDBC 驱动程序属性：queryDataSize。
queryTimeoutInterruptProcessingMode	• 2 • 1		JDBC 驱动程序属性：queryTimeoutInterruptProcessingMode。 **2** INTERRUPT_PROCESSING_MODE_CLOSE_SOCKET **1** INTERRUPT_PROCESSING_MODE_STATEMENT_CANCEL
readOnly	布尔型		JDBC 驱动程序属性：readOnly。
recordTemporalHistory	• 2 • 1		JDBC 驱动程序属性：recordTemporalHistory。 **2** NO **1** YES
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT

属性名称	数据类型	缺省值	描述
			1 HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
retryIntervalForClientReroute	精度为秒的时间段		JDBC 驱动程序属性：retryIntervalForClientReroute。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
securityMechanism	• 3 • 7 • 4 • 18 • 15 • 9 • 16		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY), 7 (ENCRYPTED_PASSWORD_SECURI

属性名称	数据类型	缺省值	描述
	• 13 • 11 • 12		TY)、9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)、11 (KERBEROS_SECURITY)、12 (ENCRYPTED_USER_AND_DATA_SECURITY)、13 (ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY)、15 (PLUGIN_SECURITY)、16 (ENCRYPTED_USER_ONLY_SECURITY)、18 (TLS_CLIENT_CERTIFICATE_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **18** TLS_CLIENT_CERTIFICATE_SECURITY **15** PLUGIN_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY

属性名称	数据类型	缺省值	描述
			16 ENCRYPTED_USER_ONLY_SECURITY **13** ENCRYPTED_USER_PASSWORD_AND_DATA_SECURITY **11** KERBEROS_SECURITY **12** ENCRYPTED_USER_AND_DATA_SECURITY
sendDataAsIs	布尔型		JDBC 驱动程序属性：sendDataAsIs。
serverName	字符串	localhost	数据库正在其中运行的服务器。
sessionTimeZone	字符串		JDBC 驱动程序属性：sessionTimeZone。
sqljCloseStmtsWithOpenResultSet	布尔型		JDBC 驱动程序属性：sqljCloseStmtsWithOpenResultSet。
sqljEnableClassLoaderSpecificProfiles	布尔型		JDBC 驱动程序属性：sqljEnableClassLoaderSpecificProfiles。
sslConnection	布尔型		JDBC 驱动程序属性：sslConnection。
streamBufferSize	整型		JDBC 驱动程序属性：streamBufferSize。
stripTrailingZerosForDecimalNumbers	• 2 • 1		JDBC 驱动程序属性：stripTrailingZerosForDecimalNumbers。 **2** NO

属性名称	数据类型	缺省值	描述
			1 YES
sysSchema	字符串		JDBC 驱动程序属性：sysSchema。
timerLevelForQueryTimeOut	• 2 • 1 • -1		JDBC 驱动程序属性：timerLevelForQueryTimeOut。 **2** QUERYTIMEOUT_CONNECTION_LEVEL **1** QUERYTIMEOUT_STATEMENT_LEVEL **-1** QUERYTIMEOUT_DISABLED
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceFileCount	整型		JDBC 驱动程序属性：traceFileCount。
traceFileSize	整型		JDBC 驱动程序属性：traceFileSize。
traceLevel	整型	0	下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURAT

属性名称	数据类型	缺省值	描述
			ION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
traceOption	• 1 • 0		JDBC 驱动程序属性：traceOption **1** 1 **0** 0
translateForBitData	• 2 • 1		JDBC 驱动程序属性：translateForBitData。 **2** SERVER_ENCODING_REPRESENTATION **1** HEX_REPRESENTATION
updateCountForBatch	• 2 • 1		JDBC 驱动程序属性：updateCountForBatch。

属性名称	数据类型	缺省值	描述
			2 TOTAL_UPDATE_COUNT **1** NO_UPDATE_COUNT
useCachedCursor	布尔型		JDBC 驱动程序属性：useCachedCursor。
useIdentityValLocalForAutoGeneratedKeys	布尔型		JDBC 驱动程序属性：useIdentityValLocalForAutoGeneratedKeys。
useJDBC41DefinitionForGetColumns	• 2 • 1		JDBC 驱动程序属性：useJDBC41DefinitionForGetColumns。 **2** NO **1** YES
useJDBC4ColumnNameAndLabelSemantics	• 2 • 1		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
useTransactionRedirect	布尔型		JDBC 驱动程序属性：useTransactionRedirect。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
xaNetworkOptimization	布尔型		JDBC 驱动程序属性：xaNetworkOptimization。

dataSource > properties.derby.client

Derby Network Client JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1527	在其中获取数据库连接的端口。
retrieveMessageText	布尔型	true	JDBC 驱动程序属性：retrieveMessageText。
securityMechanism	• 3 • 7	3	JDBC 驱动程序属性：securityMechanism

属性名称	数据类型	缺省值	描述
	• 4 • 9 • 8		。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY)、8 (STRONG_PASSWORD_SUBSTITUTE_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。 **3** CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY **8** STRONG_PASSWORD_SUBSTITUTE_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。

属性名称	数据类型	缺省值	描述
			shutdown 尝试连接时，关闭数据库。
ssl	• basic • off • peerAuthentication		JDBC 驱动程序属性：ssl。 **basic** basic **off** off **peerAuthentication** peerAuthentication
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512、TRACE_XA_CALLS=2048 和 TRACE_ALL=-1。

属性名称	数据类型	缺省值	描述
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.derby.embedded

Derby Embedded JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
connectionAttributes	字符串		JDBC 驱动程序属性：connectionAttributes。
createDatabase	• false • create		JDBC 驱动程序属性：createDatabase。 **false** 不自动创建数据库。 **create** 建立第一个连接时，会自动创建数据库（如果它不存在）。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
shutdownDatabase	• false • shutdown		JDBC 驱动程序属性：shutdownDatabase。 **false** 不关闭数据库。 **shutdown** 尝试连接时，关闭数据库。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.informix

Informix JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
databaseName	字符串		JDBC 驱动程序属性：databaseName。
ifxCLIENT_LOCALE	字符串		JDBC 驱动程序属性：ifxCLIENT_LOCALE。
ifxCPMAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMInitPoolSize	整型		JDBC 驱动程序属性：ifxCPMInitPoolSize。
ifxCPMMaxConnections	整型		JDBC 驱动程序属性：ifxCPMMaxConnections。

属性名称	数据类型	缺省值	描述
ifxCPMMaxPoolSize	整型		JDBC 驱动程序属性：ifxCPMMaxPoolSize。
ifxCPMMinAgeLimit	精度为秒的时间段		JDBC 驱动程序属性：ifxCPMMinAgeLimit。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxCPMMinPoolSize	整型		JDBC 驱动程序属性：ifxCPMMinPoolSize。
ifxCPMServiceInterval	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxCPMServiceInterval。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxDBANSIWARN	布尔型		JDBC 驱动程序属性：ifxDBANSIWARN。
ifxDBCENTURY	字符串		JDBC 驱动程序属性：ifxDBCENTURY。
ifxDBDATE	字符串		JDBC 驱动程序属性：ifxDBDATE。
ifxDBSPACETEMP	字符串		JDBC 驱动程序属性：ifxDBSPACETEMP。
ifxDBTEMP	字符串		JDBC 驱动程序属性：ifxDBTEMP。

属性名称	数据类型	缺省值	描述
ifxDBTIME	字符串		JDBC 驱动程序属性：ifxDBTIME。
ifxDBUPSPACE	字符串		JDBC 驱动程序属性：ifxDBUPSPACE。
ifxDB_LOCALE	字符串		JDBC 驱动程序属性：ifxDB_LOCALE。
ifxDELIMIDENT	布尔型		JDBC 驱动程序属性：ifxDELIMIDENT。
ifxENABLE_TYPE_CACHE	布尔型		JDBC 驱动程序属性：ifxENABLE_TYPE_CACHE。
ifxFET_BUF_SIZE	整型		JDBC 驱动程序属性：ifxFET_BUF_SIZE。
ifxGL_DATE	字符串		JDBC 驱动程序属性：ifxGL_DATE。
ifxGL_DATETIME	字符串		JDBC 驱动程序属性：ifxGL_DATETIME。
ifxIFXHOST	字符串	localhost	JDBC 驱动程序属性：ifxIFXHOST。
ifxIFX_AUTOFREE	布尔型		JDBC 驱动程序属性：ifxIFX_AUTOFREE。
ifxIFX_DIRECTIVES	字符串		JDBC 驱动程序属性：ifxIFX_DIRECTIVES。
ifxIFX_LOCK_MODE_WAIT	精度为秒的时间段	2s	JDBC 驱动程序属性：ifxIFX_LOCK_MODE_WAIT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

属性名称	数据类型	缺省值	描述
ifxIFX_SOC_TIMEOUT	具有毫秒精度的时间段		JDBC 驱动程序属性：ifxIFX_SOC_TIMEOUT。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
ifxIFX_USEPUT	布尔型		JDBC 驱动程序属性：ifxIFX_USEPUT。
ifxIFX_USE_STRENC	布尔型		JDBC 驱动程序属性：ifxIFX_USE_STRENC。
ifxIFX_XASPEC	字符串	y	JDBC 驱动程序属性：ifxIFX_XASPEC。
ifxINFORMIXCONRETRY	整型		JDBC 驱动程序属性：ifxINFORMIXCONRETRY。
ifxINFORMIXCONTIME	精度为秒的时间段		JDBC 驱动程序属性：ifxINFORMIXCONTIME。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
ifxINFORMIXOPCACHE	字符串		JDBC 驱动程序属性：ifxINFORMIXOPCACHE。
ifxINFORMIXSTACKSIZE	整型		JDBC 驱动程序属性：ifxINFORMIXSTACKSIZE。
ifxJDBCTEMP	字符串		JDBC 驱动程序属性：ifxJDBCTEMP。

属性名称	数据类型	缺省值	描述
ifxLDAP_IFXBASE	字符串		JDBC 驱动程序属性：ifxLDAP_IFXBASE。
ifxLDAP_PASSWD	字符串		JDBC 驱动程序属性：ifxLDAP_PASSWD。
ifxLDAP_URL	字符串		JDBC 驱动程序属性：ifxLDAP_URL。
ifxLDAP_USER	字符串		JDBC 驱动程序属性：ifxLDAP_USER。
ifxLOBCACHE	整型		JDBC 驱动程序属性：ifxLOBCACHE。
ifxNEWCODESET	字符串		JDBC 驱动程序属性：ifxNEWCODESET。
ifxNEWLOCALE	字符串		JDBC 驱动程序属性：ifxNEWLOCALE。
ifxNODEFDAC	字符串		JDBC 驱动程序属性：ifxNODEFDAC。
ifxOPTCOMPIND	字符串		JDBC 驱动程序属性：ifxOPTCOMPIND。
ifxOPTOFC	字符串		JDBC 驱动程序属性：ifxOPTOFC。
ifxOPT_GOAL	字符串		JDBC 驱动程序属性：ifxOPT_GOAL。
ifxPATH	字符串		JDBC 驱动程序属性：ifxPATH。
ifxPDQPRIORITY	字符串		JDBC 驱动程序属性：ifxPDQPRIORITY。
ifxPLCONFIG	字符串		JDBC 驱动程序属性：ifxPLCONFIG。
ifxPLOAD_LO_PATH	字符串		JDBC 驱动程序属性：ifxPLOAD_LO_PATH。

属性名称	数据类型	缺省值	描述
ifxPROTOCOLTRACE	整型		JDBC 驱动程序属性：ifxPROTOCOLTRACE。
ifxPROTOCOLTRACEFILE	字符串		JDBC 驱动程序属性：ifxPROTOCOLTRACEFILE。
ifxPROXY	字符串		JDBC 驱动程序属性：ifxPROXY。
ifxPSORT_DBTEMP	字符串		JDBC 驱动程序属性：ifxPSORT_DBTEMP。
ifxPSORT_NPROCS	布尔型		JDBC 驱动程序属性：ifxPSORT_NPROCS。
ifxSECURITY	字符串		JDBC 驱动程序属性：ifxSECURITY。
ifxSQLH_FILE	字符串		JDBC 驱动程序属性：ifxSQLH_FILE。
ifxSQLH_LOC	字符串		JDBC 驱动程序属性：ifxSQLH_LOC。
ifxSQLH_TYPE	字符串		JDBC 驱动程序属性：ifxSQLH_TYPE。
ifxSSLCONNECTION	字符串		JDBC 驱动程序属性：ifxSSLCONNECTION。
ifxSTMT_CACHE	字符串		JDBC 驱动程序属性：ifxSTMT_CACHE。
ifxTRACE	整型		JDBC 驱动程序属性：ifxTRACE。
ifxTRACEFILE	字符串		JDBC 驱动程序属性：ifxTRACEFILE。
ifxTRUSTED_CONTEXT	字符串		JDBC 驱动程序属性：ifxTRUSTED_CONTEXT。

属性名称	数据类型	缺省值	描述
ifxUSEV5SERVER	布尔型		JDBC 驱动程序属性：ifxUSEV5SERVER。
ifxUSE_DTENV	布尔型		JDBC 驱动程序属性：ifxUSE_DTENV。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
roleName	字符串		JDBC 驱动程序属性：roleName。
serverName	字符串		数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.informix.jcc

用于 Informix 的 IBM Data Server Driver for JDBC and SQLJ 的数据源属性。

false

属性名称	数据类型	缺省值	描述
DBANSIWARN	布尔型		JDBC 驱动程序属性：DBANSIWARN。
DBDATE	字符串		JDBC 驱动程序属性：DBDATE。
DBPATH	字符串		JDBC 驱动程序属性：DBPATH。

属性名称	数据类型	缺省值	描述
DBSPACETEMP	字符串		JDBC 驱动程序属性：DBSPACETEMP。
DBTEMP	字符串		JDBC 驱动程序属性：DBTEMP。
DBUPSPACE	字符串		JDBC 驱动程序属性：DBUPSPACE。
DELIMIDENT	布尔型		JDBC 驱动程序属性：DELIMIDENT。
IFX_DIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_DIRECTIVES。 **ON** ON **OFF** OFF
IFX_EXTDIRECTIVES	• ON • OFF		JDBC 驱动程序属性：IFX_EXTDIRECTIVES。 **ON** ON **OFF** OFF
IFX_UPDDESC	字符串		JDBC 驱动程序属性：IFX_UPDDESC。
IFX_XASTDCOMPLIANCE_XAEND	• 1 • 0		JDBC 驱动程序属性：IFX_XASTDCOMPLIANCE_XAEND。 **1** 1 **0** 0
INFORMIXOPCACHE	字符串		JDBC 驱动程序属性：INFORMIXOPCACHE。

属性名称	数据类型	缺省值	描述
INFORMIXSTACKSIZE	字符串		JDBC 驱动程序属性：INFORMIXSTACKSIZE。
NODEFDAC	• yes • no		JDBC 驱动程序属性：NODEFDAC。 **yes** yes **no** no
OPTCOMPIND	• 2 • 1 • 0		JDBC 驱动程序属性：OPTCOMPIND。 **2** 2 **1** 1 **0** 0
OPTOFC	• 1 • 0		JDBC 驱动程序属性：OPTOFC。 **1** 1 **0** 0
PDQPRIORITY	• HIGH • LOW • OFF		JDBC 驱动程序属性：PDQPRIORITY。 **HIGH** HIGH **LOW** LOW **OFF** OFF
PSORT_DBTEMP	字符串		JDBC 驱动程序属性：PSORT_DBTEMP。

属性名称	数据类型	缺省值	描述
PSORT_NPROCS	字符串 最大值：10		JDBC 驱动程序属性：PSORT_NPROCS。
STMT_CACHE	• 1 • 0		JDBC 驱动程序属性：STMT_CACHE。 **1** 1 **0** 0
currentLockTimeout	精度为秒的时间段	2s	JDBC 驱动程序属性：currentLockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
deferPrepares	布尔型		JDBC 驱动程序属性：deferPrepares。
driverType	整型	4	JDBC 驱动程序属性：driverType。
enableNamedParameterMarkers	整型		JDBC 驱动程序属性：enableNamedParameterMarkers。值为：1 (YES) 或 2 (NO)。
enableSeamlessFailover	整型		JDBC 驱动程序属性：enableSeamlessFailover。值为：1 (YES) 或 2 (NO)。
enableSysplexWLB	布尔型		JDBC 驱动程序属性：enableSysplexWLB。
fetchSize	整型		JDBC 驱动程序属性：fetchSize。

属性名称	数据类型	缺省值	描述
fullyMaterializeLobData	布尔型		JDBC 驱动程序属性：fullyMaterializeLobData。
keepDynamic	整型		JDBC 驱动程序属性：keepDynamic。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1526	在其中获取数据库连接的端口。
progressiveStreaming	• 2 • 1		JDBC 驱动程序属性：progressiveStreaming。值为：1 (YES) 或 2 (NO)。 **2** NO **1** YES
queryDataSize	整型 最小值：4096 最大值：10485760		JDBC 驱动程序属性：queryDataSize。
resultSetHoldability	• 2 • 1		JDBC 驱动程序属性：resultSetHoldability。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。

属性名称	数据类型	缺省值	描述
			2 CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
resultSetHoldabilityForCatalogQueries	• 2 • 1		JDBC 驱动程序属性：resultSetHoldabilityForCatalogQueries。值为：1 (HOLD_CURSORS_OVER_COMMIT) 或 2 (CLOSE_CURSORS_AT_COMMIT)。 **2** CLOSE_CURSORS_AT_COMMIT **1** HOLD_CURSORS_OVER_COMMIT
retrieveMessagesFromServerOnGetMessage	布尔型	true	JDBC 驱动程序属性：retrieveMessagesFromServerOnGetMessage。
securityMechanism	• 3 • 7 • 4 • 9		JDBC 驱动程序属性：securityMechanism。值为：3 (CLEAR_TEXT_PASSWORD_SECURITY)、4 (USER_ONLY_SECURITY)、7 (ENCRYPTED_PASSWORD_SECURITY) 和 9 (ENCRYPTED_USER_AND_PASSWORD_SECURITY)。

属性名称	数据类型	缺省值	描述
			3 CLEAR_TEXT_PASSWORD_SECURITY **7** ENCRYPTED_PASSWORD_SECURITY **4** USER_ONLY_SECURITY **9** ENCRYPTED_USER_AND_PASSWORD_SECURITY
serverName	字符串	localhost	数据库正在其中运行的服务器。
traceDirectory	字符串		JDBC 驱动程序属性：traceDirectory。
traceFile	字符串		JDBC 驱动程序属性：traceFile。
traceFileAppend	布尔型		JDBC 驱动程序属性：traceFileAppend。
traceLevel	整型		下列常量值的按位组合：TRACE_NONE=0、TRACE_CONNECTION_CALLS=1、TRACE_STATEMENT_CALLS=2、TRACE_RESULT_SET_CALLS=4、TRACE_DRIVER_CONFIGURATION=16、TRACE_CONNECTS=32、TRACE_DRDA_FLOWS=64、TRACE_RESULT_SET_META_DATA=128、TRACE_PARAMETER_META_DATA=256、TRACE_DIAGNOSTICS=512

属性名称	数据类型	缺省值	描述
			、TRACE_SQLJ=1024、TRACE_META_CALLS=8192、TRACE_DATASOURCE_CALLS=16384、TRACE_LARGE_OBJECT_CALLS=32768、TRACE_SYSTEM_MONITOR=131072、TRACE_TRACEPOINTS=262144 和 TRACE_ALL=-1。
useJDBC4ColumnNameAndLabelSemantics	整型		JDBC 驱动程序属性：useJDBC4ColumnNameAndLabelSemantics。值为：1 (YES) 或 2 (NO)。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.microsoft.sqlserver

Microsoft SQL Server JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
URL	字符串		用于连接至数据库的 URL。示例：jdbc:sqlserver://localhost:1433;databaseName=myDB。
applicationIntent	• ReadOnly • ReadWrite		JDBC 驱动程序属性：applicationIntent。 **ReadOnly** ReadOnly **ReadWrite** ReadWrite
applicationName	字符串		JDBC 驱动程序属性：applicationName。
authenticationScheme	• NativeAuthentication		JDBC 驱动程序属性：authenticationScheme。

属性名称	数据类型	缺省值	描述
	• JavaKerberos		**NativeAuthentication** NativeAuthentication **JavaKerberos** JavaKerberos
databaseName	字符串		JDBC 驱动程序属性：databaseName。
encrypt	布尔型		JDBC 驱动程序属性：encrypt。
failoverPartner	字符串		JDBC 驱动程序属性：failoverPartner。
hostNameInCertificate	字符串		JDBC 驱动程序属性：hostNameInCertificate。
instanceName	字符串		JDBC 驱动程序属性：instanceName。
integratedSecurity	布尔型		JDBC 驱动程序属性：integratedSecurity。
lastUpdateCount	布尔型		JDBC 驱动程序属性：lastUpdateCount。
lockTimeout	具有毫秒精度的时间段		JDBC 驱动程序属性：lockTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可

属性名称	数据类型	缺省值	描述
			以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
multiSubnetFailover	布尔型		JDBC 驱动程序属性：multiSubnetFailover。
packetSize	整型 最小值：512 最大值：32767		JDBC 驱动程序属性：packetSize。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型		在其中获取数据库连接的端口。
responseBuffering	• full • adaptive		JDBC 驱动程序属性：responseBuffering。 **full** full **adaptive** adaptive
selectMethod	• direct • cursor		JDBC 驱动程序属性：selectMethod。 **direct** direct **cursor** cursor
sendStringParametersAsUnicode	布尔型	false	JDBC 驱动程序属性：sendStringParametersAsUnicode。
sendTimeAsDatetime	布尔型		JDBC 驱动程序属性：sendTimeAsDatetime。
serverName	字符串	localhost	数据库正在其中运行的服务器。

属性名称	数据类型	缺省值	描述
trustServerCertificate	布尔型		JDBC 驱动程序属性：trustServerCertificate。
trustStore	字符串		JDBC 驱动程序属性：trustStore。
trustStorePassword	可逆向编码的密码（字符串）		JDBC 驱动程序属性：trustStorePassword。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。
workstationID	字符串		JDBC 驱动程序属性：workstationID。
xopenStates	布尔型		JDBC 驱动程序属性：xopenStates。

dataSource > properties.oracle

Oracle JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
ONSConfiguration	字符串		JDBC 驱动程序属性：ONSConfiguration。
TNSEntryName	字符串		JDBC 驱动程序属性：TNSEntryName。
URL	字符串		用于连接至数据库的 URL。示例：jdbc:oracle:thin:@//localhost:1521/sample 或 jdbc:oracle:oci:@//localhost:1521/sample。
connectionProperties	字符串		JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
driverType	• oci • thin	thin	JDBC 驱动程序属性：driverType。

属性名称	数据类型	缺省值	描述
			oci oci **thin** thin
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	字符串		JDBC 驱动程序属性：networkProtocol。
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	1521	在其中获取数据库连接的端口。
serverName	字符串	localhost	数据库正在其中运行的服务器。
serviceName	字符串		JDBC 驱动程序属性：serviceName。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

dataSource > properties.sybase

Sybase JDBC 驱动程序的数据源属性。

false

属性名称	数据类型	缺省值	描述
SERVER_INITIATED_TRANSACTIONS	• false • true	false	JDBC 驱动程序属性：SERVER_INITIATED_TRANSACTIONS。

属性名称	数据类型	缺省值	描述
			false false **true** true
connectionProperties	字符串	SELECT_OPENS_CURSOR=true	JDBC 驱动程序属性：connectionProperties。
databaseName	字符串		JDBC 驱动程序属性：databaseName。
loginTimeout	精度为秒的时间段		JDBC 驱动程序属性：loginTimeout。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
networkProtocol	• SSL • socket		JDBC 驱动程序属性：networkProtocol。 **SSL** SSL **socket** socket
password	可逆向编码的密码（字符串）		建议使用容器管理的认证别名，而不配置此属性。
portNumber	整型	5000	在其中获取数据库连接的端口。
resourceManagerName	字符串		JDBC 驱动程序属性：resourceManagerName。
serverName	字符串	localhost	数据库正在其中运行的服务器。
user	字符串		建议使用容器管理的认证别名，而不配置此属性。

属性名称	数据类型	缺省值	描述
version	整型		JDBC 驱动程序属性：version。

dataSource > recoveryAuthData

用于事务恢复的认证数据。

false

属性名称	数据类型	缺省值	描述
password	可逆向编码的密码（字符串）		连接至 EIS 时要使用的用户密码。可以采用明文或编码格式存储该值。建议您对该密码进行编码。要对该密码进行编码，请将 securityUtility 工具与编码选项配合使用。
user	字符串		连接至 EIS 时要使用的用户名称。

信任关联拦截器 (trustAssociation)

控制信任关联拦截器 (TAI) 的操作。

- *interceptors*
  - *library*
    - *file*
    - *fileset*
    - *folder*
  - *properties*

属性名称	数据类型	缺省值	描述
failOverToAppAuthType	布尔型	false	允许拦截器后退到应用程序认证机制。
id	字符串		唯一配置标识。
invokeForUnprotectedURI	布尔型	false	控制是否为不受保护的 URI 调用 TAI。

拦截器

唯一配置标识。

false

属性名称	数据类型	缺省值	描述
className	字符串		拦截器类的标准程序包名。
enabled	布尔型	true	启用或禁用拦截器。
id	字符串		唯一配置标识。
invokeAfterSSO	布尔型	true	在单点登录 (SSO) 之后调用拦截器。
invokeBeforeSSO	布尔型	false	在单点登录 (SSO) 之前调用拦截器。
libraryRef	对顶级库元素的引用（字符串）。		对共享库配置的标识的引用。

interceptors > library

对共享库配置的标识的引用。

false

属性名称	数据类型	缺省值	描述
apiTypeVisibility	字符串	spec,ibm-api,api	此库的类装入器将能够看到的 API 包的类型，格式为下列项的任何组合的逗号分隔列表：spec、ibm-api、spi 和 third-party。
description	字符串		管理员的共享库的描述
filesetRef	顶级文件集元素的引用列表（以逗号分隔的字符串）。		所引用文件集的标识
name	字符串		管理员的共享库的名称

interceptors > library > file

所引用文件的标识

false

属性名称	数据类型	缺省值	描述
id	字符串		唯一配置标识。
name	文件路径		标准文件名

interceptors > library > fileset

所引用文件集的标识

false

属性名称	数据类型	缺省值	描述
caseSensitive	布尔型	true	用于指示搜索是否应该区分大小写的布尔值（缺省值：true）。
dir	目录路径	${server.config.dir}	用于搜索文件的基本目录。
excludes	字符串		要从搜索结果中排除的文件名模式的逗号或空格分隔列表，缺省情况下不排除任何文件。
id	字符串		唯一配置标识。
includes	字符串	*	要包括在搜索结果中的文件名模式的逗号或空格分隔列表（缺省值：*）。
scanInterval	具有毫秒精度的时间段	0	检查文件集更改的扫描时间间隔，格式为长整型加上时间单位后缀（h 表示小时，m 表示分钟，s 表示秒，ms 表示毫秒），例如 2ms 或 5s。缺省情况下为已禁用 (scanInterval=0)。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

interceptors > library > folder

所引用文件夹的标识

false

属性名称	数据类型	缺省值	描述
dir	目录路径		要包含在用于定位资源文件的库类路径中的目录或文件夹
id	字符串		唯一配置标识。

interceptors > properties

拦截器属性的集合。

false

用户信息 (userInfo)

指定包括在 openID 提供程序的响应中的用户信息。

属性名称	数据类型	缺省值	描述
alias	字符串	email	指定别名。
count	整型 最小值：1	1	指定包括在 openID 提供程序的响应中的用户信息量。
id	字符串		唯一配置标识。
required	布尔型	true	指定是否需要用户信息。
uriType	字符串	http://axschema.org/contact/email	指定 URI 类型。

变量声明 (variable)

通过指定变量的名称和值来声明新变量。

属性名称	数据类型	缺省值	描述
name	字符串		变量的名称。
value	字符串		要赋给变量的值。

虚拟主机 (virtualHost)

虚拟主机将提供用于将 Web 应用程序配置为特定主机名的逻辑分组。缺省虚拟主机 (default_host) 适合于大多数简单配置。

- *allowFromEndpoint*
  - *accessLogging*
  - *httpOptions*
  - *sslOptions*
  - *tcpOptions*
- *hostAlias*

属性名称	数据类型	缺省值	描述
allowFromEndpointRef	顶级 httpEndpoint 元素的引用列表（以逗号分隔的字符串）。		指定一个或多个 HTTP 端点的标识以将此虚拟主机的入站流量限制为所指定的端点。
enabled	布尔型	true	启用此虚拟主机。
id	字符串		唯一配置标识。

allowFromEndpoint

指定一个或多个 HTTP 端点的标识以将此虚拟主机的入站流量限制为所指定的端点。

false

属性名称	数据类型	缺省值	描述
accessLoggingRef	对顶级 httpAccessLogging 元素的引用（字符串）。		端点的 HTTP 访问日志记录配置。
enabled	布尔型	true	切换端点的可用性。值为 true 时，分派器将激活此端点以处理 HTTP 请求。
host	字符串	localhost	客户机用于请求资源的 IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或仅 DNS 主机名。可使用“*”来表示所有可用网络接口。
httpOptionsRef	对顶级 httpOptions 元素的引用（字符串）。	defaultHttpOptions	端点的 HTTP 协议选项。
httpPort	整型 最小值：-1 最大值：65535		用于客户机 HTTP 请求的端口。使用 -1 来禁用此端口。
httpsPort	整型 最小值：-1 最大值：65535		用于通过 SSL (https) 来保护的客户机 HTTP 请求的端口。使用 -1 来禁用此端口。
id	字符串		唯一配置标识。
onError	• IGNORE • FAIL	WARN	启动端点失败后要执行的操作。

属性名称	数据类型	缺省值	描述
	• WARN		**IGNORE** 服务器在引发配置错误时将不会发出任何警告和错误消息。 **FAIL** 服务器在第一次发生错误时将发出警告或错误消息，然后停止服务器。 **WARN** 服务器在引发配置错误时将发出警告和错误消息。
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		端点的 SSL 协议选项。
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	端点的 TCP 协议选项。

allowFromEndpoint > accessLogging

端点的 HTTP 访问日志记录配置。

false

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	启用访问日志记录。
filePath	文件路径	${server.output.dir}/logs/http_access.log	访问日志文件的目录路径和名称。指定目录路径时可以使用标准变量替换，如 ${server.output.dir}。
logFormat	字符串	%h %u %{t}W "%r" %s %b	指定在日志记录客户机访问信息时所使用的日志格式。

属性名称	数据类型	缺省值	描述
maxFileSize	整型 最小值：0	20	回滚之前日志文件的最大大小，以兆字节计；值为 0 时意味着无限制。
maxFiles	整型 最小值：0	2	移除最旧的日志文件之前将保留的最大日志文件数；值为 0 时意味着无限制。

allowFromEndpoint > httpOptions

端点的 HTTP 协议选项。

false

属性名称	数据类型	缺省值	描述
keepAliveEnabled	布尔型	true	启用持续连接（HTTP 保持活动）。如果为 true，那么连接将保持活动状态，以供多个顺序请求和响应复用。如果为 false，那么发送响应之后会关闭连接。
maxKeepAliveRequests	整型 最小值：-1	100	启用持续连接时，单个 HTTP 连接上可接受的最大持续请求数。值为 -1 时意味着不受限制。
persistTimeout	精度为秒的时间段	30s	将允许套接字在两个请求之间保持空闲的时间量。仅当启用持续连接时，此设置才适用。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
readTimeout	精度为秒的时间段	60s	发生第一次读取之后，用于等待读请求在

属性名称	数据类型	缺省值	描述
			套接字上完成的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。
removeServerHeader	布尔型	false	从 HTTP 头中移除服务器实现信息，并且还要禁用缺省 xigemaAS 概要文件欢迎页面。
writeTimeout	精度为秒的时间段	60s	响应数据的每个部分在套接字上等待传输的时间量。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m30s 相当于 90 秒。

allowFromEndpoint > sslOptions

端点的 SSL 协议选项。

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单

属性名称	数据类型	缺省值	描述
			个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

allowFromEndpoint > tcpOptions

端点的 TCP 协议选项。

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

hostAlias

使用 host:port 语法将主机和端口与此虚拟主机关联。所指定的主机可以是 IP 地址、具有域名后缀的域名服务器 (DNS) 主机名、DNS 主机名或者与所有主机名中的通配符匹配的 *。请注意，IPv6 地址必须用 [] 括起来。

false

字符串

xigemaAS JMS 端点 (wasJmsEndpoint)

xigemaAS JMS 入局连接请求的配置属性。

- *sslOptions*
- *tcpOptions*

属性名称	数据类型	缺省值	描述
enabled	布尔型	true	切换 xigemaAS JMS 端点的可用性。
host	字符串	localhost	客户机用于请求资源的 IP 地址、带域名后缀的域名服务器 (DNS) 主机名，或仅 DNS 主机名。可使用“*”来表示所有可用网络接口。
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		xigemaAS JMS 端点的 SSL 协议选项。
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	xigemaAS JMS 端点的 TCP 协议选项。
wasJmsPort	整型 最小值：-1 最大值：65535	7276	用于 xigemaAS JMS 客户机消息传递应用程序连接请求的端口。使用 -1 来禁用此端口。
wasJmsSSLPort	整型 最小值：-1 最大值：65535	7286	用于使用 SSL 保护的 xigemaAS JMS 客户机消息传递应用程序连接请求的端口。使用 -1 来禁用此端口。

sslOptions

xigemaAS JMS 端点的 SSL 协议选项。

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可

属性名称	数据类型	缺省值	描述
			以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

tcpOptions

xigemaAS JMS 端点的 TCP 协议选项。

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

xigemaAS JMS 出站 (wasJmsOutbound)

xigemaAS JMS 传出连接请求的配置属性。

- *sslOptions*
- *tcpOptions*

属性名称	数据类型	缺省值	描述
id	字符串		WAS JMS 出站的名称。

属性名称	数据类型	缺省值	描述
sslOptionsRef	对顶级 sslOptions 元素的引用（字符串）。		WAS JMS 出站的 SSL 协议选项
tcpOptionsRef	对顶级 tcpOptions 元素的引用（字符串）。	defaultTCPOptions	WAS JMS 出站的 TCP 协议选项
useSSL	布尔型	false	将该值设置为 true 以启用安全通信信道

sslOptions

xigemaAS JMS 出站的 SSL 协议选项

false

属性名称	数据类型	缺省值	描述
sessionTimeout	精度为秒的时间段	1d	用于等待读或写请求在套接字上完成的时间量。特定于协议的超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。例如，将 30 秒指定为 30s。可以将多个值包括在单个条目中。例如，1m 30s 相当于 90 秒。
sslRef	字符串		缺省 SSL 配置指令表。缺省值为 defaultSSLSettings。
suppressHandshakeErrors	布尔型	false	禁止记录 SSL 握手错误。正常操作期间可能会发生 SSL 握手错误，但是如果 SSL 行为异常，那么这些消息很有用。

tcpOptions

xigemaAS JMS 出站的 TCP 协议选项

false

属性名称	数据类型	缺省值	描述
inactivityTimeout	具有毫秒精度的时间段	60s	用于等待读或写请求在套接字上完成的时间量。特定于协议的

属性名称	数据类型	缺省值	描述
			超时将覆盖此值。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
soReuseAddr	布尔型	true	允许立即重新绑定到没有任何处于活动状态的侦听器的端口。

Web 容器 (webContainer)

Web 容器的配置。

属性名称	数据类型	缺省值	描述
allowExpressionFactoryPerApp	布尔型	false	切换以装入该应用程序设置的 ExpressionFactory。如果您要使用的定制 EL 实现（例如，JUEL）需要设置自己 ExpressionFactory，请启用此定制属性。
allowIncludeSendError	布尔型	false	允许 RequestDispatch 在 Include 方法上发送错误。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.allowincludesenderror。
asyncMaxSizeTaskPool	整型	5000	自动清除已取消的任务之前异步任务池中任务的最大大小。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.asyncmaxsizetaskpool。
asyncPurgeInterval	整型	30000	在每次对已取消的任务池进行必需清除之间要等待的时间间隔。完整应用程序服务器概要文件中的等

属性名称	数据类型	缺省值	描述
			价定制属性为 com.ibm.ws.webcontainer.asyncpurgeinterval。
asyncTimeoutDefault	整型	30000	尚未显式指定超时值时使用的异步 servlet 超时值。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.asynctimeoutdefault。
asyncTimerThreads	整型	2	要用于异步 servlet 超时处理的最大线程数。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.asynctimerthreads。
channelWriteType	字符串	async	设为“sync”时，将以同步方式写入响应；否则，将以异步方式写入响应。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.channelwritetype。
copyAttributesKeySet	布尔型	false	Web 容器将向 servlet 返回属性列表副本的枚举，以避免该 servlet 发生并发访问错误。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.copyattributeskeyset。
decodeUrlAsUtf8	布尔型	true	使用 UTF-8 的编码设置对 URL 进行解码。
decodeUrlPlusSign	布尔型	false	当 URL 中包含加号时对加号进行解码。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.decodeurlplussign。
defaultHeadRequestBehavior	布尔型	false	恢复当 HEAD 请求不受限于为 GET 方法定义的安全性约束时的行为。完整应用程序服务器概要文

属性名称	数据类型	缺省值	描述
			件中的等价定制属性为 com.ibm.ws.webcontainer.DefaultHeadRequestBehavior。
defaultTraceRequestBehavior	布尔型	false	恢复 HTTP TRACE 处理。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.DefaultTraceRequestBehavior。
deferServletLoad	布尔型	true	将 servlet 装入和初始化延迟至第一次请求。
deferServletRequestListenerDestroyOnError	布尔型	false	如果希望在处理请求出错时延迟 ServletRequestListener 破坏，那么进行切换。缺省值为 false。完整应用程序服务器概要文件中的等效定制属性为 com.ibm.ws.webcontainer.deferServletRequestListenerDestroyOnError。
directoryBrowsingEnabled	布尔型	false	启用应用程序的目录浏览。
disableXPoweredBy	布尔型	false	禁止设置 X-Powered-By 头。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.disablexpoweredby。
disallowAllFileServing	布尔型	false	禁用通过应用程序进行的所有文件服务。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.disallowAllFileServing。
disallowServeServletsByClassName	布尔型	true	禁止在应用程序服务器级别上使用 serveServletsByClassnameEnabled。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.disallowserveservletsbyclassname。

属性名称	数据类型	缺省值	描述
dispatcherRethrowsEr	布尔型	true	Web 容器将重新抛出错误，从而允许相关资源处理这些错误。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.dispatcherRethrowser。
doNotServeByClassName	字符串		要完全禁止按类名提供的以分号定界的类列表。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.donotservebyclassname。
emptyServletMappings	布尔型	false	当未添加 Servlet 映射时，如您想要返回空的集合，而不是返回 null，那么进行切换。缺省值为 false。完整应用程序服务器概要文件中等价的定制属性为 com.ibm.ws.webcontainer.emptyservletmappings。
enableErrorExceptionTypeFirst	布尔型	false	已将 Web 容器更新为在错误代码之前搜索和使用异常类型。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst。
enableJspMappingOverride	布尔型	false	允许覆盖 JSP 映射，以便应用程序可自行提供 JSP 内容。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.enablejspmappingoverride。
enableMultiReadOfPostData	布尔型	false	保留 POST 数据以用于多次读访问。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.enablemultireadofpostdata。

属性名称	数据类型	缺省值	描述
exposeWebInfOnDispatch	布尔型	false	如果为 true，那么 servlet 可以访问 WEB-INF 目录中的文件。如果为 false（缺省值），那么 servlet 无法访问 WEB-INF 目录中的文件。
fileServingEnabled	布尔型	true	如果没有为应用程序显式指定此设置，那么启用文件服务。
fileWrapperEvents	布尔型	false	提供静态文件时，Web 容器将生成 SMF 和 PMI 数据。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.fileWrapperEvents。
httpsIndicatorHeader	字符串		对于 SSL 卸载，设为由 SSL 加速器/代理/负载均衡器插入的 HTTP 头变量的名称。
ignoreSemiColonOnRedirectToWelcomePage	布尔型	false	切换到在重定向至欢迎页面时忽略尾随分号。缺省值为 false。完整应用程序服务器概要文件中的等效定制属性为 com.ibm.ws.webcontainer.ignoreSemiColonOnRedirectToWelcomePage。
ignoreSessiononStaticFileRequest	布尔型	false	阻止 Web 容器访问涉及过滤器的静态文件请求的会话，从而改进性能。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest。
invokeFilterInitAtStartup	布尔型	true	Web 容器将在应用程序启动时调用过滤器的 init() 方法。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.invokeFilterInitAtStartup。

属性名称	数据类型	缺省值	描述
listeners	字符串		以逗号分隔的侦听器类列表。
logServletContainerInitializerClassLoadingErrors	布尔型	false	将 servlet 容器类装入错误记录为警告，而非仅在启用调试时记录这些错误。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.logservletcontainerinitializerclassloadingerrors。
metaInfResourcesCacheSize	整型	20	meta-inf 资源高速缓存的初始大小（条目数）。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.metainfresourcescachesize.name。
parseUtf8PostData	布尔型	false	Web 容器将检测非 URL 编码的 UTF-8 POST 数据，并将其包括在参数值中。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.parseutf8postdata。
serveServletsByClassnameEnabled	布尔型	false	如果未显式指定，那么将支持使用类名在 Web 应用程序中访问 servlet。
setContentLengthOnClose	布尔型	true	切换以在应用程序显式结束响应时设置内容长度。缺省值为 true；但是，如果应用程序响应包含双字节字符，请将此值设置为 false。
skipMetaInfResourcesProcessing	布尔型	false	不在 meta-inf 目录中搜索应用程序资源。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.skipmetainfresourcesprocessing。
suppressHtmlRecursiveErrorOutput	布尔型	false	当存在应用程序的已配置错误页无法处理的递归错误时，禁止异常信息出现

属性名称	数据类型	缺省值	描述
			在 HTML 输出中。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput。
symbolicLinksCacheSize	整型	1000	符号链接高速缓存的初始大小。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.SymbolicLinksCacheSize。
tolerateSymbolicLinks	布尔型	false	使 Web 容器能够支持使用符号链接。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.TolerateSymbolicLinks。
useSemiColonAsDelimiterInURI	布尔型	false	切换到使用分号作为请求 URI 中的定界符。缺省值为 false。完整应用程序服务器概要文件中的等效定制属性为 com.ibm.ws.webcontainer.useSemiColonAsDelimiterInURI。
xPoweredBy	字符串		X-Powered-By 头设置的备用字符串。完整应用程序服务器概要文件中的等价定制属性为 com.ibm.ws.webcontainer.xpoweredby。此属性没有缺省值。如果未设置此属性，那么按 Servlet 规范的定义，X-Powered-By 头的值设置为 Servlet/<servlet spec version>。

xigemaMQ 消息传递 (wmqJmsClient)

xigemaMQ 消息传递

属性名称	数据类型	缺省值	描述
connectionConcurrency	整型	1	每个连接可提供的 MDB 的最大数目

属性名称	数据类型	缺省值	描述
	最小值：0		
id	字符串		唯一配置标识。
logWriterEnabled	布尔型	true	一个标志，表示允许或禁止将诊断跟踪发送至应用程序服务器提供的 LogWriter 对象
maxConnections	整型 最小值：0	50	与 xigemaMQ 队列管理器的最大连接数
nativeLibraryPath	字符串		xigemaMQ Java JNI 库（mqjbnd.dll 或等价项）的位置的绝对路径
reconnectionRetryCount	整型 最小值：0	5	连接失败时尝试重新连接至 xigemaMQ 队列管理器的最大次数
reconnectionRetryInterval	具有毫秒精度的时间段	5m	xigemaMQ 功能部件再次尝试重新连接至 xigemaMQ 队列管理器之前等待的时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。
startupRetryCount	整型 最小值：0	0	启动时重试创建连接的次数
startupRetryInterval	具有毫秒精度的时间段	30s	在启动时每次重新尝试连接之间要等待的时间（毫秒）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s500ms 相当于 1.5 秒。

WS-AtomicTransaction (wsAtomicTransaction)

Web Service 原子事务功能部件的配置属性。

属性名称	数据类型	缺省值	描述
clientAuth	布尔型	false	指定启用 SSL 时是否为 WS-AtomicTransaction 协议启用其他客户机认证。
externalURLPrefix	字符串		（可选）指定用于路由 WS-AtomicTransaction 协议的代理服务器的主机和端口地址。
sslEnabled	布尔型	false	指定是否为 WS-AtomicTransaction 协议启用 SSL。
sslRef	字符串	defaultSSLConfig	指定启用 SSL 时要用于 WS-AtomicTransaction 协议的 SSL 配置。

WS-Security 客户机 (wsSecurityClient)

客户机的 Web Service 安全性缺省配置。

- *encryptionProperties*
- *signatureProperties*

属性名称	数据类型	缺省值	描述
ws-security.callback-handler	字符串		密码回调处理程序实现类。
ws-security.encryption.username	字符串		用于访问加密密钥库的别名。
ws-security.password	可逆向编码的密码（字符串）		用于创建用户名令牌的用户密码信息。
ws-security.signature.username	字符串		用于访问签名密钥库的别名。
ws-security.username	字符串		用于创建用户名令牌的用户信息。

encryptionProperties

必需的加密配置。

false

属性名称	数据类型	缺省值	描述
org.apache.ws.security.crypto.merlin.cert.provider	字符串		用于装入证书的提供程序。缺省为密钥库提供程序。
org.apache.ws.security.crypto.merlin.file	字符串		密钥库的位置
org.apache.ws.security.crypto.merlin.keystore.alias	字符串		要使用的缺省密钥库别名（如果未指定任何密钥库别名）。
org.apache.ws.security.crypto.merlin.keystore.password	可逆向编码的密码（字符串）		用于访问密钥库文件的密码。
org.apache.ws.security.crypto.merlin.keystore.private.password	可逆向编码的密码（字符串）		用于装入专用密钥的缺省密码。
org.apache.ws.security.crypto.merlin.keystore.provider	字符串		用于装入密钥库的提供程序。缺省为所安装的提供程序。
org.apache.ws.security.crypto.merlin.keystore.type	字符串		JKS、JCEKS 或 PKCS11
org.apache.ws.security.crypto.merlin.truststore.file	字符串		信任库的位置
org.apache.ws.security.crypto.merlin.truststore.password	可逆向编码的密码（字符串）		信任库密码。
org.apache.ws.security.crypto.merlin.truststore.type	字符串		信任库类型。
org.apache.ws.security.crypto.merlin.x509crl.file	字符串		要使用的 (X509) CRL 文件的位置。
org.apache.ws.security.crypto.provider	字符串	org.apache.ws.security.components.crypto.Merlin	用于创建密码实例的提供程序。缺省为“org.apache.ws.security.components.crypto.Merlin”。

signatureProperties

必需的签名配置。

false

属性名称	数据类型	缺省值	描述
org.apache.ws.security.crypto.merlin.cert.provider	字符串		用于装入证书的提供程序。缺省为密钥库提供程序。
org.apache.ws.security.crypto.merlin.file	字符串		密钥库的位置
org.apache.ws.security.crypto.merlin.keystore.alias	字符串		要使用的缺省密钥库别名（如果未指定任何密钥库别名）。
org.apache.ws.security.crypto.merlin.keystore.password	可逆向编码的密码（字符串）		用于访问密钥库文件的密码。
org.apache.ws.security.crypto.merlin.keystore.private.password	可逆向编码的密码（字符串）		用于装入专用密钥的缺省密码。
org.apache.ws.security.crypto.merlin.keystore.provider	字符串		用于装入密钥库的提供程序。缺省为所安装的提供程序。
org.apache.ws.security.crypto.merlin.keystore.type	字符串		JKS、JCEKS 或 PKCS11
org.apache.ws.security.crypto.merlin.truststore.file	字符串		信任库的位置
org.apache.ws.security.crypto.merlin.truststore.password	可逆向编码的密码（字符串）		信任库密码。
org.apache.ws.security.crypto.merlin.truststore.type	字符串		信任库类型。
org.apache.ws.security.crypto.merlin.x509crl.file	字符串		要使用的 (X509) CRL 文件的位置。
org.apache.ws.security.crypto.provider	字符串	org.apache.ws.security.components.crypto.Merlin	用于创建密码实例的提供程序。缺省为“org.apache.ws.security.components.crypto.Merlin”。

WS-Security 提供程序 (wsSecurityProvider)

提供程序的 Web Service 安全性缺省配置。

- *callerToken*

- *encryptionProperties*
- *samlToken*
  - *audienceRestrictions*
- *signatureProperties*

属性名称	数据类型	缺省值	描述
ws-security.callback-handler	字符串		密码回调处理程序实现类。
ws-security.enable.nonce.cache	布尔型	true	是否高速缓存 UsernameToken 的 nonce。
ws-security.encryption.username	字符串		用于访问加密密钥库的别名。
ws-security.signature.username	字符串		用于访问签名密钥库的别名。
ws-security.username	字符串		用于创建用户名令牌的用户信息。

callerToken

调用者令牌。

false

属性名称	数据类型	缺省值	描述
allowCustomCacheKey	布尔型	true	允许生成定制高速缓存密钥以访问认证高速缓存和获取主体集。
groupIdentifier	字符串		指定 SAML 属性，该属性将用作已认证主体所属的组的名称。没有缺省值。
includeTokenInSubject	布尔型	true	指定是否在主体集中包含 SAML 断言。
mapToUserRegistry	• User • No • Group	No	指定如何将身份映射至注册表用户。选项为 No、User 和 Group。缺省值为 No，不会使用用户注册表来创建用户主体集。 **User** 将 SAML 身份映射至注册

属性名称	数据类型	缺省值	描述
			表中定义的用户 **No** 不会将 SAML 身份映射至注册表中的用户或组 **Group** 将 SAML 身份映射至用户注册表中定义的组
name	字符串		指定令牌名称。选项为 Usernametoken、X509token 和 Samltoken。
realmIdentifier	字符串		指定将用作域名的 SAML 属性。缺省值为 issuer。
realmName	字符串		mapToUserRegistry 设置为 No 或 Group 时，指定域名。
userIdentifier	字符串		指定将用作主体集中的用户主体名称的 SAML 属性。缺省值为 NameID 断言。
userUniqueIdentifier	字符串		指定 SAML 属性，此属性将在应用于主体集中的 WSCredential 时用作唯一用户名。缺省值与 userIdentifier 属性值相同。

encryptionProperties

必需的加密配置。

false

属性名称	数据类型	缺省值	描述
org.apache.ws.security.crypto.merlin.cert.provider	字符串		用于装入证书的提供程序。缺省为密钥库提供程序。
org.apache.ws.security.crypto.merlin.file	字符串		密钥库的位置
org.apache.ws.security.crypto.merlin.keystore.alias	字符串		要使用的缺省密钥库别名（如果未指定任何密钥库别名）。
org.apache.ws.security.crypto.merlin.keystore.password	可逆向编码的密码（字符串）		用于访问密钥库文件的密码。
org.apache.ws.security.crypto.merlin.keystore.private.password	可逆向编码的密码（字符串）		用于装入专用密钥的缺省密码。
org.apache.ws.security.crypto.merlin.keystore.provider	字符串		用于装入密钥库的提供程序。缺省为所安装的提供程序。
org.apache.ws.security.crypto.merlin.keystore.type	字符串		JKS、JCEKS 或 PKCS11
org.apache.ws.security.crypto.merlin.truststore.file	字符串		信任库的位置
org.apache.ws.security.crypto.merlin.truststore.password	可逆向编码的密码（字符串）		信任库密码。
org.apache.ws.security.crypto.merlin.truststore.type	字符串		信任库类型。
org.apache.ws.security.crypto.merlin.x509crl.file	字符串		要使用的 (X509) CRL 文件的位置。
org.apache.ws.security.crypto.provider	字符串	org.apache.ws.security.components.crypto.Merlin	用于创建密码实例的提供程序。缺省为“org.apache.ws.security.components.crypto.Merlin”。

samlToken

指定用于评估 SAML 断言的可信任性及有效性的属性。

false

属性名称	数据类型	缺省值	描述
clockSkew	具有毫秒精度的时间段	5m	此属性用来指定验证 SAML 令牌时允许的时钟偏差（分钟）。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
requiredSubjectConfirmationMethod	• bearer	bearer	指定 SAML 断言中是否需要主体集确认方法。缺省值为 true。 **bearer** bearer
timeToLive	具有毫秒精度的时间段	30m	指定 SAML 断言未定义 NoOnOrAfter 条件时的缺省生存时间。缺省值是 30 分钟。指定后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms)。例如，将 500 毫秒指定为 500ms。可以将多个值包括在单个条目中。例如，1s 500ms 相当于 1.5 秒。
wantAssertionsSigned	布尔型	true	指示签署此服务提供程序接收的 <saml:Assertion> 元素的要求。

samlToken > audienceRestrictions

指定 SAML 断言的允许受众。缺省值为允许所有受众。

false

字符串

signatureProperties

必需的签名配置。

false

属性名称	数据类型	缺省值	描述
org.apache.ws.security.crypto.merlin.cert.provider	字符串		用于装入证书的提供程序。缺省为密钥库提供程序。
org.apache.ws.security.crypto.merlin.file	字符串		密钥库的位置
org.apache.ws.security.crypto.merlin.keystore.alias	字符串		要使用的缺省密钥库别名（如果未指定任何密钥库别名）。
org.apache.ws.security.crypto.merlin.keystore.password	可逆向编码的密码（字符串）		用于访问密钥库文件的密码。
org.apache.ws.security.crypto.merlin.keystore.private.password	可逆向编码的密码（字符串）		用于装入专用密钥的缺省密码。
org.apache.ws.security.crypto.merlin.keystore.provider	字符串		用于装入密钥库的提供程序。缺省为所安装的提供程序。
org.apache.ws.security.crypto.merlin.keystore.type	字符串		JKS、JCEKS 或 PKCS11
org.apache.ws.security.crypto.merlin.truststore.file	字符串		信任库的位置
org.apache.ws.security.crypto.merlin.truststore.password	可逆向编码的密码（字符串）		信任库密码。
org.apache.ws.security.crypto.merlin.truststore.type	字符串		信任库类型。
org.apache.ws.security.crypto.merlin.x509crl.file	字符串		要使用的 (X509) CRL 文件的位置。
org.apache.ws.security.crypto.provider	字符串	org.apache.ws.security.components.crypto.Merlin	用于创建密码实例的提供程序。缺省为“org.apache.ws.security.components.crypto.Merlin”。

2.1.4 功能部件管理

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。

使用配置文件 server.xml 来声明您想要载入的功能部件。 功能部件集含括在 <featureManager> 元素中，而每项功能部件含括在 <feature> 子元素中。例如：

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

您可以在服务器配置文件中指定任何功能部件。有些功能部件包含其他功能部件。同一个功能部件可以包含在一个或多个其他功能部件中。在运行时，功能部件管理器会计算支持所请求功能部件集所需的内容的合并列表。

有关可用的主要功能部件的信息，请参阅 *xigemaAS* 功能部件（见第 906 页）。有关适用于每个功能部件的限制的信息，请参阅 运行时环境已知问题和限制（见第 1668 页）。

功能部件配置的动态更改

更改功能部件配置时，功能部件管理器会重新计算所需捆绑软件的列表，停止并卸载不再需要的那些捆绑软件，然后安装并启动任何新增项。因此，所有功能部件设计为与其他动态添加或移除的功能部件配合使用。

单例功能部件

因为为 Java™ EE 7 提供了第一组功能部件，所以同一功能部件存在两种版本：

- servlet-3.0
- servlet-3.1

与其他应用程序服务器不同，您可选择要在服务器配置中使用的此功能部件的版本。servlet-3.0 保留现有应用程序的行为，而 servlet-3.1 为新的或已修改的应用程序提供新功能。尽管可选择规范版本，但系统不需要也不提供其他配置属性来控制两个版本之间的各项差异。

servlet 功能部件是一个单例功能部件，这意味着您只能配置一个版本以在服务器中使用：servlet-3.0 或 servlet-3.1。如果应用程序需要该 servlet 功能部件的不同版本，那么必须在不同服务器中对它们进行部署。许多其他功能部件将该 servlet 功能部件包含为依赖项。在 xigemaAS 产品中，这些功能部件已更新以与任一版本配合使用。这可确保您可在使用 servlet-3.1 中配置功能部件的完整“堆栈”，但来自其他源的功能部件可能未更新以“容许”servlet-3.1。要启用功能部件以“容许”servlet-3.1，请按如下所示修改功能部件清单：

```
Subsystem-Content: com.ibm.websphere.appserver.servlet-3.0;
 ibm.tolerates:="3.1"; type="osgi.subsystem.feature"
```

如果服务器配置包含单例功能部件的多个版本（通过在 server.xml 文件中直接配置或通过功能部件依赖项）那么该配置处于错误状态或不会装入该功能部件的任一版本。此错误导致产生类似如下的消息：

```
[ERROR ] CWWKF0033E: 单例功能部件 servlet-3.1 和 servlet-3.0
无法同时装入。所配置功能部件 servlet-3.1 和 servlet-3.0 包含导致冲突的一个或多个功能部件。
```

要解决此问题，请确保所配置功能部件全部指定或容许该单例功能部件的同一版本。如果对两个功能部件版本有硬件要求，那么必须将某些应用程序移至另一服务器。有关容许单例功能部件的更多信息，请参阅容许单例功能部件。

被取代的功能部件

功能部件上的取代标签指示有新功能部件或功能部件组合可提供比使用所取代功能部件更好的优势。例如，将使用更细粒度的新功能部件，而不是取代的功能部件，以便通过排除可能不必要的内容来减小服务器资源占用空间。新功能部件可能无法完全替换所取代功能部件的功能，因此在决定是否要更改配置之前，您必须考虑好情况。取代的功能部件仍保持完全受支持，且适合在您的配置中使用；取代标签仅提供您可能可以改进配置的指示。

很少的情况下，包括其他功能部件的功能部件被不包括所有其他功能部件的功能部件的新版本取代；不包括在新版本中的功能部件被认为是独立的。如果您的应用程序需要使用独立的功能部件的功能，那么必须将此功能部件显式添加至配置。

例如，featureA 和 featureB 具有下列条件：

- featureA-1.0 包括 featureB-1.0
- featureA-2.0 不包括 featureB-1.0（或者任何更高版本的 featureB）

如果应用程序使用 featureB 功能，那么下列任一配置将起作用：

- 将 featureA-1.0 包括在 server.xml 文件中
- 将 featureA-2.0 和 featureB-1.0 包括在 server.xml 文件中

xigemaAS 功能部件

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。

下表列示了 xigemaAS 中支持的功能部件。

- **Java™ EE 7 Web Profile**
  - *beanValidation-1.1*
  - *cdi-1.2*
  - *ejbLite-3.2*
  - *el-3.0*
  - *jaxrs-2.0*
  - *jaxrsClient-2.0*
  - *jdbc-4.1*
  - *jndi-1.0*
  - *jpa-2.1*
  - *jsf-2.2*
  - *jsonp-1.0*
  - *jsp-2.3*
  - *managedBeans-1.0*
  - *servlet-3.1*
  - *webProfile-7.0*
  - *websocket-1.0*
  - *websocket-1.1*
- **Java EE 7 Full Platform**

- *appClientSupport-1.0*
- *appSecurityClient-1.0*
- *batch-1.0*
- *concurrent-1.0*
- *ejb-3.2*
- *ejbHome-3.2*
- *ejbPersistentTimer-3.2*
- *ejbRemote-3.2*
- *j2eeManagement-1.1*
- *jacc-1.5*
- *jaspic-1.1*
- *javaee-7.0*
- *javaeeClient-7.0*
- *javaMail-1.5*
- *jaxb-2.2*
- *jaxws-2.2*
- *jca-1.7*
- *jcaInboundSecurity-1.0*
- *jms-2.0*
- *jmsMdb-3.2*
- *mdb-3.2*
- *wasJmsClient-2.0*
- *wasJmsSecurity-1.0*
- *wasJmsServer-1.0*

- **Extended Programming Models**
  - *cloudant-1.0*
  - *couchdb-1.0*
  - *distributedMap-1.0*
  - *json-1.0*
  - *microProfile-1.0*
  - *mongodb-2.0*
  - *redisdb-1.0*
  - *rtcomm-1.0*
  - *rtcommGateway-1.0*
- **Enterprise OSGi**
  - *blueprint-1.0*
  - *osgiAppIntegration-1.0*
  - *osgi.jpa-1.0*
  - *wab-1.0*
- **Operations**
  - *appSecurity-1.0*
  - *appSecurity-2.0*
  - *batchManagement-1.0*
  - *bells-1.0*
  - *eventLogging-1.0*
  - *ldapRegistry-3.0*

- *localConnector-1.0*
- *monitor-1.0*
- *oauth-2.0*
- *openid-2.0*
- *openidConnectClient-1.0*
- *openidConnectServer-1.0*
- *osgiConsole-1.0*
- *passwordUtilities-1.0*
- *requestTiming-1.0*
- *samlWeb-2.0*
- *restConnector-1.0*
- *restConnector-2.0*
- *serverStatus-1.0*
- *sessionDatabase-1.0*
- *redisSession-1.0*
- *spnego-1.0*
- *ssl-1.0*
- *timedOperations-1.0*
- *webCache-1.0*
- *wmqJmsClient-2.0*
- *wsSecurity-1.1*
- *WS-AT Service*（见第 1023 页）

- **Systems Management**
  - *adminCenter-1.0*
- **Java™ EE 6 Web Profile**
  - *beanValidation-1.0*
  - *cdi-1.0*
  - *ejbLite-3.1*
  - *jdbc-4.0*
  - *jndi-1.0*
  - *jpa-2.0*
  - *jsf-2.0*
  - *jsp-2.2*
  - *servlet-3.0*
  - *webProfile-6.0*
- **Java™ EE 6 Technologies**
  - *jaxb-2.2*
  - *jaxrs-1.1*
  - *jaxws-2.2*
  - *jca-1.6*
  - *jcaInboundSecurity-1.0*
  - *jms-1.1*
  - *jmsMdb-3.1*
  - *mdb-3.1*
  - *wasJmsClient-1.1*
  - *wasJmsSecurity-1.0*

- *wasJmsServer-1.0*
- *wmqJmsClient-1.1*

功能部件描述

以下列表包含可以添加至服务器配置的功能部件的相关信息。在配置中包含功能部件可促使自动装入一个或多个其他功能部件。例如，如果包含 wab-1.0 功能部件，那么会自动装入 servlet-3.0 和 bluepr int-1.0 功能部件。每个功能部件都包含一段简短描述，并举例说明如何在 server.xml 文件内部的 <fea tureManager> 元素中声明该功能部件。例如：

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

Java EE 7 Web 概要文件

Bean 验证

<feature>beanValidation-1.1</feature>

beanvalidation-1.1 功能部件在应用程序的每一层提供对 JavaBeans™ 的验证。可以通过使用注解或 validation.xml 部署描述符，将验证应用到应用程序中 JavaBeans™ 的所有层。

请参阅 *Bean* 验证功能部件限制（见第 1671 页）。

有关 beanValidation-1.1 功能部件配置信息，请参阅 *Bean Validation 1.1*（见第 939 页）。

CDI

<feature>cdi-1.2</feature>

cdi-1.2 功能部件在 xigemaAS 上启用对上下文和依赖性注入 1.2 规范的支持。

请参阅 在 *xigemaAS* 上管理上下文和依赖性注入应用程序（见第 1222 页）。

有关 cdi-1.2 功能部件配置信息，请参阅 *Contexts and Dependency Injection 1.2*（见第 943 页）。

Enterprise JavaBeans (EJB) Lite

<feature>ejbLite-3.2</feature>

ejbLite-3.2 功能部件为依照 EJB 3.2 规范的 EJB Lite 子集编写的 EJB 应用程序提供支持。

请注意，EJB 3.2 Lite API 组未包含可嵌入 EJB 容器，并且产品未提供 EJB 3.2 可嵌入容器。

而且，以下功能部件与 ejbLite-3.2 功能部件不兼容：

- cdi-1.0
- jmsMdb-3.1
- mdb-3.1

请参阅在 *xigemaAS* 上开发 *EJB* 应用程序（见第 1478 页）。

有关 ejbLite-3.2 功能部件配置信息，请参阅*Enterprise JavaBeans Lite 3.2*（见第 949 页）。

表达式语言 3.0

<feature>el-3.0</feature>

此功能部件启用对表达式语言 (EL) 3.0 的支持。

请参阅配置 *xigemaAS* 以使用 *Expression Language 3.0*（见第 1219 页）。

有关 `el-3.0` 功能部件配置信息，请参阅*Expression Language 3.0*（见第 952 页）。

Java™ API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-2.0</feature>
```

`jaxrs-2.0` 功能部件在 xigemaAS 上为 Java™ API for RESTful Web Services 提供支持。

请参阅 配置 *JAX-RS 2.0* 客户机（见第 1513 页）和 将 *JAX-RS 2.0* 与 *EJB* 和 *CDI* 集成（见第 1522 页）。

有关 `jaxrs-2.0` 功能部件配置信息，请参阅 *Java RESTful Services 2.0*（见第 987 页）。

Java™ EE Client API for JAX-RS 2.0

```
<feature>jaxrsClient-2.0</feature>
```

`jaxrsClient-2.0` 功能部件提供对 Java Client API for JAX-RS 2.0 的支持

请参阅配置 *JAX-RS 2.0* 客户机（见第 1513 页）和将 *JAX-RS 2.0* 与 *EJB* 和 *CDI* 集成（见第 1522 页）。

有关 `jaxrsClient-2.0` 功能部件配置信息，请参阅*Java RESTful Services Client 2.0*（见第 988 页）。

Java™ 数据库连接 (JDBC)

```
<feature>jdbc-4.1</feature>
```

您可以采用某一使用 Java™ 数据库连接 (JDBC) 和数据源的现有应用程序，然后将该应用程序部署到服务器。`jdbc-4.1` 功能部件提供对访问数据库的应用程序的支持。

请参阅在 *xigemaAS* 中配置数据库连接（见第 1196 页）。

有关 `jdbc-4.1` 功能部件配置信息，请参阅*Java Database Connectivity 4.1*（见第 969 页）。

Java™ 命名和目录接口 (JNDI)

```
<feature>jndi-1.0</feature>
```

`jndi-1.0` 功能部件为 xigemaAS 的服务器配置中的单一 JNDI 条目定义提供支持。

请参阅在 *xigemaAS* 功能部件中使用 *JNDI* 缺省命名空间开发（见第 1268 页）。

有关 `jndi-1.0` 功能部件配置信息，请参阅*Java Naming and Directory Interface*（见第 976 页）。

Java 持久性 API 2.1

```
<feature>jpa-2.1</feature>
```

`jpa-2.1` 功能部件提供对一些应用程序（这些应用程序使用依照 JPA 2.1 规范编写的应用程序管理及容器管理的 JPA）的支持，EclipseLink 支持此功能部件。

请参阅*Java Persistence API (JPA)* 功能部件概述（见第 1069 页）。

有关 `jpa-2.1` 功能部件配置信息，请参阅*Java Persistence API 2.1*（见第 979 页）。

JavaServer Faces (JSF)

```
<feature>jsf-2.2</feature>
```

此功能部件启用对使用 Java Server Faces (JSF) 2.2 框架的 Web 应用程序的支持。此框架简化了用户界面的构造。

请参阅配置 *xigemaAS* 以使用 *JavaServer Faces 2.2*（见第 1220 页）。

有关 `jsf-2.2` 功能部件配置信息，请参阅*JavaServer Faces 2.2*（见第 999 页）。

JavaScript™ 对象表示法处理

```
<feature>jsonp-1.0</feature>
```

Java™ API for JSON Processing (JSON-P) 功能部件提供一种标准化方法以构造和处理要以 JavaScript™ 对象表示法 (JSON) 呈现的数据。

有关 `jsonp-1.0` 功能部件配置信息，请参阅*JavaScript Object Notation Processing*（见第 996 页）。

JavaServer Pages (JSP)

```
<feature>jsp-2.3</feature>
```

此功能部件启用对写至 JSP 2.3 规范的 Java Server Pages (JSP) 的支持。此框架简化了用户界面的构造。启用此功能部件还会启用 Expression Language (EL) V3.0 功能部件。`el-3.0`。

请参阅配置 *xigemaAS* 以使用 *JavaServer Pages 2.3*（见第 1221 页）。

有关 `jsp-2.3` 功能部件配置信息，请参阅*JavaServer Pages 2.3*（见第 1001 页）。

受管 Bean

```
<feature>managedBeans-1.0</feature>
```

`managedBeans-1.0` 功能部件提供对受管 Bean 1.0 规范 (JSR-316) 的支持。此功能部件允许使用 `javax.annotation.ManagedBean` 注释。

有关 `managedBeans-1.0` 功能部件配置信息，请参阅*Java EE Managed Bean 1.0*（见第 971 页）。

Servlet 3.1

```
<feature>servlet-3.1</feature>
```

`servlet-3.1` 功能部件启用对依照 Java™ Servlet 3.1 规范编写的 HTTP Servlet 的支持。

请参阅配置 *xigemaAS* 以使用 *Servlet 3.1*（见第 1211 页）和*Servlet 3.1 行为更改*（见第 1212 页）。

有关 `servlet-3.1` 功能部件配置信息，请参阅*Java Servlets 3.1*（见第 990 页）。

Web 概要文件 7.0

```
<feature>webProfile-7.0</feature>
```

此功能部件提供支持 Java™ EE 7 Web 概要文件所需 xigemaAS 功能部件的便利组合。

请参阅*xigemaAS* 中的 *Java EE 7*（见第 24 页）。

有关 `webProfile-7.0` 功能部件配置信息，请参阅*Java EE Web Profile 7.0*（见第 973 页）。

WebSocket

```
<feature>websocket-1.0</feature>
```

```
<feature>websocket-1.1</feature>
```

WebSocket 是一种标准协议，它允许 Web 浏览器或客户机应用程序使用一种全双工连接与 Web 服务器应用程序通信。

请参阅 *xigemaAS: WebSocket* 和在 *xigemaAS* 中开发 *WebSocket* 应用程序（见第 1493 页）。

有关 `websocket-1.0` 功能部件配置信息，请参阅*Java WebSocket 1.0*（见第 993 页）。

有关 `websocket-1.1` 功能部件配置信息，请参阅*Java WebSocket 1.1*（见第 994 页）。

Java EE 7 完整平台

应用程序客户机支持

```
<feature>appClientSupport-1.0</feature>
```

`appClientSupport-1.0` 功能部件允许服务器处理应用程序的客户机模块内的 Java EE 元数据，例如，读取部署描述符 XML 文件和/或注释并在必要时将它们提供给应用程序中的其他模块。它还允许远程应用程序客户机进程与服务器通信以执行 JNDI 查找。

`appClientSupport-1.0` 功能部件仅在 `server.xml` 文件中启用。

有关 `appClientSupport-1.0` 功能部件配置信息，请参阅*Application Client Support for Server*（见第 931 页）。

应用程序客户机容器安全性

```
<feature>appSecurityClient-1.0</feature>
```

要在客户机容器上启用安全性，请将 `appSecurityClient-1.0` 功能部件添加至 client.xml 文件。

`appSecurityClient-1.0` 功能部件在客户机上启用 SSL、CSIv2 和 JAAS。必须配置 SSL 以确保客户机与服务器之间的通信是安全的和加密的。

请参阅 *xigemaAS* 应用程序客户机容器上的安全性（见第 1057 页） 和 为 *xigemaAS* 应用程序客户机容器及其应用程序配置安全性（见第 1395 页）。

有关 `appSecurityClient-1.0` 功能部件配置信息，请参阅 *Application Security for Client 1.0*（见第 934 页）。

批处理

```
<feature>batch-1.0</feature>
```

`batch-1.0` 功能部件允许使用 JSR-352 编程模型。

请参阅 *Java* 批处理和受管批处理概述（见第 1559 页）。

有关 `batch-1.0` 功能部件配置信息，请参阅 *Batch API 1.0*（见第 936 页）。

受管执行程序和线程工厂

```
<feature>concurrent-1.0</feature>
```

`concurrent-1.0` 功能部件支持创建受管执行程序服务，这些服务可让应用程序使用应用程序服务器所管理的线程上下文提交要同时运行的任务。此功能部件还允许创建受管线程工厂，以创建与查找该受管线程工厂的组件的线程上下文一起运行的线程。

请参阅配置受管调度执行程序（见第 1174 页）。

有关 concurrent-1.0 功能部件配置信息，请参阅 *Concurrency Utilities for Java EE 1.0*（见第 941 页）。

Enterprise JavaBeans (EJB)

```
<feature>ejb-3.2</feature>
```

ejb-3.2 功能部件为依照 EJB 3.2 规范编写的 EJB 应用程序提供支持。

此功能部件包含以下功能部件：

- `<feature>ejbLite-3.2</feature>`

  此功能部件提供对依照 EJB 3.2 规范的 EJB Lite 子集编写的 EJB 应用程序的支持。有关 ejbLite-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Lite 3.2*（见第 949 页）。
- `<feature>ejbHome-3.2</feature>`

  此功能部件提供对 EJB 2.x API 的支持。有关 ejbHome-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Home Interfaces 3.2*（见第 947 页）。
- `<feature>ejbPersistentTimer-3.2</feature>`

  此功能部件提供对持久 EJB 计时器的支持。有关 ejbPersistentTimer-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Persistent Timers 3.2*（见第 950 页）。
- `<feature>ejbRemote-3.2</feature>`

  此功能部件提供对远程 EJB 接口的支持。有关 ejbRemote-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Remote 3.2*（见第 951 页）。
- `<feature>mdb-3.2</feature>`

  此功能部件提供对消息驱动的 Bean 的支持。有关 mdb-3.2 功能部件配置信息，请参阅 *Message-Driven Beans 3.2*（见第 1006 页）。

  mdb-3.2 功能部件取代了 jmsMdb-3.2 功能部件。有关 jmsMdb-3.2 功能部件的信息，请参阅部署消息驱动 *Bean* 以连接至 *xigemaMQ*（见第 1551 页）。

如果不需要完整 EJB 3.2 支持，那么可使用这些功能部件的各种组合来提供所需支持。

请参阅在 *xigemaAS* 上开发 *EJB* 应用程序（见第 1478 页）。

有关 ejb-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans 3.2*（见第 946 页）。

J2EE 管理 1.1

```
<feature>j2eeManagement-1.1</feature>
```

j2eeManagement-1.1 功能部件提供标准接口以用于管理 Java EE 7，并允许应用程序使用 JSR 77 规范中定义的接口。

要调用管理 EJB API，服务器配置必须在功能部件管理器中同时具有 j2eeManagement-1.1 和 ejbRemote-3.2 功能部件。如果这两个功能部件都已包含在服务器配置中，那么您可通过 JNDI 名称查找来调用管理 EJB API。管理 EJB 绑定名称（JNDI 查找名称）为 ejb/mejb/MEJB。

请参阅*j2eeManagement-1.1* 功能部件限制（见第 1672 页）。

有关 j2eeManagement-1.1 功能部件配置信息，请参阅*J2EE Management 1.1*（见第 953 页）。

Java 容器授权合同 1.5

```
<feature>jacc-1.5</feature>
```

`jacc-1.5` 功能部件启用对 Java 容器授权合同 (JACC) V1.5 的支持。为将 `jacc-1.5` 功能部件添加至服务器，您需要添加 xigemaAS 中未包含的第三方 JACC 提供程序。

请参阅开发 *Java Authorization Contract for Containers (JACC)* 授权提供程序（见第 1418 页）。

有关 `jacc-1.5` 功能部件配置信息，请参阅*Java Authorization Contract for Containers 1.5*（见第 964 页）。

Java™ 容器认证 SPI 1.1

```
<feature>jaspic-1.1</feature>
```

`jaspic-1.1` 功能部件支持使用 Java™ 容器认证 SPI (JASPIC) 提供者保护服务器运行时环境和应用程序（如 JSR-196 中所定义）。

请参阅配置 *Java Authentication SPI for Containers (JASPIC)* 用户功能部件（见第 1309 页）。

有关 `jaspic-1.1` 功能部件配置信息，请参阅*Java Authentication SPI for Containers 1.1*（见第 963 页）。

Java EE

```
<feature>javaee-7.0</feature>
```

此功能部件提供支持 Java EE 7.0 完整平台时所需的 xigemaAS 功能部件的便利组合。

请参阅*xigemaAS* 中的 *Java EE 7*（见第 24 页）。

有关 `javaee-7.0` 功能部件配置信息，请参阅*Java EE Full Platform 7.0*（见第 970 页）。

Java EE 应用程序客户机 7.0

```
<feature>javaeeClient-7.0</feature>
```

此功能部件提供对 Java EE 应用程序客户机 7.0 的支持。

请参阅手动创建 *xigemaAS* 应用程序客户机（见第 1110 页）。

有关 `javaeeClient-7.0` 功能部件配置信息，请参阅*Java EE V7 Application Client*（见第 971 页）。

JavaMail™ API

```
<feature>javaMail-1.5</feature>
```

JavaMail™ API 支持外部邮件服务器与 xigemaAS 应用程序之间的通信。请参阅在 *xigemaAS* 上管理 *JavaMail*（见第 1228 页）。

有关 `javaMail-1.5` 功能部件配置信息，请参阅*JavaMail 1.5*（见第 995 页）。

Java™ XML 绑定体系结构 (JAXB)

```
<feature>jaxb-2.2</feature>
```

`jaxb-2.2` 功能部件在 xigemaAS 上为“Java™ XML 绑定体系结构”(JAXB) 提供支持。

请参阅 *jaxb-2.2* 功能部件限制（见第 1672 页）。

有关 `jaxb-2.2` 功能部件配置信息，请参阅*Java XML Bindings 2.2*（见第 994 页）。

Java™ API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

jaxws-2.2 功能部件在 xigemaAS 上为 Java™ API for XML-Based Web Services 提供支持。

- 对于支持 JAX-WS 编程模型的 Web 应用程序，必须在 server.xml 文件中启用 servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于支持 JAX-WS 编程模型的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1、servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

请参阅将 *JAX-WS 应用程序部署到 xigemaAS*（见第 1531 页）和*jaxws-2.2 功能部件限制*（见第 1673 页）。

有关 jaxws-2.2 功能部件配置信息，请参阅*Java Web Services 2.2*（见第 992 页）。

Java EE 连接器体系结构 **1.7**

```
<feature>jca-1.7</feature>
```

jca-1.7 功能部件提供了配置元素来定义连接工厂、受管对象和激活规范的实例，并将这些实例与安装的资源适配器关联。

有关 jca-1.7 功能部件配置信息，请参阅*Java Connector Architecture 1.7*（见第 967 页）。

Java™ EE 连接器体系结构入站安全性

```
<feature>jcaInboundSecurity-1.0</feature>
```

jcaInboundSecurity-1.0 功能部件针对资源适配器启用安全性流程。

有关 jcaInboundSecurity-1.0 功能部件配置信息，请参阅*Java Connector Architecture 1.0 Security Inflow*（见第 965 页）。

Java™ 消息服务 **2.0**

```
<feature>jms-2.0</feature>
```

jms-2.0 功能部件支持将资源适配器配置为使用 Java 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.6 规范的任何 JMS 资源适配器。

请参阅对*xigemaAS 启用 JMS 消息传递*和*将消息传递应用程序部署到 xigemaAS*（见第 1547 页）。

有关 jms-2.0 功能部件配置信息，请参阅 *Java 消息传递服务 2.0*。

嵌入式 **xigemaAS** 消息传递功能部件

```
<feature>wasJmsClient-2.0</feature>
```

wasJmsClient-2.0 功能部件取代了 wasJmsClient-1.1 功能部件。wasJmsClient-2.0 功能部件符合 JMS 2.0 规范并且仅在 IBM JDK 7 或更高版本上受支持。

请参阅对*xigemaAS 启用 JMS 消息传递*和*将消息传递应用程序部署到 xigemaAS*（见第 1547 页）。

有关 wasJmsClient-2.0 功能部件配置信息，请参阅*JMS 2.0 Client for Message Server*（见第 957 页）。

```
<feature>wasJmsSecurity-1.0</feature>
```

wasJmsSecurity-1.0 功能部件支持与消息传递引擎建立安全连接。如果已启用 wasJmsSecurity-1.0 功能部件，那么它会开始对尝试连接至消息传递引擎的用户进行认证和授权。针对在 server.xml 文件

中定义的注册表来认证用户。如果用户想要访问主题或队列等目标，那么必须向用户授予必需的许可权。在 server.xml 文件中的 <messagingSecurity> 元素（这是 messagingEngine 元素的子元素）中定义了对于该目标的访问权。如果添加了 wasJmsSecurity-1.0 功能部件，但是 server.xml 文件中未定义 <messagingSecurity> 元素，那么用户将无法连接至消息传递引擎，也无法执行任何消息传递操作（例如，将消息发送至目标或者从目标接收消息）。

有关 wasJmsSecurity-1.0 功能部件配置信息，请参阅*Message Server Security 1.0*（见第 1004 页）。

👉 :

- 配置用户注册表是 wasJmsSecurity-1.0 功能部件的先决条件。请确保在启用 wasJmsSecurity-1.0 功能部件之前已配置用户注册表。
- 启用 wasJmsSecurity-1.0 功能部件时，您还必须在 server.xml 文件中配置 <messagingSecurity> 元素（这是 <messagingEngine> 元素的子元素）。此配置使得授权用户能够访问消息传递目标。

```
<feature>wasJmsServer-1.0</feature>
```

wasJmsServer-1.0 功能部件使 JMS 消息传递引擎运行时能够进行初始化。消息传递运行时负责提供应用程序连接，管理目标（例如，主题或队列）的状态，以及处理服务质量、安全性和事务。此功能部件还支持来自远程消息传递应用程序的入站连接。远程消息传递应用程序可以通过基于 SSL 或者非 SSL 的 TCP/IP 来连接至 JMS 消息传递引擎。

要使用 SSL 进行连接，您必须启用 SSL 功能部件。

请参阅 对 *xigemaAS* 启用安全 *JMS* 消息传递。

有关 wasJmsServer-1.0 功能部件配置信息，请参阅*Message Server 1.0*（见第 1003 页）。

扩展编程模型

Cloudant 集成

```
<feature>cloudant-1.0</feature>
```

此功能部件通过提供服务器配置中配置的连接器实例来启用与 Cloudant 的连接。连接器实例可以通过 JNDI 注入或访问。应用程序通过 Cloudant 客户机库来使用连接器实例。请参阅 在 *xigemaAS* 中通过 *Cloudant Java* 客户机库配置 *CouchDB* 连接。

对于 cloudant-1.0 功能部件配置信息，请参阅 *Cloudant Integration 1.0*（见第 940 页）。

CouchDB

```
<feature>couchdb-1.0</feature>
```

couchdb-1.0 功能部件支持 CouchDB 实例以及相关联的数据库连接。可以通过 JNDI 查找或资源注入来提供对 CouchDB 连接的访问权。

请参阅在 *xigemaAS* 中使用 *Ektorp* 客户机库配置 *CouchDB* 连接（见第 1209 页）。

有关 couchdb-1.0 功能部件配置信息，请参阅 *CouchDB Integration 1.0*（见第 944 页）。

高速缓存服务

```
<feature>distributedMap-1.0</feature>
```

此功能部件提供可使用 DistributedMap API 访问的本地高速缓存服务。缺省高速缓存绑定在 JNDI 内的 services/cache/distributedmap 中。可通过添加网络高速缓存提供者来分配高速缓存。

有关 distributedMap-1.0 功能部件配置信息，请参阅*Distributed Map interface for Dynamic Caching*（见第 945 页）。

JavaScript™ 对象表示法 (JSON4J) 库

```
<feature>json-1.0</feature>
```

json-1.0 功能部件提供对 JSON4J 库的访问权，该库为 Java™ 环境提供一组 JSON 处理类。JSON4J 库提供简单的 Java™ 模型来构造和操控要作为 JSON 数据来呈现的数据。

有关 json-1.0 功能部件配置信息，请参阅*JavaScript Object Notation for Java*（见第 997 页）。

Micro Profile

```
<feature>microProfile-1.0</feature>
```

microProfile-1.0 功能部件对支持 Micro Profile for enterprise Java 的 xigemaAS 功能部件进行组合。

对于 microProfile-1.0 功能部件配置信息，请参阅 *Micro Profile 1.0*（见第 1007 页）

MongoDB

```
<feature>mongodb-2.0</feature>
```

mongodb-2.0 功能部件支持 MongoDB 实例以及相关联的数据库连接。可以通过 JNDI 查找或资源注入来提供对 MongoDB 连接的访问权。本机 com.mongodb API 可执行数据库处理。

请参阅在 *xigemaAS* 中配置 *MongoDB* 连接（见第 1205 页）。

有关 mongodb-2.0 功能部件配置信息，请参阅 *MongoDB Integration 2.0*（见第 1007 页）。

RedisDB

```
<feature>redisdb-1.0</feature>
```

redisdb-1.0 功能部件允许应用程序通过 Jedis API 与 redis DB 实例交互，提供对 redis 单节点和集群的数据源支持。应用程序可以通过 JNDI 查找或资源注入两种方式来访问 redis DB。此功能部件支持多数据源的配置方式。

请参阅在 *xigemaAS* 中配置 *Redis DB* 连接（见第 1207 页）。

有关 redisdb-1.0 功能部件的配置信息，请参阅 *Redis DB Integration 1.0*（见第 1016 页）。

实时通信

```
<feature>rtcomm-1.0</feature>
```

xigemaAS 实时通信功能部件将启用可高度伸缩的呼叫信号引擎，该引擎可用于将 WebRTC 客户机连接至实时音频/视频/数据呼叫。此功能部件支持客户机注册及在两个端点间创建 WebRTC 对等连接时所需的信号交换。

有关 功能部件配置信息，请参阅*Real-Time Communications*（见第 1018 页）。

RTCommGateway

```
<feature>rtcommGateway-1.0</feature>
```

rtcommGateway-1.0 功能部件添加将会话启动协议 (SIP) 与 RTComm WebRTC 端点连接的功能以交换音频和视频流。

有关 rtcommGateway-1.0 功能部件配置信息，请参阅*WebRTC Rtcomm Gateway*（见第 1026 页）。

SIP Servlet

<feature>sipServlet-1.1</feature>

sipServlet-1.1 功能部件提供对 SIP Servlet 规范 1.1（又称为 JSR 289）的支持。会话启动协议 (SIP) 是用于许多交互式服务（包括音频、视频和对等通信）的控制协议。

请参阅*xigemaAS: 会话启动协议 (SIP)*（见第 1082 页）和*在 xigemaAS 上管理会话启动协议 (SIP)*（见第 1225 页）。

有关 sipServlet-1.1 功能部件配置信息，请参阅 *SIP Servlet* （见第 1020 页）。

企业 **OSGi**

Blueprint

<feature>blueprint-1.0</feature>

blueprint-1.0 功能部件为部署使用 OSGi Blueprint Container 规范的 OSGi 应用程序启用支持。借助 xigemaAS中的 OSGi 应用程序支持，您可以开发并部署使用 Java™ EE 和 OSGi 技术的模块化应用程序。

请参阅*查找 OSGi 应用程序*（见第 1267 页）。

有关 blueprint-1.0 功能部件配置信息，请参阅 *OSGi Blueprint*（见第 1009 页）。

OSGi 应用程序集成

<feature>osgiAppIntegration-1.0</feature>

使用 osgiAppIntegration-1.0 功能部件以允许同一 Java 虚拟机内可用的 OSGi 应用程序相互共享它们的服务。

有关 osgiAppIntegration-1.0 功能部件配置信息，请参阅 *OSGi Application Integration*（见第 1009 页）。

OSGi JPA

<feature>osgi.jpa-1.0</feature>

osgi.jpa-1.0 功能部件在 xigemaAS 上为 OSGi 应用程序提供 JPA 支持。

请参阅*将 OSGi 应用程序部署到 xigemaAS*（见第 1500 页）。

有关 osgi.jpa-1.0 功能部件配置信息，请参阅 *OSGi Java Persistence API*（见第 1012 页）。

Web 应用程序捆绑软件 **(WAB)**

<feature>wab-1.0</feature>

此功能部件为下列过程启用支持：应用程序服务器中某些操作的运行速度慢于预期时对警告进行记录。wab-1.0 功能部件对企业捆绑软件内的 WAB 提供支持。

wab-1.0 功能部件对企业捆绑软件内的 WAB 提供支持。

此功能部件支持打包在 WAB 中的下列资源：

- 静态 Web 内容和 JSP。

- 按照 Servlet 3.0 规范编写的 HTTP Servlet。
- Blueprint 应用程序。

如果包含 `wab-1.0` 功能部件，那么也包含 `servlet-3.0` 和 `blueprint-1.0` 功能部件。

请参阅将 *OSGi* 应用程序部署到 *xigemaAS*（见第 1500 页）。

有关 `wab-1.0` 功能部件配置信息，请参阅*OSGi Web Application Bundles*（见第 1013 页）。

操作

API 发现

```
<feature>apiDiscovery-1.0</feature>
```

`apiDiscovery-1.0` 功能部件允许您发现 REST API 文档。使用此功能部件查找 xigemaAS 服务器上的可用 REST API，然后使用 Swagger 用户接口调用所发现的 REST 端点。请参阅 在 *xigemaAS* 服务器上发现 *REST API* 文档（见第 1609 页）。

安全性

```
<feature>appSecurity-2.0</feature>
```

此版本的 appSecurity 功能部件显式地根据其他功能部件的存在情况，仅提供某些方面的安全性。此外，它不会自动地包含 `servlet-3.0` 或 `ldapRegistry-3.0` 功能部件，从而减少了服务器占用量。要保护 Web 应用程序，必须包含 `servlet-3.0` 功能部件。要启用 EJB 安全性，必须包含 `ejbLite-3.1` 功能部件。要支持 LDAP 用户注册表，必须包含 `ldapRegistry-3.0` 功能部件。

👉 注：

- `appSecurity-2.0` 功能部件将取代 `appSecurity-1.0`。除了 `appSecurity-2.0` 不会自动包括 `servlet-3.0` 或 `ldapRegistry-3.0` 之外，这两个功能部件在其他方面都相同。可以选择在服务器配置中改用 `appSecurity-2.0` 版本。请参阅被取代的功能部件（见第 906 页）。
  - 要启用 Web 安全性，必须在 server.xml 文件中指定 `servlet-3.0` 功能部件。
  - 要启用对于 LDAP 的支持，必须在 server.xml 文件中指定 `ldapRegistry-3.0` 功能部件。

`appSecurity-1.0` 和 `appSecurity-2.0` 功能部件支持保护服务器运行时环境和应用程序。支持下列方面：

- 基本用户注册表
- 轻量级目录访问协议 (LDAP) 用户注册表
- 基本授权
- Web 应用程序安全性
  - 基本认证登录
  - 表单登录 表单注销
  - 程序化 API：`getRemoteUser`、`getUserPrincipal`、`isUserInRole`、`authenticate`、`logout` 和 `login`。
- EJB 应用程序安全性
  - 可以在 ejb-jar.xml 文件中指定的所有安全性注解及所有安全性元素。

- 程序化 API：getCallerPrincipal、isCallerInRole 和 getCallerIdentity。单独会话 Bean 不支持 getCallerIdentity API。
- ibm-ejb-jar-ext.xml 文件中用于 run-as-mode CALLER_IDENTITY 和 SPECIFIED_IDENTITY（不支持 SYSTEM_IDENTITY）的 EJB 扩展设置。

当您将 appSecurity-1.0 或 appSecurity-2.0 功能部件添加到服务器时，还必须配置用户注册表（例如，基本用户注册表或者 LDAP 用户注册表）。

请参阅保护 *xigemaAS* 及应用程序（见第 1274 页） 和 *appSecurity-2.0* 功能部件限制（见第 1670 页）。

有关 appSecurity-1.0 功能部件配置信息，请参阅 *Application Security 1.0*（见第 932 页）。

有关 appSecurity-2.0 功能部件配置信息，请参阅 *Application Security 2.0*（见第 933 页）。

受管批处理

```
<feature>batchManagement-1.0</feature>
```

batchManagement-1.0 功能部件提供 REST 接口以用于远程作业提交和 batchManager 命令行客户机实用程序。

请参阅 *Java* 批处理和受管批处理概述（见第 1559 页）

有关 batchManagement-1.0 功能部件配置信息，请参阅 *Batch Management*（见第 937 页）。

Bluemix 实用程序

```
<feature>bluemixUtility-1.0</feature>
```

此功能部件可以更方便地配置对 IBM Bluemix 管理服务的访问。

有关 bluemixUtility-1.0 功能部件配置信息，请参阅 *bluemixUtility-1.0*。

事件日志记录

```
<feature>eventLogging-1.0</feature>
```

eventLogging-1.0 功能部件对包含多个事件（例如，JDBC 请求和 servlet 请求及其持续时间）的记录进行日志记录。

请参阅事件日志记录（见第 1654 页）。

有关 eventLogging-1.0 功能部件配置信息，请参阅事件日志记录。

ldapRegistry-3.0

```
<feature>ldapRegistry-3.0</feature>
```

ldapRegistry-3.0 功能部件支持 LDAP 用户注册表。ldapRegistry-3.0 V3.0 功能部件符合 LDAP V3 规范。ldapRegistry-3.0 功能部件未由 appSecurity-2.0 功能部件自动启用。通过使用此功能部件，您可以联合多个 LDAP 存储库。可以在 server.xml 文件中配置两个或两个以上的 LDAP 存储库，并且您可以从多个存储库中获取所有 LDAP 操作的合并结果。

有关 ldapRegistry-3.0 功能部件配置信息，请参阅 *LDAP User Registry*（见第 1002 页）。

本地 **JMX** 连接器

```
<feature>localConnector-1.0</feature>
```

localConnector-1.0 功能部件提供构建到 JVM 中的本地 JMX 连接器。该 JMX 连接器只能在同一主机上供以同一用户标识和同一 JDK 运行的用户使用。它允许 JMX 客户机（例如，jConsole 或其他使用 Attach API 的 JMX 客户机）进行本地访问。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）。

有关 localConnector-1.0 功能部件配置信息，请参阅*JMX Local Connector*（见第 960 页）。

监视

```
<feature>monitor-1.0</feature>
```

monitor-1.0 功能部件在 xigemaAS 上提供性能监控基础结构 (PMI) 支持。

请参阅监视 *xigemaAS* 服务器运行时环境（见第 1612 页）。

有关 monitor-1.0 功能部件配置信息，请参阅 *Performance Monitoring*（见第 1015 页）。

OAuth

```
<feature>oauth-2.0</feature>
```

oauth-2.0 功能部件支持使用 OAuth 2.0 协议来保护对资源的访问。

有关 oauth-2.0 功能部件配置信息，请参阅 *OAuth*（见第 1008 页）。

OpenID

```
<feature>openid-2.0</feature>
```

此功能部件允许用户对多个实体认证自身而不需要管理多个帐户或多组凭证。xigemaAS 支持 OpenID 2.0 并在 Web 单点登录中充当依赖方的角色。访问 Web 站点之类的各种实体通常需要与每个实体相关联的唯一帐户。OpenID 启用由 OpenID 提供者处理的一组凭证，以授予对支持 OpenID 的任意数目实体的访问权。请参阅*OpenID*（见第 1311 页）。

有关 openid-2.0 功能部件配置信息，请参阅*OpenID*（见第 1013 页）。

OpenID Connect 客户机

```
<feature>openidConnectClient-1.0</feature>
```

此功能部件允许 Web 应用程序集成 OpenID Connect Client 1.0 以认证用户而不是（或以及）所配置用户注册表。请参阅 *OpenID Connect*（见第 1312 页）。

有关 openidConnectClient-1.0 功能部件配置信息，请参阅 *OpenID Connect Client*（见第 1014 页）。

OpenID Connect 提供者

```
<feature>openidConnectServer-1.0</feature>
```

此功能部件允许 Web 应用程序集成 OpenID Connect Server 1.0 以认证用户而不是（或以及）所配置用户注册表。请参阅 *OpenID Connect*（见第 1312 页）。

有关 openidConnectServer-1.0 功能部件配置信息，请参阅 *OpenId Connect Provider*（见第 1015 页）。

OSGi 控制台

```
<feature>osgiConsole-1.0</feature>
```

此功能部件可让 OSGi 控制台帮助调试运行时环境。它可用来显示有关捆绑软件、软件包和服务的信息。为产品扩展开发您自己的功能部件时，此信息可能很有用。

请参阅使用 *OSGi* 控制台（见第 1129 页）。

有关 `osgiConsole-1.0` 功能部件配置信息，请参阅*OSGi Debug Console*（见第 1011 页）。

密码实用程序

```
<feature>passwordUtilities-1.0</feature>
```

此功能部件支持使用安全性插接点从应用程序获取 AuthData。

有关 `passwordUtilities-1.0` 功能部件配置信息，请参阅 *passwordUtilities-1.0*。

请求计时

```
<feature>requestTiming-1.0</feature>
```

`requestTiming-1.0` 提供有关运行缓慢或挂起的请求的警告和诊断信息。

请参阅检测运行缓慢和挂起的请求（见第 1656 页）。

有关 `requestTiming-1.0` 功能部件配置信息，请参阅请求计时。

REST 连接器 1.0

```
<feature>restConnector-1.0</feature>
```

`restConnector-1.0` 功能部件提供可以通过任何 JDK 以本地或远程方式使用的安全 JMX 连接器。它支持 JMX 客户机通过基于 REST 的连接器进行远程访问，而且需要 SSL 和基本用户安全性配置。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页），有关 REST 连接器的详细信息，请参阅配置与 *xigemaAS* 的安全 *JMX* 连接（见第 1179 页）。

有关 `restConnector-1.0` 功能部件配置信息，请参阅*JMX REST Connector*（见第 961 页）。

REST 连接器 2.0

```
<feature>restConnector-2.0</feature>
```

`restConnector-2.0` 功能部件提供了安全 JMX 连接器，可以通过任何 JDK 以本地或远程方式进行使用。它支持 JMX 客户机通过基于 REST 的连接器进行远程访问，而且需要 SSL 和基本用户安全性配置。此功能部件取代了 `restConnector-1.0` 功能部件，并且未包含 `jaxrs-1.1` 功能部件。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页），有关 REST 连接器的详细信息，请参阅配置与 *xigemaAS* 的安全 *JMX* 连接（见第 1179 页）。

对于 `restConnector-2.0` 功能部件配置信息，请参阅 *JMX REST Connector 2.0*（见第 962 页）。

SAML Web 浏览器 SSO

```
<feature>samlWeb-2.0</feature>
```

`samlWeb-2.0` 功能部件允许 Web 应用程序将用户认证委派给 SAML 身份提供者（而不是已配置的用户注册表）。

有关 `samlWeb-2.0` 功能部件配置信息，请参阅*SAML Web Single Sign-on V2.0*（见第 1019 页）。

服务器状态

```
<feature>serverStatus-1.0</feature>
```

serverStatus-1.0 功能部件使 xigemaAS 服务器能够将其状态自动发布到作业管理器（它将该服务器视为作业配置中的资源）。已知状态为**已启动**和**已停止**。

有关 serverStatus-1.0 功能部件配置信息，请参阅*Job Manager Integration*（见第 1002 页）。

会话持久性

```
<feature>sessionDatabase-1.0</feature>
```

sessionDatabase-1.0 功能部件在xigemaAS 上提供会话亲缘关系和故障转移支持。

请参阅为 *xigemaAS* 配置会话持久性（见第 1145 页）。

有关 sessionDatabase-1.0 功能部件配置信息，请参阅 *Database Session Persistence*（见第 945 页）。

基于 **Redis DB** 的会话持久性

```
<feature>redisSession-1.0</feature>
```

redisSession-1.0 功能部件主要用于 HTTP Session 集中式存储，以保证在服务器意外宕机后可以进行数据恢复。开启 redisSession-1.0 功能部件并正确配置 redis 数据源，部署在 redis 单机或集群环境中的所有 Web 应用程序都会将其会话数据存储到对应的 redis 数据库中，并且不同应用程序之间的会话数据互不影响。

请参阅为 *NoSQL* 数据库 *Redis* 配置会话持久性。

有关 redisSession-1.0 功能部件配置信息，请参阅*Redis Database Session Persistence*（见第 1017 页）。

SPNEGO

```
<feature>spnego-1.0</feature>
```

此功能部件允许用户登录 Microsoft™ 域控制器一次就可访问 xigemaAS 服务器上的受保护应用程序，而不用提示用户再次进行登录。

有关在 xigemaAS 服务器上配置 SPNEGO 的更多信息，请参阅在 *xigemaAS* 中配置 *SPNEGO* 认证。

有关 spnego-1.0 功能部件配置信息，请参阅*Simple and Protected GSSAPI Negotiation Mechanism*（见第 1022 页）。

安全套接字层 **(SSL)**

```
<feature>ssl-1.0</feature>
```

ssl-1.0 功能部件为安全套接字层 (SSL) 连接提供支持。要使用安全 HTTPS 侦听器，必须启用此功能部件。xigemaAS 提供哑元密钥库和哑元信任库。不启动安全 HTTPS 侦听器，除非启用 ssl-1.0 功能部件。如果功能部件不可用，那么会停止 HTTPS 侦听器。要指定 SSL 证书，请在 server.xml 文件中添加一个指针；请参阅保护与 *xigemaAS* 的通信（见第 1278 页）。要更改 HTTPS 端口，请在 server.xml 文件中设置 <httpEndpoint> 元素的 <httpsPort> 属性；请参阅设置应用服务器的引导属性（见第 1102 页）。

有关 ssl-1.0 功能部件配置信息，请参阅*Secure Socket Layer*（见第 1021 页）。

定时操作

```
<feature>timedOperations-1.0</feature>
```

此功能部件为下列过程启用支持：应用程序服务器中某些操作的运行速度慢于预期时对警告进行记录。

请参阅定时操作和 *JDBC* 调用（见第 1653 页）。

有关 timedOperations-1.0 功能部件配置信息，请参阅*Timed Operations*（见第 1022 页）。

动态高速缓存服务

```
<feature>webCache-1.0</feature>
```

此功能部件会对 Web 响应启用本地高速缓存。它包含高速缓存服务 (distributedMap) 功能部件，并对 Web 应用程序响应执行自动高速缓存以缩短响应时间并提高吞吐量。为定制响应高速缓存，可在应用程序中包括 cache-spec.xml 文件。可通过添加网络高速缓存提供者来分配高速缓存。

有关 webCache-1.0 功能部件配置信息，请参阅*Web Response Cache*（见第 1024 页）。

xigemaMQ 消息传递功能部件

```
<feature>wmqJmsClient-2.0</feature>
```

wmqJmsClient-2.0 功能部件允许应用程序通过 JMS 2.0 API 访问 xigemaMQ 上的消息队列。

请参阅 将 *JMS* 应用程序部署到 *xigemaAS* 以使用 *xigemaAS* 消息传递提供程序（见第 1548 页）。

有关 wmqJmsClient-2.0 功能部件配置信息，请参阅 *JMS 2.0 Client for xigemaMQ*（见第 956 页）。

Web Service 安全性

```
<feature>wsSecurity-1.1</feature>
```

wsSecurity-1.1 功能部件支持在消息级别保护 Web Service。要保护 Web Service 消息，必须启用此功能部件以及 appSecurity-2.0 和 jaxws-2.2 功能部件。会忽略 WSDL 文件中所定义的 Web Service 安全策略；除非启用了 wsSecurity-1.1 功能部件，否则不会强制实施此策略。

请参阅 *Web Service* 安全性（见第 1427 页）。

有关 wsSecurity-1.1 功能部件配置信息，请参阅 *Web Service Security*（见第 1025 页）。

Web Service 原子事务

```
<feature>wsAtomicTransaction-1.2</feature>
```

wsAtomicTransaction 是一个可互操作的事务协议。它使您可通过 Web Service 消息传送分布式事务，并可在异构事务基础结构之间以可互操作方式协调。

有关 xigemaAS 中的 wsAtomicTransaction-1.2 配置信息，请参阅 *xigemaAS* 中的 *Web Service* 原子事务（见第 1542 页）。

系统管理

管理中心

```
<feature>adminCenter-1.0</feature>
```

adminCenter-1.0 功能部件是一个基于 Web 的图形界面，用于在手机、平板电脑或计算机上从 Web 浏览器管理 xigemaAS 概要文件服务器和应用程序以及其他资源。

请参阅使用管理中心管理 *xigemaAS*和管理中心功能部件限制（见第 1671 页）。

有关 adminCenter-1.0 功能部件配置信息，请参阅 *Admin Center 1.0*（见第 930 页）。

Java™ EE 6 Web Profile

Bean 验证

```
<feature>beanValidation-1.0</feature>
```

beanvalidation-1.0 功能部件在应用程序的每一层提供对 JavaBeans™ 的验证。可以通过使用注解或 validation.xml 部署描述符，将验证应用到应用程序中 JavaBeans™ 的所有层。

请参阅 *Bean* 验证功能部件限制（见第 1671 页）。

有关 beanValidation-1.0 功能部件配置信息，请参阅 *Bean Validation 1.0*（见第 938 页）。

CDI

```
<feature>cdi-1.0</feature>
```

cdi-1.0 功能部件在 xigemaAS 上启用对上下文和依赖性注入 1.0 规范的支持。

请参阅在 *xigemaAS* 上管理上下文和依赖性注入应用程序（见第 1222 页）。

有关 cdi-1.0 功能部件配置信息，请参阅*Contexts and Dependency Injection 1.0*（见第 942 页）。

Enterprise JavaBeans™ (EJB) Lite 子集

```
<feature>ejbLite-3.1</feature>
```

ejbLite-3.1 功能部件为依照 EJB 3.1 规范的 EJB Lite 子集编写的 EJB 应用程序提供支持。

支持下列功能：

- 打包在 EAR 文件中的 EJB 模块。
- 打包在 WAR 文件中的 EJB。
- @Stateful 注解、@Stateless 注解、@Singleton 注解和 @EJB 注解。
- javax.annotation.security 注解。
- 将 JPA EntityManager、EntityManagerFactory 和 JDBC DataSource 注入所有类型的会话 Bean。
- ejb-jar.xml。
- EJB 拦截器。
- 非接口视图。
- Bean 管理的事务 (UserTransaction)。

请参阅 *ejbLite-3.1* 功能部件限制（见第 1672 页）。

有关 ejbLite-3.1 功能部件配置信息，请参阅 *Enterprise JavaBeans Lite 3.1*（见第 948 页）。

Java™ 数据库连接 (JDBC)

```
<feature>jdbc-4.0</feature>
```

您可以采用某一使用 Java™ 数据库连接 (JDBC) 和数据源的现有应用程序，然后将该应用程序部署到服务器。 jdbc-4.0 功能部件为访问数据库的应用程序提供支持。

请参阅将现有 *JDBC* 应用程序部署到 *xigemaAS*（见第 1501 页）。

有关 `jdbc-4.0` 功能部件配置信息，请参阅 *Java Database Connectivity 4.0*（见第 968 页）。

Java™ 命名和目录接口 (JNDI)

```
<feature>jndi-1.0</feature>
```

`jndi-1.0` 功能部件为 xigemaAS 的服务器配置中的单一 JNDI 条目定义提供支持。

有关 `jndi-1.0` 功能部件配置信息，请参阅 *Java Naming and Directory Interface*（见第 976 页）。

Java™ 持久性 API (JPA)

```
<feature>jpa-2.0</feature>
```

`jpa-2.0` 功能部件为应用程序（使用依照 JPA 2.0 规范编写的应用程序管理及容器管理的 JPA）提供支持。此支持基于 Apache OpenJPA，提供扩展来支持容器管理的编程模型。

扩展持久性上下文现在适用于有状态会话 Bean。

请参阅将 *JPA* 应用程序部署到 *xigemaAS*（见第 1509 页）。

有关 `jpa-2.0` 功能部件配置信息，请参阅 *Java Persistence API 2.0*（见第 977 页）。

JavaServer Faces (JSF)

```
<feature>jsf-2.0</feature>
```

`jsf-2.0` 功能部件为使用 JSF 框架的 Web 应用程序提供支持。此框架简化了用户界面的构造。如果包含 `jsf-2.0` 功能部件，那么也包含 `jsp-2.2` 功能部件，因为 JSF 框架是 JSP 框架的扩展。

有关 `jsf-2.0` 功能部件配置信息，请参阅 *JavaServer Faces 2.0*（见第 998 页）。

JavaServer Pages (JSP)

```
<feature>jsp-2.2</feature>
```

如果包含 `jsf-2.0` 功能部件，那么也包含 `jsp-2.2` 功能部件，因为 JSF 框架是 JSP 框架的扩展。如果包含 `jsp-2.2` 功能部件，那么也包含 `servlet-3.0` 功能部件。

请参阅 *jsp-2.2* 功能部件限制（见第 1673 页）。

有关 `jsp-2.2` 功能部件配置信息，请参阅 *JavaServer Pages 2.2*（见第 1000 页）。

Servlet 3.0

```
<feature>servlet-3.0</feature>
```

`servlet-3.0` 功能部件为依照 Java™ Servlet 3.0 规范编写的 HTTP Servlet 提供支持。

请参阅保护 *xigemaAS* 及应用程序（见第 1274 页）。

有关 `servlet-3.0` 功能部件配置信息，请参阅 *Java Servlets 3.0*（见第 988 页）。

Web 概要文件

```
<feature>webProfile-6.0</feature>
```

此功能部件提供支持 Java™ EE 6.0 Web 概要文件所需 xigemaAS 功能部件的便利组合。

有关 `webProfile-6.0` 功能部件配置信息，请参阅 *Java EE Web Profile 6.0*（见第 972 页）。

Java EE 6 技术

Java™ XML 绑定体系结构 (JAXB)

```
<feature>jaxb-2.2</feature>
```

jaxb-2.2 功能部件在 xigemaAS 上为“Java™ XML 绑定体系结构”(JAXB) 提供支持。

请参阅 和 *jaxb-2.2 功能部件限制*（见第 1672 页）。

有关 jaxb-2.2 功能部件配置信息，请参阅 *Java XML Bindings 2.2*（见第 994 页）。

Java™ API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-1.1</feature>
```

jaxrs-1.1 功能部件在 xigemaAS 上为 Java™ API for RESTful Web Services 提供支持。

- 对于使用 jaxrs-1.1 服务器功能部件的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1 功能部件。
- 对于使用 CDI 的 JAX-RS 应用程序，必须在 server.xml 文件中启用 cdi-1.0 功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

有关 jaxrs-1.1 功能部件配置信息，请参阅 *Java RESTful Services 1.1*（见第 984 页）。

Java™ API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

jaxws-2.2 功能部件在 xigemaAS 上为 Java™ API for XML-Based Web Services 提供支持。

- 对于支持 JAX-WS 编程模型的 Web 应用程序，必须在 server.xml 文件中启用 servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于支持 JAX-WS 编程模型的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1、servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

请参阅 *jaxws-2.2 功能部件限制*（见第 1673 页）。

有关 jaxws-2.2 功能部件配置信息，请参阅 *Java Web Services 2.2*（见第 992 页）。

Java™ EE 连接器体系结构

```
<feature>jca-1.6</feature>
```

jca-1.6 功能部件提供了配置元素来定义连接工厂、受管对象和激活规范的实例，并将这些实例与安装的资源适配器关联。

有关 jca-1.6 功能部件配置信息，请参阅 *Java Connector Architecture 1.6*（见第 966 页）。

Java™ EE 连接器体系结构入站安全性

```
<feature>jcaInboundSecurity-1.0</feature>
```

jcaInboundSecurity-1.0 功能部件针对资源适配器启用安全性流程。

有关 jcaInboundSecurity-1.0 功能部件配置信息，请参阅*Java Connector Architecture 1.0 Security Inflow*（见第 965 页）。

Java™ 消息服务 1.1

```
<feature>jms-1.1</feature>
```

jms-1.1 功能部件支持将资源适配器配置为使用 Java™ 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.6 规范的任何 JMS 资源适配器。

有关 jms-1.1 功能部件配置信息，请参阅 *Java Message Service 1.1*（见第 974 页）。

消息驱动的 Bean

```
<feature>jmsMdb-3.1</feature>
```

jmsMdb-3.1 功能部件支持部署和配置 JMS 资源，在 xigemaAS 中运行 消息驱动 Bean (MDB) 时需要这些 JMS 资源。此功能部件使 MDB 能够与嵌入式 xigemaAS 消息传递或 xigemaMQ 进行交互。

有关 jmsMdb-3.1 功能部件配置信息，请参阅 *JMS Message-Driven Beans 3.1*（见第 959 页）。

消息驱动的 Bean 3.1

```
<feature>mdb-3.1</feature>
```

mdb-3.1 功能部件支持使用消息驱动的 Enterprise JavaBeans™。MDB 允许异步处理 Java™ EE 组件中的消息。

有关 mdb-3.1 功能部件配置信息，请参阅 *Message-Driven Beans 3.1*（见第 1005 页）。

嵌入式 xigemaAS消息传递功能部件

```
<feature>wasJmsClient-1.1</feature>
```

wasJmsClient-1.1 功能部件支持 JMS 资源配置（例如，连接工厂、激活规范以及队列和主题资源），它还提供了客户机库，消息传递应用程序需要这些客户机库才能连接至 xigemaAS 上的 JMS 服务器。

请参阅对*xigemaAS 启用 JMS 消息传递*和*将消息传递应用程序部署到 xigemaAS*（见第 1547 页）。

有关 wasJmsClient-1.1 功能部件配置信息，请参阅 *JMS 1.1 Client for Message Server*（见第 955 页）。

```
<feature>wasJmsSecurity-1.0</feature>
```

wasJmsSecurity-1.0 功能部件支持与消息传递引擎建立安全连接。如果已启用 wasJmsSecurity-1.0 功能部件，那么它会开始对尝试连接至消息传递引擎的用户进行认证和授权。针对在 server.xml 文件中定义的注册表来认证用户。如果用户想要访问主题或队列等目标，那么必须向用户授予必需的许可权。在 server.xml 文件中的 <messagingSecurity> 元素（这是 messagingEngine 元素的子元素）中定义了对于该目标的访问权。如果添加了 wasJmsSecurity-1.0 功能部件，但是 server.xml 文件中未定义 <messagingSecurity> 元素，那么用户将无法连接至消息传递引擎，也无法执行任何消息传递操作（例如，将消息发送至目标或者从目标接收消息）。

有关 wasJmsSecurity-1.0 功能部件配置信息，请参阅 *Message Server Security 1.0*（见第 1004 页）。

👉 注:

- 配置用户注册表是 wasJmsSecurity-1.0 功能部件的先决条件。请确保在启用 wasJmsSecurity-1.0 功能部件之前已配置用户注册表。
- 启用 wasJmsSecurity-1.0 功能部件时，您还必须在 server.xml 文件中配置 <messagingSecurity> 元素（这是 <messagingEngine> 元素的子元素）。此配置使得授权用户能够访问消息传递目标。

```
<feature>wasJmsServer-1.0</feature>
```

wasJmsServer-1.0 功能部件使 JMS 消息传递引擎运行时能够进行初始化。消息传递运行时负责提供应用程序连接，管理目标（例如，主题或队列）的状态，以及处理服务质量、安全性和事务。此功能部件还支持来自远程消息传递应用程序的入站连接。远程消息传递应用程序可以通过基于 SSL 或者非 SSL 的 TCP/IP 来连接至 JMS 消息传递引擎。

要使用 SSL 进行连接，您必须启用 SSL 功能部件。

请参阅对 *xigemaAS* 启用安全 *JMS* 消息传递。

有关 wasJmsServer-1.0 功能部件配置信息，请参阅 *Message Server 1.0*（见第 1003 页）。

xigemaMQ 消息传递功能部件

```
<feature>wmqJmsClient-1.1</feature>
```

wmqJmsClient-1.1 功能部件使应用程序能够使用连接至 xigemaMQ 服务器的 JMS 消息传递。

请参阅将 *JMS* 应用程序部署到 *xigemaAS* 以使用 *xigemaAS* 消息传递提供程序（见第 1548 页）。

有关 wmqJmsClient-1.1 功能部件配置信息，请参阅 *JMS 1.1 Client for xigemaMQ*（见第 954 页）。

API Discovery 1.0

允许在 xigemaAS 中发现和展示 REST API。

启用此功能部件

要启用 API 发现 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>apiDiscovery-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 API 发现 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.apiDiscovery-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

此功能部件提供的第三方 API 包

- io.swagger.annotations

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 API 发现 1.0 功能部件：

- *administrator-role*
- *apiDiscovery*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *channelfw*
- *classloading*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *mimeTypes*
- *quickStartSecurity*
- *tcpOptions*
- *trustAssociation*
- *virtualHost*

Admin Center 1.0

adminCenter-1.0 功能部件是基于 Web 的图形界面，用于在智能手机、平板电脑或计算机上从 Web 浏览器管理 xigemaAS 服务器和应用程序以及其他资源。管理中心启用可扩展的管理控制台，该管理控制台提供用于帮助进行管理和监视活动的面向任务的工具。当启用 WebSocket 功能部件时，管理中心会利用 WebSocket 技术进行高性能推送通知而不是轮询。对于大型部署，建议启用 WebSocket 功能部件。

启用此功能部件

要启用管理中心功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>adminCenter-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于管理中心功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.adminCenter-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jaxrs-1.1 - Java RESTful Services 1.1*
- *jaxrs-2.0 - Java RESTful Services 2.0*
- *json-1.0 - JavaScript Object Notation for Java*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *restConnector-1.0 - JMX REST Connector*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置管理中心功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *channelfw*
- *classloading*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *mimeTypes*
- *quickStartSecurity*
- *tcpOptions*
- *trustAssociation*
- *virtualHost*

Application Client Support for Server

appClientSupport-1.0 功能部件允许 xigemaAS 服务器处理客户机模块并支持远程客户机容器。

启用此功能部件

要启用“应用程序客户机的服务器支持”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>appClientSupport-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于“应用程序客户机的服务器支持”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.appClientSupport-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“应用程序客户机的服务器支持”功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *channelfw*
- *classloading*
- *enterpriseApplication*
- *iiopEndpoint*
- *iiopServerPolicies*
- *javaPermission*
- *library*
- *orb*
- *tcpOptions*
- *webApplication*

Application Security 1.0

此功能部件被 appSecurity-2.0 取代。支持保护服务器运行时环境和应用程序。此功能部件启用 servlet-3.0 和 Web 应用程序安全性，支持 LDAP 用户注册表、基本用户注册表及 SSL。为支持安全 EJB 应用程序，必须添加 ejbLite-3.1 功能部件。将此功能部件添加至服务器时，您需要配置用户注册表，例如，基本用户注册表或 LDAP 用户注册表。

启用此功能部件

要启用应用程序安全性 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>appSecurity-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于应用程序安全性 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.appSecurity-1.0; type="osgi.subsystem.feature"
```

取代此功能部件的功能部件

- *ldapRegistry-3.0 - LDAP User Registry*
- *servlet-3.0 - Java Servlets 3.0*
- *appSecurity-2.0 - Application Security 2.0*

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *ldapRegistry-3.0 - LDAP User Registry*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *passwordUtilities-1.0 - Password Utilities*

Application Security 2.0

此功能部件启用对保护服务器运行时环境和应用程序的支持；它包括基本用户注册表。此功能部件取代 appSecurity-1.0 并且未包含 servlet-3.0 或对 LDAP 用户注册表的支持。要保护 Web 应用程序，请添加 servlet-3.0 功能部件。要保护 EJB 应用程序，请添加 ejbLite-3.1 功能部件。要使用 LDAP，请添加 ldapRegistry-3.0 功能部件。将 appSecurity-2.0 功能部件添加至服务器时，您需要配置用户注册表，例如，基本用户注册表或 LDAP 用户注册表。

启用此功能部件

要启用应用程序安全性 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>appSecurity-2.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于应用程序安全性 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.appSecurity-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *appSecurity-1.0 - Application Security 1.0*
- *constrainedDelegation-1.0 - Kerberos Constrained Delegation for SPNEGO*
- *jacc-1.5 - Java Authorization Contract for Containers 1.5*
- *jaspic-1.1 - Java Authentication SPI for Containers 1.1*
- *oauth-2.0 - OAuth*
- *openid-2.0 - OpenID*
- *passwordUtilities-1.0 - Password Utilities*
- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism*
- *webProfile-6.0 - Java EE Web Profile 6.0*
- *webProfile-7.0 - Java EE Web Profile 7.0*
- *wsSecurity-1.1 - Web Service 安全性*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置应用程序安全性 2.0 功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *basicRegistry*
- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *quickStartSecurity*

Application Security for Client 1.0

支持保护客户机容器运行时环境和应用程序。此功能部件在客户机上启用应用程序安全性、CSIv2 和 SSL。

启用此功能部件

要启用“用于客户机的应用程序安全性”1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>appSecurityClient-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于“用于客户机的应用程序安全性”1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.appSecurityClient-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ssl-1.0 - Secure Socket Layer*

此功能部件提供的 API 包

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“用于客户机的应用程序安全性”1.0 功能部件：

- *channelfw*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *orb*
- *tcpOptions*

Basic Extensions using xigemaAS Libraries

此功能部件允许配置使用 xigemaAS 库的基本扩展 (BELL)。

启用此功能部件

要启用使用 xigemaAS 库功能部件的基本扩展，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>bells-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您开发的功能部件依赖于使用 xigemaAS 库功能部件的基本扩展，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.bells-1.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置使用 xigemaAS 库功能部件的基本扩展：

- *bell*
- *classloading*
- *library*

Batch API 1.0

此功能部件能够支持由 JSR-352 指定的 Java 批处理 1.0 API。

启用此功能部件

要启用批处理 API 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>batch-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于批处理 API 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.batch-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *batchManagement-1.0 - Batch Management*
- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.batch.api
- javax.batch.api.chunk
- javax.batch.api.chunk.listener
- javax.batch.api.listener
- javax.batch.api.partition
- javax.batch.operations
- javax.batch.runtime
- javax.batch.runtime.context
- javax.inject

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置批处理 API 1.0 功能部件：

- *batchPersistence*
- *classloading*
- *databaseStore*
- *transaction*

Batch Management

此功能部件对 Java 批处理容器提供了受管批处理支持。这包括批处理 REST 管理界面、作业日志记录支持以及命令行实用程序，以进行外部调度程序集成。

启用此功能部件

要启用批处理管理功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>batchManagement-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于批处理管理功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.batchManagement-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *batch-1.0 - Batch API 1.0*
- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

此功能部件提供的标准 API 包

- javax.batch.api
- javax.batch.api.chunk
- javax.batch.api.chunk.listener
- javax.batch.api.listener
- javax.batch.api.partition
- javax.batch.operations
- javax.batch.runtime
- javax.batch.runtime.context

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置批处理管理功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *batchJobLogging*

- *channelfw*
- *classloading*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *mimeTypes*
- *quickStartSecurity*
- *tcpOptions*
- *transaction*
- *trustAssociation*
- *virtualHost*

Bean Validation 1.0

Bean Validation 1.0 规范为验证 JavaBeans 提供了基于注解的模型。它可用于断言和维护数据在遍历应用程序时的完整性。

启用此功能部件

要启用 Bean Validation 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>beanValidation-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Bean Validation 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.beanValidation-1.0;
 type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *jpa-2.0 - Java Persistence API 2.0*
- *jsf-2.0 - JavaServer Faces 2.0*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *webProfile-6.0 - Java EE Web Profile 6.0*

此功能部件提供的标准 API 包

- javax.validation
- javax.validation.bootstrap
- javax.validation.constraints
- javax.validation.groups

- javax.validation.metadata
- javax.validation.spi

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Bean Validation 1.0 功能部件：

- *classloading*
- *library*

Bean Validation 1.1

Bean Validation 1.1 规范为验证 JavaBeans 提供了基于注解的模型。它可用于断言和维护数据在遍历应用程序时的完整性。

启用此功能部件

要启用 Bean Validation 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>beanValidation-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Bean Validation 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.beanValidation-1.1;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *el-3.0 - Expression Language 3.0*

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *jpa-2.0 - Java Persistence API 2.0*
- *jsf-2.0 - JavaServer Faces 2.0*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.validation
- javax.validation.bootstrap
- javax.validation.constraints
- javax.validation.constraintvalidation
- javax.validation.executable
- javax.validation.groups
- javax.validation.metadata
- javax.validation.spi

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Bean Validation 1.1 功能部件：

- *classloading*
- *library*
- *transaction*

Bluemix Utilities 1.0

Bluemix 实用程序功能部件可用于快速轻松地配置对 IBM Bluemix 受管服务的访问。

启用此功能部件

要启用 Bluemix 实用程序 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>bluemixUtility-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Bluemix 实用程序 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.bluemixUtility-1.0;
 type="osgi.subsystem.feature"
```

Cloudant Integration 1.0

此功能部件通过提供服务器配置中配置的连接器实例来启用与 Cloudant 的连接，此实例可通过 JNDI 插入或访问。应用程序使用 Cloudant 客户机库来利用连接器实例。

启用此功能部件

要启用 Cloudant 集成 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>cloudant-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Cloudant 集成 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.cloudant-1.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Cloudant 集成 1.0 功能部件：

- *authData*
- *classloading*
- *cloudant*
- *library*

Concurrency Utilities for Java EE 1.0

concurrent-1.0 功能部件支持创建受管执行程序，该程序可让应用程序在应用程序服务器所管理的线程上下文中提交要并发运行的任务。此功能部件还允许创建受管线程工厂，以创建与查找该受管线程工厂的组件的线程上下文一起运行的线程。

启用此功能部件

要启用“用于 Java EE 1.0 的并行实用程序”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>concurrent-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于“用于 Java EE 1.0 的并行实用程序”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.concurrent-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.enterprise.concurrent

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“用于 Java EE 1.0 的并行实用程序”功能部件：

- *classloading*
- *contextService*
- *managedExecutorService*
- *managedScheduledExecutorService*
- *managedThreadFactory*

Contexts and Dependency Injection 1.0

上下文和依赖性注入规范简化了集成不同类型的 Java EE 组件的过程。它提供了一种常用机制以将 EJB 或受管 bean 之类的组件插入至 JSP 或其他 EJB 之类的组件。

启用此功能部件

要启用上下文和依赖性注入 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>cdi-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于上下文和依赖性注入 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.cdi-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *webProfile-6.0 - Java EE Web Profile 6.0*

此功能部件提供的标准 API 包

- javax.decorator
- javax.enterprise.context
- javax.enterprise.context.spi
- javax.enterprise.event
- javax.enterprise.inject
- javax.enterprise.inject.spi
- javax.enterprise.util
- javax.inject
- javax.interceptor

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置上下文和依赖性注入 1.0 功能部件：

- *cdiContainer*
- *classloading*

- *transaction*

Contexts and Dependency Injection 1.2

上下文和依赖性注入规范简化了集成不同类型的 Java EE 组件的过程。它提供了一种常用机制以将 EJB 或受管 bean 之类的组件插入至 JSP 或其他 EJB 之类的组件。

启用此功能部件

要启用上下文和依赖性注入 1.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>cdi-1.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于上下文和依赖性注入 1.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.cdi-1.2; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.decorator
- javax.enterprise.context
- javax.enterprise.context.spi
- javax.enterprise.event
- javax.enterprise.inject
- javax.enterprise.inject.spi
- javax.enterprise.util
- javax.inject
- javax.interceptor

此功能部件提供的第三方 API 包

- org.jboss.weld.context
- org.jboss.weld.context.api
- org.jboss.weld.context.beanstore
- org.jboss.weld.context.bound
- org.jboss.weld.context.conversation

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置上下文和依赖性注入 1.2 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *cdi12*
- *classloading*
- *javaPermission*
- *library*
- *transaction*

CouchDB Integration 1.0

此功能部件通过提供服务器配置中配置的连接器实例来启用与 CouchDB 的连接，此实例可通过 JNDI 插入或访问。应用程序使用 ektorp 客户机库以通过此连接器实例与 CouchDB 配合工作。

启用此功能部件

要启用 CouchDB 集成 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>couchdb-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 CouchDB 集成 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.couchdb-1.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 CouchDB 集成 1.0 功能部件：

- *classloading*
- *couchdb*
- *library*

Database Session Persistence

此功能部件允许使用 JDBC 将 HTTP 会话保存至数据源。将 HTTP 会话数据保存至数据库允许在服务器重新启动或发生意外服务器故障后恢复数据。可通过配置多个服务器实现 HTTP 会话的故障转移以将数据保存至同一位置

启用此功能部件

要启用数据库会话持久性功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>sessionDatabase-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于数据库会话持久性功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.sessionDatabase-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置数据库会话持久性功能部件：

- *classloading*
- *httpSession*
- *httpSessionDatabase*
- *transaction*

Distributed Map interface for Dynamic Caching

`distributedMap-1.0` 功能部件提供可通过 DistributedMap API 访问的本地缓存服务。缺省缓存绑定在 JNDI 内的 services/cache/distributedmap 中。可通过添加网络高速缓存提供程序来分发缓存。

启用此功能部件

要启用“动态高速缓存的分布式映射接口”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>distributedMap-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发“依赖于动态高速缓存的分布式映射接口”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.distributedMap-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *apiDiscovery-1.0 - API Discovery 1.0*
- *batchManagement-1.0 - Batch Management*
- *oauth-2.0 - OAuth*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *restConnector-1.0 - JMX REST Connector*
- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *scim-1.0 - System for Cross-domain Identity Management*
- *webCache-1.0 - Web Response Cache*
- *wsAtomicTransaction-1.2 - WS-AT Service*

此功能部件提供的 API 包

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“动态高速缓存的分布式映射接口”功能部件：

- *classloading*
- *distributedMap*
- *library*

Enterprise JavaBeans 3.2

此功能部件启用对依照 EJB 3.2 规范编写的 Enterprise JavaBeans 的支持。

启用此功能部件

要启用 Enterprise JavaBeans 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejb-3.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejb-3.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ejbHome-3.2 - Enterprise JavaBeans Home Interfaces 3.2*
- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*
- *ejbPersistentTimer-3.2 - Enterprise JavaBeans Persistent Timers 3.2*
- *ejbRemote-3.2 - Enterprise JavaBeans Remote 3.2*
- *mdb-3.2 - Message-Driven Beans 3.2*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans 3.2 功能部件：

- *classloading*
- *transaction*

Enterprise JavaBeans Home Interfaces 3.2

此功能部件允许在 Enterprise JavaBeans 中使用 home 接口。

启用此功能部件

要启用 Enterprise JavaBeans Home 接口 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejbHome-3.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans Home 接口 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejbHome-3.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*

启用此功能部件的功能部件

- *ejb-3.2 - Enterprise JavaBeans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans Home 接口 3.2 功能部件：

- *classloading*
- *transaction*

Enterprise JavaBeans Lite 3.1

此功能部件启用对依照 EJB 3.1 规范的 EJB Lite 子集编写的 Enterprise JavaBeans 的支持。

启用此功能部件

要启用 Enterprise JavaBeans Lite 3.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejbLite-3.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans Lite 3.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejbLite-3.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *webProfile-6.0 - Java EE Web Profile 6.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans Lite 3.1 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*

- *javaPermission*
- *library*
- *transaction*
- *webApplication*

Enterprise JavaBeans Lite 3.2

此功能部件启用对依照 EJB 3.2 规范的 EJB Lite 子集编写的 Enterprise JavaBeans 的支持。

启用此功能部件

要启用 Enterprise JavaBeans Lite 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejbLite-3.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans Lite 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejbLite-3.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *ejb-3.2 - Enterprise JavaBeans 3.2*
- *ejbHome-3.2 - Enterprise JavaBeans Home Interfaces 3.2*
- *ejbPersistentTimer-3.2 - Enterprise JavaBeans Persistent Timers 3.2*
- *ejbRemote-3.2 - Enterprise JavaBeans Remote 3.2*
- *webProfile-7.0 - Java EE Web Profile 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans Lite 3.2 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*
- *javaPermission*

- *library*
- *transaction*
- *webApplication*

Enterprise JavaBeans Persistent Timers 3.2

此功能部件允许在 Enterprise JavaBeans 中使用持久性计时器。

启用此功能部件

要启用 Enterprise JavaBeans 持久性计时器 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejbPersistentTimer-3.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans 持久性计时器 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejbPersistentTimer-3.2;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *ejb-3.2 - Enterprise JavaBeans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans 持久性计时器 3.2 功能部件：

- *classloading*
- *databaseStore*
- *persistentExecutor*
- *transaction*

Enterprise JavaBeans Remote 3.2

此功能部件允许在 Enterprise JavaBeans 中使用远程接口。

启用此功能部件

要启用 Enterprise JavaBeans Remote 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ejbRemote-3.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Enterprise JavaBeans Remote 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ejbRemote-3.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*

启用此功能部件的功能部件

- *ejb-3.2 - Enterprise JavaBeans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Enterprise JavaBeans Remote 3.2 功能部件：

- *channelfw*
- *classloading*
- *iiopEndpoint*
- *iiopServerPolicies*
- *orb*
- *tcpOptions*
- *transaction*

Event Logging

记录事件（例如，JDBC 请求和 servlet 请求）及它们的持续时间。

启用此功能部件

要启用事件日志记录功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>eventLogging-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于事件日志记录功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.eventLogging-1.0; type="osgi.subsystem.feature"
```

此功能部件提供的 SPI 包

- com.ibm.wsspi.event.logging

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置事件日志记录功能部件：

- *eventLogging*

Expression Language 3.0

此功能部件启用对 Expression Language (EL) 3.0 的支持。

启用此功能部件

要启用表达式语言 3.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>el-3.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于表达式语言 3.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.el-3.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *beanValidation-1.1 - Bean Validation 1.1*
- *jsp-2.3 - JavaServer Pages 2.3*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.el

Federated User Registry

此功能部件提供对多个用户注册表的联合的支持。

启用此功能部件

要启用联合用户注册表功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>federatedRegistry-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于联合用户注册表功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.federatedRegistry-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *ldapRegistry-3.0 - LDAP User Registry*
- *scim-1.0 - System for Cross-domain Identity Management*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置联合用户注册表功能部件：

- *federatedRepository*

J2EE Management 1.1

此功能部件允许应用程序使用 JSR77 规范中定义的接口。

启用此功能部件

要启用 J2EE 管理 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>j2eeManagement-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 J2EE 管理 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.j2eeManagement-1.1;
 type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.management.j2ee
- javax.management.j2ee.statistics
- org.omg.stub.javax.management.j2ee

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 J2EE 管理 1.1 功能部件：

- *channelfw*
- *classloading*
- *orb*
- *tcpOptions*
- *transaction*

JMS 1.1 Client for xigemaMQ

此功能部件允许应用程序通过 JMS 1.1 API 访问 xigemaMQ 上的消息队列。

启用此功能部件

要启用 xigemaMQ 的 JMS 1.1 客户机功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wmqJmsClient-1.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 xigemaMQ 的 JMS 1.1 客户机功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wmqJmsClient-1.1; type="osgi.subsystem.feature"
```

此功能部件提供的标准 API 包

- javax.jms

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 xigemaMQ 的 JMS 1.1 客户机功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *transaction*
- *wmqJmsClient*

JMS 1.1 Client for Message Server

wasJmsClient-1.1 功能部件向应用程序提供通过 JMS 1.1 API 访问 xigemaAS 上的消息队列的权限。它支持通过 wasJmsServer 功能部件访问所启用的消息传递引擎。

启用此功能部件

要启用“用于消息服务器的 JMS 1.1 客户机”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wasJmsClient-1.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于“用于消息服务器的 JMS 1.1 客户机”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wasJmsClient-1.1; type="osgi.subsystem.feature"
```

此功能部件提供的标准 API 包

- javax.jms

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“用于消息服务器的 JMS 1.1 客户机”功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *channelfw*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *tcpOptions*
- *transaction*
- *wasJmsOutbound*

JMS 2.0 Client for xigemaMQ

此功能部件允许应用程序通过 JMS 2.0 API 访问 xigemaMQ 上托管的消息队列。

启用此功能部件

要启用 xigemaMQ 的 JMS 2.0 客户机功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wmqJmsClient-2.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 xigemaMQ 的 JMS 2.0 客户机功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wmqJmsClient-2.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 xigemaMQ 的 JMS 2.0 客户机功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *transaction*
- *wmqJmsClient*

JMS 2.0 Client for Message Server

wasJmsClient-2.0 功能部件向应用程序提供通过 JMS 2.0 API 访问 xigemaAS上的消息队列的权限。此功能部件将取代 wasJmsClient-1.1。wasJmsClient-2.0 功能部件符合 JMS 2.0 规范，并且仅在 JDK 7 或更高版本上受支持。它支持访问通过 wasJmsServer 功能部件启用的消息传递引擎。

启用此功能部件

要启用“用于消息服务器的 JMS 2.0 客户机”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wasJmsClient-2.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于“用于消息服务器的 JMS 2.0 客户机”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wasJmsClient-2.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*
- *javaeeClient-7.0 - Java EE V7 Application Client*

此功能部件提供的标准 API 包

- javax.jms

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置“用于消息服务器的 JMS 2.0 客户机”功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *channelfw*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *tcpOptions*
- *transaction*
- *wasJmsOutbound*

JMS Message-Driven Beans 3.1

此功能部件允许使用依照 EJB 3.1 规范编写的 JMS 消息驱动的 Enterprise JavaBeans。MDB 允许异步处理 Java EE 组件中的消息。

启用此功能部件

要启用 JMS 消息驱动的 Bean 3.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jmsMdb-3.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 JMS 消息驱动的 Bean 3.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jmsMdb-3.1; type="osgi.subsystem.feature"
```

取代此功能部件的功能部件

- *mdb-3.1 - Message-Driven Beans 3.1*

此功能部件启用的功能部件

- *jca-1.6 - Java Connector Architecture 1.6*
- *jndi-1.0 - Java Naming and Directory Interface*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JMS 消息驱动的 Bean 3.1 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*
- *javaPermission*
- *library*
- *transaction*
- *webApplication*

JMS Message-Driven Beans 3.2

此功能部件被 mdb-3.2 取代。这两个功能部件提供了相同功能；仅功能部件名称不同。mdb-3.2 是首选功能部件名称。

启用此功能部件

要启用 JMS 消息驱动的 Bean 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jmsMdb-3.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 JMS 消息驱动的 Bean 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jmsMdb-3.2; type="osgi.subsystem.feature"
```

取代此功能部件的功能部件

- *mdb-3.2 - Message-Driven Beans 3.2*

此功能部件启用的功能部件

- *mdb-3.2 - Message-Driven Beans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JMS 消息驱动的 Bean 3.2 功能部件：

- *classloading*
- *transaction*

JMX Local Connector

此功能部件允许使用内置到 JVM 中的本地 JMX 连接器访问服务器中的 JMX 资源。此 JMX 连接器只能由同一主机上其用户标识和 JDK 与服务器进程相同的客户机使用。它允许 JMX 客户机（例如，jConsole）或其他使用连接 API 的 JMX 客户机进行本地访问。

启用此功能部件

要启用 JMX 本地连接器功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>localConnector-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 JMX 本地连接器功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.localConnector-1.0;
 type="osgi.subsystem.feature"
```

JMX REST Connector

可用于在本地或远程使用任何 JDK 的安全 JMX 连接器。它支持 JMX 客户机通过基于 REST 的连接器进行远程访问，而且需要 SSL 和基本用户安全性配置。

启用此功能部件

要启用 JMX REST Connector 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>restConnector-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JMX REST Connector 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.restConnector-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jaxrs-1.1 - Java RESTful Services 1.1*
- *jaxrs-2.0 - Java RESTful Services 2.0*
- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*

功能部件配置元素

可在 server.xml 文件中使用以下元素来配置 JMX REST Connector 1.0 功能部件：

- *administrator-role*
- *authCache*

- *authentication*
- *basicRegistry*
- *channelfw*
- *classloading*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *mimeTypes*
- *quickStartSecurity*
- *remoteFileAccess*
- *tcpOptions*
- *trustAssociation*
- *virtualHost*

JMX REST Connector 2.0

可通过任何 JDK 在本地或远程使用的安全 JMX 连接器。它允许 JMX 客户机通过基于 REST 的连接器进行远程访问，并需要 SSL 和基本用户安全配置。此功能部件将取代 restConnector-1.0 功能部件，并且不包括 jaxrs-1.1 功能部件。

启用此功能部件

要启用 JMX REST Connector 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>restConnector-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JMX REST Connector 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.restConnector-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

adminCenter-1.0 - Admin Center

- *adminCenter-1.0 - Admin Center*

功能部件配置元素

可在 server.xml 文件中使用以下元素来配置 JMX REST Connector 2.0 功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *channelfw*
- *classloading*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *mimeTypes*
- *quickStartSecurity*
- *remoteFileAccess*
- *tcpOptions*
- *trustAssociation*
- *virtualHost*

Java Authentication SPI for Containers 1.1

此功能部件启用对按 JSR-196 中的定义使用 Java Authentication SPI for Containers (JASPIC) 提供程序保护服务器运行时环境和应用程序的支持

启用此功能部件

要启用 Java 容器认证 SPI 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaspic-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 容器认证 SPI 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaspic-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.security.auth.message
- javax.security.auth.message.callback
- javax.security.auth.message.config
- javax.security.auth.message.module

此功能部件提供的 SPI 包

- com.ibm.wsspi.security.jaspi

Java Authorization Contract for Containers 1.5

此功能部件启用对 Java Authorization Contract for Containers (JACC) V1.5 的支持。为了将 jacc-1.5 功能部件添加至服务器，需要添加第三方 JACC 提供程序，该提供程序没有包含在 xigemaAS 概要文件中。

启用此功能部件

要启用 Java 容器授权合同 1.5 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jacc-1.5</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 容器授权合同 1.5 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jacc-1.5; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.security.jacc

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 容器授权合同 1.5 功能部件：

- *classloading*

Java Connector Architecture 1.0 Security Inflow

`jcaInboundSecurity-1.0` 功能部件为资源适配器启用安全性流程。

启用此功能部件

要启用 Java 连接器体系结构 1.0 安全性流入功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jcaInboundSecurity-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 连接器体系结构 1.0 安全性流入功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jcaInboundSecurity-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jca-1.6 - Java Connector Architecture 1.6*
- *jca-1.7 - Java Connector Architecture 1.7*
- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

此功能部件提供的标准 API 包

- javax.security.auth.message.callback

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 连接器体系结构 1.0 安全性流入功能部件：

- *administrator-role*

- *authCache*
- *authentication*
- *basicRegistry*
- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *quickStartSecurity*
- *transaction*

Java Connector Architecture 1.6

`jca-1.6` 功能部件启用资源适配器配置，以从应用程序访问企业信息系统（EIS）。资源适配器配置也包括连接工厂、受管对象以及激活规范的配置。任何符合 JCA 1.6 规范的资源适配器都可使用，并且提供高性能连接池。

启用此功能部件

要启用 Java 连接器体系结构 1.6 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jca-1.6</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 连接器体系结构 1.6 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jca-1.6; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *jcaInboundSecurity-1.0 - Java Connector Architecture 1.0 安全性流程*
- *jms-1.1 - Java Message Service 1.1*
- *jmsMdb-3.1 - JMS Message-Driven Beans 3.1*
- *mdb-3.1 - Message-Driven Beans 3.1*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 连接器体系结构 1.6 功能部件：

- *activationSpec*
- *adminObject*
- *application*

- *applicationManager*
- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *library*
- *resourceAdapter*
- *transaction*

Java Connector Architecture 1.7

此功能部件允许配置资源适配器以通过应用程序访问企业信息系统 (EIS)。配置资源适配器也包括配置连接工厂、受管对象和激活规范。可使用符合 JCA 1.7 规范或更低版本的任何资源适配器。还将提供高性能连接池。

启用此功能部件

要启用 Java 连接器体系结构 1.7 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jca-1.7</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 连接器体系结构 1.7 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jca-1.7; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*
- *jcaInboundSecurity-1.0 - Java Connector Architecture 1.0 安全性流程*
- *jms-2.0 - Java Message Service 2.0*
- *mdb-3.2 - Message-Driven Beans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 连接器体系结构 1.7 功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*

- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *library*
- *resourceAdapter*
- *transaction*

Java Database Connectivity 4.0

jdbc-4.0 功能部件支持数据源的配置以从应用程序访问数据库。 任何符合 JDBC 4.0 规范的数据库都可以使用；包含多个特定提供程序的自定义配置。并且提供高性能连接池。

启用此功能部件

要启用 Java 数据库连接 4.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jdbc-4.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 数据库连接 4.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jdbc-4.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *batch-1.0 - Batch API 1.0*
- *batchManagement-1.0 - Batch Management*
- *jpa-2.0 - Java Persistence API 2.0*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *sessionDatabase-1.0 - Database Session Persistence*
- *webProfile-6.0 - Java EE Web Profile 6.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 数据库连接 4.0 功能部件：

- *authData*

- *classloading*
- *connectionManager*
- *dataSource*
- *jdbcDriver*
- *library*
- *transaction*

Java Database Connectivity 4.1

此功能部件允许配置数据源以通过应用程序访问数据库。可使用符合 JDBC 4.1、4.0、3.0 或 2.x 规范的任何 JDBC 驱动程序；包括许多特定提供程序的定制配置。还将提供高性能连接池。

启用此功能部件

要启用 Java 数据库连接 4.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jdbc-4.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 数据库连接 4.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jdbc-4.1; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *batch-1.0 - Batch API 1.0*
- *batchManagement-1.0 - Batch Management*
- *ejbPersistentTimer-3.2 - Enterprise JavaBeans Persistent Timers 3.2*
- *javaeeClient-7.0 - Java EE V7 Application Client*
- *jpa-2.0 - Java Persistence API 2.0*
- *jpa-2.1 - Java Persistence API 2.1*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *sessionDatabase-1.0 - Database Session Persistence*
- *webProfile-6.0 - Java EE Web Profile 6.0*
- *webProfile-7.0 - Java EE Web Profile 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 数据库连接 4.1 功能部件：

- *authData*
- *classloading*
- *connectionManager*
- *dataSource*

- *jdbcDriver*
- *library*
- *transaction*

Java EE Full Platform 7.0

此功能部件组合了支持 Java EE 7.0 Full Platform 的 xigemaAS 功能部件。

启用此功能部件

要启用 Java EE 完整平台 7.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>javaee-7.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java EE 完整平台 7.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.javaee-7.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appClientSupport-1.0 - Application Client Support for Server*
- *batch-1.0 - Batch API 1.0*
- *concurrent-1.0 - Concurrency Utilities for Java EE 1.0*
- *ejb-3.2 - Enterprise JavaBeans 3.2*
- *j2eeManagement-1.1 - J2EE Management 1.1*
- *jacc-1.5 - Java Authorization Contract for Containers 1.5*
- *jaspic-1.1 - Java Authentication SPI for Containers 1.1*
- *javaMail-1.5 - JavaMail 1.5*
- *jaxws-2.2 - Java Web Services 2.2*
- *jca-1.7 - Java Connector Architecture 1.7*
- *jcaInboundSecurity-1.0 - Java Connector Architecture 1.0 安全性流程*
- *servlet-3.1 - Java Servlets 3.1*
- *wasJmsClient-2.0 - JMS 2.0 Client for Message Server*
- *wasJmsSecurity-1.0 - Message Server Security 1.0*
- *wasJmsServer-1.0 - Message Server 1.0*
- *webProfile-7.0 - Java EE Web Profile 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java EE 完整平台 7.0 功能部件：

- *classloading*
- *transaction*

Java EE Managed Bean 1.0

此功能部件启用对 Managed Beans 1.0 规范的支持。受管 bean 为容器管理的不同 Java EE 组件类型提供公共基础。为受管 bean 提供的公共服务包括资源注入、生命周期管理和拦截器的使用。

启用此功能部件

要启用 Java EE 受管 Bean 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>managedBeans-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java EE 受管 Bean 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.managedBeans-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *webProfile-6.0 - Java EE Web Profile 6.0*
- *webProfile-7.0 - Java EE Web Profile 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java EE 受管 Bean 1.0 功能部件：

- *classloading*
- *ejbContainer*
- *library*
- *transaction*

Java EE V7 Application Client

此功能部件启用对 Java EE 7 应用程序客户机的支持。

启用此功能部件

要启用 Java EE V7 应用程序客户机功能部件，请在 client.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>javaeeClient-7.0</feature>
```

此功能部件启用的功能部件

- *beanValidation-1.1 - Bean Validation 1.1*
- *cdi-1.2 - Contexts and Dependency Injection 1.2*
- *javaMail-1.5 - JavaMail 1.5*
- *jaxb-2.2 - Java XML Bindings 2.2*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *jpa-2.1 - Java Persistence API 2.1*
- *jsonp-1.0 - JavaScript Object Notation Processing*
- *managedBeans-1.0 - Java EE Managed Bean 1.0*
- *wasJmsClient-2.0 - JMS 2.0 Client for Message Server*

功能部件配置元素

可在 client.xml 文件中使用以下元素以配置 Java EE V7 应用程序客户机功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *channelfw*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*
- *javaPermission*
- *library*
- *orb*
- *tcpOptions*
- *transaction*
- *webApplication*

Java EE Web Profile 6.0

webProfile-6.0 功能部件提供一个支持 Java™

启用此功能部件

要启用 Java EE Web 概要文件 6.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>webProfile-6.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java EE Web 概要文件 6.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.webProfile-6.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *beanValidation-1.0 - Bean Validation 1.0*
- *cdi-1.0 - Contexts and Dependency Injection 1.0*
- *ejbLite-3.1 - Enterprise JavaBeans Lite 3.1*
- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *jpa-2.0 - Java Persistence API 2.0*
- *jsf-2.0 - JavaServer Faces 2.0*
- *jsp-2.2 - JavaServer Pages 2.2*
- *managedBeans-1.0 - Java EE Managed Bean 1.0*
- *servlet-3.0 - Java Servlets 3.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java EE Web 概要文件 6.0 功能部件：

- *classloading*
- *transaction*

Java EE Web Profile 7.0

此功能部件组合了支持 Java EE 7.0 Web 概要文件的 xigemaAS 功能部件。

启用此功能部件

要启用 Java EE Web 概要文件 7.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>webProfile-7.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java EE Web 概要文件 7.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.webProfile-7.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *beanValidation-1.1 - Bean Validation 1.1*
- *cdi-1.2 - Contexts and Dependency Injection 1.2*
- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*
- *el-3.0 - Expression Language 3.0*
- *jaxrs-2.0 - Java RESTful Services 2.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *jpa-2.1 - Java Persistence API 2.1*

- *jsf-2.2 - JavaServer Faces 2.2*
- *jsonp-1.0 - JavaScript Object Notation Processing*
- *jsp-2.3 - JavaServer Pages 2.3*
- *managedBeans-1.0 - Java EE Managed Bean 1.0*
- *servlet-3.1 - Java Servlets 3.1*
- *websocket-1.1 - Java WebSocket 1.1*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java EE Web 概要文件 7.0 功能部件：

- *classloading*
- *transaction*

Java Message Service 1.1

此功能部件允许配置资源适配器以使用 Java 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.6 规范的任何 JMS 资源适配器。

启用此功能部件

要启用 Java 消息服务 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jms-1.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 消息服务 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jms-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jca-1.6 - Java Connector Architecture 1.6*

此功能部件提供的标准 API 包

- javax.jms

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 消息服务 1.1 功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *transaction*

Java Message Service 2.0

此功能部件允许配置资源适配器以使用 Java 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.7 规范的任何 JMS 资源适配器。

启用此功能部件

要启用 Java 消息服务 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jms-2.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 消息服务 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jms-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jca-1.7 - Java Connector Architecture 1.7*

此功能部件提供的标准 API 包

- javax.jms

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 消息服务 2.0 功能部件：

- *activationSpec*
- *adminObject*
- *application*
- *applicationManager*
- *applicationMonitor*
- *authData*
- *classloading*
- *connectionFactory*
- *connectionManager*
- *javaPermission*
- *jmsActivationSpec*
- *jmsConnectionFactory*
- *jmsDestination*
- *jmsQueue*
- *jmsQueueConnectionFactory*
- *jmsTopic*
- *jmsTopicConnectionFactory*
- *library*
- *resourceAdapter*
- *transaction*

Java Naming and Directory Interface

jndi-1.0 功能部件允许使用 Java 命名和目录接口 (JNDI) 访问服务器配置资源，例如，数据源或 JMS 连接工厂。它还允许访问服务器中配置的 Java 基本类型，例如，jndiEntry。

启用此功能部件

要启用 Java 命名和目录接口功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jndi-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 命名和目录接口功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jndi-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *appClientSupport-1.0 - Application Client Support for Server*
- *batch-1.0 - Batch API 1.0*
- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *ejbLite-3.1 - Enterprise JavaBeans Lite 3.1*
- *ejbLite-3.2 - Enterprise JavaBeans Lite 3.2*
- *ejbPersistentTimer-3.2 - Enterprise JavaBeans Persistent Timers 3.2*
- *javaeeClient-7.0 - Java EE V7 Application Client*
- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jmsMdb-3.1 - JMS Message-Driven Beans 3.1*
- *jpa-2.0 - Java Persistence API 2.0*
- *jpa-2.1 - Java Persistence API 2.1*
- *managedBeans-1.0 - Java EE Managed Bean 1.0*
- *mdb-3.1 - Message-Driven Beans 3.1*
- *mdb-3.2 - Message-Driven Beans 3.2*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *sessionDatabase-1.0 - Database Session Persistence*
- *webProfile-6.0 - Java EE Web Profile 6.0*
- *webProfile-7.0 - Java EE Web Profile 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 命名和目录接口功能部件：

- *classloading*
- *jndiEntry*
- *jndiObjectFactory*
- *jndiReferenceEntry*
- *jndiURLEntry*
- *library*

Java Persistence API 2.0

此功能部件为应用程序（使用依照 Java Persistence API 2.0 规范编写的应用程序管理的和容器管理的 JPA）提供支持。此支持基于 Apache OpenJPA，提供扩展来支持容器管理的编程模型。

启用此功能部件

要启用 Java 持久性 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jpa-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 持久性 API 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jpa-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *beanValidation-1.0 - Bean Validation 1.0*
- *beanValidation-1.1 - Bean Validation 1.1*
- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *webProfile-6.0 - Java EE Web Profile 6.0*

此功能部件提供的标准 API 包

- javax.persistence
- javax.persistence.criteria
- javax.persistence.metamodel
- javax.persistence.spi

此功能部件提供的第三方 API 包

- org.apache.openjpa.abstractstore
- org.apache.openjpa.ant
- org.apache.openjpa.audit
- org.apache.openjpa.conf
- org.apache.openjpa.datacache
- org.apache.openjpa.ee
- org.apache.openjpa.enhance
- org.apache.openjpa.event
- org.apache.openjpa.instrumentation
- org.apache.openjpa.instrumentation.jmx
- org.apache.openjpa.jdbc.ant
- org.apache.openjpa.jdbc.conf
- org.apache.openjpa.jdbc.identifier
- org.apache.openjpa.jdbc.kernel
- org.apache.openjpa.jdbc.kernel.exps
- org.apache.openjpa.jdbc.meta
- org.apache.openjpa.jdbc.meta.strats
- org.apache.openjpa.jdbc.schema
- org.apache.openjpa.jdbc.sql
- org.apache.openjpa.kernel
- org.apache.openjpa.kernel.exps
- org.apache.openjpa.kernel.jpql
- org.apache.openjpa.lib.ant
- org.apache.openjpa.lib.conf
- org.apache.openjpa.lib.encryption
- org.apache.openjpa.lib.graph

- org.apache.openjpa.lib.identifier
- org.apache.openjpa.lib.instrumentation
- org.apache.openjpa.lib.jdbc
- org.apache.openjpa.lib.log
- org.apache.openjpa.lib.meta
- org.apache.openjpa.lib.rop
- org.apache.openjpa.lib.util
- org.apache.openjpa.lib.util.concurrent
- org.apache.openjpa.lib.util.svn
- org.apache.openjpa.lib.xml
- org.apache.openjpa.meta
- org.apache.openjpa.persistence
- org.apache.openjpa.persistence.criteria
- org.apache.openjpa.persistence.jdbc
- org.apache.openjpa.persistence.meta
- org.apache.openjpa.persistence.osgi
- org.apache.openjpa.persistence.query
- org.apache.openjpa.persistence.util
- org.apache.openjpa.persistence.validation
- org.apache.openjpa.slice
- org.apache.openjpa.slice.jdbc
- org.apache.openjpa.util
- org.apache.openjpa.validation
- org.apache.openjpa.xmlstore

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 持久性 API 2.0 功能部件：

- *classloading*
- *jpa*
- *library*
- *transaction*

Java Persistence API 2.1

此功能部件为应用程序（使用依照 Java Persistence API 2.1 规范编写的应用程序管理的和容器管理的 JPA）提供支持。此支持基于 EclipseLink 以支持容器管理的编程模型。

启用此功能部件

要启用 Java 持久性 API 2.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jpa-2.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 Java 持久性 API 2.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jpa-2.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.persistence
- javax.persistence.criteria
- javax.persistence.metamodel
- javax.persistence.spi

此功能部件提供的第三方 API 包

- org.eclipse.persistence
- org.eclipse.persistence.annotations
- org.eclipse.persistence.config
- org.eclipse.persistence.core.descriptors
- org.eclipse.persistence.core.mappings
- org.eclipse.persistence.core.mappings.converters
- org.eclipse.persistence.core.queries
- org.eclipse.persistence.core.sessions
- org.eclipse.persistence.descriptors
- org.eclipse.persistence.descriptors.copying
- org.eclipse.persistence.descriptors.invalidation
- org.eclipse.persistence.descriptors.partitioning
- org.eclipse.persistence.dynamic
- org.eclipse.persistence.eis
- org.eclipse.persistence.eis.interactions
- org.eclipse.persistence.eis.mappings
- org.eclipse.persistence.exceptions
- org.eclipse.persistence.exceptions.i18n
- org.eclipse.persistence.expressions
- org.eclipse.persistence.expressions.spatial
- org.eclipse.persistence.history
- org.eclipse.persistence.internal.codegen
- org.eclipse.persistence.internal.core.databaseaccess
- org.eclipse.persistence.internal.core.descriptors

- org.eclipse.persistence.internal.core.helper
- org.eclipse.persistence.internal.core.queries
- org.eclipse.persistence.internal.core.sessions
- org.eclipse.persistence.internal.databaseaccess
- org.eclipse.persistence.internal.descriptors.changetracking
- org.eclipse.persistence.internal.dynamic
- org.eclipse.persistence.internal.expressions
- org.eclipse.persistence.internal.helper
- org.eclipse.persistence.internal.helper.linkedlist
- org.eclipse.persistence.internal.history
- org.eclipse.persistence.internal.indirection
- org.eclipse.persistence.internal.jpa
- org.eclipse.persistence.internal.jpa.deployment
- org.eclipse.persistence.internal.jpa.deployment.xml.parser
- org.eclipse.persistence.internal.jpa.jdbc
- org.eclipse.persistence.internal.jpa.jpql
- org.eclipse.persistence.internal.jpa.metadata
- org.eclipse.persistence.internal.jpa.metadata.accessors
- org.eclipse.persistence.internal.jpa.metadata.accessors.classes
- org.eclipse.persistence.internal.jpa.metadata.accessors.mappings
- org.eclipse.persistence.internal.jpa.metadata.accessors.objects
- org.eclipse.persistence.internal.jpa.metadata.additionalcriteria
- org.eclipse.persistence.internal.jpa.metadata.cache
- org.eclipse.persistence.internal.jpa.metadata.changetracking
- org.eclipse.persistence.internal.jpa.metadata.columns
- org.eclipse.persistence.internal.jpa.metadata.converters
- org.eclipse.persistence.internal.jpa.metadata.copypolicy
- org.eclipse.persistence.internal.jpa.metadata.inheritance
- org.eclipse.persistence.internal.jpa.metadata.listeners
- org.eclipse.persistence.internal.jpa.metadata.locking
- org.eclipse.persistence.internal.jpa.metadata.mappings
- org.eclipse.persistence.internal.jpa.metadata.multitenant
- org.eclipse.persistence.internal.jpa.metadata.nosql
- org.eclipse.persistence.internal.jpa.metadata.partitioning
- org.eclipse.persistence.internal.jpa.metadata.queries
- org.eclipse.persistence.internal.jpa.metadata.sequencing
- org.eclipse.persistence.internal.jpa.metadata.structures
- org.eclipse.persistence.internal.jpa.metadata.tables
- org.eclipse.persistence.internal.jpa.metadata.transformers
- org.eclipse.persistence.internal.jpa.metadata.xml
- org.eclipse.persistence.internal.jpa.metamodel
- org.eclipse.persistence.internal.jpa.parsing
- org.eclipse.persistence.internal.jpa.parsing.jpql
- org.eclipse.persistence.internal.jpa.parsing.jpql.antlr
- org.eclipse.persistence.internal.jpa.querydef
- org.eclipse.persistence.internal.jpa.transaction
- org.eclipse.persistence.internal.jpa.weaving

- org.eclipse.persistence.internal.libraries.antlr.runtime
- org.eclipse.persistence.internal.libraries.antlr.runtime.debug
- org.eclipse.persistence.internal.libraries.antlr.runtime.misc
- org.eclipse.persistence.internal.libraries.antlr.runtime.tree
- org.eclipse.persistence.internal.libraries.asm
- org.eclipse.persistence.internal.libraries.asm.commons
- org.eclipse.persistence.internal.libraries.asm.signature
- org.eclipse.persistence.internal.libraries.asm.tree
- org.eclipse.persistence.internal.libraries.asm.tree.analysis
- org.eclipse.persistence.internal.libraries.asm.util
- org.eclipse.persistence.internal.libraries.asm.xml
- org.eclipse.persistence.internal.localization
- org.eclipse.persistence.internal.localization.i18n
- org.eclipse.persistence.internal.oxm
- org.eclipse.persistence.internal.oxm.accessor
- org.eclipse.persistence.internal.oxm.conversion
- org.eclipse.persistence.internal.oxm.documentpreservation
- org.eclipse.persistence.internal.oxm.mappings
- org.eclipse.persistence.internal.oxm.record
- org.eclipse.persistence.internal.oxm.record.deferred
- org.eclipse.persistence.internal.oxm.record.json
- org.eclipse.persistence.internal.oxm.record.namespaces
- org.eclipse.persistence.internal.oxm.schema
- org.eclipse.persistence.internal.oxm.schema.model
- org.eclipse.persistence.internal.oxm.unmapped
- org.eclipse.persistence.internal.platform.database
- org.eclipse.persistence.internal.queries
- org.eclipse.persistence.internal.security
- org.eclipse.persistence.internal.sequencing
- org.eclipse.persistence.internal.sessions
- org.eclipse.persistence.internal.sessions.coordination
- org.eclipse.persistence.internal.sessions.coordination.broadcast
- org.eclipse.persistence.internal.sessions.coordination.corba
- org.eclipse.persistence.internal.sessions.coordination.corba.sun
- org.eclipse.persistence.internal.sessions.coordination.jms
- org.eclipse.persistence.internal.sessions.coordination.rmi
- org.eclipse.persistence.internal.sessions.coordination.rmi.iiop
- org.eclipse.persistence.internal.sessions.factories
- org.eclipse.persistence.internal.sessions.factories.model
- org.eclipse.persistence.internal.sessions.factories.model.event
- org.eclipse.persistence.internal.sessions.factories.model.log
- org.eclipse.persistence.internal.sessions.factories.model.login
- org.eclipse.persistence.internal.sessions.factories.model.platform
- org.eclipse.persistence.internal.sessions.factories.model.pool
- org.eclipse.persistence.internal.sessions.factories.model.project
- org.eclipse.persistence.internal.sessions.factories.model.property
- org.eclipse.persistence.internal.sessions.factories.model.rcm

- org.eclipse.persistence.internal.sessions.factories.model.rcm.command
- org.eclipse.persistence.internal.sessions.factories.model.sequencing
- org.eclipse.persistence.internal.sessions.factories.model.session
- org.eclipse.persistence.internal.sessions.factories.model.transport
- org.eclipse.persistence.internal.sessions.factories.model.transport.discovery
- org.eclipse.persistence.internal.sessions.factories.model.transport.naming
- org.eclipse.persistence.internal.sessions.remote
- org.eclipse.persistence.jpa.dynamic
- org.eclipse.persistence.jpa.jpql
- org.eclipse.persistence.jpa.jpql.parser
- org.eclipse.persistence.jpa.jpql.tools
- org.eclipse.persistence.jpa.jpql.tools.model
- org.eclipse.persistence.jpa.jpql.tools.model.query
- org.eclipse.persistence.jpa.jpql.tools.resolver
- org.eclipse.persistence.jpa.jpql.tools.spi
- org.eclipse.persistence.jpa.jpql.tools.utility
- org.eclipse.persistence.jpa.jpql.tools.utility.filter
- org.eclipse.persistence.jpa.jpql.tools.utility.iterable
- org.eclipse.persistence.jpa.jpql.tools.utility.iterator
- org.eclipse.persistence.jpa.jpql.utility
- org.eclipse.persistence.jpa.jpql.utility.filter
- org.eclipse.persistence.jpa.jpql.utility.iterable
- org.eclipse.persistence.jpa.jpql.utility.iterator
- org.eclipse.persistence.jpa.metadata
- org.eclipse.persistence.logging
- org.eclipse.persistence.mappings
- org.eclipse.persistence.mappings.converters
- org.eclipse.persistence.mappings.foundation
- org.eclipse.persistence.mappings.querykeys
- org.eclipse.persistence.mappings.structures
- org.eclipse.persistence.mappings.transformers
- org.eclipse.persistence.mappings.xdb
- org.eclipse.persistence.oxm
- org.eclipse.persistence.oxm.annotations
- org.eclipse.persistence.oxm.attachment
- org.eclipse.persistence.oxm.documentpreservation
- org.eclipse.persistence.oxm.mappings
- org.eclipse.persistence.oxm.mappings.converters
- org.eclipse.persistence.oxm.mappings.nullpolicy
- org.eclipse.persistence.oxm.platform
- org.eclipse.persistence.oxm.record
- org.eclipse.persistence.oxm.schema
- org.eclipse.persistence.oxm.sequenced
- org.eclipse.persistence.oxm.unmapped
- org.eclipse.persistence.platform.database
- org.eclipse.persistence.platform.database.converters
- org.eclipse.persistence.platform.database.events

- org.eclipse.persistence.platform.database.jdbc
- org.eclipse.persistence.platform.database.oracle.annotations
- org.eclipse.persistence.platform.database.oracle.jdbc
- org.eclipse.persistence.platform.database.oracle.plsql
- org.eclipse.persistence.platform.database.partitioning
- org.eclipse.persistence.platform.server
- org.eclipse.persistence.platform.xml
- org.eclipse.persistence.platform.xml.jaxp
- org.eclipse.persistence.sequencing
- org.eclipse.persistence.services
- org.eclipse.persistence.services.websphere
- org.eclipse.persistence.sessions.broker
- org.eclipse.persistence.sessions.changesets
- org.eclipse.persistence.sessions.coordination
- org.eclipse.persistence.sessions.coordination.broadcast
- org.eclipse.persistence.sessions.coordination.corba
- org.eclipse.persistence.sessions.coordination.corba.sun
- org.eclipse.persistence.sessions.coordination.jms
- org.eclipse.persistence.sessions.coordination.rmi
- org.eclipse.persistence.sessions.factories
- org.eclipse.persistence.sessions.interceptors
- org.eclipse.persistence.sessions.remote
- org.eclipse.persistence.sessions.remote.corba.sun
- org.eclipse.persistence.sessions.remote.rmi
- org.eclipse.persistence.sessions.remote.rmi.iiop
- org.eclipse.persistence.sessions.serializers
- org.eclipse.persistence.sessions.server
- org.eclipse.persistence.tools
- org.eclipse.persistence.tools.file
- org.eclipse.persistence.tools.profiler
- org.eclipse.persistence.tools.schemaframework
- org.eclipse.persistence.tools.tuning
- org.eclipse.persistence.tools.weaving.jpa
- org.eclipse.persistence.transaction
- org.eclipse.persistence.transaction.was

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java 持久性 API 2.1 功能部件：

- *classloading*
- *jpa*
- *library*
- *transaction*

Java RESTful Services 1.1

启用 jaxrs-1.1 功能部件将对 RESTful Web Services 1.1 提供支持的 Java API。可以使用 JAX-RS 注解以定义遵守 REST 体系结构样式的 Web Service 客户机和端点。可以通过基于 HTTP 标准方法的公共接口来访问端点。

启用此功能部件

要启用 Java RESTful 服务 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaxrs-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java RESTful 服务 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaxrs-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *restConnector-1.0 - JMX REST Connector*

此功能部件提供的标准 API 包

- javax.ws.rs
- javax.ws.rs.core
- javax.ws.rs.ext

此功能部件提供的第三方 API 包

- org.apache.wink.client
- org.apache.wink.client.handlers
- org.apache.wink.client.internal
- org.apache.wink.client.internal.handlers
- org.apache.wink.client.internal.log
- org.apache.wink.common
- org.apache.wink.common.annotations
- org.apache.wink.common.categories
- org.apache.wink.common.http
- org.apache.wink.common.internal
- org.apache.wink.common.internal.application
- org.apache.wink.common.internal.contexts
- org.apache.wink.common.internal.http
- org.apache.wink.common.internal.i18n
- org.apache.wink.common.internal.lifecycle
- org.apache.wink.common.internal.log
- org.apache.wink.common.internal.model
- org.apache.wink.common.internal.model.admin

- org.apache.wink.common.internal.properties
- org.apache.wink.common.internal.providers.entity
- org.apache.wink.common.internal.providers.entity.app
- org.apache.wink.common.internal.providers.entity.atom
- org.apache.wink.common.internal.providers.entity.csv
- org.apache.wink.common.internal.providers.entity.json
- org.apache.wink.common.internal.providers.entity.xml
- org.apache.wink.common.internal.providers.error
- org.apache.wink.common.internal.providers.header
- org.apache.wink.common.internal.providers.multipart
- org.apache.wink.common.internal.registry
- org.apache.wink.common.internal.registry.metadata
- org.apache.wink.common.internal.runtime
- org.apache.wink.common.internal.uri
- org.apache.wink.common.internal.uritemplate
- org.apache.wink.common.internal.utils
- org.apache.wink.common.model
- org.apache.wink.common.model.app
- org.apache.wink.common.model.atom
- org.apache.wink.common.model.csv
- org.apache.wink.common.model.multipart
- org.apache.wink.common.model.opensearch
- org.apache.wink.common.model.rss
- org.apache.wink.common.model.synd
- org.apache.wink.common.model.wadl
- org.apache.wink.common.utils
- org.apache.wink.jcdi.server.internal
- org.apache.wink.jcdi.server.internal.extension
- org.apache.wink.jcdi.server.internal.lifecycle
- org.apache.wink.providers.abdera
- org.apache.wink.providers.jackson
- org.apache.wink.providers.json4j
- org.apache.wink.server.handlers
- org.apache.wink.server.internal
- org.apache.wink.server.internal.application
- org.apache.wink.server.internal.contexts
- org.apache.wink.server.internal.handlers
- org.apache.wink.server.internal.handlers.ejb
- org.apache.wink.server.internal.lifecycle
- org.apache.wink.server.internal.lifecycle.metadata
- org.apache.wink.server.internal.log
- org.apache.wink.server.internal.providers.entity.html
- org.apache.wink.server.internal.providers.exception
- org.apache.wink.server.internal.registry
- org.apache.wink.server.internal.resources
- org.apache.wink.server.internal.servlet
- org.apache.wink.server.internal.servlet.contentencode

- org.apache.wink.server.internal.utils
- org.apache.wink.server.utils
- org.codehaus.jackson
- org.codehaus.jackson.annotate
- org.codehaus.jackson.impl
- org.codehaus.jackson.io
- org.codehaus.jackson.jaxrs
- org.codehaus.jackson.map
- org.codehaus.jackson.map.annotate
- org.codehaus.jackson.map.deser
- org.codehaus.jackson.map.exc
- org.codehaus.jackson.map.introspect
- org.codehaus.jackson.map.jsontype
- org.codehaus.jackson.map.jsontype.impl
- org.codehaus.jackson.map.ser
- org.codehaus.jackson.map.type
- org.codehaus.jackson.map.util
- org.codehaus.jackson.node
- org.codehaus.jackson.schema
- org.codehaus.jackson.sym
- org.codehaus.jackson.type
- org.codehaus.jackson.util
- org.codehaus.jackson.xc

Java RESTful Services 2.0

启用 jaxrs-2.0 功能部件将对 RESTful Web Services 2.0 提供支持的 Java API。可以使用 JAX-RS 注解以定义遵守 REST 体系结构样式的 Web Service 客户机和端点。可以通过基于 HTTP 标准方法的公共接口来访问端点。

启用此功能部件

要启用 Java RESTful 服务 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaxrs-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java RESTful 服务 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaxrs-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jaxrsClient-2.0 - Java RESTful Services Client 2.0*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*

- *restConnector-1.0 - JMX REST Connector*
- *webProfile-7.0 - Java EE Web Profile 7.0*

Java RESTful Services Client 2.0

此功能部件启用对 Java Client API for JAX-RS 2.0 的支持。

启用此功能部件

要启用 Java RESTful 服务客户机 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaxrsClient-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java RESTful 服务客户机 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaxrsClient-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *jaxrs-2.0 - Java RESTful Services 2.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java RESTful 服务客户机 2.0 功能部件：

- *classloading*
- *library*

Java Servlets 3.0

此功能部件启用对依照 Java Servlet 3.0 规范编写的 HTTP servlet 的支持。可将 servlet 打包到 Java EE 指定的 WAR 或 EAR 文件中。如果需要 servlet 安全性，那么还应配置 appSecurity 功能部件；如果缺少安全性功能部件，那么针对应用程序的所有安全性约束将被忽略。

启用此功能部件

要启用 Java Servlet 3.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>servlet-3.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java Servlet 3.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.servlet-3.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *apiDiscovery-1.0 - API Discovery 1.0*
- *appSecurity-1.0 - Application Security 1.0*
- *batchManagement-1.0 - Batch Management*
- *httpWhiteboard-1.0 - OSGi Http Whiteboard*
- *jaspic-1.1 - Java Authentication SPI for Containers 1.1*
- *jaxrs-1.1 - Java RESTful Services 1.1*
- *jpa-2.0 - Java Persistence API 2.0*
- *jsf-2.0 - JavaServer Faces 2.0*
- *jsp-2.2 - JavaServer Pages 2.2*
- *oauth-2.0 - OAuth*
- *openid-2.0 - OpenID*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *openidConnectServer-1.0 - OpenID Connect Provider*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *passwordUtilities-1.0 - Password Utilities*
- *restConnector-1.0 - JMX REST Connector*
- *rtcommGateway-1.0 - WebRTC Rtcomm Gateway*
- *rtcommGateway-1.0 - WebRTC Rtcomm Gateway*
- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *scim-1.0 - System for Cross-domain Identity Management*
- *sipServlet-1.1 - SIP Servlet*
- *spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism*
- *wab-1.0 - OSGi Web Application Bundles*
- *webCache-1.0 - Web Response Cache*
- *webProfile-6.0 - Java EE Web Profile 6.0*
- *wsAtomicTransaction-1.2 - WS-AT Service*
- *wsSecuritySaml-1.1 - WSSecurity SAML*

此功能部件提供的标准 API 包

- javax.servlet
- javax.servlet.annotation
- javax.servlet.descriptor
- javax.servlet.http
- javax.servlet.resources

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java Servlet 3.0 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *channelfw*
- *classloading*
- *cors*
- *enterpriseApplication*
- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *httpSession*
- *javaPermission*
- *library*
- *mimeTypes*
- *pluginConfiguration*
- *tcpOptions*
- *virtualHost*
- *webApplication*
- *webContainer*

Java Servlets 3.1

此功能部件启用对依照 Java Servlet 3.1 规范编写的 HTTP servlet 的支持。可将 servlet 打包到 Java EE 指定的 WAR 或 EAR 文件中。如果需要 servlet 安全性，那么还应配置 appSecurity 功能部件。如果没有安全性功能部件，那么针对该应用程序的所有安全性约束会被忽略。

启用此功能部件

要启用 Java Servlet 3.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>servlet-3.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java Servlet 3.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.servlet-3.1; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *apiDiscovery-1.0 - API Discovery 1.0*
- *appSecurity-1.0 - Application Security 1.0*
- *batch-1.0 - Batch API 1.0*
- *batchManagement-1.0 - Batch Management*
- *httpWhiteboard-1.0 - OSGi Http Whiteboard*

- *jaspic-1.1 - Java Authentication SPI for Containers 1.1*
- *javaee-7.0 - Java EE Full Platform 7.0*
- *jaxrs-1.1 - Java RESTful Services 1.1*
- *jaxrsClient-2.0 - Java RESTful Services Client 2.0*
- *jpa-2.0 - Java Persistence API 2.0*
- *jsf-2.0 - JavaServer Faces 2.0*
- *jsf-2.2 - JavaServer Faces 2.2*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *oauth-2.0 - OAuth*
- *openid-2.0 - OpenID*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *openidConnectServer-1.0 - OpenID Connect Provider*
- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *passwordUtilities-1.0 - Password Utilities*
- *restConnector-1.0 - JMX REST Connector*
- *rtcomm-1.0 - Real-Time Communications*
- *rtcommGateway-1.0 - WebRTC Rtcomm Gateway*
- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *scim-1.0 - System for Cross-domain Identity Management*
- *sipServlet-1.1 - SIP Servlet*
- *spnego-1.0 - Simple and Protected GSSAPI Negotiation Mechanism*
- *wab-1.0 - OSGi Web Application Bundles*
- *webCache-1.0 - Web Response Cache*
- *webProfile-7.0 - Java EE Web Profile 7.0*
- *websocket-1.0 - Java WebSocket 1.0*
- *websocket-1.1 - Java WebSocket 1.1*
- *wsAtomicTransaction-1.2 - WS-AT Service*
- *wsSecuritySaml-1.1 - WSSecurity SAML*

此功能部件提供的标准 API 包

- javax.servlet
- javax.servlet.annotation
- javax.servlet.descriptor
- javax.servlet.http
- javax.servlet.resources

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java Servlet 3.1 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *channelfw*
- *classloading*
- *cors*
- *enterpriseApplication*

- *httpAccessLogging*
- *httpDispatcher*
- *httpEncoding*
- *httpEndpoint*
- *httpOptions*
- *httpProxyRedirect*
- *httpSession*
- *javaPermission*
- *library*
- *mimeTypes*
- *pluginConfiguration*
- *tcpOptions*
- *virtualHost*
- *webApplication*
- *webContainer*

Java Web Services 2.2

jaxws-2.2 功能部件提供对基于 Java API for XML 的 Web Service 2.2 的支持。JAX-WS Web Service 和客户机使用 XML 进行通信。注解可用于简化 JAX-WS 客户机和端点的开发。

启用此功能部件

要启用 Java Web Service 2.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaxws-2.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java Web Service 2.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaxws-2.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jaxb-2.2 - Java XML Bindings 2.2*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*
- *wsAtomicTransaction-1.2 - WS-AT Service*
- *wsSecurity-1.1 - Web Service 安全性*

此功能部件提供的标准 API 包

- javax.wsdl
- javax.wsdl.extensions
- javax.wsdl.extensions.http
- javax.wsdl.extensions.mime
- javax.wsdl.extensions.schema

- javax.wsdl.extensions.soap
- javax.wsdl.extensions.soap12
- javax.wsdl.factory
- javax.wsdl.xml
- javax.xml.ws
- javax.xml.ws.handler
- javax.xml.ws.handler.soap
- javax.xml.ws.http
- javax.xml.ws.soap
- javax.xml.ws.spi
- javax.xml.ws.spi.http
- javax.xml.ws.wsaddressing

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java Web Service 2.2 功能部件：

- *classloading*
- *library*

Java WebSocket 1.0

此功能部件启用对依照 Java API for WebSocket 1.0 规范编写的 WebSocket 应用程序的支持。

启用此功能部件

要启用 Java WebSocket 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>websocket-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java WebSocket 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.websocket-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *servlet-3.1 - Java Servlets 3.1*

此功能部件提供的标准 API 包

- javax.websocket
- javax.websocket.server

Java WebSocket 1.1

此功能部件启用对依照 Java API for WebSocket 1.1 规范编写的 WebSocket 应用程序的支持。

启用此功能部件

要启用 Java WebSocket 1.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>websocket-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java WebSocket 1.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.websocket-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.websocket
- javax.websocket.server

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java WebSocket 1.1 功能部件：

- *wsocOutbound*

Java XML Bindings 2.2

jaxb-2.2 功能部件为 Java Architecture for XML Binding 2.2 规范提供支持，简化从Java类到XML文件的映射。

启用此功能部件

要启用 Java XML 绑定 2.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jaxb-2.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Java XML 绑定 2.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jaxb-2.2; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *jaxws-2.2 - Java Web Services 2.2*

此功能部件提供的标准 API 包

- javax.xml.bind
- javax.xml.bind.annotation
- javax.xml.bind.annotation.adapters
- javax.xml.bind.attachment
- javax.xml.bind.helpers
- javax.xml.bind.util

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Java XML 绑定 2.2 功能部件：

- *classloading*
- *library*

JavaMail 1.5

此功能部件允许应用程序使用 JavaMail 1.5 API。

启用此功能部件

要启用 JavaMail 1.5 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>javaMail-1.5</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaMail 1.5 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.javaMail-1.5; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*
- *javaeeClient-7.0 - Java EE V7 Application Client*

此功能部件提供的标准 API 包

- javax.mail
- javax.mail.event

- javax.mail.internet
- javax.mail.search
- javax.mail.util

此功能部件提供的第三方 API 包

- com.sun.mail
- com.sun.mail.auth
- com.sun.mail.handlers
- com.sun.mail.iap
- com.sun.mail.imap
- com.sun.mail.imap.protocol
- com.sun.mail.pop3
- com.sun.mail.smtp
- com.sun.mail.util
- com.sun.mail.util.logging

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JavaMail 1.5 功能部件：

- *mailSession*

JavaScript Object Notation Processing

Java API for JSON Processing (JSON-P) 功能部件提供一种标准化方法以构造和处理要以 JavaScript 对象表示法 (JSON) 呈现的数据。

启用此功能部件

要启用 JavaScript 对象表示法处理功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jsonp-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaScript 对象表示法处理功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jsonp-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaeeClient-7.0 - Java EE V7 Application Client*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.json
- javax.json.spi
- javax.json.stream

JavaScript Object Notation for Java

此功能部件提供对 JavaScript 对象表示法 (JSON4J) 库的访问。此 JSON4J 库提供简单 Java 模型以构造和处理要呈示为 JSON 数据的数据。

启用此功能部件

要启用“用于 Java 的 JavaScript 对象表示法”功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>json-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于“用于 Java 的 JavaScript 对象表示法”功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.json-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *apiDiscovery-1.0 - API Discovery 1.0*
- *batchManagement-1.0 - Batch Management*
- *jaxrs-1.1 - Java RESTful Services 1.1*
- *jaxrsClient-2.0 - Java RESTful Services Client 2.0*
- *oauth-2.0 - OAuth*
- *restConnector-1.0 - JMX REST Connector*
- *rtcomm-1.0 - Real-Time Communications*
- *scim-1.0 - System for Cross-domain Identity Management*

JavaServer Faces 2.0

此功能部件启用对使用 Java Server Faces (JSF) 框架的 Web 应用程序的支持。此框架简化了用户界面的构造。

启用此功能部件

要启用 JavaServer Faces 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jsf-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaServer Faces 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jsf-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *beanValidation-1.0 - Bean Validation 1.0*
- *beanValidation-1.1 - Bean Validation 1.1*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *webProfile-6.0 - Java EE Web Profile 6.0*

此功能部件提供的标准 API 包

- javax.faces
- javax.faces.application
- javax.faces.bean
- javax.faces.component
- javax.faces.component.behavior
- javax.faces.component.html
- javax.faces.component.visit
- javax.faces.context
- javax.faces.convert
- javax.faces.el
- javax.faces.event
- javax.faces.lifecycle
- javax.faces.model
- javax.faces.render
- javax.faces.validator
- javax.faces.view
- javax.faces.view.facelets

- javax.faces.webapp
- javax.persistence

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JavaServer Faces 2.0 功能部件：

- *classloading*
- *transaction*

JavaServer Faces 2.2

jsf-2.2 功能部件提供对使用 Java Server Faces(JSF) 2.2 框架的 Web 应用程序的支持。此框架会简化用户界面的构造。

启用此功能部件

要启用 JavaServer Faces 2.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jsf-2.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaServer Faces 2.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jsf-2.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jsp-2.3 - JavaServer Pages 2.3*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.faces
- javax.faces.application
- javax.faces.bean
- javax.faces.component
- javax.faces.component.behavior
- javax.faces.component.html
- javax.faces.component.visit
- javax.faces.context
- javax.faces.convert
- javax.faces.el
- javax.faces.event
- javax.faces.flow
- javax.faces.flow.builder

- javax.faces.lifecycle
- javax.faces.model
- javax.faces.render
- javax.faces.validator
- javax.faces.view
- javax.faces.view.facelets
- javax.faces.webapp

JavaServer Pages 2.2

jsp-2.2 功能部件提供对 JSP 2.2 规范中 Java Server Pages (JSP) 的支持。此框架会简化用户界面的构造。

启用此功能部件

要启用 JavaServer Pages 2.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jsp-2.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaServer Pages 2.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jsp-2.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *jsf-2.0 - JavaServer Faces 2.0*
- *oauth-2.0 - OAuth*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *openidConnectServer-1.0 - OpenID Connect Provider*
- *webCache-1.0 - Web Response Cache*
- *webProfile-6.0 - Java EE Web Profile 6.0*

此功能部件提供的标准 API 包

- javax.el
- javax.servlet.jsp
- javax.servlet.jsp.el
- javax.servlet.jsp.jstl.core
- javax.servlet.jsp.jstl.fmt
- javax.servlet.jsp.jstl.sql
- javax.servlet.jsp.jstl.tlv
- javax.servlet.jsp.resources
- javax.servlet.jsp.tagext

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JavaServer Pages 2.2 功能部件：

- *jspEngine*

JavaServer Pages 2.3

此功能部件启用对写至 JSP 2.3 规范的 Java Server Pages (JSP) 的支持。此框架简化了用户界面的构造。启用此功能部件还会启用 Expression Language (EL) V3.0 功能部件。

启用此功能部件

要启用 JavaServer Pages 2.3 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>jsp-2.3</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 JavaServer Pages 2.3 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.jsp-2.3; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *el-3.0 - Expression Language 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *jsf-2.0 - JavaServer Faces 2.0*
- *jsf-2.2 - JavaServer Faces 2.2*
- *oauth-2.0 - OAuth*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *openidConnectServer-1.0 - OpenID Connect Provider*
- *webCache-1.0 - Web Response Cache*
- *webProfile-7.0 - Java EE Web Profile 7.0*

此功能部件提供的标准 API 包

- javax.el
- javax.servlet.jsp
- javax.servlet.jsp.el
- javax.servlet.jsp.jstl.core
- javax.servlet.jsp.jstl.fmt
- javax.servlet.jsp.jstl.sql
- javax.servlet.jsp.jstl.tlv
- javax.servlet.jsp.tagext

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 JavaServer Pages 2.3 功能部件：

- *jspEngine*

Job Manager Integration

此功能部件允许 xigemaAS 概要文件服务器将其状态自动发布至 xigemaAS 作业管理器。作业管理器需要此功能部件来发现它未启动的服务器实例。

启用此功能部件

要启用作业管理器集成功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>serverStatus-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于作业管理器集成功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.serverStatus-1.0; type="osgi.subsystem.feature"
```

LDAP User Registry

此功能部件启用对将 LDAP 服务器用作用户注册表的支持。可使用支持 LDAP V3.0 的任何服务器。可配置多个 LDAP 注册表，然后联合它们以实现单一逻辑注册表视图。

启用此功能部件

要启用 LDAP 用户注册表功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ldapRegistry-3.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 LDAP 用户注册表功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ldapRegistry-3.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *federatedRegistry-1.0 - Federated User Registry*

启用此功能部件的功能部件

- *appSecurity-1.0 - Application Security 1.0*
- *oauth-2.0 - OAuth*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 LDAP 用户注册表功能部件：

- *activedLdapFilterProperties*
- *administrator-role*
- *classloading*
- *customLdapFilterProperties*
- *domino50LdapFilterProperties*
- *edirectoryLdapFilterProperties*
- *idsLdapFilterProperties*
- *iplanetLdapFilterProperties*
- *ldapRegistry*
- *library*
- *netscapeLdapFilterProperties*
- *securewayLdapFilterProperties*

Message Server 1.0

此功能部件启用符合 JMS 的嵌入式消息传递服务器。应用程序可使用 wasJmsClient 功能部件发送和接收消息。

启用此功能部件

要启用消息服务器 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wasJmsServer-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于消息服务器 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wasJmsServer-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置消息服务器 1.0 功能部件：

- *channelfw*
- *classloading*
- *messagingEngine*
- *tcpOptions*
- *transaction*
- *wasJmsEndpoint*
- *wasJmsOutbound*

Message Server Security 1.0

此功能部件允许嵌入式消息传递服务器对来自 JMS 客户机的访问进行认证和授权。

启用此功能部件

要启用消息服务器安全性 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wasJmsSecurity-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于消息服务器安全性 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wasJmsSecurity-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *javaee-7.0 - Java EE Full Platform 7.0*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置消息服务器安全性 1.0 功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *basicRegistry*
- *classloading*

- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *quickStartSecurity*

Message-Driven Beans 3.1

此功能部件允许使用消息驱动的 Enterprise JavaBeans。MDB 允许异步处理 Java EE 组件中的消息。

启用此功能部件

要启用消息驱动的 Bean 3.1 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>mdb-3.1</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于消息驱动的 Bean 3.1 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.mdb-3.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jca-1.6 - Java Connector Architecture 1.6*
- *jndi-1.0 - Java Naming and Directory Interface*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置消息驱动的 Bean 3.1 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*
- *javaPermission*
- *library*
- *transaction*
- *webApplication*

Message-Driven Beans 3.2

此功能部件允许使用依照 EJB 3.2 规范编写的消息驱动的 Enterprise JavaBeans。MDB 允许异步处理 Java EE 组件中的消息。

启用此功能部件

要启用消息驱动的 Bean 3.2 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>mdb-3.2</feature>
```

受支持的 Java™ 版本

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于消息驱动的 Bean 3.2 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.mdb-3.2; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jca-1.7 - Java Connector Architecture 1.7*
- *jndi-1.0 - Java Naming and Directory Interface*

启用此功能部件的功能部件

- *ejb-3.2 - Enterprise JavaBeans 3.2*
- *jmsMdb-3.2 - JMS Message-Driven Beans 3.2*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置消息驱动的 Bean 3.2 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *classloading*
- *ejbApplication*
- *ejbContainer*
- *enterpriseApplication*
- *javaPermission*
- *library*
- *transaction*
- *webApplication*

Micro Profile 1.0

此功能部件组合了支持 Micro Profile for Enterprise Java 的 xigemaAS 功能部件。

启用此功能部件

要启用 Micro Profile 1.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>microProfile-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Micro Profile 1.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.microProfile-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *cdi-1.2 - Contexts and Dependency Injection 1.2*
- *jaxrs-2.0 - Java RESTful Services 2.0*
- *jsonp-1.0 - JavaScript Object Notation Processing*

MongoDB Integration 2.0

此功能部件允许使用 MongoDB Java 驱动程序，并允许在服务器配置中配置数据库实例、将数据库实例插入到受管组件（例如，EJB）中以及通过 JNDI 访问数据库实例。应用程序通过 MongoDB API 与这些数据库实例交互。

启用此功能部件

要启用 MongoDB 集成 2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>mongodb-2.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 MongoDB 集成 2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.mongodb-2.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 MongoDB 集成 2.0 功能部件：

- *classloading*

- *library*
- *mongo*
- *mongoDB*

OAuth

此功能部件允许 Web 应用程序集成 OAuth 2.0 以对用户进行认证和授权。

启用此功能部件

要启用 OAuth 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>oauth-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OAuth 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.oauth-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *json-1.0 - JavaScript Object Notation for Java*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *ldapRegistry-3.0 - LDAP User Registry*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *openidConnectClient-1.0 - OpenID Connect Client*
- *openidConnectServer-1.0 - OpenID Connect Provider*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OAuth 功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*

- *ltpa*
- *oauth-roles*
- *oauthProvider*
- *quickStartSecurity*
- *trustAssociation*

OSGi Application Integration

此功能部件为 OSGi 应用程序添加本地应用程序间集成。

启用此功能部件

要启用 OSGi 应用程序集成功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>osgiAppIntegration-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 OSGi 应用程序集成功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.ws.eba.app.integration-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *blueprint-1.0 - OSGi Blueprint*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OSGi 应用程序集成功能部件：

- *classloading*
- *transaction*

OSGi Blueprint

此功能部件启用对部署 OSGi 应用程序（这些应用程序使用 OSGi 计划容器规范）的支持。通过 xigemaAS 中的 OSGi 应用程序支持，可开发和部署那些使用 Java EE 和 OSGi 技术的模块应用程序。

启用此功能部件

要启用 OSGi Blueprint 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>blueprint-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6

- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 OSGi Blueprint 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.blueprint-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *osgi.jpa-1.0 - OSGi Java Persistence API*
- *osgiAppIntegration-1.0 - OSGi Application Integration*
- *wab-1.0 - OSGi Web Application Bundles*

此功能部件提供的标准 API 包

- org.apache.aries.blueprint
- org.apache.aries.blueprint.ext
- org.apache.aries.blueprint.ext.evaluator
- org.apache.aries.blueprint.mutable
- org.apache.aries.blueprint.services
- org.apache.aries.blueprint.utils
- org.apache.aries.util
- org.osgi.framework
- org.osgi.framework.hooks.bundle
- org.osgi.framework.hooks.resolver
- org.osgi.framework.hooks.service
- org.osgi.framework.hooks.weaving
- org.osgi.framework.launch
- org.osgi.framework.namespace
- org.osgi.framework.startlevel
- org.osgi.framework.wiring
- org.osgi.resource
- org.osgi.service.blueprint
- org.osgi.service.blueprint.container
- org.osgi.service.blueprint.reflect
- org.osgi.service.component
- org.osgi.service.condpermadmin
- org.osgi.service.jndi
- org.osgi.service.packageadmin
- org.osgi.service.startlevel
- org.osgi.service.url
- org.osgi.util.tracker

此功能部件提供的第三方 API 包

- org.apache.aries.transaction.exception

此功能部件提供的 SPI 包

- org.apache.aries.blueprint
- org.apache.aries.blueprint.ext
- org.apache.aries.blueprint.ext.evaluator
- org.apache.aries.blueprint.mutable
- org.apache.aries.blueprint.services
- org.apache.aries.util
- org.osgi.service.cm
- org.osgi.service.subsystem

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OSGi Blueprint 功能部件：

- *application*
- *applicationManager*
- *applicationMonitor*
- *bundleRepository*
- *classloading*
- *javaPermission*
- *library*
- *osgiApplication*
- *osgiApplications*
- *osgiLibrary*
- *transaction*

OSGi Debug Console

此功能部件允许 OSGi 控制台帮助调试运行时。它可用来显示有关捆绑软件、软件包和服务的信息，为产品扩展开发您自己的功能部件时，此信息可能很有用。

启用此功能部件

要启用 OSGi 调试控制台功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>osgiConsole-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OSGi 调试控制台功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.osgiConsole-1.0; type="osgi.subsystem.feature"
```

OSGi Java Persistence API

此功能部件被 blueprint-1.0 和 jpa-2.0 功能部件取代。这些功能部件都添加至服务器后，将自动添加此功能部件。

启用此功能部件

要启用 OSGi Java 持久性 API 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>osgi.jpa-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OSGi Java 持久性 API 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.osgi.jpa-1.0; type="osgi.subsystem.feature"
```

取代此功能部件的功能部件

- *blueprint-1.0 - OSGi Blueprint*
- *jpa-2.0 - Java Persistence API 2.0*

此功能部件启用的功能部件

- *beanValidation-1.0 - Bean Validation 1.0*
- *beanValidation-1.1 - Bean Validation 1.1*
- *blueprint-1.0 - OSGi Blueprint*
- *jdbc-4.0 - Java Database Connectivity 4.0*
- *jdbc-4.1 - Java Database Connectivity 4.1*
- *jndi-1.0 - Java Naming and Directory Interface*
- *jpa-2.0 - Java Persistence API 2.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OSGi Java 持久性 API 功能部件：

- *classloading*
- *transaction*

OSGi Web Application Bundles

此功能部件启用包含 Web 应用程序捆绑软件 (WAB) 的 OSGi 应用程序。Web 应用程序捆绑软件是一些 OSGi 捆绑软件，这些捆绑软件是按构造 WAR 文件的方式在内部构造的，并且支持相同 Web 组件。

启用此功能部件

要启用 OSGi Web 应用程序捆绑软件功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wab-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OSGi Web 应用程序捆绑软件功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wab-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *blueprint-1.0 - OSGi Blueprint*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *wsAtomicTransaction-1.2 - WS-AT Service*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OSGi Web 应用程序捆绑软件功能部件：

- *classloading*
- *transaction*

OpenID

此功能部件允许 Web 应用程序集成 OpenID 2.0 以认证用户而不是（或以及）所配置用户注册表。

启用此功能部件

要启用 OpenID 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>openid-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OpenID 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.openid-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OpenID 功能部件：

- *authFilter*
- *openId*
- *userInfo*

OpenID Connect Client

此功能部件允许 Web 应用程序集成 OpenID Connect Client 1.0 以认证用户而不是（或以及）所配置用户注册表。

启用此功能部件

要启用 OpenID Connect 客户机功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>openidConnectClient-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OpenID Connect 客户机功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.openidConnectClient-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *oauth-2.0 - OAuth*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OpenID Connect 客户机功能部件：

- *administrator-role*
- *authCache*
- *authFilter*
- *authentication*
- *authorization-roles*
- *basicRegistry*

- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *openidConnectClient*
- *quickStartSecurity*
- *trustAssociation*

OpenId Connect Provider

此功能部件允许 Web 应用程序集成 OpenID Connect Server 1.0 以认证用户而不是（或以及）所配置用户注册表。

启用此功能部件

要启用 OpenID Connect 提供者功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>openidConnectServer-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 OpenID Connect 提供者功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.openidConnectServer-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *oauth-2.0 - OAuth*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 OpenID Connect 提供者功能部件：

- *openidConnectProvider*

Performance Monitoring

此功能部件对服务器正在运行的其他功能部件启用性能监视基础结构 (PMI)。可通过标准 MBean 访问监视数据。

启用此功能部件

要启用性能监视功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>monitor-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于性能监视功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.monitor-1.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置性能监视功能部件：

- *classloading*
- *monitor*

Redis DB Integration 1.0

`redisdb-1.0` 功能部件允许应用程序通过 Jedis API 与 redis DB 实例交互，提供对 redis 单节点和集群的数据源支持。应用程序可以通过 JNDI 查找或资源注入两种方式来访问 redis DB。此功能部件支持多数据源的配置方式。

启用此功能部件

要启用基于 Redis 数据库的会话持久性功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>redisdb-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于 RedisDB Integration 1.0 的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.vsettan.xigema.as.redis.datasource.redisdb-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *Java Naming and Directory Interface*（见第 976 页）

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 RedisDB Integration 1.0 功能部件：

- *redisPool*
- *redisDB*

Redis Database Session Persistence

`redisSession-1.0` 功能部件主要用于 HTTP Session 集中式存储，以保证在服务器意外宕机后可以进行数据恢复。开启 `redisSession-1.0` 功能部件并正确配置 redis 数据源，部署在 redis 单机或集群环境中的所有 Web 应用程序都会将其会话数据存储到对应的 redis 数据库中，并且不同应用程序之间的会话数据互不影响。

启用此功能部件

要启用基于 Redis 数据库的会话持久性功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>redisSession-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于数据库会话持久性功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.vsettan.xigema.as.session.redis.redisSession-1.0;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *Redis DB Integration 1.0*（见第 1016 页）
- *Java Naming and Directory Interface*（见第 976 页）
- *Java Servlets 3.0*（见第 988 页）

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置数据库会话持久性功能部件：

- *httpSessionRedis*
- *sessionPolicy*

Real-Time Communications

xigemaAS Real-Time Communications 功能部件 (rtcomm-1.0) 启用高度可缩放的信令基础结构，您可使用此基础结构将 WebRTC 客户机及其他类型的 IoT 节点连接到实时音频/视频/数据交互中。

启用此功能部件

要启用实时通信功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>rtcomm-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于实时通信功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.rtcomm-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *json-1.0 - JavaScript Object Notation for Java*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *rtcommGateway-1.0 - WebRTC Rtcomm Gateway*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置实时通信功能部件：

- *rtcomm*

Request Timing

提供有关缓慢请求或挂起请求的警告和诊断信息。

启用此功能部件

要启用请求计时功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>requestTiming-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于请求计时功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.requestTiming-1.0; type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置请求计时功能部件：

- *requestTiming*

SAML Web Single Sign-on V2.0

此功能部件允许 Web 应用程序使用 SAML Web Single Sign-on V2.0 功能。

启用此功能部件

要启用 SAML Web 单点登录 V2.0 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>samlWeb-2.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 SAML Web 单点登录 V2.0 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.samlWeb-2.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*

启用此功能部件的功能部件

- *wsSecuritySaml-1.1 - WSSecurity SAML*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 SAML Web 单点登录 V2.0 功能部件：

- *administrator-role*
- *authCache*
- *authFilter*
- *authentication*
- *authorization-roles*
- *basicRegistry*

- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*
- *library*
- *ltpa*
- *quickStartSecurity*
- *samlWebSso20*
- *trustAssociation*

SIP Servlet

此功能部件启用对依照 Java SipServlet 1.1 规范编写的 SIP servlet 的支持。可将 servlet 打包到 Java EE 指定的 WAR、SAR 或 EAR 文件中。如果需要 servlet 安全性，那么还应配置 appSecurity 功能部件；如果缺少安全性功能部件，那么针对应用程序的所有安全性约束将被忽略。

启用此功能部件

要启用 SIP Servlet 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>sipServlet-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 SIP Servlet 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.sipServlet-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

启用此功能部件的功能部件

- *rtcommGateway-1.0 - WebRTC Rtcomm Gateway*

此功能部件提供的标准 API 包

- javax.servlet.sip
- javax.servlet.sip.annotation
- javax.servlet.sip.ar
- javax.servlet.sip.ar.spi

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 SIP Servlet 功能部件：

- *channelfw*
- *domainResolver*
- *sipApplicationRouter*
- *sipContainer*

- *sipEndpoint*
- *sipStack*
- *tcpOptions*

Secure Socket Layer

此功能部件启用对安全套接字层 (SSL) 连接的支持。安全 HTTPS 侦听器不会启动，除非已启用 ssl-1.0 功能部件并已配置密钥库。

启用此功能部件

要启用安全套接字层功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>ssl-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果您要开发依赖于安全套接字层功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.ssl-1.0; type="osgi.subsystem.feature"
```

启用此功能部件的功能部件

- *adminCenter-1.0 - Admin Center*
- *apiDiscovery-1.0 - API Discovery 1.0*
- *appSecurity-2.0 - Application Security 2.0*
- *appSecurityClient-1.0 - Application Security for Client 1.0*
- *batchManagement-1.0 - Batch Management*
- *federatedRegistry-1.0 - Federated User Registry*
- *jcaInboundSecurity-1.0 - Java Connector Architecture 1.0 安全性流程*
- *logstashCollector-1.0 - Logstash Collector 1.0*
- *oauth-2.0 - OAuth*
- *openidConnectClient-1.0 - OpenID Connect Client*
- *restConnector-1.0 - JMX REST Connector*
- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *scim-1.0 - System for Cross-domain Identity Management*
- *wasJmsSecurity-1.0 - Message Server Security 1.0*
- *wsAtomicTransaction-1.2 - WS-AT Service*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置安全套接字层功能部件：

- *channelfw*

- *keyStore*
- *ssl*
- *sslDefault*
- *sslOptions*
- *tcpOptions*

Simple and Protected GSSAPI Negotiation Mechanism

此功能部件允许 Web 应用程序集成 SPNEGO 1.0 以认证用户而不是（或以及）所配置用户注册表。

启用此功能部件

要启用简单且受保护的 GSSAPI 协商机制功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>spnego-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于简单且受保护的 GSSAPI 协商机制功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.spnego-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置简单且受保护的 GSSAPI 协商机制功能部件：

- *authFilter*
- *spnego*

Timed Operations

此功能部件启用对应用程序服务器中的某些操作的运行速度比预期慢时的日志记录警告的支持。

启用此功能部件

要启用定时操作功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>timedOperations-1.0</feature>
```

受支持的 Java™ 版本

- JavaSE-1.6
- JavaSE-1.7
- JavaSE-1.8

开发依赖于此功能部件的功能

如果要开发依赖于定时操作功能部件的功能部件，请在功能部件清单文件的 Subsystem-Content 头中添加用于新功能部件的以下项：

```
com.ibm.websphere.appserver.timedOperations-1.0;
 type="osgi.subsystem.feature"
```

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置定时操作功能部件：

- *timedOperation*

WS-AT Service

此功能部件支持基于 Atomic Transaction for XML 的 Web Service 2.2。

启用此功能部件

要启用 WS-AT 服务功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wsAtomicTransaction-1.2</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 WS-AT 服务功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wsAtomicTransaction-1.2;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jaxws-2.2 - Java Web Services 2.2*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *ssl-1.0 - Secure Socket Layer*
- *wab-1.0 - OSGi Web Application Bundles*

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 WS-AT 服务功能部件：

- *administrator-role*
- *authCache*
- *authentication*
- *authorization-roles*
- *basicRegistry*
- *classloading*
- *jaasLoginContextEntry*
- *jaasLoginModule*

- *library*
- *ltpa*
- *quickStartSecurity*
- *transaction*
- *trustAssociation*
- *wsAtomicTransaction*

WSSecurity SAML

此功能部件在 Web Service 安全性 V1.1 功能中启用 SAML 断言。

启用此功能部件

要启用 WS-Security SAML 功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wsSecuritySaml-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 WSSecurity SAML 功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wsSecuritySaml-1.1;
 type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *samlWeb-2.0 - SAML Web Single Sign-on V2.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *wsSecurity-1.1 - Web Service 安全性*

Web Response Cache

webCache-1.0 功能部件会对 Web 响应启用本地高速缓存。它包括 distributedMap 功能部件，并自动高速缓存 Web 应用程序响应以缩短响应时间和提高吞吐量。应用程序可包括 cache-spec.xml 文件以定制响应高速缓存。可通过添加网络高速缓存提供程序分发高速缓存。

启用此功能部件

要启用 Web 响应高速缓存功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>webCache-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Web 响应高速缓存功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.webCache-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *distributedMap-1.0 - Distributed Map interface for Dynamic Caching*
- *jsp-2.2 - JavaServer Pages 2.2*
- *jsp-2.3 - JavaServer Pages 2.3*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*

Web Service Security

此功能部件提供对使用 WS-Security 策略保护 JAX-WS Web Service 的支持。

启用此功能部件

要启用 Web Service 安全性功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>wsSecurity-1.1</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 Web Service 安全性功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.wsSecurity-1.1; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *appSecurity-2.0 - Application Security 2.0*
- *jaxws-2.2 - Java Web Services 2.2*

启用此功能部件的功能部件

- *wsSecuritySaml-1.1 - WSSecurity SAML*

此功能部件提供的第三方 API 包

- org.apache.ws.security
- org.apache.ws.security.action
- org.apache.ws.security.cache
- org.apache.ws.security.components.crypto
- org.apache.ws.security.conversation
- org.apache.ws.security.conversation.dkalgo
- org.apache.ws.security.handler
- org.apache.ws.security.message
- org.apache.ws.security.message.token
- org.apache.ws.security.processor
- org.apache.ws.security.saml
- org.apache.ws.security.saml.ext
- org.apache.ws.security.saml.ext.bean
- org.apache.ws.security.saml.ext.builder
- org.apache.ws.security.spnego
- org.apache.ws.security.str

- org.apache.ws.security.transform
- org.apache.ws.security.util
- org.apache.ws.security.validate

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 Web Service 安全性功能部件：

- *wsSecurityClient*
- *wsSecurityProvider*

WebRTC Rtcomm Gateway

此功能部件允许 Rtcomm 框架连接至 SIP 网络。

启用此功能部件

要启用 WebRTC Rtcomm 网关功能部件，请在 server.xml 文件的 featureManager 元素内添加以下元素声明：

```
<feature>rtcommGateway-1.0</feature>
```

开发依赖于此功能部件的功能

如果您要开发依赖于 WebRTC Rtcomm 网关功能部件的功能部件，请在新功能部件的功能部件清单文件的 Subsystem-Content 头中添加以下项：

```
com.ibm.websphere.appserver.rtcommGateway-1.0; type="osgi.subsystem.feature"
```

此功能部件启用的功能部件

- *rtcomm-1.0 - Real-Time Communications*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.0 - Java Servlets 3.0*
- *servlet-3.1 - Java Servlets 3.1*
- *sipServlet-1.1 - SIP Servlet*

2.1.5 xigemaAS Kernel

xigemaAS 内核。

此功能部件提供的 API 包

- com.ibm.websphere.config.mbeans
- com.ibm.websphere.logging
- com.ibm.websphere.logging.hpel
- com.ibm.websphere.logging.hpel.reader
- com.ibm.websphere.logging.hpel.reader.filters
- com.ibm.websphere.logging.hpel.writer

此功能部件提供的 SPI 包

- com.ibm.websphere.crypto

- com.ibm.websphere.ras
- com.ibm.websphere.ras
- com.ibm.websphere.ras.annotation
- com.ibm.websphere.ras.annotation
- com.ibm.ws.ffdc
- com.ibm.ws.ffdc
- com.ibm.wsspi.config
- com.ibm.wsspi.kernel.filemonitor
- com.ibm.wsspi.kernel.service.location
- com.ibm.wsspi.kernel.service.utils
- com.ibm.wsspi.logging
- com.ibm.wsspi.logging
- com.ibm.wsspi.security.crypto
- com.ibm.wsspi.threading
- org.eclipse.equinox.log
- org.eclipse.osgi.framework.console
- org.eclipse.osgi.framework.eventmgr
- org.eclipse.osgi.framework.log
- org.eclipse.osgi.service.datalocation
- org.eclipse.osgi.service.debug
- org.eclipse.osgi.service.environment
- org.eclipse.osgi.service.localization
- org.eclipse.osgi.service.resolver
- org.eclipse.osgi.service.runnable
- org.eclipse.osgi.service.security
- org.eclipse.osgi.service.urlconversion
- org.eclipse.osgi.signedcontent
- org.eclipse.osgi.storagemanager
- org.eclipse.osgi.util
- org.osgi.service.cm
- org.osgi.service.component
- org.osgi.service.coordinator
- org.osgi.service.event
- org.osgi.service.log
- org.osgi.service.metatype

功能部件配置元素

可在 server.xml 文件中使用以下元素以配置 xigemaAS 内核功能部件：

- *config*
- *executor*
- *featureManager*
- *fileset*
- *include*
- *logging*
- *variable*

2.1.6 产品扩展

可以通过编写产品扩展来扩展 xigemaAS 的功能。

可以编写您自己的 xigemaAS 功能部件，并将其安装到现有 xigemaAS 服务器，也可以将它们打包以交付给用户。

xigemaAS 提供各种可用来扩展运行时环境的系统编程接口 (SPI)；您也可以使用更多高级功能，例如，以程序化方式从 Java™ 应用程序操作 xigemaAS 服务器。

产品扩展

产品扩展是磁盘上的目录，它的结构类似于 xigemaAS 安装目录 ${wlp.install.dir}。它通过 ${wlp.install.dir}/etc/extensions 目录中名为 *extension-name*.properties 的文件定义到 xigemaAS 安装过程中。随后使用产品扩展目录的内容来扩展 xigemaAS。可将多个产品扩展一起安装，但是它们必须具有唯一名称；这可通过为属性文件命名来强制实施。缺省产品扩展位置 ${wlp.user.dir}/extension 不需要属性文件；此位置下的内容是自动检测到的，并且运行时将其识别为“usr”产品扩展。

产品扩展通常包含一个或多个 xigemaAS 功能部件，但是可以具有任何用于扩展 xigemaAS 环境的内容（例如，脚本或资源）。

注：将产品扩展安装到不受 xigemaAS 环境更新影响的目录。有关更多信息，请参阅应用服务或升级时可修改哪些内容?（见第 37 页）。

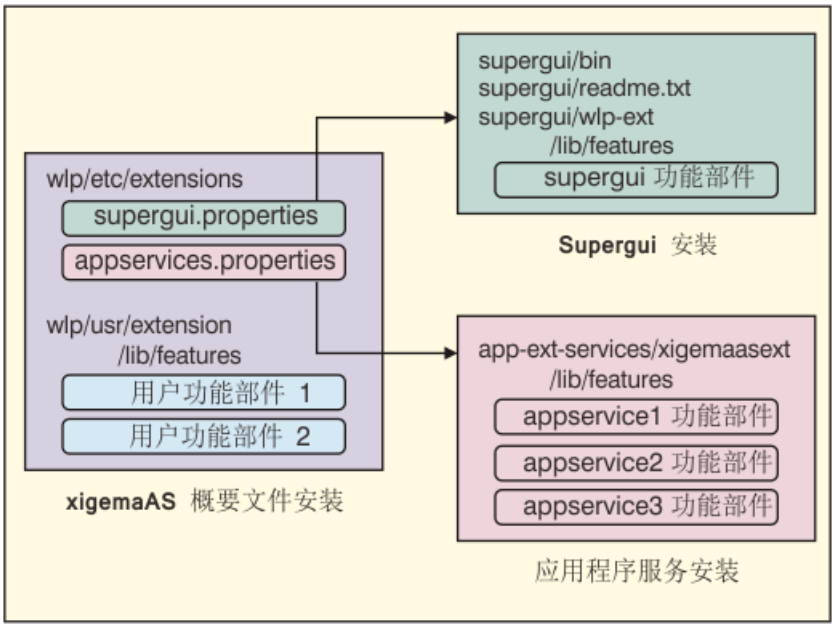


图 14: xigemaAS 产品扩展

产品扩展文件具有下列属性：

```
com.ibm.websphere.productId=your_product_id
com.ibm.websphere.productInstall=absolute_or_relative_file_path
```

注：如果使用了相对文件路径，那么它必须是 `${wlp.install.dir}` 值的对等项。

例如：

```
com.ibm.websphere.productId=org.acme.supergui
com.ibm.websphere.productInstall=supergui/wlp-ext
```

功能部件

xigemaAS 功能部件由定义文件（功能部件清单）组成，是 OSGi 捆绑软件的集合，这些捆绑软件提供对应于 xigemaAS 运行时环境中特定功能的类和服务。

使用xigemaAS 功能部件取代常规应用程序的场景

在一些场景中，将功能实现为 xigemaAS 功能部件，而不是实现为应用程序，可能比较合适。以下列表描述了使用功能部件的一些优点：

- 通过功能部件管理器配置来控制功能部件，因此功能部件独立于用户应用程序管理。
- 功能部件代码可以访问 xigemaAS SPI，这允许与运行时环境进行更深层次的集成。
- 功能部件可以从 server.xml 文件接收用户指定的配置，并且可以在开发工具中提供其配置设置，而不必更改开发工具。
- 功能部件可以轻松地将类和服务提供给彼此以及提供给用户应用程序。
- 功能部件可以非常轻，不含任何应用程序容器依赖性。
- 功能部件可以用来扩充特定编程模型。例如，用户功能部件可以对 OSGi 应用程序编程模型添加对于定制 Blueprint 命名空间处理程序或 Blueprint 注解的支持。

👉 注：功能部件通常无法直接移植到其他应用程序服务器；如果可移植性很重要，那么您应该使用符合规范的应用程序。

开发简单的功能部件

请参阅手动开发 *xigemaAS* 功能部件（见第 1233 页）和*xigemaAS* 功能部件清单文件（见第 1234 页）。

在服务器中使用功能部件

要在 xigemaAS 服务器中使用用户编写的功能部件，必须将该功能部件安装到产品扩展目录。这可以是预定义的“用户产品扩展”位置，也可以是位于 xigemaAS 安装目录外部的扩展。然后，您可以将功能部件名称添加到服务器配置。

用户产品扩展是服务器在其中查找其他功能部件的预定义目录。必须将功能部件定义 .mf 文件复制到 ${wlp.user.dir}/extension/lib/features 目录，并且必须将捆绑软件 .jar 文件复制到 ${wlp.user.dir}/extension/lib 目录。

会按照添加产品功能部件的相同方式，将用户编写的功能部件添加到服务器配置。作为产品扩展一部分的功能部件必须以扩展名称为前缀，以避免与其他提供程序的功能部件产生名称冲突。对于 usr/extension/lib 目录中的功能部件，缺省前缀为 `usr:`。

例如，如果已将称为 `simple-1.0` 的功能部件安装到 ${wlp.user.dir}/extension/lib 目录，那么必须将该功能部件配置到 server.xml，如下所示：

```
<featureManager>
    <feature>usr:simple-1.0</feature>
</featureManager>
```

如果已在您自己的位置安装名为 myFeature 的功能部件，并且已在 ${wlp.install.dir}/etc/extensions/myExt.properties 文件中定义了产品扩展，那么必须将该功能部件配置到 server.xml 文件中，如下所示：

```
<featureManager>
    <feature>myExt:myFeature</feature>
</featureManager>
```

当您启动服务器时，功能部件管理器会检测到该功能部件，而且捆绑软件会安装到 OSGi 框架并启动。

另请参阅添加和移除 *xigemaAS* 功能部件（见第 1129 页）和功能部件管理（见第 905 页）。

以程序化方式从应用程序使用功能部件

功能部件可以将类和服务提供给应用程序。

要提供 Java™ 类以供应用程序使用，必须在功能部件清单的 IBM-API-Package 头中列出类包。在 IBM-API-Package 头中列出类包可使这些类对于应用程序类加载器可见。可通过 API 可见性类型来控制 API 包的可见性。

要允许 OSGi 应用程序使用服务，必须在功能部件清单的 IBM-API-Service 头中列出这些服务。功能部件提供 OSGi 服务，以便您能以程序化方式从应用程序引用这些服务。

通常应该向 OSGi 服务注册表 (SR) 注册服务，以允许应用程序（或其他功能部件）找到这些服务。OSGi 应用程序和其他功能部件可从 SR 执行直接查询，也可以使用 Blueprint 或 OSGi 声明式服务之类的功能来插入其服务依赖关系。Java™ EE 应用程序更有可能通过 JNDI 来查找服务；在 xigemaAS 中，联合了 SR 和 JNDI，从而允许 Java™ EE 应用程序使用 JNDI 在 SR 中查找服务。有关更多信息，请参阅使用 *OSGi* 服务注册表（见第 1244 页）。

将功能部件作为 Web 应用程序来提供

要将 xigemaAS 功能部件作为 Web 应用程序来提供，您可以将功能部件中的 OSGi 捆绑软件发布为 Web 应用程序捆绑软件 (WAB)。除了捆绑软件所需要的 OSGi 头以外，您还可以使用 Web-ContextPath 头来指定 Web 应用程序上下文路径。

例如：

```
Web-ContextPath: myWABapp
Bundle-ClassPath: WEB-INF/classes
```

配置插入和处理

使用功能部件的主要好处在于，用户在 server.xml 文件（以及所包括的文件）中可以轻松配置这些功能部件。配置文件由 xigemaAS 内核进行监视和解析，而且每次更改配置文件时，产生的属性集可以插入到相关组件。

xigemaAS 配置由内核中的 OSGi 配置管理 (CA) 服务管理，而且可以根据该规范来访问。配置属性集由持久存储的身份 (PID) 加以标识，而 PID 用于将 server.xml 文件中的元素与注册用来接收属性的组件相关联。

例如，如果您使用 com.acme.console 的 PID 向 CA 服务注册您的功能部件，那么用户可以在 server.xml 文件中指定以下配置：

```
<com.acme.console color="blue" size="17"/>
```

而且您的功能部件将接收属性：

- `color="blue"`
- `size="17"`

（可选）您可以提供使用 OSGi 元类型描述符来描述配置属性的元数据。使用描述符会导致将配置元数据包含在 xigemaAS 所生成的配置模式中，因此在应用程序开发者配置其服务器时，将自动向其呈现配置属性。

有关接收和描述配置属性的更多详细信息，请参阅启用服务来接收配置数据（见第 1249 页）。

xigemaAS 中的声明式服务

更大或更复杂的功能部件往往受益于使用 OSGi 声明式服务 (DS) 来让功能部件由多项服务组成，以及管理依赖性和配置属性的注入。使用 DS 允许延迟激活服务，延迟装入 Java™ 类直到使用服务，以及根据依赖性对服务激活进行排序。xigemaAS 产品中的大多数功能部件由 DS 管理。

另请参阅使用 *OSGi* 声明式服务来编写高级功能部件（见第 1247 页）。

2.1.7 安全性

xigemaAS 概要文件安全性依照 Servlet 3.0 规范提供对 Web 资源的保护，依照 ejbLite 3.1 规范提供对 EJB 资源的保护。xigemaAS 概要文件安全性还在您使用 REST 连接器的情况下提供对 JMX 连接的保护。

下图显示了访问受保护 Web 资源时涉及的典型安全性过程。要使安全性过程工作，必须配置适当的安全性功能部件，以及设置认证和授权所需的配置。

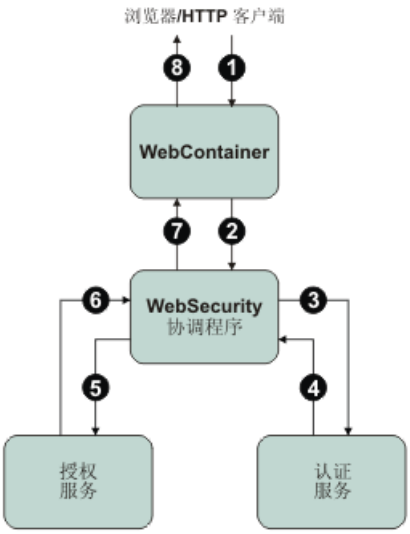


图 15: Web 资源的典型安全性流程

1. HTTP 客户端请求 WebContainer 中的 Web 资源。
2. WebContainer 将安全性检查委派给 WebSecurity 协调程序。
3. WebSecurity 协调程序提示用户输入凭证（如果没有），并使用 认证服务 来认证用户。
4. 认证服务认证、创建和返回主体（如果认证成功）。否则，认证服务会报告认证失败异常。
5. WebSecurity 协调程序使用 授权服务 来执行用户授权检查。
6. 授权服务将授权结果返回给 WebSecurity 协调程序。
7. WebSecurity 协调程序返回用户是否已获授权的安全性检查结果。
8. WebContainer 处理或拒绝所请求的资源。

下列各节概括了 xigemaAS 中的所有主要安全性组件：

快速入门

使用 `quickStartSecurity` 元素，可以在 xigemaAS 概要文件中配置单用户安全性环境。请参阅 安全性快速概述（见第 1034 页）以了解当您使用 `quickStartSecurity` 元素时安全性工作流程的详细信息；请参阅 *xigemaAS* 安全性入门（见第 1275 页）以获取样本任务。

认证

认证确认用户的身份。最常用的认证形式是用户名和密码，例如通过基本认证或者用于 Web 应用程序的表单登录。认证用户时，请求源在运行时表示为 `Subject` 对象。此过程涉及在用户访问资源时，根据为资源配置的授权规则来执行访问控制检查。请参阅 认证（见第 1035 页）以了解更多概念；参阅在 *xigemaAS* 中认证用户（见第 1293 页）以获取详细任务。

授权

授权确定是否要授予用户对系统中资源的访问权。Java™ EE 模型使用主体集、资源和角色来确定哪些可以做，哪些不可以做。此过程涉及检查用户凭证（例如用户标识和密码、证书以及令牌），以及根据已认证的用户来创建主体集。请参阅 授权（见第 1055 页）以了解更多概念；参阅授予对 *xigemaAS* 中资源的访问权（见第 1365 页）以获取详细任务。

安全套接字层 (SSL)

SSL 提供传输级别安全性。请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）以获取详细任务。

单点登录 (SSO)

SSO 支持在不多次提示用户进行登录的情况下访问应用程序。请参阅 *SSO* 的概念以了解更多详细信息；参阅针对 *xigemaAS* 使用 *LTPA cookie* 来定制 *SSO* 配置（见第 1316 页）以获取详细的任务。

Web 安全性相关属性

可以将许多配置属性（例如，SSO 和客户机证书认证）配置为应用程序的 Web 安全性的一部分。参阅为 *xigemaAS* 概要文件配置与 *Web* 安全性相关的属性（见第 1404 页）以获取一些示例。

安全性公用 API

xigemaAS 包含可用来实现安全性功能的公用 API。主要类是 `WSSecurityHelper`、`WSSubject` 和 `RegistryHelper`。此外，还有一个新类 `WebSecurityHelper`。 请参阅安全性公共 *API*（见第 1060 页）。

请参阅开发 *xigemaAS* 概要文件安全性基础结构的扩展（见第 1407 页）以了解一些示例。

管理安全性

管理安全性意味着您可以使用远程 JMX 客户机来管理 xigemaAS。要使用 REST 连接器来保护远程连接，请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）。您还可以开发您自己的 JMX 客户机应用程序，如为 *xigemaAS* 开发 *JMX Java* 客户机（见第 1181 页）中所述。

认证别名

认证数据别名为数据库连接提供安全性支持。请参阅 为 *xigemaAS* 配置认证别名（见第 1406 页）。

配置轻量级目录访问协议 (LDAP)

选择要添加至服务器配置的 LDAP 用户注册表项后，**LDAP 用户注册表详细信息**面板将显示受支持 LDAP 服务器类型列表。如果选择受支持的 LDAP 服务器类型，那么系统不会自动预填充与所选 LDAP 服务器类型相关联的 LDAP 过滤器。

每个受支持 LDAP 服务器类型都定义了一组缺省过滤器。添加 **LDAP 用户注册表**项和服务器类型后，可通过选择 **LDAP 用户注册表**配置并添加所需 LDAP 过滤器来配置关联 LDAP 过滤器：

- Active Directory LDAP 过滤器
- Custom LDAP 过滤器
- Domino LDAP 过滤器
- eDirectory LDAP 过滤器
- IBM Directory Server LDAP 过滤器
- iPlanet LDAP 过滤器
- Netscape LDAP 过滤器
- SecureWay LDAP 过滤器

选择任何 LDAP 过滤器时都会显示过滤器类型的缺省值：

- 用户过滤器
- 组过滤器
- 用户标识映射
- 组标识映射
- 组成员标识映射

如果使用缺省过滤器，那么系统不会使用任何过滤器信息来更新 server.xml 文件。如果任何过滤器发生更改，那么只会在 server.xml 中更新所更改过滤器类型。

注：如果未指定引用标识或未使用**浏览**按钮选择引用标识，那么将使用与所选 LDAP 服务器类型相关联的缺省过滤器。

或者，可向服务器配置添加 LDAP 过滤器。必须指定标识以使该引用与此特定过滤器配置相关联，从而使其与 LDAP 用户注册表配置相关联。如果使用此方法来配置 LDAP 过滤器，那么该引用标识在 **LDAP 用户注册表详细信息**面板上将会被选中（在对应 LDAP 过滤器类型下使用**浏览**按钮定位）。

如果要使用基于 Eclipse 的开发者工具来配置 LDAP，请针对 wlp/templates/config/ldapRegistry.xml 中的样本来验证所保存配置。

有关更多信息，请参阅使用 *xigemaAS* 概要文件来配置 *LDAP* 用户注册表（见第 1294 页）。

故障诊断

使用故障诊断信息来解决使用 xigemaAS 概要文件时的安全性相关问题。请参阅安全性故障诊断（见第 1639 页）和*LDAP* 故障诊断（见第 1641 页）。

安全性快速概述

为了解 xigemaAS 中安全性的基本工作流程，详细介绍了一些常见安全性术语以及示例。

安全性关键术语

认证

认证确认用户的身份。最常用的认证形式是用户名和密码，例如通过基本认证或者用于 Web 应用程序的表单登录。认证用户时，请求源在运行时表示为 `Subject` 对象。

权限

授权决定用户是否能够访问系统中所给定的角色。Java™ EE 模型使用主体、角色和角色映射来确定是否允许访问。

角色

在 Java™ EE 应用程序中定义了角色。系统预定义了一些角色（例如，管理员角色）；另一些角色由应用程序开发者定义。在 Java™ EE 中，通常根据主体 在应用程序中承担的角色来授权或拒绝主体 访问某一角色。

主体

主体（Subject）是常规术语，也是 Java™ 对象：`javax.security.auth.Subject`。通常，术语主体表示系统中的活动实体，例如系统上的用户，甚至是系统进程自身。

安全性工作流示例

以下示例说明了在用户请求访问资源时如何应用安全性。例如，用户 `Bob` 想要访问 servlet `myWebApp`。请参阅 *xigemaAS* 安全性入门（见第 1275 页）中的代码样本。

下列条件必须成立，才能访问 servlet `myWebApp`：

1. `Bob` 必须能登录系统，因为 servlet 是受保护的。
2. `Bob` 必须在 `testing` 角色中，因为 servlet 已使用部署描述符中的 `auth-constraint` 元素进行限制。

如果 `Bob` 无法登录系统，或者 `Bob` 不在 `testing` 角色中，那么将拒绝对 servlet `myWebApp` 进行访问。

另一个用户 `Alice` 可登录系统，因为 `Alice` 是有效用户。但 `Alice` 不在 `testing` 角色中。`Alice` 登录时，会显示 HTTP 403 错误（拒绝/禁止访问）。

认证

在 xigemaAS 安全性中，认证就是确认用户的身份。

要访问受保护的 Web 资源，用户必须提供凭证数据（例如，用户标识和密码）。认证过程涉及收集此用户凭证信息（基于 Web 应用程序配置成收集此数据的方式）并针对所配置注册表验证此信息。验证凭证信息时，会为该用户创建 JAAS 主体（Subject）。该主体包含有关用户的其他信息，例如用户所属的组以及为该用户创建的令牌。然后在授权过程中使用此主体中的信息来确定用户是否可以访问资源。

下图说明 Web 资源的典型认证过程流程。

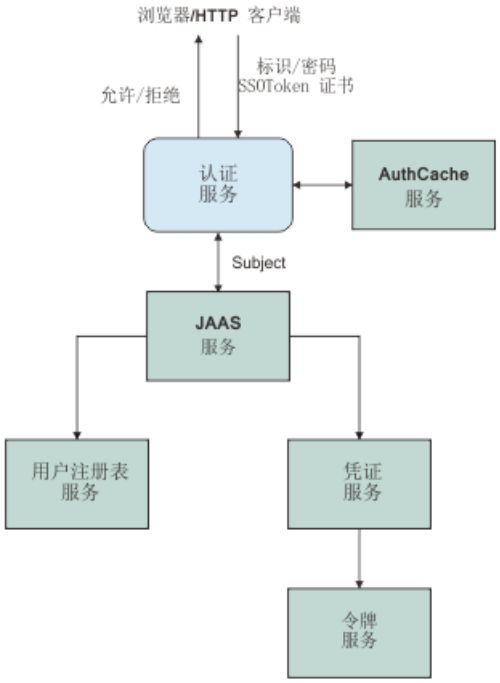


图 16: 认证过程的概述

认证过程涉及从用户收集凭证数据，检查高速缓存以查看是否存在该用户的主体，如果不存在，那么调用 JAAS 服务来执行认证以创建主题。JAAS 服务会调用一组登录模块来处理认证。一个或多个登录模块会根据凭证数据来创建主体。登录模块随后会调用已配置成验证凭证信息的注册表。如果验证成功，那么认证过程会为该用户收集和创建相关信息，其中包括用户所属的组以及用于 SSO 功能的单点登录 (SSO) 令牌，然后将这些信息存储为主体中的相关凭证。在此过程中，您也可以通过插入定制登录模块来定制主体中保存的信息。

如果认证成功，那么会使用一个 cookie 将此过程中创建的 SSO 令牌发送回浏览器。可配置 cookie 的缺省名称为 ltpaToken2。在后续调用中，使用令牌信息来认证用户。如果此认证失败，那么认证服务会尝试使用其他认证数据，例如用户标识和密码（如果它们仍存在于请求中）。

注：Web 应用程序需要表单登录方法来支持包含非 US-ASCII 字符的用户标识和密码。

下列各节详细描述了这些概念：

- 用户注册表（见第 1036 页）
- 认证高速缓存（见第 1036 页）

用户注册表

验证用户的认证数据时，登录模块会调用已配置成验证用户信息的用户注册表。xigemaAS 概要文件支持基于配置的简单用户注册表及基于 LDAP 的更稳健的存储库。有关更多信息，请参阅为 *xigemaAS* 概要文件配置用户注册表（见第 1293 页）。

通过使用 LDAP 注册表，您还可以联合多个存储库以及对两个或两个以上的注册表执行 LDAP 操作。xigemaAS 概要文件用户可以直接在 server.xml 文件中配置 LDAP 注册表联合功能部件，也可以在开发者工具中的 **LDAP 注册表联合**部分进行配置。配置联合存储库之后，您可以通过想要执行的任何操作来获得联合存储库的合并结果。例如，如果您想要对所有以 *test* 开头的用户名执行搜索操作，那么可以对一组 LDAP 注册表执行搜索，您将获得合并搜索结果，然后可以将此搜索结果发送回调用程序。

认证高速缓存

因为创建主体代价颇高，所以 xigemaAS 概要文件会在成功认证用户之后提供认证高速缓存来存储主体。高速缓存的缺省到期时间为 10 分钟。如果用户未在 10 分钟之内重新登录，那么会移除主体并重复认证过程以便为该用户创建主体。如果对配置所作的更改（例如，添加登录模块或者更改 LTPA 密钥）会影响主体的创建，那么这些更改将导致清除认证高速缓存。如果对主体进行了高速缓存，但注册表中的信息发生更改，那么会使用注册表中的信息来更新高速缓存。您可以配置高速缓存超时时间段和高速缓存大小，还可以禁用或启用高速缓存。有关更多信息，请参阅在 *xigemaAS* 上配置认证高速缓存（见第 1306 页）。

JAAS 配置

JAAS 配置定义了一组登录模块来创建主体。xigemaAS 概要文件支持下列 JAAS 配置：

system.WEB_INBOUND

访问 Web 资源（例如 servlet 和 JSP）时使用。

WSLogin

使用程序化登录时供应用程序使用。正在应用程序客户机容器中运行的应用程序也可使用它，但与 `ClientContainerJAAS` 配置不同，它不识别客户机应用程序模块的部署描述符中指定的 `CallbackHandler` 处理程序。

system.DEFAULT

未指定任何 JAAS 配置时用于登录。

system.DESERIALIZE_CONTEXT

反序列化安全上下文时使用。此 JAAS 配置处理认证以在序列化安全上下文时重新创建在线程上处于活动状态的主体。通过在 server.xml 文件中编辑 JAAS 配置条目，可指定此 JAAS 配置并添加您自己的定制 JAAS 登录模块以确保所传播主体包含您的定制信息。

ClientContainer

由正在应用程序客户机容器中运行的应用程序使用。此 JAAS 登录配置识别客户机应用程序模块的部署描述符中指定的 `CallbackHandler` 处理程序（如果已指定）。

system.WEB_INBOUND 和 system.DEFAULT 配置具有下列缺省登录模块，顺序如下：**hashtable**、**userNameAndPassword**、**certificate** 和 **token**。WSLogin 配置具有代理登录模块作为缺省登录模块，并且代理会将所有操作都委派给 system.DEFAULT 中的实际登录模块。

除非您想要使用定制登录模块来进行定制，否则不需要任何显式配置。您可以根据需求来定制特定登录配置。例如，如果您想要定制所有 Web 资源登录，那么只能向 `system.WEB_INBOUND` 配置添加定制登录模块。请参阅为 *xigemaAS* 概要文件配置 *JAAS* 定制登录模块（见第 1307 页）。

JAAS 登录模块

JAAS 配置使用一组登录模块来创建主体。xigemaAS 会在每项登录配置中提供一组登录模块。特定登录模块会根据认证数据来创建主体。将如 JAAS 规范中所规定的那样，使用回调处理程序将认证数据传递到登录模块。例如，如果正在使用用户标识和密码回调处理程序进行认证，那么 **userNameAndPassword** 登录模块会处理认证。如果提供了 `SingleSignonToken` 凭证作为认证数据，那么只有令牌登录模块会处理认证。

xigemaAS 支持下列缺省登录模块：

userNameAndPassword

将用户名和密码用作认证数据时处理认证。

certificate

将 X509 证书用作相互 SSL 的认证数据时处理认证。

token

提供 SSO 令牌作为认证数据时处理认证。在认证过程中，会创建 SSO 令牌并通过 Cookie 将其发送回 HTTP 客户端（浏览器）。在后续请求中，浏览器会发送回此 cookie，并且服务器会从此 cookie 中抽取令牌以认证用户（如果已启用单点登录）。

hashtable

通过预定义的散列表来发送认证数据时使用。有关散列表登录的更多信息，请参阅散列表登录模块（见第 1042 页）。如果仅使用身份来执行认证（例如，在使用 `runAs` 的情况下），那么安全性运行时也使用此登录模块。

proxy

WSLogin 的缺省登录模块。请参阅代理登录模块（见第 1041 页）。

登录模块以其配置顺序进行调用。缺省顺序为 **hashtable**、**userNameAndPassword**、**certificate** 和 **token**。如果必须使用定制登录模块来定制登录过程，那么可以提供这些模块并按照所需顺序进行配置。通常，将定制

登录模块放在登录模块列表的最前面，以便最先调用该定制登录模块。如果使用定制登录模块，那么必须在配置中指定所有登录模块信息并将定制登录模块置于必需顺序。

如果登录模块确定它可以处理认证，那么它会先确保所传入的认证数据是有效的。例如，对于用户名和密码认证，会调用所配置的用户注册表来验证认证信息。对于令牌认证，令牌必须加以解密且有效，才能使验证成功。

如果验证认证数据，那么登录模块会使用用户的其他数据（包括组和 SSO 令牌）来创建凭证。定制登录模块可以通过创建它自己的凭证，在主体中添加其他数据。要使 xigemaAS 概要文件授权工作，主体必须包含 `WSCredential`、`WSPrincipal` 和 `SingleSignonToken` 凭证。`WSCredential` 凭证包含组信息，以及安全性运行时环境所需的其他信息。

回调处理程序

xigemaAS 概要文件支持各种回调处理程序在 JAAS 认证过程中向登录模块提供数据。定制登录模块可以使用回调处理程序信息对自身进行认证。例如，如果回调处理程序需要访问 `HttpServletRequest` 对象中的一些信息，那么可以使用该特定回调处理程序来这样做。

xigemaAS 概要文件支持程序化 JAAS 登录的下列回调处理程序和工厂：

- `com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl`
- `com.ibm.wsspi.security.auth.callback.WSCallbackHandlerFactory`

请参阅为系统登录配置开发 *JAAS* 定制登录模块（见第 1410 页）。

凭证和令牌

如登录模块部分中所述，会在创建主体过程中创建凭证。xigemaAS 概要文件会创建 `WSCredential`、`SingleSignonToken` 和 `WSPrincipal` 凭证。`SingleSignonToken` 凭证包含使用 cookie 发送回浏览器以使 SSO 工作的令牌。此令牌包含用户信息和到期时间。使用服务器第一次启动期间生成的轻量级第三方认证 (LTPA) 密钥对此令牌进行签名和加密。缺省到期时间是 2 小时，是绝对时间，并不取决于用户活动。2 小时之后，令牌会到期，并且用户必须重新登录才能访问资源。

LTPA

LTPA 预期用于分布式及多应用程序服务器环境。在 xigemaAS 概要文件中，LTPA 通过密码术来支持分布式环境中的 SSO 和安全性。此支持使 LTPA 能对与认证相关的数据进行加密、数字签名以及安全传输，并在以后对签名进行解密和验证。

应用程序服务器可以使用 LTPA 协议安全地通信。此协议还提供 SSO 功能部件，因此，用户只有在连接至域名系统 (DNS) 时才需要进行认证。随后，用户可以访问同一个域中其他 xigemaAS 服务器中的资源，而不用提示用户再次进行登录。DNS 域中每个系统上的域名区分大小写，因此必须完全匹配。

LTPA 协议使用密钥来对服务器之间传递的用户数据进行加密和解密。必须在不同服务器之间共享这些密钥，才能让一个服务器中的资源访问其他服务器中的资源，这假定涉及的所有服务器都使用同一个用户注册表。LTPA 要求所配置的用户注册表必须是集中共享的存储库，这样用户和组都相同，而无论是哪个服务器。

使用 LTPA 时，会创建一个包含用户信息和到期时间的令牌，并且使用密钥对该令牌进行签名。LTPA 令牌是时间敏感的。所有参与服务器都必须同步其时间和日期。如果不同步，那么 LTPA 令牌会提早到期，并导致认证或验证失败。缺省情况下使用全球标准时间 (UTC)，并且所有其他服务器都必须具有相同的 UTC 时间。有关如何确保服务器使用相同 UTC 时间的信息，请参阅操作系统文档。

如果已启用 SSO，那么会通过 Web 资源的 Cookie 将 LTPA 令牌传递到其他服务器。

如果接收服务器和发端服务器使用相同的密钥，那么可以对令牌进行解密以获取用户信息，此信息随后会加以验证以确保令牌不会过期，并确保令牌中的用户信息在其注册表中是有效的。成功验证时，便可以在授权检查之后访问接收服务器中的资源。

每个服务器都必须具有有效的凭证。当凭证到期时，需要服务器来与用户注册表通信以进行认证。如果延长 LTPA 令牌保持高速缓存的时间，那么会促使定义安全策略时考虑到安全风险会稍微提升。

如果需要在不同的 xigemaAS 概要文件服务器之间进行密钥共享，请将密钥从一个服务器复制到另一个服务器。为了安全起见，会使用随机生成的密钥对密钥进行加密，并使用用户定义的密码来保护密钥。将密钥导入另一个服务器时，需要此相同密码。该密码仅用来保护密钥，而不用来生成密钥。

如果已启用安全性，那么缺省情况下会在 xigemaAS 概要文件服务器启动时配置 LTPA。有关 LTPA 支持的更多信息，请参阅在 *xigemaAS* 概要文件上配置 *LTPA*（见第 1310 页）。

SPNEGO

简单且受保护 GSS-API 协商机制 (SPNEGO) 允许用户登录 Microsoft™ 域控制器一次就可访问 xigemaAS 服务器上的受保护应用程序，而不用提示用户再次进行登录。

如果已启用 SPNEGO Web 认证，并且浏览器客户机访问 xigemaAS 服务器上的受保护资源，那么 SPNEGO 负责认证对 HTTP 请求中标识的受保护资源的访问。浏览器客户机创建 SPNEGO 令牌并将其作为 HTTP 请求的一部分发送至 xigemaAS 概要文件服务器。xigemaAS 通过 SPNEGO 令牌验证并检索用户身份。此身份用于在用户与应用程序服务器之间建立安全上下文。

有关 SPNEGO 的更多信息，请参阅 *SPNEGO*。有关在 xigemaAS 概要文件服务器上配置 SPNEGO 的进一步信息，请参阅在 *xigemaAS* 中配置 *SPNEGO* 认证。

单点登录 (SSO)

SSO 使用户能够登录到一个位置（例如，一个服务器）来访问其他服务器上的应用程序，而不用提示用户再次进行登录。要使 SSO 工作，必须在不同的 xigemaAS 概要文件服务器之间交换 LTPA 密钥，用户注册表必须相同，而且令牌不得过期。要交换 LTPA 密钥，您可以将 `ltpa.keys` 文件从一个服务器复制到另一个服务器，然后重新启动服务器以使用新的 LTPA 密钥。参与 SSO 域的所有服务器所使用的注册表都必须相同。

如果在一个 xigemaAS 概要文件服务器中对用户进行了认证，那么会将认证过程中为该用户创建的 SSO 令牌放入发送至 HTTP 客户端（例如，浏览器）的 Cookie。如果该客户机中发出另一个请求，要求访问位于另一服务器上但位于同一 DNS（配置为第一个服务器中的 SSO 配置的一部分）中的另一组应用程序，那么会随该请求一起发送 Cookie。接收服务器会尝试使用 cookie 中的令牌来认证用户。如果这两个条件都满足，那么接收服务器会验证令牌并根据此服务器中的用户创建主体，而不会提示用户再次登录。如果无法验证令牌（例如，由于 LTPA 密钥不匹配而无法对令牌进行解密或验证），那么会提示用户重新输入凭证信息。

配置成使用 `form-login` 属性的任何应用程序都必须针对该服务器配置 SSO。针对 `form-login` 认证用户时，会将令牌发送回浏览器，并在用户访问资源时将此令牌用于为用户授权。

请参阅针对 *xigemaAS* 使用 *LTPA cookie* 来定制 *SSO* 配置（见第 1316 页）。

SAML Web 浏览器 SSO

SAML Web 浏览器 SSO 允许 Web 应用程序将用户认证授权给 SAML 身份提供者（而不是所配置的用户注册表）。

有关在 xigemaAS 服务器上配置 SAML Web 浏览器 SSO 的更多信息，请参阅*SAML 2.0 Web* 浏览器单点登录（见第 1051 页）。

插入式认证

使用下列方法来定制认证过程：

- 提供定制登录模块。大部分认证过程都基于 JAAS 登录模块，因此您可以在 xigemaAS 概要文件所提供的登录模块之前、之后或之间插入定制登录模块。请参阅为 *xigemaAS* 概要文件配置 *JAAS* 定制登录模块（见第 1307 页）。
- 实现信任关联拦截器 (TAI) 以处理所有基于 Web 资源的认证。请参阅为 *xigemaAS* 开发定制 *TAI*（见第 1407 页）。

身份断言

除了要求请求实体来证明其身份的认证过程之外，xigemaAS 概要文件还支持身份断言。这是一种形式宽松的认证，不需要身份证明，但根据与实体（为断言身份作证）之间的信任关系来接受身份。

使用下列方法来断言 xigemaAS 概要文件中的身份：

1. 使用散列表登录。请参阅为系统登录配置开发 *JAAS* 定制登录模块（见第 1410 页）。
2. 使用 `IdentityAssertionLoginModule`。您可以允许应用程序或系统提供者通过信任验证来断言身份。要使用 `IdentityAssertionLoginModule`，请使用 JAAS 登录框架，在这种情况下，将在一个定制登录模块中完成信任验证，而在 `IdentityAssertionLoginModule` 中完成创建凭证。您可以使用两个登录模块来创建可用来断言身份的 JAAS 登录配置。

   需要下列两个定制登录模块：

用户实现的登录模块（信任验证）

用户实现的登录模块会执行用户需要在信任验证中执行的任何操作。验证信任时，必须在登录模块的共享状态下将信任验证状态和登录身份放置在映射中，以便凭证创建登录模块能使用该信息。将此映射存储在下列属性中：

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

此属性由下列属性组成：

- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted
  ```

  此属性会设置为 true（如果可信）或 false（如果不可信）。
- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal
  ```

  此属性包含身份的主体。
- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates
  ```

  此属性包含身份的证书。

身份断言登录模块（凭证创建）

下列模块会创建凭证：

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule
```
```

此模块依靠登录上下文的共享状态中的信任状态信息。

身份断言登录模块会在共享状态属性中查找信任信息：

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

此属性包含信任状态及用来登录的身份，并且必须包含下列属性：

- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted
  ```
  此属性会设置为 true（如果可信）或 false（如果不可信）。
- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal
  ```
  此属性包含使用主体时用来登录的身份的主体。
- ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates
  ```
  此属性包含一组证书链，内含使用证书时用来登录的身份。

如果缺少状态、信任或身份信息，那么会返回 `WSLoginFailedException` 消息。登录模块随后会使用该身份来登录，并且主体包含新身份。

请参阅定制应用程序登录以使用 *JAAS* 来执行身份断言（见第 1422 页）。

RunAs() 认证

如果您在调用 Servlet 之后已成功完成认证，那么 Servlet 可以进行后续调用（例如，对其他 Servlet 进行调用）。通常是使用您最初用来登录 servlet 的相同安全身份来进行这些后续调用。此身份称为调用者身份。此外，您还可以选择通过使用 RunAs 规范来委派给其他身份，以便 servlet 进行的任何后续调用都以此其他身份运行。概括地说，您具有两个选项来传播该安全身份：

- 传播调用者身份（缺省行为）。
- 委派给 RunAs 身份，可以通过使用 RunAs 规范来指定 RunAs 身份。

在服务器对原始用户进行认证之后，服务器随后会对 RunAs 用户进行认证。如果此认证失败，那么服务器会后退以传播调用者身份。

要使用 RunAs 规范，必须更新应用程序的部署描述符以包含 `run-as` 元素或 `@RunAs` 注解。将此元素设置为您想要委派给的安全角色。

请参阅在 *xigemaAS* 概要文件中配置 *RunAs* 认证（见第 1316 页）。

代理登录模块

代理登录模块类会装入应用程序服务器登录模块，并将所有操作都委派给实际的登录模块实现。实际的登录模块实现是作为选项配置中的委托选项指定的。该代理登录模块是必需的，因为应用程序类装入器看不到应用程序服务器产品的共享库类装入器。借助应用程序的程序化登录（将 `LoginContext` 类的 `Login()` 方法与 JAAS 登录上下文条目 WSLogin 配合使用），代理登录模块会将所有工作都委派给 JAAS 登录上下文条目 system.DEFAULT。

证书登录

借助证书登录功能，您可以通过使用客户端 X509 证书取代提供用户标识和密码来认证诸如 servlet 等 Web 请求。
  ```

证书认证的工作方式是将用户注册表中的用户与 Web 请求的客户机证书中的专有名称相关联。通过让服务器信任客户机证书来建立信任，例如，客户机证书的签署者必须位于服务器的信任库中。使用此机制，用户不需要提供密码即可建立信任。

请参阅保护与 *xigemaAS* 的通信（见第 1278 页）。

散列表登录模块

在用户注册表中查找必需的属性，将属性放入散列表中，然后将该散列表添加到共享状态。如果在此登录模块中切换了身份，那么您必须将散列表添加到共享状态。如果未切换身份，但是 `requiresLogin` 代码的值为 true，那么您可以创建属性的散列表。在这种情况下，您不必创建散列表，因为 xigemaAS 会为您处理登录。但是，在特殊情况下，您可以考虑创建散列表来收集属性。例如，如果您正在使用自己的专用用户注册表，那么简单的解决方案是创建 `UserRegistry` 实现，使用散列表，然后让服务器为您收集用户属性。

下列规则更详细地定义了如何完成散列表登录。必须在主体集（公用或专用凭证集合）或共享状态散列映射中使用 `java.util.Hashtable` 对象。`com.ibm.wsspi.security.token.AttributeNameConstants` 类定义包含用户信息的键。如果使用列在散列表登录模块之前的定制登录模块将散列表对象放入登录上下文的共享状态，请使用下列键在共享状态散列映射中搜索 `java.util.Hashtable` 对象的值：

属性

com.ibm.wsspi.security.cred.propertiesObject

对属性的引用

AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY

说明

此键搜索在登录上下文的共享状态中包含必需属性的散列表对象。

期望的结果

java.util.Hashtable 对象。

如果在主体集或者共享状态区域中找到 `java.util.Hashtable` 对象，请验证散列表中是否存在下列属性：

- `com.ibm.wsspi.security.cred.uniqueId`

对属性的引用

AttributeNameConstants.WSCREDENTIAL_UNIQUEID

返回

java.util.String

说明

此属性值必须是用户的唯一表示。对于 xigemaAS 缺省实现，此属性表示应用程序授权配置中存储的信息。部署信息并执行用户到角色的映射后，此信息位于应用程序部署描述符中。如果通过查找 xigemaAS 的用户注册表实现来执行用户到角色的映射，请参阅所需要格式的示例。

所需要格式的示例

表 7: uniqueId 的格式示例

此表给出期望的格式示例。

用户注册表	格式 **(UniqueUserId)** 值
LDAP	ldapRegistryRealm/cn=kevin,o=mycompany,c=use
Basic	basicRegistryRealm/kelvin

`com.ibm.wsspi.security.cred.uniqueId` 属性是必需的。

- `com.ibm.wsspi.security.cred.securityName`

对属性的引用

AttributeNameConstants. WSCREDENTIAL_ SECURITYNAME

返回

java.util.String

说明

此属性将搜索认证用户的 securityName。此名称通常称为显示名称或短名称。xigemaAS 对 `getRemote User`、`getUserPrincipal` 和 `getCallerPrincipal` 应用程序编程接口 (API) 使用 securityName 属性。要确保与 securityName 值的缺省实现的兼容性，请调用 `public String getUserSecurityName (String uniqueUserId) UserRegistry` 方法。

所需要格式的示例

表 8: securityName 的格式示例

此表给出期望的格式示例。

用户注册表	格式 **(securityName)** 值
LDAP	kevin
Basic	kevin

`com.ibm.wsspi.security.cred.securityname` 属性是必需的。

- `com.ibm.wsspi.security.cred.group`

对属性的引用

AttributeNameConstants. WSCREDENTIAL_GROUP

返回

java.util.ArrayList

说明

此键搜索此用户所属的组的数组列表。以 *realm_name/user_name* 格式指定这些组。这些组的格式很重要，因为 xigemaAS 授权引擎使用这些组在部署描述符中进行组到角色的映射。提供的格式必须与 xigemaAS 缺省实现需要的格式匹配。在使用第三方权限提供者时，必须使用第三方提供者所需的格式。要确保与唯

一组标识值的缺省实现的兼容性，请调用 `public List getUniqueGroupIds(String uniqueUserId) UserRegistry` 方法。

所需要格式的示例

表 9: 组的格式示例

此表提供了配置入站标识映射时的某些格式示例。

用户注册表	格式（组）值
LDAP	ldapRegistryRealm/cn=group1,o=Groups,c=US
Basic	basicRegistryRealm/group1

`com.ibm.wsspi.security.cred.group` 属性是必需的。用户不需要具有关联组。

- `com.ibm.wsspi.security.cred.cacheKey`

对属性的引用

AttributeNameConstants. WSCREDENTIAL_CACHE_KEY

返回

java.lang.Object

说明

这个键属性可指定表示登录唯一属性（包括可能影响唯一性的特定于用户的信息以及用户动态属性）的对象。例如，当用户从位置 A 登录时，这可能影响他们的访问控制，那么高速缓存键必须包括位置 A，以便接收到的主体集是当前位置的正确主体集。

`com.ibm.wsspi.security.cred.cacheKey` 属性不是必需的。未指定此属性时，高速缓存查找是对 WSCREDENTIAL_UNIQUEID 指定的值。在 `java.util.Hashtable` 对象中找到此信息时，xigemaAS 创建的主体集类似于通过正常登录过程的主体集（至少对于 LTPA 来说如此）。新的主体集包含 `WSCredential` 对象和 `WSPrincipal` 对象，其使用散列表对象中找到的信息进行完全填充。

使用 SPNEGO Web 认证针对 HTTP 请求进行单点登录

可通过对 xigemaAS 使用简单且受保护的 GSS-API 协商机制 (SPNEGO) 作为 Web 认证服务，以安全地协商和认证针对 xigemaAS 中的受保护资源的 HTTP 请求。

下列各节更详细地描述了 SPNEGO Web 认证：

- 什么是 *SPNEGO?*（见第 1044 页）
- 单个 *Kerberos* 域中的 *SPNEGO Web* 认证（见第 1045 页）
- 可信 *Kerberos* 域中的 *SPNEGO Web* 认证（见第 1047 页）
- 有关向浏览器客户端进行 *SPNEGO Web* 认证的支持信息（见第 1048 页）

什么是 SPNEGO?

SPNEGO 是 简单且受保护的 *GSS-API* 协商机制 *(IETF RFC 2478)* 中定义的一种标准规范。

如果启用了 xigemaAS 概要文件服务器安全性和 SPNEGO Web 认证，那么系统会在处理第一个入站 HTTP 请求时初始化 SPNEGO。如果未指定认证过滤器，或指定了认证过滤器并且满足条件，那么 SPNEGO 负责认证对 HTTP 请求中标识的受保护资源的访问。

要启用 SPNEGO 操作，除了需要 xigemaAS 安全性运行时服务以外，还需要一些外部组件。这些外部组件包括下列各项：

- 具有 Active Directory 域和相关联的 Kerberos 密钥分发中心 (KDC) 的 Microsoft™ Windows™ Server。
- 支持 IETF RFC 2478 中定义的 SPNEGO Web 认证机制的客户机应用程序（例如，Microsoft™ .NET 或 Web Service 和 J2EE 客户机）。Microsoft™ Internet Explorer 和 Mozilla Firefox 是浏览器示例。要使用的任何浏览器必须配置为使用 SPNEGO Web 认证机制。

HTTP 请求的认证由用户（客户端）触发，这会生成 SPNEGO 令牌。xigemaAS 接收此令牌。具体地说，SPNEGO Web 认证通过 SPNEGO 令牌对用户身份解码并进行检索。然后使用该身份制定权限决策。

SPNEGO Web 认证是 xigemaAS 中的服务器端解决方案。客户端应用程序负责生成 SPNEGO 令牌供 SPNEGO Web 认证使用。xigemaAS 安全注册表中的用户身份必须与 SPNEGO Web 认证检索的身份完全相同。当 Microsoft™ Windows™ Active Directory 服务器是 xigemaAS 中使用的“轻量级目录访问协议”(LDAP) 服务器时，该标识就会完全匹配。定制登录模块是作为插件提供的，用来支持从 Active Directory 到 xigemaAS 安全性注册表的定制身份映射。

xigemaAS 会对照其安全注册表验证身份。如果验证成功，那么系统会检索客户机 GSS 授权凭证并放置在客户机主体集中，并创建轻量级第三方认证 (LTPA) 安全令牌。然后，它会在 HTTP 响应中将 LTPA cookie 返回给用户。来自同一用户的后续 HTTP 请求（要求访问 xigemaAS 中的更多受保护资源）使用先前创建的 LTPA 安全令牌以避免重复登录提问。

单个 Kerberos 域中的 SPNEGO Web 认证

SPNEGO Web 认证在单个 Kerberos 域中受支持。下图中显示了提问应答握手过程：

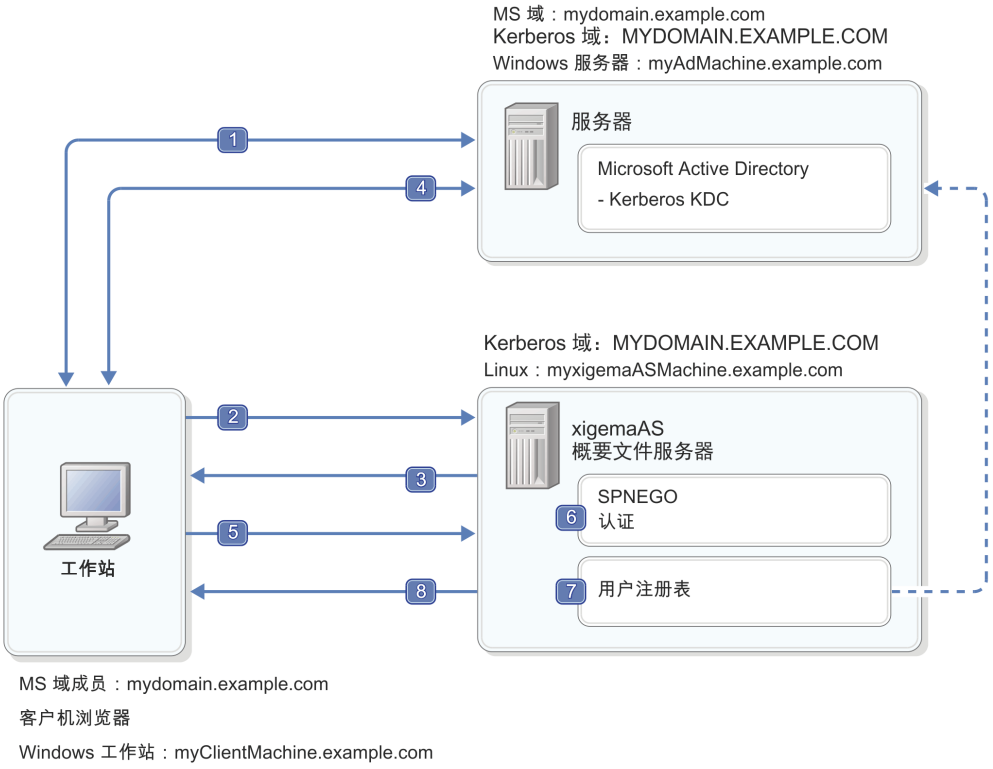


图 17: 单个 Kerberos 域中的 SPNEGO Web 认证

在上图中，发生了下列事件：

1. 首先，用户从工作站登录 Microsoft™ 域控制器 `MYDOMAIN.EXAMPLE.COM`。
2. 然后，用户尝试访问 Web 应用程序。用户使用客户机浏览器请求受保护的 Web 资源，这会向 xigemaAS 概要文件服务器发送 `HTTP GET` 请求。
3. xigemaAS 概要文件服务器中的 SPNEGO 认证使用包含 `Authenticate: Negotiate` 状态的 `HTTP 401` 提问头应答客户机浏览器。
4. 客户机浏览器会识别协商头，因为客户机浏览器配置为支持集成 Windows™ 认证。客户机针对主机名解析所请求的 URL。客户机使用主机名构成目标 Kerberos 服务主体名称 (SPN) `HTTP/myxigemaASMachine.example.com`，以请求来自 Microsoft™ Kerberos KDC (`TGS_REQ`) 中的 Kerberos 凭单授予服务 (`TGS`) 的 Kerberos 服务凭单。然后，`TGS` 对客户机发出 Kerberos 服务凭单 (`TGS_REP`)。Kerberos 服务凭单（SPNEGO 令牌）对该服务（xigemaAS 概要文件服务器）证明用户的身份和许可权。
5. 然后，客户机浏览器响应 xigemaAS 概要文件服务器认证：使用执行先前步骤时在请求 HTTP 头中获取的 SPNEGO 令牌协商提问。
6. xigemaAS 概要文件服务器中的 SPNEGO 认证查看带有 SPNEGO 令牌的 HTTP 头，验证 SPNEGO 令牌，并获取用户的身份（主体）。
7. xigemaAS 概要文件服务器获取用户身份后，它会在其用户注册表中验证该用户并执行授权检查。
8. 如果授予访问权，那么 xigemaAS 服务器会发送带有 `HTTP 200` 的响应。xigemaAS 服务器还会在响应中包含 LTPA cookie。此 LTPA cookie 用于后续请求。

注：支持 SPNEGO 的其他客户机（例如，Web Service、.NET 和 J2EE）不必遵循前面所显示的提问应答握手过程。这些客户机可获取用于目标服务器的授予凭单的凭单 (TGT) 和 Kerberos 服务凭单，创建 SPNEGO 令牌，将其插入到 HTTP 头中，然后遵循用于创建 HTTP 请求的正常过程。

可信 Kerberos 域中的 SPNEGO Web 认证

SPNEGO Web 认证在可信 Kerberos 域中也受支持。下图中显示了提问应答握手过程：

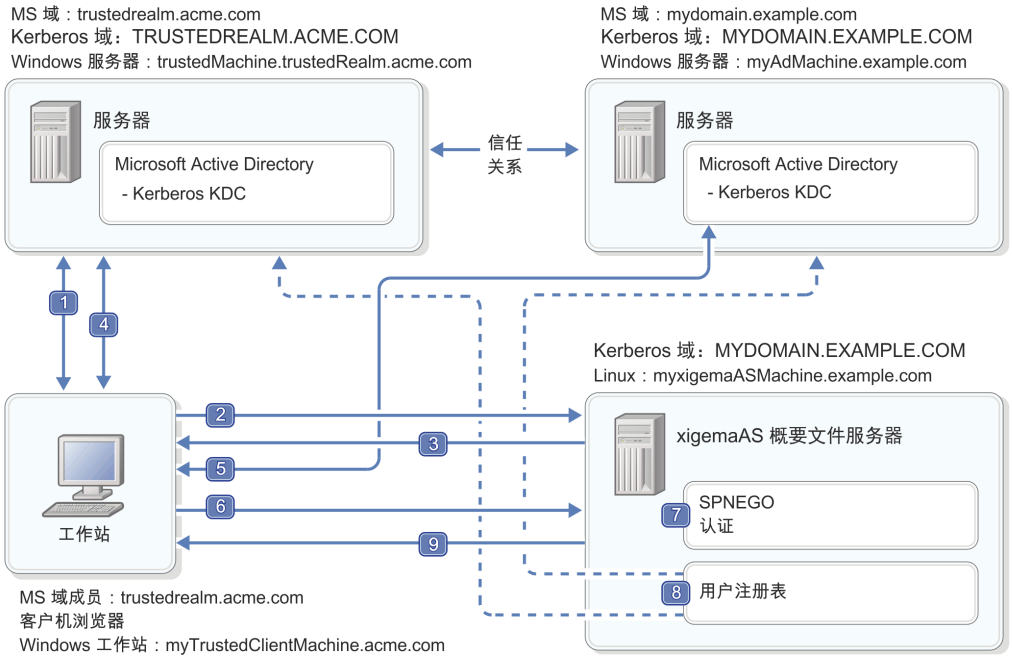


图 18: 可信 Kerberos 域中的 SPNEGO Web 认证

在上图中，发生了下列事件：

1. 用户登录 Microsoft™ 域控制器 `TRUSTEDREALM.ACME.COM`。
2. 在客户机浏览器中，用户发出针对原始 Microsoft™ 域控制器 `MYDOMAIN.EXAMPLE.COM` 中的 xigemaAS 服务器上的受保护 Web 资源的请求。
3. xigemaAS 服务器使用包含 `Authenticate: Negotiate` 状态的 HTTP 401 提问头应答客户机浏览器。
4. URL。客户机浏览器使用主机名作为属性，以请求来自域 `TRUSTEDREALM.ACME.COM` 的用于 `MYDOMAIN.EXAMPLE.COM` 的 Kerberos 跨域凭单 (`TGS_REQ`)。
5. 客户机浏览器使用步骤 4 中的 Kerberos 跨域凭单请求来自域 `MYDOMAIN.EXAMPLE.COM` 的 Kerberos 服务凭单。Kerberos 服务凭单（SPNEGO 令牌）对该服务（xigemaAS 概要文件服务器）证明用户的身份和许可权。
6. 然后，客户机浏览器响应 xigemaAS 概要文件服务器认证：使用执行先前步骤时在请求 HTTP 头中获取的 SPNEGO 令牌协商提问。
7. xigemaAS 概要文件服务器接收请求并检查带有 SPNEGO 令牌的 HTTP 头。然后，它会抽取 Kerberos 服务凭单，验证该凭单，并获取用户的身份（主体）。
8. xigemaAS 概要文件服务器获取用户身份后，它会在其用户注册表中验证该用户并执行授权检查。
9. 如果授予访问权，那么 xigemaAS 概要文件服务器会发送带有 `HTTP 200` 的响应。xigemaAS 概要文件服务器还会在响应中包含 LTPA cookie。此 LTPA cookie 用于后续请求。

👉 注：不需要对 xigemaAS 概要文件服务器进行任何修改就可支持更多可信域。SPNEGO 只需要必需 Active Directory 域之间的信任关系就可使用可信域。

在可信 Kerberos 域环境中，应注意必须在每个 Kerberos KDC 上完成 Kerberos 可信域设置。请参阅 Kerberos Administrator and User's Guide，以获取有关如何设置 Kerberos 可信域的更多信息。

有关向浏览器客户端进行 SPNEGO Web 认证的支持信息

支持以下方案：

- 跨林信任
- 同一林中的域信任
- Kerberos 域信任

不支持以下方案：

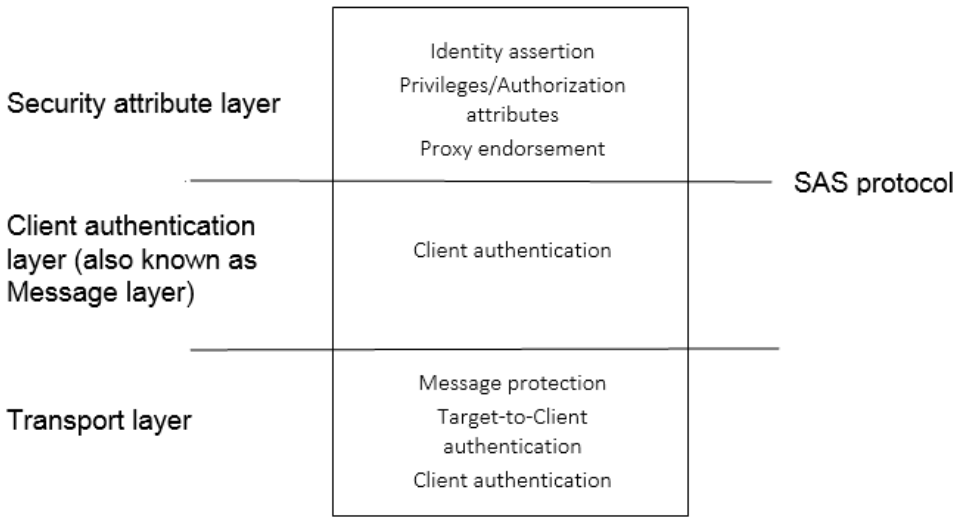
- 林外部信任
- 域外部信任

有关在 xigemaAS 服务器上配置 SPNEGO 的进一步信息，请参阅在 *xigemaAS* 中配置 *SPNEGO* 认证。

公共安全互操作性 V2 (CSIv2)

公共安全互操作性 V2 (CSIv2) 是用于满足认证、授权和权限的 CORBA 安全互操作性需求的体系结构。在 CSIv2 体系结构中，将使用 SAS 协议在 GIOP 请求和回复消息的服务上下文中交换令牌以建立安全上下文。传输层安全性 (SSL/TLS) 是 SAS 必需的，它在上面多提供了两层用于客户机认证和授权。

SAS 协议分为两层。认证层用于在传输时未能完成足够认证的情况下执行客户机认证。客户机可使用属性层将安全属性（例如，身份）推送或提交至目标服务器，在目标服务器中，可在访问控制决策中应用这些属性。在 CSIv2 文档中，传输又称为另一层（以便于进行讨论），尽管它不在 SAS 协议消息中，并且 SAS 消息位于传输之上。



CSIv2 身份断言

使用 RMI/IIOP 执行请求时，属性层中的 CSIv2 身份断言支持用于在从客户机进程至服务器进程的传输过程中断言身份。

断言是一个实体对另一个实体的声明，用于代表它自身接受身份。客户机可断言表示主体集的身份，该主体集在启动远程资源时生效。除表示该用户的身份令牌外，客户机进程还会在认证层或传输层中发送它自己的身份。目标服务器通过执行信任验证确保客户机进程能够断言身份。如果目标服务器信任客户机，那么服务器使用已断言身份以创建服务器端主体集，该主体集表示调用时在客户机进程上生效的用户。

客户机可断言使用主体名称身份令牌的用户。主体名称的格式取决于客户机进程上配置的用户注册表。匿名身份令牌类型也是受支持的，服务器接收到这类令牌时会使用未认证主体集。

有关使用身份断言配置 CSIV2 属性层的信息，请参阅配置入站 *CSIv2* 属性层（见第 1385 页）或配置出站 *CSIv2* 属性层（见第 1390 页）。

CSIv2 认证层

使用 RMI/IIOP 执行请求时，系统使用 CSIv2 认证层将认证信息从客户机进程传输至服务器进程。

CSIv2 认证层可包含客户机发送的令牌，服务器随后可使用该令牌认证客户机。各种令牌类型在认证层都受支持。例如，系统使用 GSSUP 令牌传输客户机的用户名和密码，系统将针对目标服务器的用户注册表进行验证。轻量级第三方认证 (LTPA) 令牌表示无需传输密码的客户机用户，但在进行远程方法调用之前，系统必须在客户机进程上认证该用户，并且客户机和服务器进程必须共享 LTPA 密钥。

对于任一令牌类型，系统使用令牌以在服务器进程上认证远程用户，并在客户机启动远程对象之前创建在客户端生效的主体集的主体集表示。如果还启用了身份断言，那么认证层可包含安全信息，此信息表示客户机身份，而身份断言令牌表示调用时的实际远程用户。

有关配置 CSIV2 认证层的信息，请参阅配置入站 *CSIv2* 认证层（见第 1387 页）或配置出站 *CSIv2* 认证层（见第 1392 页）。

CSIv2 传输层

公共安全互操作性 V2 (CSIv2) 传输层支持用于保护 SAS 协议请求消息及支持从客户机进程至服务器进程的客户机证书认证。

传输层的主要功能是在从客户机进程至服务器进程的传输过程中提供 SAS 协议消息传输的安全特征。可使用加密和/或签名来保护消息。 xigemaAS SSL 支持用作提供这类特征的底层机制。

传输层的第二个功能是在未使用认证层时提供认证材料来源。如果已启用身份断言并且未启用认证层，那么系统从传输的客户机证书链获取客户机进程身份。目标服务器进程通过将客户机证书链映射至其用户注册表中的用户来认证客户机证书链。证书链颁发者专有名称用于确定是否信任客户机断言身份。

如果未启用任何身份断言和认证层，那么在目标服务器进程上启动实际远程方法调用时，通过映射客户机证书链获取的主体集将用作调用者主体集。即使目标服务器的认证层受支持但并非必需，并且客户机未发送认证令牌和身份令牌，这一点也适用。

有关使用身份断言配置 CSIV2 属性层的信息，请参阅配置入站 *CSIv2* 传输层或配置出站 *CSIv2* 传输层。

关键术语

ORB – 对象请求代理程序。

它在可能已并置或未并置在同一进程中的实体间调解对象方法调用。

安全上下文

一些信息，用于规定对于 ORB 中对象的特定操作的安全特征。例如，要在调用对象操作期间使用的身份。

客户机安全服务 **(CSS)**

一个实体，此实体启动 SAS 协议请求，以在目标安全服务中建立针对目标 ORB 内对象的操作的安全上下文。

目标安全服务 **(TSS)**

一个实体，此实体接收 SAS 协议请求，以建立与针对目标 ORB 内对象的操作相关联的安全上下文。它接受或拒绝要建立或使用安全上下文的请求。

客户机认证

基于令牌的机制，用于认证客户机。GSSUP（用户名密码 GSS）是最低要求，但可能有其他要求，例如，LTPA。

身份断言

一种机制，中介实体通过此机制为另一实体担保，TSS 通过此机制对调用主体使用所断言身份。TSS 可决定它是否信任用于断言该身份的代理。

无状态

仅在处理单个请求期间使用该安全上下文，不会对后续请求重复使用该安全上下文。

有状态

建立该安全上下文后，多个请求可复用该安全上下文，直到 TSS 或 CSS 使其失效。

传输层安全性

底层传输提供的安全性支持。

xigemaAS 应用程序客户机容器上的认证

客户机上的认证需求与服务器上相同，但客户机上用于认证的某些机制与服务器上不同。

访问服务器上的受保护资源时，需要在客户机上认证。请使用下列其中一个方法来提供认证信息：

- 在 client.xml 文件中指定**用户**和**密码**：使用 CSIv2 协议将凭证发送至服务器，建议对密码进行加密或编码。有关进一步详细信息，请参阅在 *xigemaAS* 概要文件应用程序客户机容器中配置出站 *CSIv2* 认证层（见第 1399 页）。
- 客户机证书认证：客户机向服务器提供证书，此证书已认证并映射至注册表中的用户以用于授权检查。要配置服务器，请参阅配置入站 *CSIv2* 传输层。要配置客户机，请参阅在 *xigemaAS* 应用程序客户机容器中配置出站 *CSIv2* 传输层（见第 1400 页）。
- 执行程序化登录：程序化登录是一种表单登录类型，它支持用于认证的应用程序表示登录表单。此方法要求应用程序开发者收集用户的凭证并认证该用户。有关进一步详细信息，请参阅在 *xigemaAS* 应用程序客户机容器中配置 *JAAS* 程序化登录（见第 1396 页）。

与在服务器上一样，您可使用定制登录模块来制定更多认证决策，或向主体集添加信息以在客户机应用程序内制定更详细的授权决策。有关进一步详细信息，请参阅为 *xigemaAS* 应用程序客户机容器配置 *JAAS* 定制登录模块（见第 1398 页）。

SAML 2.0 Web 浏览器单点登录

SAML Web 浏览器单点登录 (SSO) 允许 Web 应用程序将用户认证授权给 SAML 身份提供程序（而不是所配置的用户注册表）。

安全性断言标记语言 (SAML) 是一项 OASIS 开放式标准，用于表示和交换用户身份、认证和属性信息。SAML 断言是一种 XML 格式的令牌，用于在单点登录请求的完成过程中将用户的身份提供程序中的用户身份和属性信息传输至可信服务提供程序。SAML 断言提供了一种在联合业务合作伙伴之间传输信息的方法，此方法与供应商无关。通过使用 SAML，企业服务提供程序可联系另一企业身份提供程序以认证尝试访问安全内容的用户。

xigemaAS 支持带有 HTTP 发布绑定的 SAML Web 浏览器单点登录概要文件并充当 SAML 服务提供程序。Web 用户向产生 SAML 断言的 SAML 身份提供程序认证，SAML 服务提供程序使用此 SAML 断言对 Web 用户建立安全上下文。

SAML Web SSO 流程包含三个参与者：最终用户、身份提供程序 (IdP) 和服务提供程序 (SP)。此用户始终向 IdP 认证，SP 依赖于 IdP 断言来标识该用户。

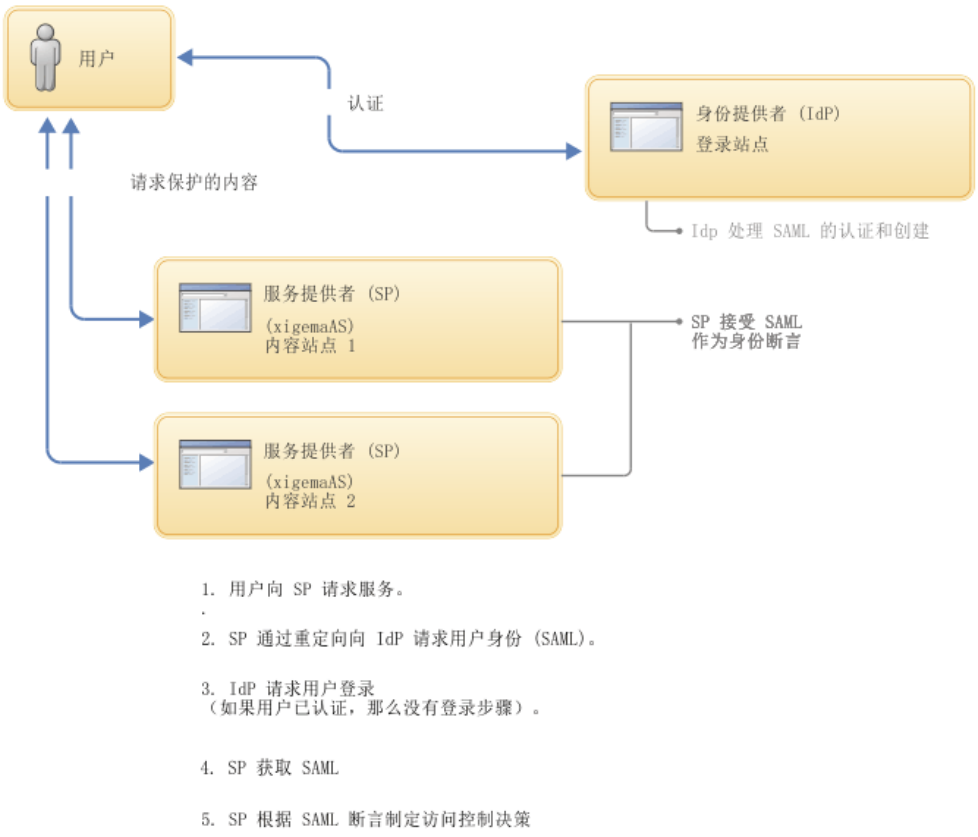


图 19: Web 单点登录中的关键概念

xigemaAS SAML Web 浏览器 SSO 方案

方案 1：SP 启动的请求式 Web SSO（最终用户在 SP 上启动）

1. 最终用户访问 SP。
2. SP 将用户重定向至 IdP。

3. 最终用户向 IdP 认证。
4. IdP 将 SAML 响应和断言发送至 SP。
5. SP 验证 SAML 响应并对用户请求授权。

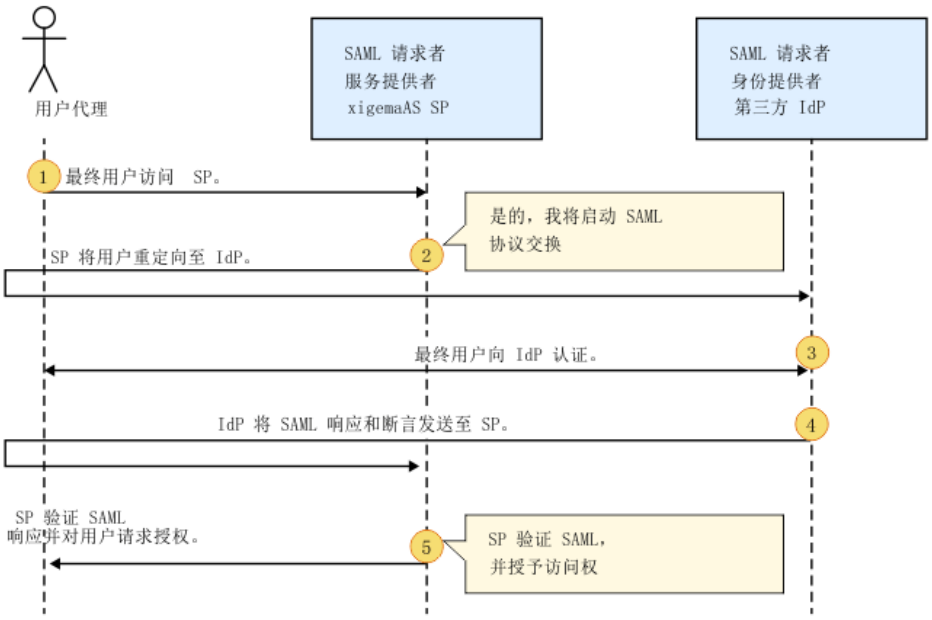


图 20: SP 启动的请求式 Web SSO

方案 2：IdP 启动的非请求式 Web SSO（最终用户在 IdP 上启动）

1. 用户代理访问 SAML IdP。
2. IdP 认证用户并发出 SAML 断言。
3. IdP 将用户重定向至 SP 并产生 SAMLResponse。
4. SP 验证 SAML 响应并对用户请求授权。

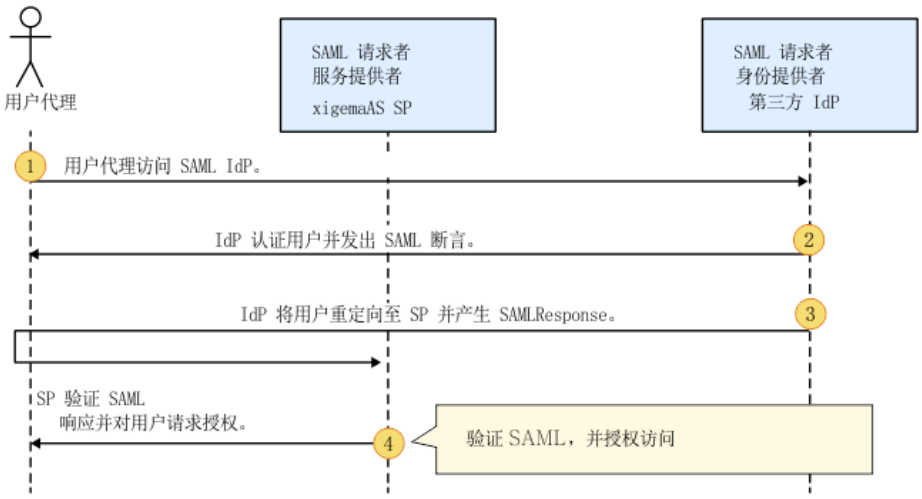


图 21: IdP 启动的非请求式 Web SSO

方案 3：OpenID Connect 提供程序和 SAML 服务提供程序

1. 最终用户访问 OpenID Connect 依赖方 (RP)。
2. RP 将最终用户重定向至 OpenID Connect 提供程序 (OP)。
3. OP（又称为 SAML SP）将最终用户重定向至 SAML IdP。
4. 最终用户向 SAML IdP 认证。
5. IdP 将最终用户重定向至 OP 或 SP 并产生 SAMLResponse。
6. OP/SP 验证 SAML，并将授权代码发送至 RP。
7. RP 针对 `id_token` 和 `access_token` 交换代码。
8. RP 验证 `id_token` 并向最终用户授权。

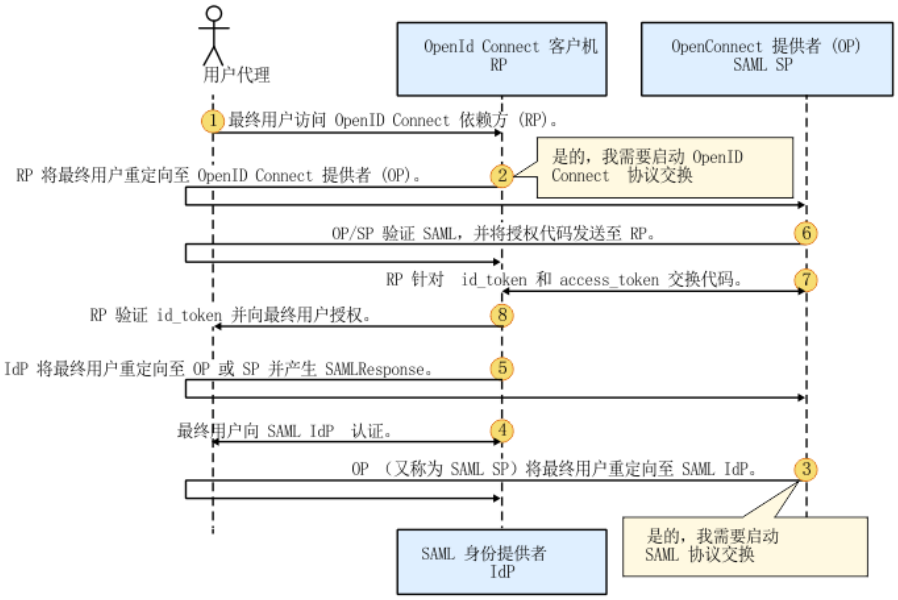


图 22: OpenID Connect 提供程序和 SAML 服务提供程序

用户注册表的联合

多个注册表中散布有 user 和 group 信息时，将使用用户注册表联合。例如，该信息可能在两个不同 LDAP 中、同一 LDAP 的两个子树中或某个文件中，或者这些用户属于某个系统。该信息甚至可能在定制用户数据存储库中。通过联合注册表，可以统一方式搜索和使用这些分布式用户信息，并不断存储这些信息。通过使用联合注册表，可在 xigemaAS 中使用统一视图来对用户进行认证和授权。

👉 注：快速启动安全性 (quickStartSecurity) 无法与其他用户注册表（例如，LDAP 注册表、基本注册表、SAF 注册表和定制注册表）联合。

如何联合用户注册表?

LDAP 注册表在缺省情况下是联合的。如果在 server.xml 中配置了多个 LDAP 注册表，那么这些注册表将自动联合，不需要附加配置。它们可以是两个独立 LDAP 或同一 LDAP 中的两个子树。

如果一个或多个 LDAP 注册表配置有一个或多个基本注册表、SAF 注册表或定制注册表，那么这些注册表也会联合而不需要任何附加配置。以下示例显示与 LDAP 联合的基本注册表：

```
<server description="Federation">
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>servlet-3.0</feature>
  <feature>ldapRegistry-3.0</feature>
    </featureManager>

    <basicRegistry id="basic" realm="SampleBasicRealm">
        <user name="admin" password="password" />
        <user name="user1" password="password" />
            <user name="user2" password="password" />
        <group name="memberlessGroup" />
        <group name="adminGroup">
            <member name="admin"/>
        </group>
        <group name="users">
            <member name="user1" />
            <member name="user2" />
        </group>
    </basicRegistry>
```

```
    <ldapRegistry id="LDAP1" realm="SampleLdapIDSRealm" host="LDAPHOST1.vsettan.com.cn" port="389"
 ignoreCase="true"
 baseDN="o=vsettan,c=cn"
 ldapType="IBM Tivoli Directory Server"
 searchTimeout="8m"
 recursiveSearch="true">
    </ldapRegistry>
</server>
```

如果要联合两个基本注册表或将基本注册表与 SAF 注册表或定制注册表联合，那么可按以下示例中所示添加 `federatedRegistry-1.0` 和 `appSecurity-2.0` 功能部件来完成任务。

```
    <feature>appSecurity-2.0</feature>
    <feature>federatedRegistry-1.0</feature>
```

以下示例显示如何联合两个基本注册表：

```
<server description="Federation">
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>federatedRegistry-1.0</feature>
    </featureManager>

    <basicRegistry id="basic1" realm="SampleBasicRealm1">
        <user name="admin" password="password" />
        <user name="user1" password="password" />
            <user name="user2" password="password" />
        <group name="memberlessGroup" />
        <group name="adminGroup">
            <member name="admin"/>
        </group>
        <group name="users">
            <member name="user1" />
            <member name="user2" />
        </group>
    </basicRegistry>

    <basicRegistry id="basic2" realm="SampleBasicRealm2">
        <user name="user3" password="password123" />
        <user name="user4" password="password123" />
        <group name="memberlessGroup2" />
        <group name="users2">
            <member name="user3"/>
            <member name="user4" />
        </group>
    </basicRegistry>
</server>
```

授权

xigemaAS 中的授权决定用户是否能够访问系统中的某个角色。

授权指定对资源的访问权。通常，在执行认证来确认身份之后，会执行授权。认证回答的问题是：“您就是您所声明的用户吗？”；而授权回答的问题是：“您有权执行您正在尝试执行的操作吗？”

下列各节详细描述了这些概念：

- 管理功能授权（见第 1055 页）
- 应用程序授权（见第 1056 页）
- 特殊主体（见第 1056 页）
- 访问标识和授权（见第 1057 页）

管理功能授权

当实体尝试访问资源时，授权服务会确定该实体是否具有必需的权限来访问资源。不管实体是访问应用程序还是执行管理功能，授权服务都会执行此操作。授权访问应用程序和授权访问管理功能的主要差别在于将用户映射到角色的方式。如果是应用程序授权，请使用 server.xml 文件或 ibm-application-bnd.xml/xmi 文件中的 `application-bnd` 元素将用户映射至角色。如果是管理功能授权，请使用 server.xml 文件中的

`administrator-role` 元素将用户映射至管理员角色。有关管理安全性的更多信息，请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）。

应用程序授权

下图描述了应用程序授权的工作原理：

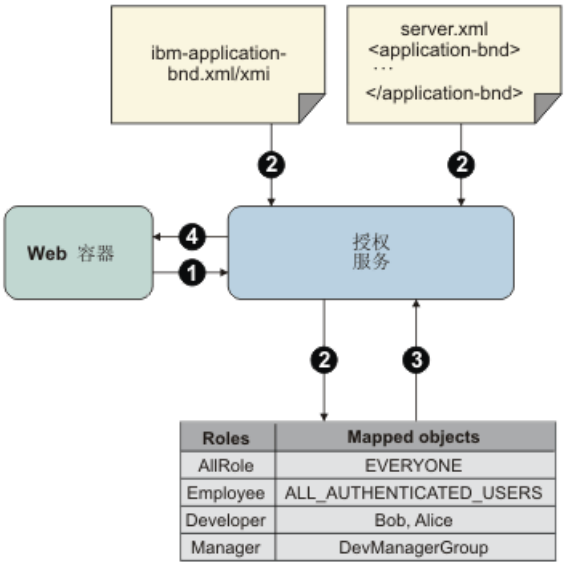


图 23: 授权过程概述

1. 当实体尝试访问 xigemaAS 所服务的应用程序中的资源时，会执行授权。Web 容器会调用授权服务来确定在给定必需角色集（包含一个或多个必需角色）的情况下，用户是否有权访问特定资源。必需角色是由部署描述符中的 `auth-constraint` 元素以及 `@ServletSecurity` 注解所确定。
2. 授权服务确定必需角色所映射到的对象。通过处理 ibm-application-bnd.xmi 文件或 ibm-application-bnd.xml 文件中以及 server.xml 文件的 `application-bnd` 元素中所定义的映射来完成此步骤。将合并来自这两个源的映射。如果两个源中存在同一角色，那么仅使用 server.xml 文件中的角色映射。使用 server.xml 文件将角色映射到用户的优点是您的应用程序不需要打包成 EAR 文件，并且更容易更新。此外，使用 ibm-application-bnd.xmi/xml 文件时，可以使您的应用程序移植到其他服务器。
3. 如果已将必需角色映射到 EVERYONE 特殊主体集，那么授权服务会立即返回以允许任何人访问。如果已将该角色映射至 ALL_AUTHENTICATED_USERS 特殊主体集并且已认证用户，那么授权服务会将为用户授予访问权。如果这些条件都未得到满足，那么授权服务会确定映射到必需角色的用户和组。如果用户已映射到必需角色或者用户属于映射到角色的组，那么授权服务会授予资源访问权。
4. 授权服务会将结果返回到 Web 容器以指示是已授予用户访问权还是已拒绝用户访问。

特殊主体

将实体映射到角色时，可以映射特殊主体，而不是特定用户或组。特殊主体是主体概念的扩展。特殊主体可以表示归为特定类别的用户组。

可以使用下列两种特殊主体：

- `EVERYONE`：表示系统上的任何实体，这意味着任何安全性都不适用，因为每个人都可以访问，而且不会提示您输入凭证。

- ALL_AUTHENTICATED_USERS：表示向服务器成功认证的所有实体。

要将特殊主体映射至用户，请更新 ibm-application-bnd.xmi/xml 文件或 server.xml 文件，其中 application-bnd 在 application 元素下。在此示例中，名为 AllAuthenticated 的角色映射至特殊主体 ALL_AUTHENTICATED_USERS：

```
    <application-bnd>
           <security-role name="AllAuthenticated">
               <special-subject type="ALL_AUTHENTICATED_USERS" />
           </security-role>
       </application-bnd>
```

请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。

访问标识和授权

对用户或组授权时，服务器需要一种方法来唯一标识该用户或组。用户和组的唯一标识可用作此用途，并且用来构建授权配置。由用户注册表实现来确定这些标识：唯一用户标识是 getUniqueUserId() 的值，唯一组标识是 getUniqueGroupId() 的值。您也可以选择在授权配置中对用户或组显式地指定访问标识。将使用这些显式访问标识，而不使用用户注册表实现所返回的值。要在 ibm-application-bnd.xml/xmi 文件或 server.xml 文件中指定访问标识（其中 application-bnd 在 application 元素下），请对 user 或 group 元素使用 access-id 属性。

在此示例中，会对用户 Bob 和组 developers 指定访问标识：

```
    <application-bnd>
           <security-role name="Employee">
               <user name="Bob" access-id="user:MyRealm/Bob"/>
               <group name="developers" access-id="group:myRealm/developers"/>
           </security-role>
    </application-bnd>
```

注：access-id 属性用于授权检查。如果未指定此属性，那么系统从通过使用该用户或组名配置的注册表确定此属性。但是，用户或组不属于活动注册表时，必须按示例中所示指定 access-id 属性。例如，使用程序化登录时。

xigemaAS 应用程序客户机容器上的安全性

xigemaAS 应用程序客户机容器上的安全性包括 SSL、JAAS 和 CSIv2。

应用程序客户机是在自己的 Java™ 虚拟机中运行的客户机程序。xigemaAS 应用程序客户机容器为这些客户机提供系统服务，包括安全性。客户机上的安全服务是服务器上可用的安全服务的子集。

在客户机上启用安全性

要在客户机上启用安全性，请将 appSecurityClient-1.0 功能部件添加至 client.xml 文件。

```
<featureManager>
       <feature>javaeeClient-7.0</feature>
       <feature>appSecurityClient-1.0</feature>
</featureManager>
```

appSecurityClient-1.0 功能部件在客户机上启用 SSL、CSIv2 和 JAAS。必须配置 SSL 以确保客户机与服务器之间的通信是安全的和加密的。有关更多信息，请参阅对 *xigemaAS* 应用程序客户机容器启用 *SSL* 通信（见第 1395 页）。CSIv2 为客户机提供用于将认证信息发送至服务器的协议。xigemaAS 概要文件应用程序客户机容器中的客户机无法断言身份或传播安全性属性。要了解有关 CSIv2 及如何在客户机上配置

CSIv2 的更多信息，请参阅公共安全互操作性 *V2 (CSIv2)* 和在 *xigemaAS* 应用程序客户机容器中配置公共安全互操作性 *V2 (CSIv2)*（见第 1399 页）。客户机上的 JAAS 框架允许客户机应用程序从使用回调的用户处收集凭证并使用登录模块认证该用户。有关在客户机上认证用户的更多信息，请参阅*xigemaAS* 应用程序客户机容器上的认证（见第 1050 页）。

xigemaAS：Java™ 2 安全性

Java™ 2 安全性功能在 xigemaAS 中受支持。Java™ 2 安全性提供基于策略的细颗粒度访问控制机制，此机制通过在允许对某些受保护的系统资源进行访问前检查许可权来提高整体系统完整性。

Java™ 2 安全性独立于 Java™ Platform Enterprise Edition 的基于角色的授权。Java™ 2 安全性保护对系统资源（例如，文件输入和输出、套接字和属性）的访问；而 Java™ Platform Enterprise Edition 安全性保护对 Web 资源（例如，servlet 和 JSP 文件）的访问。

用于部署者和管理员的 Java™ 2 安全性

启用 Java™ 2 安全性之前，需要确保所有应用程序被授予所需许可权，否则，应用程序可能运行失败。缺省情况下，系统会针对每个 Java™ Platform Enterprise Edition 7.0 规范向应用程序授予许可权。如果应用程序未准备好使用 Java™ 2 安全性 或者应用程序提供程序未在应用程序中提供 permissions.xml 文件，那么启用 Java™ 2 安全性时，该应用程序可能导致运行时发生 Java™ 2 安全性 访问控制异常。即使应用程序正在运行，它也可能不会正常运行。

用于应用程序开发者的 Java™ 2 安全性

应用程序开发者必须了解 Java™ SDK API 的许可权需求。您需要知道应用程序调用的 API 是否需要额外许可权。有关哪些 Java™ API 需要许可权的更多信息，请参阅 *Java™ 2 SDK* 中的许可权。

许可权是通过 permissions.xml 文件添加至应用程序的，与所列示许可权相关联的代码库基于该文件的位置。对于独立 .war 应用程序，permissions.xml 文件捆绑在 META-INF 目录中，并且所有指定许可权适用于 .war 文件中包含的所有模块。对于 .ear 应用程序，permissions.xml 直接捆绑在 .ear 本身的 META-INF 目录下，指定许可权适用于 .ear 文件中包含的所有模块。

👉 注：对于 .ear 应用程序，捆绑在 .ear 以外的任何模块的 META-INF 目录下的 permissions.xml 文件被忽略。

启用 Java™ 2 安全性

Java™ 2 安全性功能包含在内核扩展中，在引导时通过使用 `websphere.java.security` 属性更新 bootstrap.properties 文件启用。

如果在 bootstrap.properties 文件中指定了 `websphere.java.security` 属性，那么系统会强制实施 Java™ 2 安全性；否则，不会进行许可权检查。

指定受限许可权

xigemaAS 提供一种机制以在运行 Web 或 EJB 应用程序组件时指定受限许可权。受限许可权确保不向捆绑软件或应用程序授予该许可权的任何实例。它们提供一种机制以阻止应用程序向其自身授予必须允许的许可权之外的其他许可权，例如，退出 VM 的许可权。

受限许可权是在 server.xml 文件和 client.xml 文件中指定的。以下示例显示如何限制用于编写系统属性 `os.name` 的 `PropertyPermission`。此语法在 server.xml 文件和 client.xml 文件中完全相同：

```
<javaPermission className="java.security.PropertyPermission" name="os.name" actions="write"
 restriction="true" />
```

授予许可权

OSGi 捆绑软件可自我调整通过 permissions.perm 文件向该捆绑软件内的库/类授予的许可权。

应用程序也可自我调整通过 permissions.xml 文件授予或通过在 server.xml 文件和 client.xml 文件中指定许可权授予来授予的许可权。

OSGi 捆绑软件许可权

OSGi 规范提供一种机制以通过捆绑软件的 OSGI-INF 目录中的 permissions.perm 文件指定对捆绑软件的许可权。此机制允许对捆绑软件的许可权进行细颗粒度访问控制。

permissions.perm 文件指定捆绑软件需要的最大许可权。

👉 重要：空 permissions.perm 文件不等于没有 permissions.perm 文件。如果您想要获取受限许可权，请确保您具有非空 permissions.perm 文件。

在 server.xml 和 client.xml 中对应用程序声明许可权

server.xml 文件和 client.xml 文件中定义的不带指定代码库的许可权适用于该 xigemaAS 服务器上的所有应用程序。

可按以下示例中所示在 server.xml 文件和 client.xml 文件中指定要授予的许可权：在此示例中，授予允许读取所有系统属性的 `PropertyPermission`：

```
<javaPermission className="java.util.PropertyPermission"  name="*" actions="read" />
```

可在 server.xml 文件和 client.xml 文件中指定要限制的许可权。以下示例显示如何限制用于编写系统属性 `os.name` 的 `PropertyPermission`。此语法在 server.xml 文件和 client.xml 文件中完全相同：

```
<javaPermission className="java.security.PropertyPermission" name="os.name"
 actions="write" restriction="true" />
```

👉 注：

- 受限许可权的 **`restriction`** 设置为 *true*。
- 如果应用程序尝试向其自身授予定义为受限许可权的许可权，那么受限许可权优先于授权并且不允许授权。

在 permissions.xml 中对应用程序声明许可权

permissions.xml 文件是 Java™ EE7 规范引入的新文件。它被打包在应用程序目录的 META-INF 目录下。

对于打包为独立 .war 文件的应用程序，在 META-INF WAR 级别指定的许可权适用于打包在 .war 文件中的所有模块和库。

对于打包在 .ear 文件中的应用程序，必须在 .ear 文件级别声明许可权。此许可权集适用于打包在 .ear 文件或其所包含模块中的所有模块和库。这类打包模块中的任何 permissions.xml 文件将被忽略，不管是否对 .ear 文件提供了 permissions.xml 文件都是如此。

对于打包在 .rar 文件中的应用程序，必须在 META-INF RAR 级别声明许可权。

no-rethrow 选项

启用 Java™ 2 安全性后，发生许可权违例时，缺省情况下 JDK 安全管理器会抛出 `java.security.AccessControl` 异常。如果不处理此异常，那么可能导致运行时失败。如果开发者在准备应用程序以使用 Java™ 2 安全性时需要帮助，可使用了 no-rethrow 选项。no-rethrow 选项允许将

AccessControl 异常记录在 console.log 和 messages.log 中而不会导致应用程序失败。no-rethrow 选项是通过在 bootstrap.properties 文件中指定 websphere.java.security.norethrow=true 启用的。缺省情况下未启用 no-rethrow 选项，因此，必须在 bootstrap.properties 文件中指定它以启用此属性。

注：因为此属性不允许安全管理器抛出该异常，所以从技术上讲，安全管理器不会强制实施 Java™ 2 安全性。不能在生产环境中使用 no-rethrow 属性。

动态更新

系统不支持对 permissions.perm、permissions.xml、server.xml 和 client.xml 之类的许可权文件的动态更新。对许可权的更新要求重新启动 xigemaAS 服务器。

安全性公共 API

xigemaAS 中的安全性公用 API 提供了一种扩展安全性基础结构的方法。

xigemaAS 包含可用来实现安全性功能的公用 API。主要类是 WSSecurityHelper、WSSubject 和 RegistryHelper。此外，还有一个新类 WebSecurityHelper。

下列部分描述了这些主要类。还有其他类，例如 UserRegistry、WSCredential 和其他异常类。

xigemaAS 支持的所有安全性公用 API 都在 Java™ API 文档中。每个 xigemaAS API 的 Java™ API 文档在 ${wlp.install.dir}/dev 目录的某个 javadoc 子目录下的单独 .zip 文件中提供。

WSSecurityHelper

此类仅包含方法 isServerSecurityEnabled() 和 isGlobalSecurityEnabled()。如果此外还启用了 appSecurity-2.0，那么这些调用返回 true。否则，这两个方法都将返回 false。

注：

- xigemaAS 中没有单元，所以 xigemaAS 在全局安全性和服务器安全性之间没有差别。因此，两个方法返回相同值。
- xigemaAS 中不支持方法 revokeSSOCookies(javax.servlet.http.HttpServletRequest req,javax.servlet.http.HttpServletResponse res)。
- getLTPACookieFromSSOToken() 方法已重命名为新的公用 API 类：WebSecurityHelper。

WSSubject

此类提供实用程序方法来查询和设置安全性线程上下文。

注：Java™ 2 安全性在 xigemaAS 概要文件中受支持，但缺省情况下未启用。所以，在缺省情况下，不会在 WSSubject 中执行 Java™ 2 安全性检查。

RegistryHelper

此类提供对 UserRegistry 对象及可信域信息的访问权。

WebSecurityHelper

此类包含重命名的 getLTPACookieFromSSOToken() 方法，该方法已从 WSSecurityHelper 中移出：

```
public static Cookie getLTPACookieFromSSOToken() throws Exception
```

安全性公用 API 代码示例

以下示例说明如何在 xigemaAS 中使用安全性公用 API 来执行程序化登录并操作 Subject 。

- 示例 *1:* 创建主体并将其用于授权
- 示例 *2:* 创建主体并使其成为线程上的当前 *Subject*
- 示例 *3:* 获取线程上当前主体的信息

示例 **1:** 创建主体并将其用于授权

此示例说明如何使用 `WSSecurityHelper`、`WSSubject` 和 `UserRegistry` 来执行程序化登录以创建 Java™ 主体，然后执行操作并将该主体用于任何需要的授权。

注：下列代码在执行进一步的安全处理之前使用 `WSSecurityHelper` 来检查是否启用了安全性。此检查由于 xigemaAS 的模块化性质而得到广泛使用：如果未启用安全性，那么不会载入安全性运行时。`WSSecurityHelper` 总是会载入，即使安全性并未启用也是如此。

```
import java.rmi.RemoteException;
import java.security.PrivilegedAction;

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.CustomRegistryException;
import com.ibm.websphere.security.UserRegistry;
import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
import com.ibm.wsspi.security.registry.RegistryHelper;
public class myServlet {

 ...
 if (WSSecurityHelper.isServerSecurityEnabled()) {
  UserRegistry ur = null;
  try {
   ur = RegistryHelper.getUserRegistry(null);
   } catch (WSSecurityException e1) {
      // record some diagnostic info
   return;
  }
   String userid = "user1";
   String password = "user1password";
  try {
   if (ur.isValidUser(userid)) {
    // create a Subject, authenticating with
    // a userid and password
    CallbackHandler wscbh = new WSCallbackHandlerImpl(userid, password);
    LoginContext ctx;
    ctx = new LoginContext("WSLogin", wscbh);
    ctx.login();
    Subject subject = ctx.getSubject();
    // Perform an action using the Subject for
    // any required authorization
    WSSubject.doAs(subject, action);
   }
  } catch (CustomRegistryException e) {
   // record some diagnostic info
   return;
  } catch (RemoteException e) {
   // record some diagnostic info
   return;
   } catch (LoginException e) {
   // record some diagnostic info
   return;
  }
 }
 ...
```

```
  private final PrivilegedAction action = new PrivilegedAction() {
  @Override
   public Object run() {
         // do something useful here
   return null;
  }
 };

}
```

示例 **2:** 创建主体并使其成为线程上的当前 **Subject**

以下示例说明如何使用 `WSSecurityHelper` 和 `WSSubject` 来执行程序化登录以创建 Java™ 主体，使该主体成为线程上的当前主体，最后复原原始安全性线程上下文。

👉 注**:** 下列代码在执行进一步的安全处理之前使用 `WSSecurityHelper` 来检查是否启用了安全性。

```
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
...
if (WSSecurityHelper.isServerSecurityEnabled()) {
 CallbackHandler wscbh = new WSCallbackHandlerImpl("user1", "user1password");
  LoginContext ctx;
         try {
          // create a Subject, authenticating with
          // a userid and password
   ctx = new LoginContext("WSLogin", wscbh);
   ctx.login();
    Subject mySubject = ctx.getSubject();
    Subject oldSubject = null;
          try {
    // Save a ref to the current Subject on the thread
    oldSubject = WSSubject.getRunAsSubject();
    // Make mySubject the current Subject on the thread
    WSSubject.setRunAsSubject(mySubject);
    // Do something useful here. Any authorization
          // required will be performed using mySubject
    } catch (WSSecurityException e) {
     // record some diagnostic info
     return;
  } finally {
    // Put the original Subject back on the thread context
    if (oldSubject != null) {
            try {
     WSSubject.setRunAsSubject(oldSubject);
      } catch (WSSecurityException e) {
      // record some diagnostic info
                  }
           }
  }
  } catch (LoginException e) {
  // record some diagnostic info
  return;
 }
}
```

示例 **3:** 获取线程上当前主体的信息

以下示例说明如何使用 `WSSecurityHelper`、`WSSubject` 和 `WSCredential` 来获取有关线程上当前主体的信息。

👉 注：下列代码在执行进一步的安全处理之前使用 WSSecurityHelper 来检查是否启用了安全性。

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Set;

import javax.security.auth.Subject;
import javax.security.auth.login.CredentialExpiredException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.CredentialDestroyedException;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.cred.WSCredential;
...
 if (WSSecurityHelper.isServerSecurityEnabled()) {
  // Get the caller's subject
  Subject callerSubject;
 try {
    callerSubject = WSSubject.getCallerSubject();
  } catch (WSSecurityException e) {
     // record some diagnostic info
  return;
 }
  WSCredential wsCred = null;
  Set<WSCredential> wsCredentials =
    callerSubject.getPublicCredentials(WSCredential.class);
  Iterator<WSCredential> wsCredentialsIterator = wsCredentials.iterator();
  if (wsCredentialsIterator.hasNext()) {
   wsCred = wsCredentialsIterator.next();
          try {
    // Print out the groups
    ArrayList<String> groups = wsCred.getGroupIds();
    for (String group : groups) {
    System.out.println("Group name: " + group);
           }
          } catch (CredentialExpiredException e) {
      // record some diagnostic info
      return;
    } catch (CredentialDestroyedException e) {
      // record some diagnostic info
      return;
           }
          }
        }
}
```

通过密码加密进行保护时存在的限制

xigemaAS 概要文件支持对存储在 server.xml 文件中的密码进行高级加密标准 (AES) 或国密算法 (SM4) 加密。当您使用此选项来保护 xigemaAS 概要文件配置中的系统密码时，您需要了解它所提供的保护存在的限制。

对 xigemaAS 概要文件配置中的密码进行加密并不保证密码是安全的或者受保护的；它仅仅意味着可以查看加密密码、但是不知道加密密钥的人员无法轻易恢复密码。应用程序服务器流程要求访问加密密码和解密密钥，因此，这些数据项都需要存储在服务器运行时环境可访问的文件系统上。对放置在服务器配置中的密码进行加密的人员也需要加密密钥。对于能够访问与 xigemaAS 概要文件服务器实例所访问的完全相同的一组文件的攻击者，对密码应用 AES 加密或 SM4 加密，因此，除了“异或”(XOR) 编码以外，未提供其他安全性。

但是，还有一些原因会导致您可能考虑对 xigemaAS 概要文件配置中的密码进行加密。xigemaAS 概要文件配置设计为高度可组合且可共享， 任何管理员都可以看到任何采用 XOR 编码的密码。对于这些设计功能部件，考虑下列情况：

- 密码不具有机密性，因此对它们进行编码几乎没有价值。
- 密码具有机密性，因此，包含密码的配置文件对安全性敏感并且需要控制访问权，或者已对密码进行加密，并且编码密钥因对安全性敏感而受到保护。

可以通过设置 `wlp.password.encryption.key` 或 `wlp.password.encryption.sm4.key` 属性来覆盖用于解密的缺省加密密钥。不应在用于存储密码的 server.xml 文件中设置此属性，而是在 server.xml 文件所引用的独立配置文件中设置。此独立配置文件应当只包含单个属性声明，并且应当存储在服务器的正常配置目录外部。这将确保当您运行服务器转储或打包命令时不引用包含此密钥的文件。还可以将加密密钥属性指定为引导属性。如果您选择此选项，那么应将加密密钥放入服务器 bootstrap.properties 文件中所引用的独立属性文件中。

有关使用 XOR 、 AES 或 SM4 来保护密码的信息，请参阅相关链接，尤其是 *securityUtility* 命令（见第 1287 页）。

2.1.8 JMS 消息传递

xigemaAS 概要文件支持异步消息传递作为基于 Java™ 消息服务 (JMS) 编程接口的通信方法。JMS 接口为 Java™ 程序（客户机和 Java™ EE 应用程序）提供了创建、发送、接收和读取异步请求（作为 JMS 消息）的公共方法。使用 xigemaAS，可以配置多个 JMS 消息传递提供程序，这些提供程序可供 JMS 应用程序使用。

在 xigemaAS 概要文件中，支持以下 JMS 消息传递提供程序：

- 作为 JMS 消息传递提供程序的 xigemaAS 嵌入式消息传递引擎
- xigemaMQ 消息传递提供程序，它将 xigemaMQ 系统用作提供者

您的应用程序可以使用来自这两个 JMS 提供者的消息传递资源。根据有关消息传递的特定需求或者是否要与现有消息传递系统进行集成来选择 JMS 提供者。例如，您可能想让应用程序在本地连接至 xigemaAS 消息传递提供程序，而不需要配置任何外部消息传递提供程序，在这种情况下，您会使用 xigemaAS 嵌入式消息传递引擎。此外，您可能想要与现有消息传递基础结构（例如，xigemaMQ）进行集成。在这种情况下，可以使用 xigemaMQ 消息传递提供程序直接进行连接。

xigemaAS JMS 消息传递提供程序

xigemaAS 支持不同 JMS 消息传递提供程序。

xigemaAS 嵌入式 JMS 消息传递提供程序

xigemaAS 消息传递是 xigemaAS 中的一个嵌入式消息传递功能部件。它是在 xigemaAS 中运行的一个可组合且灵活的动态 JMS 消息传递引擎。xigemaAS 消息传递同时符合 JMS 1.1 和 JMS 2.0 规范，并且同时支持点到点消息传递模型和发布/预订消息传递模型。

xigemaAS 消息传递仅在 xigemaAS 运行时运行，并且您可以使用 xigemaAS 功能部件管理器，根据需要来启用或禁用消息传递功能部件。因为消息传递运行时高度可组合，所以您可以对运行时启用基本消息传递功能部件，并且可以根据需求来动态启用更多消息传递功能部件（例如，安全性、事务和远程通信）。

xigemaAS 消息传递可以分类为两个部件：

- JMS 服务器运行时：为连接、事务、持久性和安全性等提供所有运行时功能。
- JMS 客户机连接：提供资源适配器支持以允许 JMS 客户机执行同步和异步消息传递活动。

消息传递引擎作为 xigemaAS 中的单实例运行，这意味着在任何特定时间，xigemaAS 内核中只能有一个消息传递引擎处于运行状态。

xigemaAS 消息传递体系结构

xigemaAS 消息传递具有高度可组合和动态的性质。xigemaAS 消息传递包含若干其他内部消息传递子组件（这些子组件是作为 OSGi 捆绑包实现的），可以根据用户需求来启用或禁用这些子组件。使用 OSGi 服务来管理组件生命周期，以及管理依赖性和配置的注入过程。

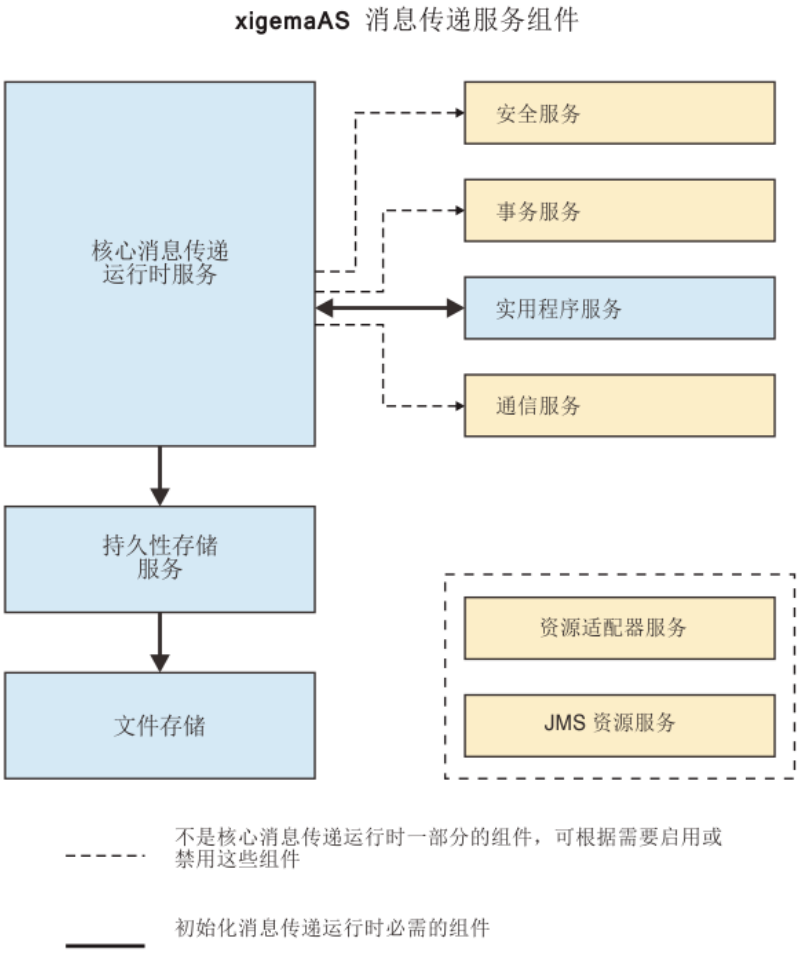


图 24: xigemaAS 消息传递体系结构

消息传递运行时和其他消息传递子组件作为 OSGi 捆绑软件在 OSGi 框架中运行。这可让 xigemaAS 内核根据使用情况来装入或卸载消息传递捆绑软件。例如，如果用户未使用消息传递安全性，那么不会初始化与消息传递安全性相关的捆绑软件。

应用程序部署

xigemaAS 消息传递支持三种类型的 JMS 应用程序连接。应用程序可以采用下列任何方式运行：

- 在主管消息传递引擎的 xigemaAS 中运行。
- 在另一个未主管任何消息传递引擎的 xigemaAS 中运行。
- 在其他未主管任何消息传递引擎的服务器中运行。

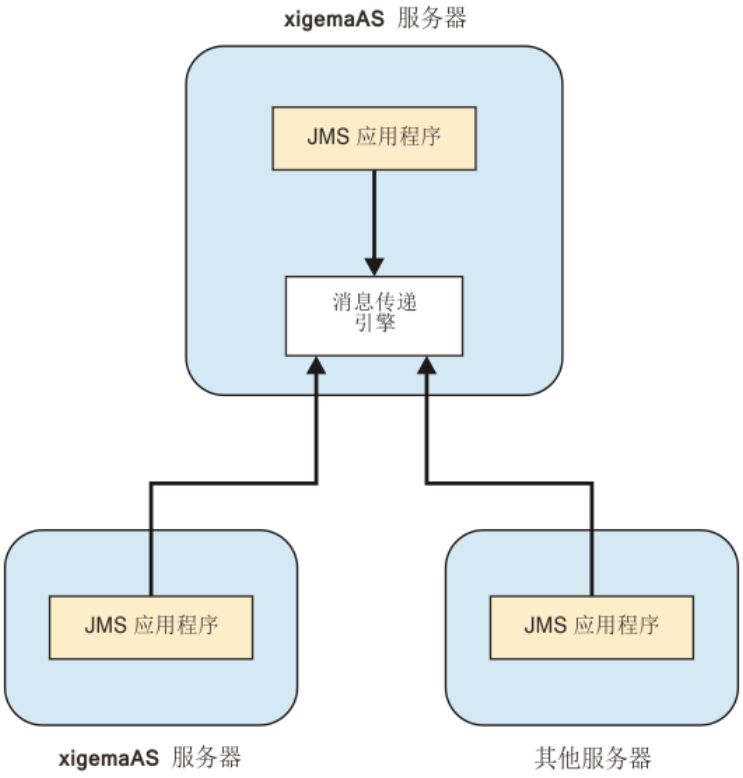


图 25: 应用程序部署模型

xigemaAS 消息传递支持应用程序进行进程内及网络 *TCP/IP* 连接。如果 JMS 应用程序部署在正运行消息传递引擎的同一个 JVM 中，那么该应用程序可与进程内消息传递引擎进行通信，而不需要通过 TCP/IP 层进行通信。这会为应用程序提供显著的性能优势来收发消息。

如果 JMS 应用程序正在一个未主管消息传递引擎的 xigemaAS 上运行，那么该应用程序必须通过 TCP/IP 进行连接，以便与消息传递引擎进行通信。

消息处理

目标（队列或主题）总是定位到其中定义了目标的消息传递引擎。如果应用程序需要将消息发送到目标或从目标接收消息，那么它必须总是连接至可以定位目标的消息传递引擎。

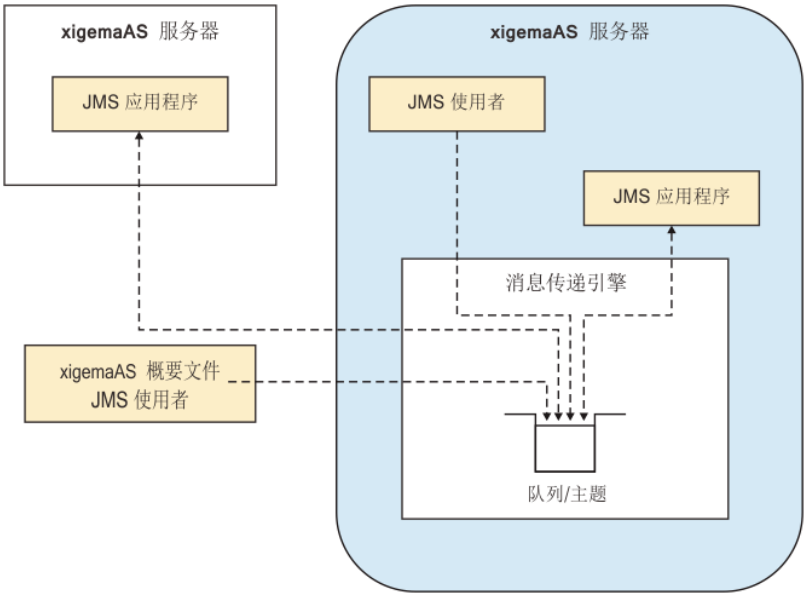


图 26: xigemaAS 消息传递中的消息处理

xigemaMQ 消息传递提供程序

通过 xigemaAS 中的 xigemaMQ 消息传递提供程序，Java™ 消息服务 (JMS) 消息传递应用程序可以使用 xigemaMQ 系统作为 JMS 消息传递资源的外部提供者。xigemaMQ 同时符合 JMS 1.1 和 JMS 2.0 规范。

可以配置应用程序的 xigemaMQ 资源（例如，队列连接工厂）以及管理与 xigemaAS 中的 JMS 目标相关联的消息和预订。可以通过 xigemaMQ 来管理安全性。在 xigemaMQ 中可能需要其他配置，这视应用程序所使用的资源而定。

xigemaMQ 的特征为如下所示：

- 消息传递由队列管理器网络进行处理，每个队列管理器都在自己的进程集中运行，并且可以自行管理。
- 许多 Vsettan 产品和 Vsettan 合作伙伴产品支持具有监视和控制，以及高可用性的 xigemaMQ。
- xigemaMQ 客户机可在 xigemaAS (JMS) 或者几乎所有使用各种 API 的消息传递环境中运行。
- xigemaMQ 客户机可以在各种各样的使用各种 API 的环境、平台和操作系统中运行。通过 xigemaMQ 资源适配器进行集成即可连接至其他 Java™ EE 系统。

👉 注：在xigemaAS 中部署的 xigemaMQ 功能部件具有下列限制：

- xigemaMQ 消息传递功能部件中不包括 xigemaMQ Java™ 类（通常称为“基本 Java™”）。（这包括在其他应用程序服务器的资源适配器中，但是建议不要将其用于 Java™, Enterprise Edition 环境中的基本 Java™ API。）
- xigemaMQ 资源适配器具有传输类型 BINDINGS_THEN_CLIENT。在 xigemaMQ 消息传递功能部件中不支持此传输类型。
- xigemaMQ 消息传递功能部件中不包括高级消息传递安全性 (AMS) 功能部件。

JMS 消息传递 (wasJmsClient-2.0) 行为更改

如果在 JMS 消息传递应用程序中使用 `wasJmsClient-1.1` 功能部件，那么可继续使用 `wasJmsClient-1.1` 以避免迁移应用程序。如果要创建新的 JMS 消息传递应用程序，那么

可使用 `wasJmsClient-2.0` 功能部件以利用新的可用功能。如果要迁移现有应用程序以使用 `wasJmsClient-2.0` 功能部件，请注意影响 Java Platform Enterprise Edition 7 中的某些 API 的行为更改。

下表显示这些 API 的行为在两个版本之间的差别：

表 10: `wasJmsClient-1.1` 与 `wasJmsClient-2.0` 之间的行为差别

接口名称	异常的条件	**wasJmsClient-1.1**	**wasJmsClient-2.0**
Session.createDurableSubscriber	未设置 ClientID	InvalidClientIDException	IllegalStateException
TopicSession.createDurableSubscriber	未设置 ClientID	InvalidClientIDException	JMSException
Connection.setClientID	具有相同 ClientID 的另一连接已处于活动状态	IllegalStateException	InvalidCliendIDException

2.1.9 Java™ Persistence API (JPA)

应用程序可通过数据持久性在非易失性存储系统中保存和检索信息。对于企业应用程序而言，由于需要访问关系数据库，因此持久性至关重要。为此环境开发的应用程序必须自行管理持久性或使用第三方解决方案，以使用持久性来处理数据库更新和检索。自 EJB 3.0 规范开始，Java™ Persistence API (JPA) 提供了一种机制来管理持久性及对象关系映射和功能。

JPA 规范在内部定义了对象关系映射，而不依赖于特定于供应商的映射实现。JPA 基于适用于 Java™ Enterprise Edition (Java EE) 环境的 Java™ 编程模型，但 JPA 可以在 Java™ SE 环境中工作以测试应用程序功能。

JPA 表示持久性编程模型的简化。JPA 规范显式地定义了对象关系映射，而不依赖于特定于供应商的映射实现。JPA 通过使用注解或 XML 将对象映射至数据库的一个或多个表，对对象关系映射的重要任务进行了标准化。为了进一步简化持久性编程模型：

- EntityManager API 可以在数据库中保存、更新、检索或移除对象。
- EntityManager API 和对象关系映射元数据可处理大多数数据库操作，而不需要您编写 JDBC 或 SQL 代码来维护持久性。
- JPA 提供了查询语言，对独立 EJB 查询语言进行了扩展（扩展后的语言又称为 JPQL），您可以使用 JPQL 来检索对象，而不必针对您所使用的数据库编写 SQL 查询。

JPA 设计为可在 Java™ Enterprise Edition (Java™ EE) 容器的内外进行操作。当您在容器中运行 JPA 时，应用程序可使用该容器来管理持久性上下文。 如果没有容器管理 JPA，那么应用程序必须自行管理持久性上下文。为容器管理的持久性设计的应用程序不需要这么多的代码实现即可处理持久性，但这些应用程序无法在容器外部使用。管理它们自己的持久性的应用程序可在容器环境或 Java™ SE 环境中工作。

支持 EJB 3.x 编程模型的 Java™ EE 容器必须支持 JPA 实现，后者也称为持久性提供程序。JPA 持久性提供程序使用以下元素以便能够简化 EJB 3.x 环境中的持久性管理：

持久性单元

使用关系数据库定义完整的对象关系模型映射 Java™ 类（实体 + 支持结构）。EntityManagerFactory 使用此数据创建可通过 EntityManager 来访问的持久性上下文。

EntityManagerFactory

用来为数据库交互创建 EntityManager。应用程序服务器容器通常提供此功能，但如果您使用的是 JPA 应用程序管理的持久性，那么 EntityManagerFactory 是必需的。EntityManagerFactory 的实例表示持久性上下文。

持久性上下文

定义应用程序当前正在操作的一组活动实例。可手动创建持久性上下文或通过注入创建。

EntityManager

该资源管理器维护应用程序正在使用的活动实体对象集合。EntityManager 处理对象关系映射的数据库交互和元数据。EntityManager 的实例表示持久性上下文。容器中的应用程序可通过注入到应用程序或通过在 Java™ 组件命名空间中查询 EntityManager 来获取该 EntityManager。如果由该应用程序管理它自己的持久性，那么将从 EntityManagerFactory 获取 EntityManager。

实体对象

一个简单的 Java™ 类，以最简单的形式表示数据库表中的一行。实体对象可以是具体类，也可以是抽象类。它们通过使用属性或字段来维护状态。

Java™ Persistence API (JPA) 功能部件概述

可对您的应用程序使用两个 JPA 功能部件。`jpa-2.0` 基于 Apache OpenJPA 开放式源代码项目。`jpa-2.1` 基于 EclipseLink 开放式源代码项目。

- *jpa-2.0*
- *jpa-2.1*
- *JPA 功能部件兼容性*

jpa-2.0

Java™ Persistence API (JPA) 2.0 for xigemaAS 基于 Apache OpenJPA 2.2.x 开放式源代码项目。

Apache OpenJPA 是符合 JPA 1.0 和 2.0 规范的实现。通过将 OpenJPA 用于基本实现，xigemaAS 采用扩展来为 xigemaAS 客户提供其他功能部件和实用程序。由于 JPA for xigemaAS 根据 OpenJPA 构建，因此所有 OpenJPA 功能、扩展和配置均不受 xigemaAS 扩展的影响。您不必对 OpenJPA 应用程序进行任何更改，即可在 xigemaAS 中使用这些应用程序。

JPA for xigemaAS 提供了与 OpenJPA 的更广泛兼容性。JPA for xigemaAS 包含一组适用于应用程序开发和部署的工具。 JPA for xigemaAS 的其他功能部件包括对 DB2®Optim pureQuery Runtime、DB2® 优化、JPA 访问意向、增强跟踪功能、命令脚本和已翻译消息文件的支持。此产品的 JPA 的提供程序为 `com.ibm.websphere.persistence.PersistenceProviderImpl`。

Apache OpenJPA 支持使用属性来配置持久性环境。可使用 `openjpa` 或 `wsjpa` 前缀指定 JPA for xigemaAS 属性。可根据需要对公共属性集同时使用 `openjpa` 和 `wsjpa` 前缀。此规则的例外是特定于 `wsjpa` 的配置属性，这些属性使用 `wsjpa` 前缀。将特定于 JPA for xigemaAS 的属性与 `openjpa` 前缀配合使用时，将记录一条警告消息，指示违例属性被视为 `wsjpa` 属性。对于 `openjpa` 前缀，反之则不然。在这种情况下，将忽略违例属性。

jpa-2.1

Java™ Persistence API (JPA) 2.1 for xigemaAS 基于 EclipseLink 开放式源代码项目。EclipseLink 是所有版本的 JPA 规范的引用实现。此产品的 JPA 的提供程序为 `org.eclipse.persistence.jpa.PersistenceProvider`。

JPA 2.1 规范添加了 JPA 2.0 规范中未提供的新功能部件。这些功能部件包括：

- 模式生成
- 类型转换方法
- 查询和查找操作中的实体图形
- 未同步化持久性上下文
- 存储过程调用
- 注入至实体侦听器类
- JPQL 增强功能
- Criteria API 增强功能
- 本机查询的映射

有关这些功能部件的更多详细信息，请参阅 *JPA 2.1 规范*。此产品还提供了一部分 EclipseLink API。请参阅 xigemaAS 功能部件页面 *Java Persistence API 2.1* 以了解详细信息。

👉 注：JPA 2.1 向下兼容 JPA 2.0。

JPA 功能部件兼容性

`jpa-2.0`

`jpa-2.0` 功能部件是 JPA 2.0 规范实现，由 Apache OpenJPA 提供支持。此功能部件是 Java Platform Enterprise Edition (Java EE) 6 系列技术的一部分，但它很特殊，因为它与其他 Java EE 7 功能部件兼容。例如，`servlet-3.1` 功能部件（一个 Java EE 7 功能部件）与 `jpa-2.0` 功能部件配合使用。这允许应用程序与现有 JPA 提供程序保持同步，同时可使用新的 Java EE 7 功能部件。

`jpa-2.1`

`jpa-2.1` 功能部件是 JPA 2.1 规范实现，由 EclipseLink 提供支持。此功能部件仅与其他 Java EE 7 功能部件兼容。如果此 `jpa-2.1` 功能部件与其他 Java EE 6 功能部件配合使用，那么 message.log 文件中将记录以下错误。

```
CWWKF0033E:
不能同时装入单例功能部件 com.ibm.websphere.appserver.javaeeCompatible-7.0 和 com.ibm.websphere.appserver.javaeeCompatible-6.0。所配置功能部件 jpa-2.1 和 servlet-3.0 包含的一个或多个功能部件导致该冲突。您的配置不受支持；请更新 server.xml 以移除不兼容的功能部件。
```

Java™ Persistence API 2.1 行为更改

如果已对应用程序使用 `jpa-2.0` 功能部件，那么我们鼓励您继续对现有应用程序使用 `jpa-2.0` 功能部件以避免任何迁移问题。对于新应用程序，我们鼓励您使用 `jpa-2.1` 功能部件，这允许您使用 JPA 2.1 规范中提供的新功能部件。如果要将现有应用程序更改为使用 `jpa-2.1` 功能部件而不是 `jpa-2.0` 功能部件，那么您可能需要在迁移过程中调整应用程序。

jpa-2.0 与 jpa-2.1 之间的差别

您需要注意 jpa-2.0 与 jpa-2.1 功能部件之间的一些主要差别：

PersistenceProvider 类名

jpa-2.0

- Vsettan 提供程序：com.ibm.websphere.persistence.PersistenceProviderImpl
- OpenJPA 提供程序：org.apache.openjpa.persistence.PersistenceProviderImpl

jpa-2.1

- org.eclipse.persistence.jpa.PersistenceProvider

缓存行为

jpa-2.0：缺省情况下缓存被禁用。如果应用程序需要使用二级高速缓存，那么必须显式启用二级高速缓存。

jpa-2.1：缺省情况下，EclipseLink 提供程序已启用二级高速缓存和 QueryCache。必须确保此设置是您的应用程序的最佳选项。

增强功能/织入差别

jpa-2.0：OpenJPA 要求增强实体。请参阅有关 *JPA 实体的增强功能*的文档以了解更多信息。

jpa-2.1：EclipseLink 使用未增强实体。xigemaAS 支持静态增强功能。

某些功能部件（例如，延迟装入和一些性能增益）可能未提供。

- 如果已静态增强实体类以与 jpa-2.0 (OpenJPA) 提供程序配合使用，那么必须在使用 jpa-2.1 提供程序前重新编译这些类。

数据源用法差别

jpa-2.0 功能部件以保守方式使用非 JTA 数据源，所以调整应用程序时需要的非 JTA 数据源连接极少。

读取数据并且事务未处于活动状态时，jpa-2.1 使用非 JTA 数据源连接。这意味着使用此功能部件时，非 JTA 数据源连接池需要增大。

有关这两个 JPA 提供程序之间的更多差别，请参阅 *OpenJPA 至 EclipseLink 迁移指南页面*。

OpenJPA 中可用的 JPA 2.1 功能部件

OpenJPA 是 JPA 2.0 提供程序，它包含的功能部件的功能与新 JPA 2.1 功能部件类似。这意味着，如果您的现有应用程序使用 jpa-2.0 功能部件并想要使用 JPA 2.1 的一些新功能部件，那么不必切换至 jpa-2.1 功能部件。您可改为使用 OpenJPA 提供的新功能部件的等价项。OpenJPA 中可用的一些 JPA 2.1 功能部件包括：

模式生成

此功能部件允许您生成 DDL 或直接与数据库交互以根据 JPA 实体定义来定义表模式。有关更多信息，请参阅 JPA 2.1 规范的 9.4 节。

OpenJPA 等价功能部件：模式映射器

实体图形

此功能部件允许您指定实体对象图形的访存或处理。有关更多信息，请参阅 JPA 2.1 规范的 3.7 节。

OpenJPA 等价功能部件：*FetchPlan* 和 *FetchGroup*

存储过程查询

此功能部件允许您调用数据库中存储的过程。有关更多信息，请参阅 JPA 2.1 规范的 3.10.17 节。

OpenJPA 等价功能部件：查询调用

基本属性类型转换

此功能部件允许您在基本类型属性的属性实体表示与数据库表示之间转换。有关更多信息，请参阅 JPA 2.1 规范的 3.8 节。

OpenJPA 等价功能部件：具体化程序功能部件

@Index 和 **@ForeignKey** 注解

请参阅 JPA 2.1 规范的 11.1.19 和 11.1.23 节。

OpenJPA 等价功能部件：OpenJPA 的 *@Index* 和 *@ForeignKey*

EntityManager 的解包实用程序方法和高速缓存

请参阅 JPA 2.1 规范的 3.1.1 和 7.10 节。

OpenJPA 等价功能部件：*EntityManagerImpl.unwrap()* 和 *OpenJPAPersistence.cast()*

从本机 **SQL** 映射结果时的对象构造

请参阅 JPA 2.1 规范的 3.10.16.2.2 节。

OpenJPA 等价功能部件：*ResultShape* 对象

2.1.10 二进制日志记录

二进制日志记录是一个高性能日志和跟踪工具。

概述

日志和跟踪存储

二进制日志记录提供日志数据存储库和跟踪数据存储库。请参阅下图以了解应用程序和应用程序服务器如何存储日志和跟踪信息。

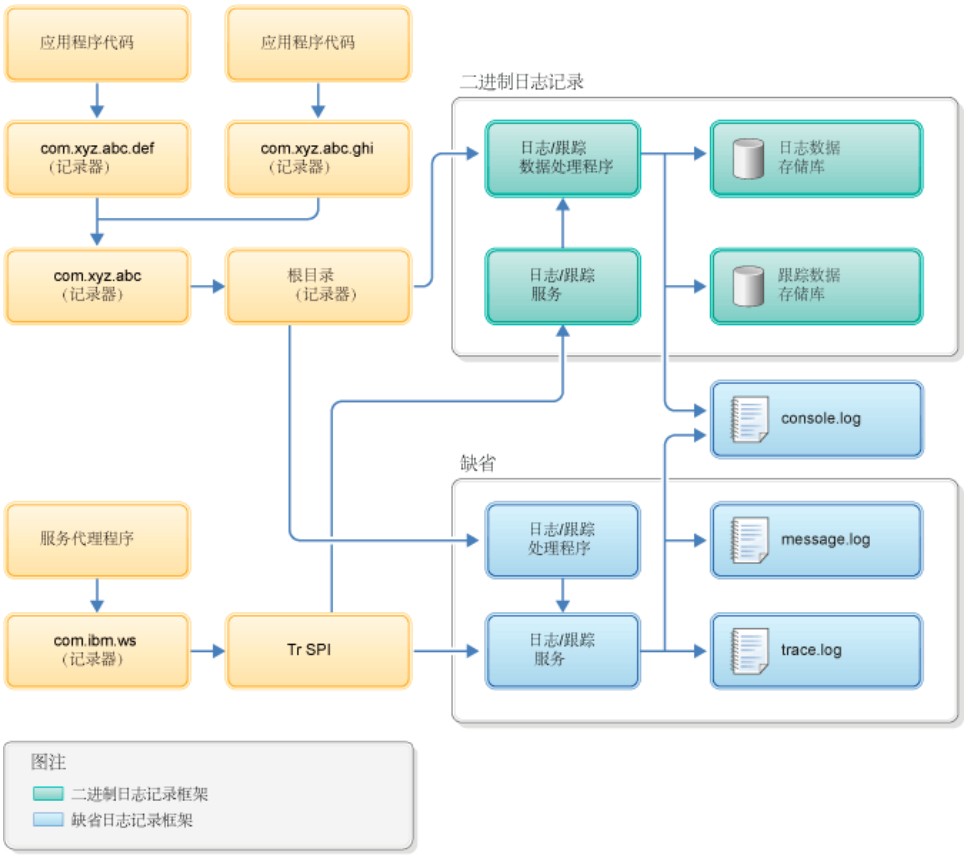


图 27: 用于HPEL日志记录和基本日志记录的日志和跟踪存储器

日志数据存储库

日志数据存储库是用于存储日志记录的存储工具。通常由管理员复查日志数据。这包括应用程序或服务器写入 System.out、System.err、OSGi 记录服务（LOG_INFO 级别或更高级别，包括 LOG_INFO、LOG_WARNING 和 LOG_ERROR）或 java.util.logging（“详细”级别或更高级别，包括“详细”、“配置”、“参考”、“审计”、“警告”、“严重”、“致命”和任何定制级别）的任何信息。

跟踪数据存储库

跟踪数据存储库是用于存储跟踪记录的存储工具。跟踪数据通常供应用程序员或者 xigemaAS 支持团队使用。这包括将应用程序或服务器写入 OSGi 记录服务（LOG_DEBUG 级别）或 `java.util.logging`（“详细”以下的级别，包括精细、较精细、最精细和任何定制级别）的任何信息。

每个日志和跟踪事件都只存储在一个位置

日志事件 System.out 和 System.err 存储在日志数据存储库中。跟踪事件存储在跟踪数据存储库中。仅将每种类型的事件存储在一个位置可确保不会因重复进行数据存储而影响性能。

注：在日志记录性能较重要的情况下，应该禁用控制台日志。写入控制台日志中的任何内容都已经存储在日志数据存储库中。

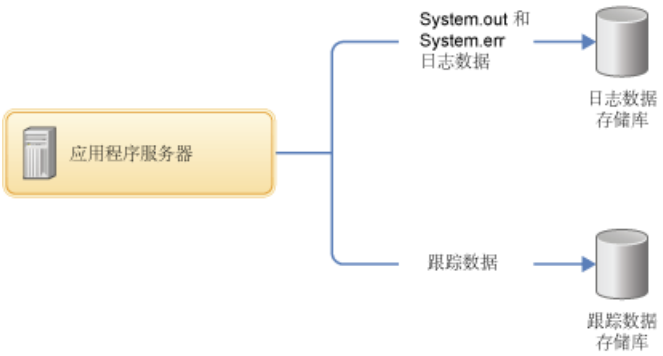


图 28: 连接至日志数据库和跟踪数据存储库的应用程序服务器

除非数据需要格式化，否则不会将数据格式化

格式化数据以便用户阅读会占用处理器时间。日志和跟踪数据会更快速地以专用二进制表示来存储，而不是在运行时格式化日志事件和跟踪事件数据。这将提高日志和跟踪工具的性能。通过延迟日志和跟踪格式化直到 `binaryLog` 命令处于运行状态，从未查看的日志或跟踪部分就决不会格式化。

将日志和跟踪数据写入磁盘之前对其进行缓存

将大块数据写入磁盘比将相同数量的数据写入小块的效率更高。二进制日志记录工具使您能够在将日志和跟踪数据写入磁盘之前对数据进行缓存。缺省情况下，在将日志和跟踪数据写入磁盘之前，会将数据存储在 8 KB 缓冲区中。如果缓冲区在 10 秒之内填满，那么会将缓冲区写入磁盘。如果缓冲区未在 10 秒之内装满，那么会自动地将缓冲区写入磁盘以确保日志具有 最新信息。

日志和跟踪的管理

二进制日志记录的目的是为了使之容易配置和理解。例如，管理员很容易配置供日志和跟踪专用的磁盘空间量、日志和跟踪记录的保留时间以及由服务器来管理日志和跟踪内容。又比如，可以使用一个容易使用的命令 (`binaryLog`) 来访问所有日志、跟踪、System.out 和 System.err 内容，从而避免对于要在哪个文件中访问特定内容可能发生任何混淆。

从日志数据存储库和跟踪数据存储库中读取

日志数据存储库和跟踪数据存储库以 xigemaAS 专用格式进行存储，并且无法使用“记事本”或 VI 之类的文本文件编辑器来阅读。可以使用 `binaryLog` 命令将日志数据存储库和跟踪数据存储库复制到纯文本格式。

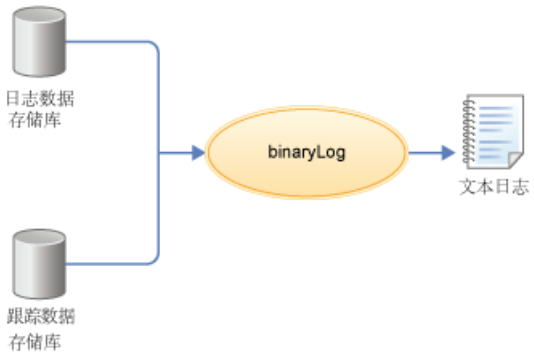


图 29: 拉取数据到文本日志中

binaryLog 命令

binaryLog 是为用户提供的一个容易使用的命令行工具，用来处理日志数据存储库和跟踪数据存储库。binaryLog 提供过滤和格式化选项，使在日志数据存储库和跟踪数据存储库中查找重要内容更容易。例如，用户可以过滤任何错误或警告，然后过滤在 10 秒钟之内发生的所有日志和跟踪条目，以找出同一线程上的关键错误消息。

使用日志和跟踪记录扩展内容进行过滤

二进制日志记录工具可让开发者使用日志记录上下文 API (com.ibm.websphere.logging.hpel.LogRecordContext) 将定制扩展添加到日志和跟踪记录。可以使用 binaryLog 命令行工具根据日志和跟踪记录扩展的内容来过滤记录。

开发资源

已经对二进制日志记录进行设计，以便比缺省日志记录工具更灵活高效地处理日志和跟踪内容。很容易对日志和跟踪内容进行过滤以仅显示感兴趣的记录。您可以使用命令行（请参阅 binaryLog 命令的描述），或者开发者可以使用 HPEL API 来创建功能强大的日志处理程序。

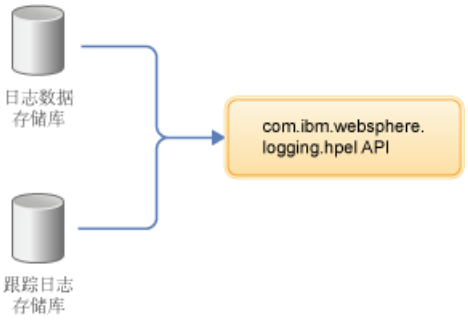


图 30: 连接至 HPEL API 的日志数据存储库和跟踪数据存储库

读取日志数据和跟踪数据

已经提供了一个 API，以使开发者容易开发一些工具来使用二进制日志和跟踪存储库中的内容。例如，开发者可编写 Java™ 程序以搜索日志和跟踪内容来查找其消息标识与已知重要消息标识列表匹配的任何消息。此 API 位于 com.ibm.websphere.logging.hpel 包中。请参阅 API 文档以了解有关 HPEL 日志读取 API 的详细信息。

日志和跟踪记录可扩展性

开发者可以通过日志记录上下文 API (`com.ibm.websphere.logging.hpel.LogRecordContext`) 向日志和跟踪记录添加定制扩展。当二进制日志记录存储日志和跟踪记录时，它会将存在于日志记录上下文中的任何扩展包括在同一线程上。例如，开发者可以编写 Servlet 过滤器，以向日志记录上下文添加重要的 HTTP 请求参数。当该 Servlet 运行时，HPEL API 会向在同一线程上创建的任何日志和跟踪记录添加这些扩展。

与其他日志和跟踪记录字段一样，开发者可以使用 HPEL API 来访问记录扩展。这在编写工具以从日志和跟踪存储库中读取时很有用。在运行时，开发者还可以利用日志记录上下文 API 来访问定制日志处理程序、过滤器和格式化程序中的扩展。

binaryLog 命令选项

使用 `binaryLog` 命令来查看或复制二进制日志记录存储库的内容，或者列示该存储库中可用的服务器进程实例。

二进制日志和跟踪工具以二进制格式写入至存储库。可以使用 binaryLog 命令来查看、查询和过滤存储库。binaryLog 命令提供的选项可快速地将存储库内容转换为各种格式的文本文件，例如基本格式和高级格式。该命令还提供可更方便地从记录中获取所需数据的选项；例如，允许您按级别、记录器名称或日期和时间过滤所需的日志记录。

语法

命令语法如下所示：

```
binaryLog action {serverName | repositoryPath} [options]
```

options 的值随 action 的值不同而不同。

参数

下列操作可用于 binaryLog 命令：

view

读取存储库，可以选择对其进行过滤，并创建用户可以读取的版本。

命令语法如下所示：

```
binaryLog view {serverName | repositoryPath} [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是同时包含 logdata 和 tracedata 目录的目录。

👉 注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

  根据最早记录创建日期进行过滤。必须按日期（例如，`--minDate="2/20/13"`）或者日期和时间（例如，`--minDate="2/20/13 16:47:21:445 EST"`）指定值。
- `--maxDate=`*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，`--maxDate="2/20/13"`）或者日期和时间（例如，`--maxDate="2/20/13 16:47:21:445 EST"`）指定值。
- --minLevel=*value*

  根据最低级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- --maxLevel=*value*

  根据最高级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- --includeLogger=*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- --includeMessage=*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- --includeThread=*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，`--includeThread=2a`）。
- --includeExtension=*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- --includeInstance=*value*

  包括所指定服务器实例中的记录。值必须为 `"latest"` 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

监视选项：

--monitor

连续监视存储库并输出所生成的新内容。

输出选项：

- --format={`basic`|`advanced`|`CBE-1.0.1`}

  指定要使用的输出格式。`"basic"` 为缺省格式。
- --encoding=*value*

  指定要用于输出的字符编码。

copy

读取存储库，可以选择对其进行过滤，并将内容写入新的存储库。

命令语法如下所示：

```
binaryLog copy {serverName | repositoryPath} targetPath [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

targetPath

指定要在其中创建新的存储库的路径。必须指定 *targetPath*。

👉 注：必须指定 *serverName* 或 *repositoryPath*，以及 *targetPath*。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

  根据最早记录创建日期进行过滤。必须按日期（例如，`--minDate="2/20/13"`）或者日期和时间（例如，`--minDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--maxDate=`*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，`--maxDate="2/20/13"`）或者日期和时间（例如，`--maxDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--minLevel=`*value*

  根据最低级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--maxLevel=`*value*

  根据最高级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--includeLogger=`*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--excludeLogger=`*value*[,*value*]*

  排除具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--includeMessage=`*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- `--includeThread=`*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，`--includeThread=2a`）。
- `--includeExtension=`*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- `--includeInstance=`*value*

  包括所指定服务器实例中的记录。值必须为 `"latest"` 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

listInstances

列示存储库中的服务器实例的标识。服务器实例是从服务器启动直到停止期间写入的所有日志记录/跟踪记录的集合。可将服务器实例标识与 binaryLog 查看操作的 --includeInstance 选项配合使用。

命令语法如下所示：

```
binaryLog listInstances {serverName | repositoryPath}
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

请了解 binaryLog 过滤优化。binaryLog 工具在与下列过滤选项配合使用时，能够最高效地过滤日志和跟踪数据：

- --minDate
- --maxDate
- --includeThread
- --minLevel
- --maxLevel

用法示例

请参阅 binaryLog 命令的下列示例。

- 显示 defaultServer 存储库中在 2013 年 7 月 19 日到 2013 年 8 月 2 日期间发生的所有事件。

```
binaryLog view --minDate=07/19/13
--maxDate=08/02/13
```

- 显示 myServer 服务器中发生的、其指定级别为 WARNING 或更高级别的新事件，并在服务器将其写入日志存储库时使用高级格式。

```
binaryLog view myServer --monitor --minLevel=WARNING
--format=advanced
```

- 写入来自存储库（位于 /apps/server1/logs）中的日志消息；仅包括已写入特定存储库的错误流的那些消息。

```
binaryLog
view /apps/server1/logs --includeLogger=SystemErr
```

- 查看 defaultServer 存储库中在 2012 年 9 月 14 日美国东部夏令时下午 4:28 之前发生的事件。

```
binaryLog view
--maxDate="09/14/12 16:28:00:000 EDT"
```

- 写入 defaultServer 存储库中包含“thread”扩展名并且具有以下值的事件：'Default Executor -thread-4'

```
binaryLog view --includeExtension=thread="Default Executor-thread-4" --format=advanced
```

- 查看 defaultServer 存储库中的服务器实例的列表：

```
binaryLog listInstances

使用 D:\wlp\usr\servers\defaultServer\logs 作为存储库目录。

实例标识                        开始日期
1358809441761                   1/21/13 18:04:01:761 EST
1358864476191                   1/22/13 9:21:16:191 EST
1358869523192                   1/22/13 10:45:23:192 EST
1358871281166                   1/22/13 11:14:41:166 EST
1358879829000                   1/22/13 13:37:09:000 EST
1358892222067                   1/22/13 17:03:42:067 EST
```

- 查看 defaultServer 中的使用前一示例中的其中一个实例标识的事件：

```
binaryLog view --includeInstance=1358871281166
```

- 将 defaultServer 中的指定级别为 WARNING 或更高级别的事件从最新服务器实例复制到 d:\toSupport 目录中的新存储库。

```
binaryLog copy defaultServer d:\toSupport --minLevel=warning --includeInstance=latest
```

在 xigemaAS 中配置二进制日志记录

使用此信息作为在 xigemaAS 中配置二进制日志记录的指南。

与缺省 xigemaAS 日志和跟踪框架相比，二进制日志记录提供处理速度更快的日志和跟踪功能以及更灵活的方式来使用日志和跟踪内容。

服务器配置由 bootstrap.properties 文件、server.xml 文件以及这些文件随附的任何可选文件组成。bootstrap.properties 文件指定在处理主要配置之前必须提供的属性，这些属性会保持最少。server.xml 文件是服务器的主要配置文件。

server.xml 文件及其关联文件使用适合于大多数文本编辑器的简单 XML 格式。

bootstrap.properties 文件指定服务器是否应该将二进制日志记录用作日志和跟踪框架，或者用作缺省日志和跟踪框架。

可以通过服务器配置或者 bootstrap.properties 文件来配置二进制日志记录。

- 服务器配置：要从您自己的代码获取日志记录（进行服务器配置处理之后载入），请使用服务器配置来配置二进制日志记录。
- bootstrap.properties 文件：可能需要设置日志记录属性以使其在服务器配置文件得到处理之前生效。例如，如果需要分析服务器启动或配置处理早期发生的问题。在这种情况下，您可以在 bootstrap.properties 文件中配置二进制日志记录。

可以在 bootstrap.properties 或者 server.xml 文件中设置日志记录属性。使用 server.xml 文件中的属性，或者使用 bootstrap.properties 文件中的等价属性。从服务器读取 bootstrap.properties 文件开始，使用 bootstrap.properties 文件中的任何设置，直到完成处理 server.xml 文件为止。如果bootstrap.properties 文件中的日志记录属性未在 server.xml 文件中进行替换或重置，那么将继续使用 bootstrap.properties 文件中的属性值。

如果启用了二进制日志记录，那么会忽略 maxFileSize、maxFiles、messageFileName、traceFileName 和 traceFormat 日志记录元素属性（因为二进制日志记录是在没有 trace.log 和 messages.log 文件的情况下运行）。traceSpecification、consoleLogLevel 和 logDirectory 属性继续用来设置跟踪规范、控制台日志的级别以及日志和跟踪文件的布置。

如果您在 server.xml 文件中设置日志记录或二进制日志记录属性，那么可以通过在 bootstrap.properties 文件中将相应的属性设置为同一值来避免在启动时与运行时之间更改配置。请注意，如果在 bootstrap.properties 文件中未设置任何日志记录属性或者二进制日志记录属性，那么服务器将使用缺省日志记录设置。

- 通过更新 bootstrap.properties 文件对服务器启用二进制日志记录。

  在 bootstrap.properties 文件中，单独添加下列文本行：

  ```
  websphere.log.provider=binaryLogging-1.0
  ```

- 使用下列参数来配置二进制日志记录。

  列出的所有子元素都是 server.xml 文件中日志记录元素的子元素。

  下表列出了可在 server.xml 文件中配置的属性，以及可在 bootstrap.properties 文件中设置的等价属性：

表 11: 可在 server.xml 中配置的二进制日志记录属性以及可在 bootstrap.properties 中设置的等价属性

日志记录子元素	属性	等价的 **bootstrap.properties** 属性
binaryLog	purgeMaxSize	com.ibm.hpel.log.purgeMaxSize
	purgeMinTime	com.ibm.hpel.log.purgeMinTime
	fileSwitchTime	com.ibm.hpel.log.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.log.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.log.outOfSpaceAction
binaryTrace	purgeMaxSize	com.ibm.hpel.trace.purgeMaxSize
	purgeMinTime	com.ibm.hpel.trace.purgeMinTime
	fileSwitchTime	com.ibm.hpel.trace.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.trace.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.trace.outOfSpaceAction

以下示例显示配置为启用二进制日志记录的 bootstrap.properties 文件：

```
websphere.log.provider=binaryLogging-1.0
```

以下示例显示了具有二进制日志记录子元素的 server.xml 文件。日志内容设为在 96 小时后过期，跟踪内容设为最大保留 1024MB：

```
<server description="new server">

    <logging>
        <binaryLog purgeMinTime="96"/>
        <binaryTrace purgeMaxSize="1024"/>
    </logging>
```

```
</server>
```

在重新启动服务器之后，将启用并配置二进制日志记录。

2.1.11 xigemaAS: 会话启动协议 (SIP)

会话启动协议 (SIP) 是用于许多交互式服务（包括音频、视频和对等通信）的控制协议。SIP 以及 SIP 的标准提供了一些机制，用于查询、 协商和管理与基于任何其他协议的任何网络上对等体的连接。

通过 `sipServlet-1.1` 功能部件，xigemaAS 包含对 SIP Servlet 规范 1.1（又称为 Java 规范请求 (JSR) 289）的支持。SIP Servlet 规范提供了适用于会话启动协议 (SIP) 的 Java API 标准。JSR 289 是现有的 SIP Servlet 规范的更新，可解决行业用户确定的新需求。此产品继续支持 SIP Servlet 1.0 规范。

本产品符合以下 SIP 标准：

IETF
JCP

要获取受支持的因特网工程任务组织 (IETF) 和 Java Community Process (JCP) 业界标准的完整列表，请参阅 *xigemaAS* 上的 *SIP* 业界标准一致性（见第 1084 页）。

在 xigemaAS 服务器中，Web 容器和 SIP 容器已汇聚并且能够共享会话管理、安全性和其他属性。在此模型中，包括 SIP Servlet、HTTP Servlet 和 portlet 的应用程序可以无缝交互，而与使用的协议无关。xigemaAS 服务器中的 HTTP 和 SIP 的紧密集成意味着这些聚合应用程序高度可用。

xigemaAS 上的会话启动协议 (SIP) 应用程序

SIP 应用程序是一个至少使用了一个会话启动协议 (SIP) Servlet 的 Java 程序。

SIP Servlet 是由 SIP Servlet 容器管理的 Java™ 应用程序组件，用于执行 SIP 信号发送。与其他 Java 的组件一样，servlet 是独立于平台的 Java 类，它们编译为独立于平台的字节码，可动态装入到支持 Java 的 SIP 应用程序服务器（例如，xigemaAS 概要文件服务器），并可由该应用程序服务器运行。容器（有时称为 Servlet 引擎）是用来处理 Servlet 交互的服务器扩展。SIP Servlet 通过 Servlet 容器与客户机交换请求和响应消息，从而实现与客户机的交互。

SIP 可用来建立、修改和终止多媒体 IP 会话（包括 IP 电话、用户状态和即时消息传递）。在此上下文中，“用户状态”是指诸如“联机”、“离开”或“请勿打扰”之类的用户状态。定义用于编写基于 SIP 的 Servlet 应用程序的编程模型的标准是 *JSR 289*。该规范的基本思想是，提供一个与 HTTP Servlet 类似的 Java™ 应用程序编程接口 (API)，从而提供一个易于使用的 SIP 编程模型。与 HTTP Servlet 编程模型一样，有些灵活性仅限于优化易用性和时间价值比。

但是，由于协议差别很大，因此 SIP Servlet API 在许多方面都与 HTTP Servlet 不同。虽然 SIP 是请求/响应协议，但每个请求并不一定只有一个响应。这种复杂性以及对高效解决方案的需求意味着更容易实现 SPI Servlet 本机异步执行。而且，与 HTTP servlet 不同，SIP Servlet 的编程模型力求简化创建客户机请求和其他逻辑的过程，因为许多应用程序充当其他服务器/代理的客户机或代理。

SipServlet 请求

SIP Servlet 与 HTTP Servlet 有相同点，就是每个 SIP Servlet 都扩展一个基本 javax.servlet.sip.SipServlet 类。所有消息都通过服务方法传入，您可以扩展该服务方法。但是，因为 SIP 中不存在请求与响应的一对一映射，

所以建议的做法是改为扩展 doRequest 或 doResponse 方法。扩展 doRequest 或 doResponse 方法时，一定要调用经过扩展的方法才能完成处理。

规范必须支持的每个请求方法都有一个 doxxx 方法，这点类似于 HTTP。在 HTTP 中，GET 和 POST 请求具有类似于 doGet 和 doPost 的方法。在 SIP 中，每个 SIP 请求方法都具有 doInvite、doAck、doOptions、doBye、doCancel、doRegister、doSubscribe、doNotify、doMessage、doInfo 和 doPrack 方法。

与 HTTP Servlet 不同，SIP Servlet 对于支持的每种响应类型都有一些方法。因此，SIP Servlet 包括 doProvisionalResponse、doSuccessResponse、doRedirectResponse 和 doErrorResponse 响应。确切地说，临时响应（1xx 响应）用于指示状态，成功响应（2xx 响应）用于指示成功完成事务，重定向响应（3xx 响应）用于将客户机重定向至已移动的资源或实体，而错误响应（4xx、5xx 和 6xx 响应）用于指示故障或特定错误情况。这些类型的响应消息与 HTTP 类似，但由于 SIP Servlet 编程模型包括客户机编程模型，所以还需要以编程方式处理响应。

JSR 289 概述

通过 `sipServlet-1.1` 功能部件，xigemaAS 包含对 SIP Servlet 规范 1.1（又称为 Java™ 规范请求 (JSR) 289）的支持。SIP 是一个信号协议，用于创建、修改和终止 IP 通信会话，例如，电话、现场和多媒体应用程序。

SIP Servlet 规范提供会话启动协议 (SIP) 的 Java™ API 标准。JSR 289 是现有的 SIP Servlet 规范的更新，可解决行业用户确定的新需求。

JSR 289 规范添加以下功能部件：

- 用于应用程序选择的应用程序路由器

  应用程序路由可使开发人员在较小型的应用程序外部构建复杂服务。在执行初始请求时，容器调用应用程序路由器来根据请求类型确定要调用的应用程序。应用程序路由器是用于选择应用程序顺序的中央集线器。有关更多信息，请参阅*xigemaAS 上的 SIP 应用程序路由器*（见第 1084 页）和*在 xigemaAS 上管理会话启动协议 (SIP)*（见第 1225 页）。
- 基于注释的编程

  注释提供了一种快速方法来通过直接在应用程序中嵌入元数据开发应用程序。 例如，您可以使用 `@SipServlet` 注释以指示类为 SIP Servlet。`@SipApplication` 为软件包级别注释。软件包中的所有 servlet 属于相同应用程序，除非 servlet 使用 `@SipServlet(applicationName)`。有关注释的更多信息，请参阅 JSR 289 的 18 节。
- 汇聚的应用程序

  JSR 289 提供了一个新的标准化机制来构建聚合应用程序。聚合应用程序包含 SIP Servlet 组件和其他 Java™ EE 组件（如 HTTP servlet 和企业 Bean）。规范包含两个新的类以支持汇聚。

  - `ConvergedHttpSession` 为汇聚应用程序的 HttpSession 的扩展。
  - `SipSessionUtil` 处理汇聚应用程序的会话管理。

  有关汇聚应用程序的更多信息，请参阅 JSR 289 的 13 节。
- 背靠背用户代理 (B2BUA) API

  JSR 289 使用 B2BUA 辅助类简化了应用程序中的 B2BUA 模式。 B2BUA 为常用应用程序模式。B2BUA 充当两个或更多的对话的端点、转发请求以及在这些对话之间响应。B2BUA 帮助程序可以创建传入请求的副本。它还会自动维护 B2BUA 两侧的会话之间的链路。有关 B2BUA 的更多信息，请参阅 JSR 289 的 12 节。

xigemaAS 上的 SIP 应用程序路由器

会话启动协议 (SIP) 容器使用 SIP 应用程序路由器以选择在容器内运行应用程序的顺序。

SIP 容器可以调用多个应用程序，以部署完整的服务或功能。这种模块化组合方法使应用程序开发者能够轻松地开发新应用程序。模块化应用程序的组合和管理更为方便，而各个应用程序实现将保持相互独立。

应用程序路由器负责按正确的顺序选择正确的应用程序，以便处理入局消息。应用程序路由器是容器正常工作所必需的，但它是独立于容器的逻辑实体。应用程序路由器基于 JSR 289 规范。有关应用程序路由器功能的更多详细信息，请参阅该规范。

可使用标准配置文件来配置缺省应用程序路由器 (DAR)，按照 JSR 289 中的定义，它将提供给 SIP 容器。

您可以通过多种方法来处理应用程序路由（也称为应用程序组合）：

- 通过对 `library` 元素指定包含应用程序路由器实现和提供程序的 Java™ 归档 (JAR) 文件的路径来配置定制应用程序路由器 (CAR)，将添加 `bell` 元素以在 server.xml 文件中引用此库。特定提供程序是在 `sipApplicationRouter` 元素的 `carProvider` 属性上定义的。有关示例，请参阅在 *xigemaAS* 上管理会话启动协议 *(SIP)*（见第 1225 页）的文档。
- 通过在 server.xml 文件的 `sipApplicationRouter` 元素的 `sipDarConfiguration` 属性中提供 DAR 的位置来配置 DAR。

xigemaAS 上的会话启动协议 (SIP) 容器

SIP 容器是一个 Web 应用程序服务器组件，它通过调用会话启动协议 (SIP) 操作 Servlet 并与该操作 Servlet 交互来处理 SIP 请求。

Servlet 容器提供了一些网络服务，通过这些网络服务来发送和接收请求和响应。它将决定调用哪些应用程序及调用顺序。Servlet 容器中还包含一些 Servlet，并通过它们的生命周期来进行管理。

SIP Servlet 容器还管理它侦听入局 SIP 流量所在的网络侦听器点。一个侦听器点是传输协议、IP 地址和端口号的组合。SIP Servlet 容器支持以下传输协议：

- 用户数据报协议 (UDP)
- 传输控制协议 (TCP)
- 基于 TCP 的传输层安全性 (TLS)

xigemaAS 上的 SIP 业界标准一致性

xigemaAS 上的会话启动协议 (SIP) 符合 SIP 容器和 SIP 应用程序的业界标准。

SIP 容器

xigemaAS 符合以下 SIP 标准：

IETF
JCP

xigemaAS 还符合因特网工程任务组织 (IETF) 和 Java™ 社区进程 (JCP) 的 SIP 业界标准。下表列出了 IETF 和 JCP 标准。

表 12: xigemaAS 符合这些 SIP 标准

一个包含两列的表，此表描述 xigemaAS 上的 SIP 容器符合的标准。

标准	描述
JR116	SIP：SIP Servlet API
JR289	SIP：SIP Servlet API V1.1
RFC 2543	SIP：会话启动协议
RFC 3261	SIP：会话启动协议
RFC 3262	SIP 中临时响应的可靠性
RFC 3263	查找 SIP 服务器
RFC 3265	特定于 SIP 的事件通知
RFC 3311	SIP UPDATE 方法
RFC 3325	SIP 的专用扩展，用于可信网络中断言的标识
RFC 3326	SIP 的原因头字段
RFC 3515	SIP Refer 方法
RFC 3581	对称响应路由的 SIP 扩展
RFC 3824	将 E.164 号码与 SIP 一起使用
RFC 3891	SIP 替换头
RFC 3903	用于事件状态发布的 SIP 扩展
RFC 3911	SIP 连接头
RFC 4475	SIP 稳定性测试消息
RFC 5057	多对话在 SIP 中的使用
RFC 5626	在 SIP 中管理客户机启动的连接 👉 注：按照 RFC 5626 第 5、6 和 7 节中的规定，SIP 服务器可以充当代理或注册器。RFC 5626 是 RFC 3261 的扩展。SIP 服务器全面支持 RFC 3261。但是，对 RFC 5626 的支持存在下列局限性： • 按照 RFC 3261 中的定义，SIP 服务器可以充当用户代理；但是，按照 RFC 5626 第 4 节中的定义，它不能充当用户代理。 • 按照 RFC 5626 中的规定，SIP 服务器不支持 STUN 保持活动。
RFC 5658	SIP 中的寻址记录路由问题

SIP 应用程序

xigemaAS 符合 SIP 应用程序的标准。

表 13: 符合 SIP 应用程序的标准

一个包含两列的表，此表描述 xigemaAS 上的 SIP 应用程序符合的标准。

标准	描述
RFC 2848	PINT 服务协议：因特网协议 (IP) 接入电话呼叫业务的 SIP 和会话描述协议 (SDP) 扩展
RFC 2976	SIP INFO 方法
RFC 3050	SIP 的公共网关接口
RFC 3087	使用 SIP 请求 URI 控制服务上下文
RFC 3264	使用 SDP 的请求和应答模型
RFC 3266	SDP 中对 IPv6 的支持
RFC 3312	资源管理和 SIP 的集成
RFC 3313	媒体权限的专用 SIP 扩展
RFC 3319	SIP 服务器的动态主机配置协议 (DHCPv6) 选项
RFC 3327	用于注册非相邻联系人的 SIP 扩展头字段
RFC 3372	电话 SIP (SIP-T)：上下文和体系结构
RFC 3398	综合业务数字网 (ISDN) 用户部分 (ISUP) 至 SIP 的映射
RFC 3428	用于即时消息传递的 SIP 扩展
RFC 3455	用于第三代合作伙伴计划 (3GPP) 的 SIP 的专用头（P 头）扩展
RFC 3578	综合业务数字网 (ISDN) 用户部分 (ISUP) 重叠信令至 SIP 的映射
RFC 3603	用于支持 PacketCable 分布式呼叫信令体系结构的专用 SIP 代理至代理扩展
RFC 3608	注册期间用于服务路由发现的 SIP 扩展头字段
RFC 3665	SIP 基本呼叫流示例
RFC 3666	SIP 公共交换电话网 (PSTN) 呼叫流
RFC 3680	用于注册的 SIP 事件包
RFC 3725	SIP 中用于第三方呼叫控制 (3pcc) 的最新最佳实践
RFC 3840	在 SIP 中指示用户代理功能
RFC 3842	SIP 的消息摘要和消息等待指示事件包
RFC 3856	SIP 的存在事件包
RFC 3857	SIP 的观察器信息事件模板包
RFC 3959	SIP 的早期会话处置类型
RFC 3960	SIP 中的早期媒体和铃音生成
RFC 3976	使 SIP 和智能网络 (IN) 应用程序互相配合
RFC 4032	SIP 前置条件框架的更新

标准	描述
RFC 4092	在 SIP 中使用 SDP 备用网络地址类型 (ANAT) 语义
RFC 4117	在 SIP 中使用第三方呼叫控制 (3pcc) 进行代码转换服务调用
RFC 4235	SIP 的邀请启动对话事件包
RFC 4240	使用 SIP 的基本网络媒体服务
RFC 4353	使用 SIP 的会议框架
RFC 4354	支持无线一键通 (PoC) 服务的各种设置的 SIP 事件包和数据格式
RFC 4411	扩展优先抢占事件的 SIP 原因头
RFC 4457	SIP P 用户数据库专用头（P 头）
RFC 4458	诸如语音邮件和交互式语音响应 (IVR) 之类的应用程序的 SIP URI
RFC 4483	用于间接获取 SIP 消息内容的机制
RFC 4497	使 SIP 与 QSIG 互相配合
RFC 4508	使用 SIP REFER 方法输送特征标记

2.2 安装xigemaAS

有两种方法可用来安装 xigemaAS，分别为图形界面方法和安装 xigemaAS 概要文件。

2.2.1 安装 xigemaAS

本部分介绍xigemaAS 的安装过程，包括安装、注册许可证和验证安装。

安装

使用两种安装方法来安装 xigemaAS 。

使用图形界面安装xigemaAS

在主机上安装 xigemaAS 并注册许可证。

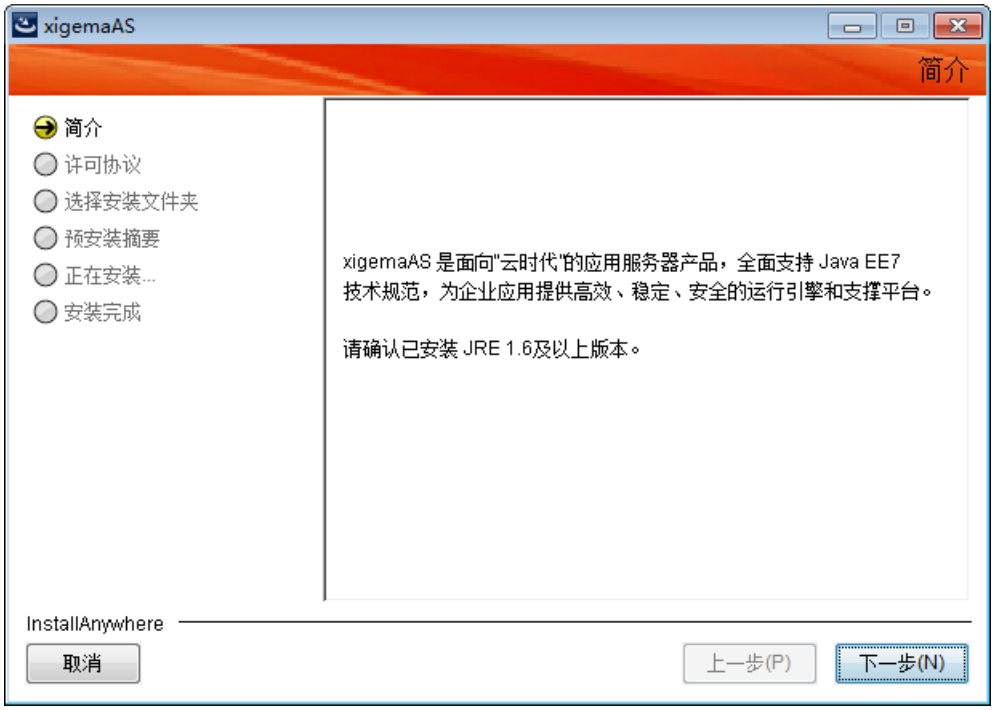
在安装之前，需要以下用户权限：

- Windows 平台：`Administrator` 组
- Linux 平台：`root` 用户

请按以下步骤进行安装。

1. 开始安装。根据所在平台选择下列安装步骤之一：
   - 在 Windows 平台上：双击安装文件 xigemaASInstall.exe
   - 在 Linux 平台上：进入到 xigemaAS 安装目录，执行命令 `./xigemaASInstall.bin`

   进入安装页面：



2. 单击**下一步**，显示“许可协议”信息。
3. 选择“我接受许可协议条款”。单击**下一步**。
4. 选择安装路径。

   在“您想在哪一位置安装？”输入框中，可以选择该安装的目标路径。默认安装路径为 C:\Program Files \xigemaAS。
5. 单击**下一步**，查看预安装摘要。
6. 确认安装信息无误，单击**安装**。
7. 显示安装成功后，单击**完成**。

通过解压 JAR 文件来安装 xigemaAS

通过运行其中包含产品镜像的自解压 JAR 文件，您可以安装 xigemaAS，并且准备创建服务器。

系统必须满足使用 xigemaAS 的操作系统和 Java™ 要求。

1. 从合法经销商处获取 xigemaAS *version*（例如：V9.1.0.1）的产品镜像，以及试用版或正式版的 license 文件。
2. 将产品镜像解压到首选目录。

   要使用交互式安装向导来解压分发映像，请运行 `java -jar xigemaAS-version.jar`。所有应用程序服务器文件都存储在 `wlp` 目录的子目录中。
3. 可选：解压编程模型扩展。必须安装 xigemaAS 的产品许可版本，然后才能安装编程模型扩展。

   编程模型扩展打包为名为 `wlp-extended-version.jar` 的 JAR 文件。要使用交互式安装向导解压编程模型扩展，请运行以下命令：

   ```
   java -jar wlp-extended-version.jar
   ```

   有关可用解压选项的列表，请参阅*#unique_428*。

4. 可选： 设置环境的 JAVA_HOME 属性。

   xigemaAS 在 Java 运行时环境 (JRE) 中运行。可以使用 server.env 文件中的 JAVA_HOME 属性来指定 JDK 或 JRE 位置，如定制 *xigemaAS* 环境（见第 1114 页）中所述。在 server.env 文件中设置 JAVA_HOME 属性时，xigemaAS 使用同一 Java™ 运行时位置而不考虑 xigemaAS 服务器运行时使用的用户概要文件。

   在 Linux™ 或 UNIX™ 系统上，可以改为在用户 .bashrc 文件中设置 JAVA_HOME，或者将 JDK 或 JRE 路径附加到 PATH 环境变量。在 Windows™ 系统上，可以改为将 JAVA_HOME 设置为系统环境变量或者对 PATH 系统变量追加 JDK 或 JRE 路径。例如，在 Windows™ 系统上，可以使用以下命令来设置 JAVA_HOME 属性，以及将 Java™ /bin 目录添加到路径：

   ```
   set JAVA_HOME=C:\Progra~1\Java\JDK16
   set PATH=%JAVA_HOME%\bin;%PATH%
   ```

   👉 注：xigemaAS 运行时环境会以此顺序搜索 `java` 命令：JAVA_HOME 属性、JRE_HOME 属性和系统 PATH 属性。

   有关受支持的 Java™ 环境及如何获取这些环境的更多信息，请参阅支持的最低 *Java* 级别（见第 1669 页）。

JAR 文件解压选项

您可通过解压 JAR 文件来安装 xigemaAS。通过运行包含产品镜像的自解压 JAR 文件，您可以安装 xigemaAS 并准备创建服务器。

语法

命令语法如下所示：

```
java -jar archive-file-name.jar [Options] [install location]
```

其中 *archive-file-name*.jar 是您要解压的 JAR 文件的名称。

👉 注：

- 如果未指定任何选项，那么分发映像是使用交互式安装程序解压的。
- 如果未指定安装位置，那么缺省目标目录为归档文件的位置。

选项

解压命令有以下可用选项：

--help

显示有关如何使用该命令的简短说明及可用选项列表

--acceptLicense

自动指示接受许可条款和条件

--verbose

显示归档解压期间的详细信息

--viewLicenseAgreement

查看许可协议

--viewLicenseInfo

查看许可证信息

用法

以下示例说明了正确的语法：

```
java -jar myarchivefile.jar
java -jar myarchivefile.jar --help
java -jar myarchivefile.jar --acceptLicense C:\xigemaAS-profile-install
java -jar myarchivefile.jar --verbose
java -jar myarchivefile.jar --viewLicenseAgreement
java -jar myarchivefile.jar --viewLicenseInfo
```

注册许可证

本部分介绍如何注册许可证。

注册许可证

在使用 xigemaAS 前，必须注册试用版许可证或正式版许可证。

在注册许可证之前，确定系统上已经安装 xigemaAS。

以下示例假设 license.dat 是许可证文件。

1. 注册许可证。

   在 ${wlp_install_dir}/bin 目录下执行以下命令：

   ```
   productInfo registerlic [options]
   ```

   其中，*options* 的可用选项为：`--path = " license file path "`，即指定需要注册的 license 文件路径。

   示例如下：

   ```
   productInfo registerlic --path=E:\cache\recv\license.dat
   ```

注册成功后，可以通过 `productInfo version` 命令查看许可证信息，以验证是否注册成功。详细信息，参见验证安装 on page 1090。

验证安装

可以使用命令实用程序来验证 xigemaAS 概要文件的安装完整性。

安装 xigemaAS 概要文件之后，必须确保安装已成功完成，并且所有必需功能部件或临时修订都已安装。*productInfo* 命令提供了不同的选项来完成下列任务：

- 比较 APAR 修订与当前安装之间的差异。
- 验证服务器安装和每个功能部件的 MD5 校验和文件。
- 验证当前安装的版本信息。
- 验证当前安装的功能部件列表。

productInfo 命令

使用 `productInfo` 命令可验证产品完整性，比较 xigemaAS 服务器的不同版本，以及验证当前产品版本。

语法

命令语法如下所示：

```
productInfo action --[options]
```

其中，*options* 的可能值随 *action* 参数的值而不同。

参数

下列 *action* 参数和 *options* 值可用于 `productInfo` 命令：

compare

可让您将当前安装中所安装的 APAR 修订，与不同版本的 xigemaAS 相比较。

--target=*path to directory or archive file*

指定要与当前安装进行比较的目标文件。--target 的值可以是目录或归档文件（必须是有效的 xigemaAS 安装位置）。如果未指定 --apars，那么此选项是必需的。

--apars=*a comma separated list of APAR IDs*

根据这个以逗号分隔的 APAR 标识列表来检查当前安装是否包含这些 APAR，然后列出任何未包括的 APAR。如果未指定 --target，那么此选项是必需的。

--output=*path to an output file*

将此命令的结果输出到所提供的文件名。缺省情况下，输出定向到标准输出。

--verbose

发生错误时，显示详细的错误消息。

👉 注：必须至少提供 --target 或 --apars 中的一个。

featureInfo

列出当前 xigemaAS 服务器上安装的所有功能部件，及任何已安装的产品扩展。

--output=*path to an output file*

将此命令的结果输出到所提供的文件名。缺省情况下，输出定向到标准输出。

validate

验证 xigemaAS 服务器。

--output=*path to an output file*

将此命令的结果输出到所提供的文件名。缺省情况下，输出定向到标准输出。

version

显示产品名称和版本。

如果产品扩展安装版本目录中提供的 *product_extension*.properties 文件包含下列属性，那么输出会包括产品扩展的名称和版本：

`com.vesttan.xigema.productVersion=your_product_version`。

`com.vesttan.xigema.productName=your_product_name`。

--output=*filename*

将此命令的结果输出到所提供的文件名。缺省情况下，输出定向到标准输出。

--ifixes

显示应用于系统的 APAR 修订及应用 APAR 修订的临时修订。

有关更多信息，请参阅为功能部件扩展提供产品信息（见第 1271 页）。

viewLicenseAgreement

显示已安装的 xigemaAS 版本的许可协议。

viewLicenseInfo

显示已安装的 xigemaAS 版本的许可证信息。

registerlic

注册 xigemaAS 许可文件

help

显示特定操作的帮助信息。

用法

以下示例说明了正确的语法：

```
productInfo compare --target=C:\wlp\newInstall\wlp
productInfo compare --target=C:\wlp\newInstall.jar --output=C:\wlp\compareOutput.txt
productInfo compare --
apars=com.vesttan.xigema.apar.PM39074,com.vesttan.xigema.apar.PM39075,com.vesttan.xigema.apar.PM39080
productInfo featureInfo --output=c:\wlp\featureListOutput.txt
productInfo validate
productInfo help compare
productInfo version
productInfo viewLicenseAgreement
productInfo viewLicenseInfo
```

xigemaAS 产品信息

根据您所使用的 xigemaAS 类型（试用版或正式版）不同，使用 `productInfo` 命令显示的产品信息也有所差异。

正式版 license 文件示例

```
************************License Info*********************************

                    Product : xigemaAS
                    Version : 9.1.x.x
                    LicenseType : Production
                    SN-KEY : DBA#A000001
                    Authorized : XX公司
```

```
                    Project : XX项目
**************************License Info*********************************
```

试用版 license 文件示例

```
**************************License Info*********************************
                    Product : xigemaAS
                    Version : 9.1.x.x
                    LicenseType : Trial
                    SN-KEY : DBA#A000000
                    StartTime : 2016-7-15
**************************License Info*********************************
There are 63 days left in the trial period for this copy of xigemaAS.
```

表 14: xigemaAS license 文件字段

字段	描述
Product	产品名称
Version	产品版本
LicenseType	License 类型 **Production** 正式版 **Trial** 试用版
SN-KEY	License 序列号
Authorized	License 授权用户（仅在正式版显示此字段）
Project	License 授权项目（仅在正式版显示此字段）
StartTime	License 起始时间（仅在试用版显示此字段）

安装目录说明

本部分讲述 xigemaAS 安装成功后安装根目录及其包含的各子目录的相关介绍。

目录位置和属性

xigemaAS 安装的根目录为 wlp。该根目录及其子目录的相关介绍，请见下表。

表说明：第 1 列包含文件和目录树。如果目录有关联属性，那么在第 2 列中给出。每个文件或目录的描述在第 3 列中给出。

目录或文件	属性	描述
wlp/	wlp.install.dir	安装根目录

目录或文件	属性	描述
+- bin/		用来管理安装的脚本。例如，server。
+- clients/		xigemaAS 客户机和瘦客户机库。例如 restConnector.jar。
+- jython/		基于 Jython 的脚本
+- dev/		开发者资源（API、SPI、规范和工具）的根目录
+- api/		缺省情况下，编译时和运行时可用的公用 API
+- vsettan/		xigemaAS 中的可用 API
+- javadoc/		Java™ 文档归档
+- spec/		缺省情况下，编译时和运行时可用的公用规范 API
+- third-party/		缺省情况下，编译时可用的第三方 API，而且在运行时，必须在配置中使用 classloader 元素的 apiTypeVisibility 属性，为应用程序指定这些 API。
+- spi/		缺省情况下，编译时和运行时可用的公用 SPI
+- vsettan/		xigemaAS 中的可用 SPI
+- javadoc/		SPI 的 Java™ 文档归档
+- spec/		缺省情况下，编译时和运行时可用的公用规范 SPI
+- etc/		适用于所有服务器的用户定制服务器变量 *(可选)*
+- server.env		缺省服务器脚本环境变量（可选）
+- client.env		缺省客户机脚本环境变量（可选）
+- jvm.options		缺省 JVM 选项（可选）
+- lafiles/		许可证信息文件
+- lib/		平台运行时环境
+- samples		用例程序。目的是辅助用户初步了解相关管理对象的使用。这些用例包括 jsp 和 servlet 组件用例、数据连接池用例

目录或文件	属性	描述
		等。关于如何使用，请参考同目录下的自述文件 readme.pdf。
+- templates/		运行时定制模板和示例
+- client/		创建客户机时的客户机模板
+- server/		创建服务器时的服务器模板
+- usr/	wlp.user.dir	用户目录
+- extension/	usr.extension.dir	用户开发的功能部件
+- shared/		
+- apps/	shared.app.dir	共享应用程序
+- config/	shared.config.dir	共享配置文件
+- resources/	shared.resource.dir	共享资源定义：适配器、数据源
+- stackGroups/	shared.stackgroup.dir	用于程序包和可安装文件的远程部署的共享堆栈组
+- servers/		共享服务器目录
+- *server_name*	server.config.dir	服务器配置目录。使用 ${server.config.dir} 来引用特定于服务器的配置（应用程序）。
+- bootstrap.properties		服务器引导属性（可选）
+- jvm.options		服务器 JVM 选项，用以替换 wlp/etc/jvm.options 中的值（可选）
+- server.env		服务器脚本环境变量，与 *wlp/etc/server.env* 合并 *(* 可选 *)*
+- server.xml		服务器配置重叠（必需）
+- apps/		应用程序的服务器配置
+- dropins/		服务器缺省应用程序 dropins 文件夹（可选）
+- *application_name*		应用程序文件夹或归档（可选）
+- configDropins/		服务器配置 dropins 文件夹（可选）
+- defaults		缺省服务器配置 dropins 文件夹（可选）

目录或文件	属性	描述
+- overrides		服务器配置覆盖 dropins 文件夹（可选）
+- *server_name*	server.output.dir	服务器输出目录。使用 ${server.output.dir} 来描述服务器所生成的工件（日志文件和工作区）。
+- logs/		服务器日志文件，包括 FFDC 日志（首次运行服务器之后存在的目录）
+- console.log		基本服务器状态和操作消息
+- trace_*timestamp*.log		带有时间戳记的跟踪消息，详细级别由当前跟踪配置确定
+- ffdc/		首次故障数据捕获 (FFDC) 输出目录
+- ffdc_*timestamp*/		首次故障数据捕获 (FFDC) 输出，通常包含与所请求操作失败相关的诊断数据的选择性内存转储
+- workarea/		服务器在运作时创建的文件（首次运行服务器之后存在的目录）
+- clients/		共享客户机目录
+- *client_name*		客户机配置目录。
+- bootstrap.properties		客户机引导属性（可选）
+- client.jvm.options		客户机 JVM 选项，它将替换 wlp/etc/client.jvm.options 中的值（可选）
+- client.xml		客户机配置重叠（必需）
+- apps/		应用程序的客户机配置
+- logs/		客户机日志文件，包括 FFDC 日志（首次运行客户机之后存在的目录）
+- trace_*timestamp*.log		带有时间戳记的跟踪消息，详细级别由当前跟踪配置确定
+- ffdc/		首次故障数据捕获 (FFDC) 输出目录
+- ffdc_*timestamp*/		首次故障数据捕获 (FFDC) 输出，通常包含与所请求操作失败相关的诊断数据的选择性内存转储
+- workarea/		客户机运作时创建的文件（首次运行客户机之后存在的目录）

2.2.2 升级 xigemaAS 安装

升级 xigemaAS 安装包括升级 xigemaAS 产品版本和升级许可证。

关于升级 xigemaAS 安装的示例场景具体如下：

- 将现有产品版本升级至更高版本以访问高级功能部件。
- 将试用版产品的许可证升级为正式版，以使产品升级至正式版。

升级产品版本

可以通过图形界面方法和解压 JAR 概要文件两种方法升级 xigemaAS 产品版本。

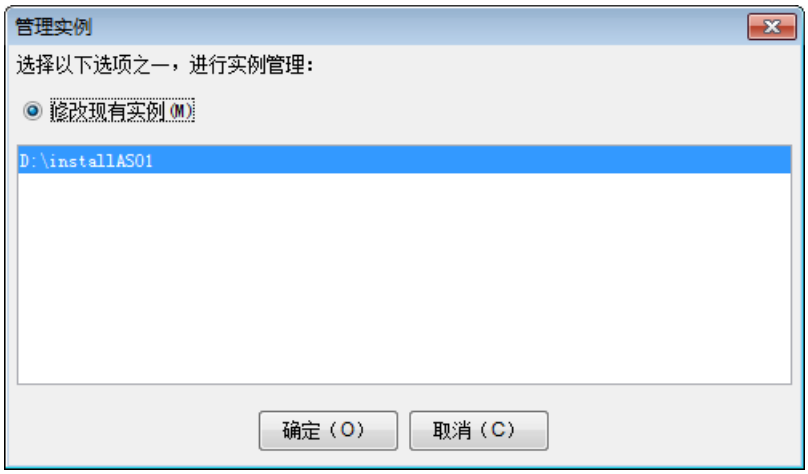
通过图形界面升级 xigemaAS 产品版本

通过图形界面方法来升级 xigemaAS 产品版本。

在获取新版本的 xigemaAS 安装包之后，可以升级已有的产品版本。升级前无需卸载已有安装。

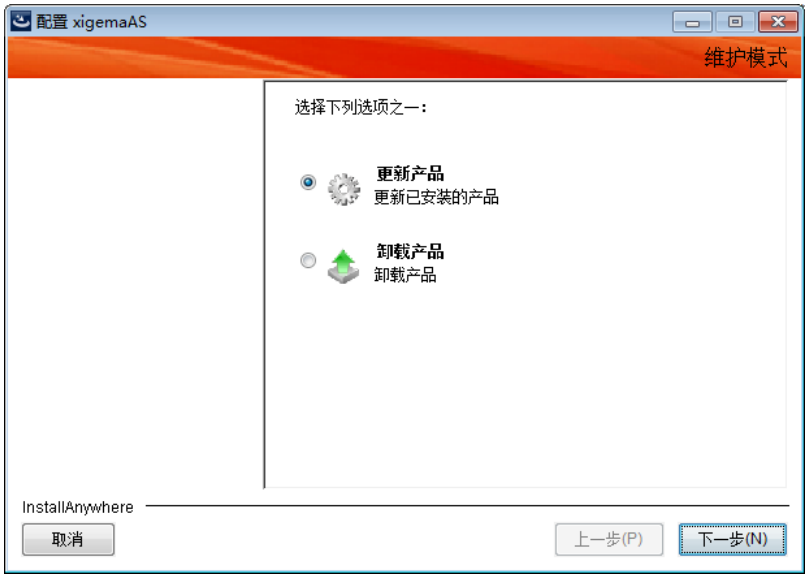
1. 开始升级。根据所在平台选择下列开始步骤之一：
   - 在 Windows 平台上：双击安装文件 xigemaASInstall.exe
   - 在 Linux 平台上：进入到 xigemaAS 安装目录，执行命令 `./xigemaASInstall.bin`

   稍后，进入以下**“管理实例”**页面：



2. 单击**确定**。

   进入**“配置”** xigemaAS 页面。



3. 选择更新产品。单击**下一步**，进入维护模式页面。
4. 单击**下一步**，显示“许可协议”信息。
5. 选择“我接受许可协议条款”。单击**下一步**。

   页面显示已有实例的安装路径。此时用户无需重新选择安装路径。此安装将按照指定的原有安装路径进行安装。
6. 单击**下一步**，查看预安装摘要。
7. 确认安装信息无误，单击**下一步**。
8. 单击**安装**。页面将显示“正在修复xigemaAS”。
9. 在**“安装完成”**页面，单击**完成**。

xigemaAS 产品版本升级完成。

通过自解压 JAR 概要文件升级 xigemaAS 产品版本

可使用自解压 Java™ 概要 (JAR) 文件来升级 xigemaAS 版本。

系统必须满足使用 xigemaAS 的操作系统和 Java™ 要求。

若您需要将现有正式版 xigemaAS 产品升级至更高版本，请直接将更高版本产品的 JAR 文件进行解压、安装并配置，之后将已部署的应用迁移至该概要文件即可。

升级许可证

升级许可证包括通过更改许可参数来改变部署范围，以及将试用版 xigemaAS 升级为正式版。

通过更改许可参数升级

可以通过更改许可参数来升级 xigemaAS。

假设用户需要在新增的某一数量的节点上部署 xigemaAS，此时只需获取一个新的许可证并进行注册，即可扩展产品的部署范围。在此情况下，用户无需每个部署节点都注册许可证。

以下示例假设 license.dat 是许可证文件。

1. 注册许可证。

在 ${wlp_install_dir}/bin 目录下执行以下命令：

```
productInfo registerlic [options]
```

其中，*options* 的可用选项为：`--path = " license file path "`，即指定需要注册的 license 文件路径。

示例如下：

```
productInfo registerlic --path=E:\cache\recv\license.dat
```

注册成功后，可以通过 `productInfo version` 命令查看许可证信息，以验证是否注册成功。详细信息，参见验证安装（见第 1090 页）。

升级试用版到正式版

升级试用版许可证到正式版许可证。

若使用的是试用版 xigemaAS，可按照如下步骤升级至正式版产品：

1. 从Vsettan获取正式版 xigemaAS 的许可证。
2. 将许可证应用于现有 xigemaAS。

   通过 `productInfo registerlic` 命令注册正式版产品许可证，该 license 文件将覆盖原试用版 license 文件，使产品升级至正式版。例如：

   ```
   productInfo registerlic --path=E:\cache\recv\license.dat
   ```

2.2.3 卸载 xigemaAS

本部分介绍如何卸载 xigemaAS。

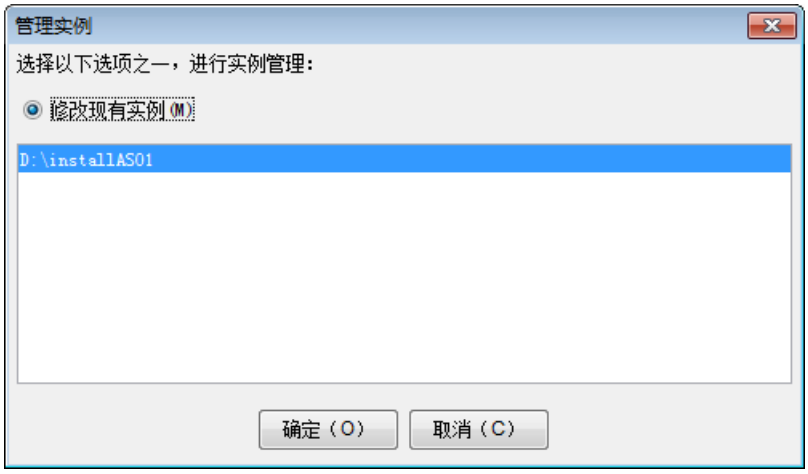
在 Windows 平台上，使用图形界面卸载 xigemaAS。

在 Linux 平台上，除图形界面方法外，还可以通过删除安装所在文件夹的方式直接卸载安装。

本部分介绍使用图形界面方法卸载 xigemaAS。

👉 注：只有通过图形界面方法安装的 xigemaAS 才可以使用该方法卸载。

1. 开始卸载。根据所在平台选择下列安装步骤之一：
   - 在 Linux 平台上：进入到 xigemaAS 安装目录，执行命令 `./xigemaASInstall.bin`
   - 在 Windows 平台上：选择以下两种方式之一开始卸载 xigemaAS。
     - 方式一：
       1. 双击安装文件 xigemaASInstall.exe。稍后，进入以下**“管理实例”**页面：

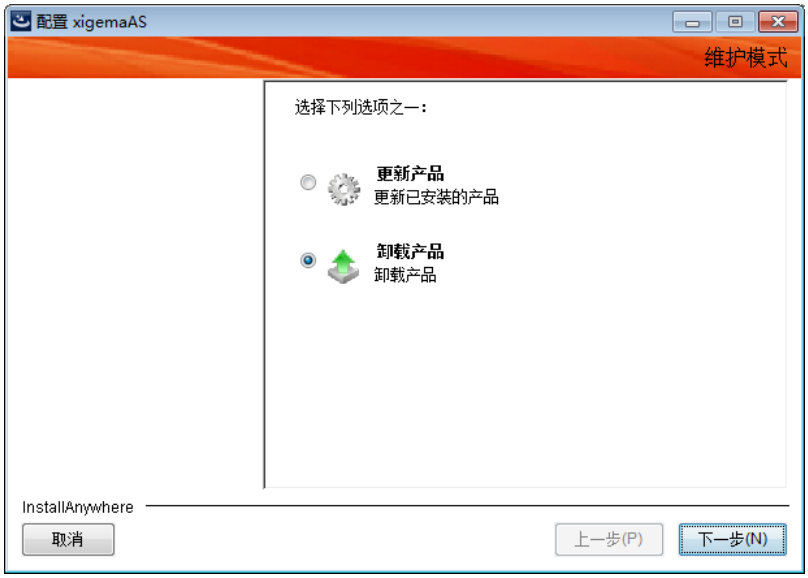


2. 单击**确定**。

- 方式二：

  单击**开始 > xigemaAS > uninstallxigemaAS**。

2. 进入以下“**配置** xigemaAS” 页面。默认选择卸载产品。



3. 单击**下一步**，进入“**卸载** xigemaAS”页面。
4. 单击**下一步**。

   注：卸载后，页面可能会显示一个“无法删除下列项”列表，这是因为目录树 wlp/usr和 wlp/lib 下包含有用户数据的文件。如果不再需要，后续可手动删除。

5. 卸载成功，单击**完成**。

2.3 设置 xigemaAS 应用服务器

定义目录位置和变量，创建并配置服务器，以及添加和移除用来指定服务器功能的 xigemaAS 功能部件。

2.3.1 手动创建应用服务器

可以从命令提示符创建服务器。

1. 打开命令行，然后将目录切换至 `wlp/bin` 目录。

   其中 *path_to_xigemaas* 是您在操作系统上安装 xigemaAS 的位置。

   Windows 上的示例：C:\Users\mo> cd *path_to_xigemaas*\wlp\bin

   Linux 上的示例：[mo@*machine01* ~]# cd *path_to_xigemaas*/wlp/bin
2. 运行以下命令来创建服务器。如果未指定服务器名称，那么会使用 defaultServer。

   其中 *server_name* 是您想要为服务器指定的名称。

   Windows 上的示例：C:\wlp\bin> server create *server_name*

   Linux 上的示例：[mo@*machine01* bin]# ./server create *server_name*

   ```
   server create server_name
   ```

   server_name 只能使用 Unicode 字母数字（例如，`0-9`、`a-z`、`A-Z`）、下划线 (_)、连字符 (-)、加号 (+) 和句点 (.) 字符。名称不能以连字符或句点开头。您的文件系统、操作系统或压缩文件目录可能会施加其他额外限制。

如果已成功创建服务器，您将接收到以下消息：`Server server_name created.`

如果所指定的服务器已存在，那么未创建服务器，您将接收到异常消息：

```
CWWKE0005E: 未能启动运行时环境。
CWWKE0045E: It was not possible to create the server called server_name
 because
the server directory C:\wlp\usr\servers\server_name already exists.
```

系统将在 ${wlp.user.dir}/servers 目录下创建带有新服务器名称的目录，其中包含新服务器的配置。新服务器的 HTTP 端口号被指定为缺省值，显示在所生成 server.xml 文件中以便于编辑。还可在同一目录中的 bootstrap.properties 文件内使用变量来设置这些值。有关更多信息，请参阅设置应用服务器的引导属性（见第 1102 页）。

配置服务器以具有应用程序所需的功能部件。请参阅手动管理 *xigemaAS*（见第 1113 页）。

2.3.2 目录位置和属性

在 xigemaAS 中，许多目录具有关联属性。配置服务器时，可以使用这些属性来指定文件位置。

创建后的服务器及配置信息位于 wlp/usr/servers 目录下。关于目录 usr 及其 servers 等子目录的位置、属性和描述信息，请参见目录位置和属性。

配置服务器时，可以使用与每个目录相关联的属性（如果有）来指定文件位置。要获取示例，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

提示：要确保配置的可移植性，请使用合适的最特定属性，而且不要依赖于资源之间的关系。例如，在某些配置中，安装位置 `${wlp.install.dir}` 可能不是定制实例 `${wlp.user.dir}` 的父代。

位置属性的程序化访问

通过在 server.xml 文件中使用 jndiEntry 配置元素，可将位置属性绑定至所选名称下的 JNDI 命名空间，例如：

```
<jndiEntry jndiName="serverName" value="${wlp.server.name}"/>
```

通过 JNDI 查找，在服务器（应用程序、共享库或功能部件）中运行的任何代码都可访问这类条目：

```
Object serverName = new InitialContext().lookup("serverName");
```

有关如何在配置中使用 JNDI 条目的更多信息，请参阅从服务器配置文件将 *JNDI* 绑定用于常量（见第 1498 页）。

功能部件代码还可使用内核提供的系统编程接口 (SPI) 来解析这些属性的值，例如：

```
ServiceReference <WsLocationAdmin>locationAdminRef =
 bundleContext.getServiceReference(WsLocationAdmin.class);
WsLocationAdmin locationAdmin = bundleContext.getService(locationAdminRef);
String serverName = locationAdmin.resolveString("${wlp.server.name}");
```

2.3.3 设置应用服务器的引导属性

引导属性会初始化特定服务器的运行时环境。通常，它们是会影响运行时核心的配置和初始化的属性。

在名称为 bootstrap.properties 的文本文件中设置引导属性。此文件不是必需的，所以它不存在，除非您创建了它。必须在服务器目录中创建此文件，此目录中还包含配置根文件 server.xml。 缺省情况下，服务器目录是 usr/servers/*server_name*。您可以更改服务器目录，如定制 *xigemaAS* 环境（见第 1114 页）中所述。

bootstrap.properties 文件包含两种属性：

- 较小的预定义初始化属性集。
- 您选择定义的任何定制属性。然后，您可以在其他配置文件（例如，server.xml 及随附的文件）中将这些定制属性用作变量。

可以使用文本编辑器来编辑 bootstrap.properties 文件。

如果更新 bootstrap.properties 文件，那么必须重新启动服务器才能使更改生效。

- 使用预定义的属性来配置跟踪和日志记录。

  例如：

  - 要更改跟踪文件的名称，请使用所选文件名指定 com.ibm.ws.logging.trace.file.name 属性，如下所示：

    ```
    com.ibm.ws.logging.trace.file.name = trace.log
    ```

  - 要启用二进制日志记录，请指定 websphere.log.provider 属性，如下所示：

    ```
    websphere.log.provider = binaryLogging-1.0
    ```

    有关更多信息，请参阅在 *xigemaAS* 中配置二进制日志记录（见第 1080 页）

  有关跟踪和日志记录的预定义属性的列表，请参阅 日志记录和跟踪（见第 1649 页）。
- 使用预定义的属性进行 OSGi 框架诊断。

例如，按如下所示设置 OSGi 控制台的端口：

```
osgi.console = 5678
```

有关更多信息，请参阅 使用 *OSGi* 控制台（见第 1129 页）。

- 使用 OSGi 框架扩展的预定义属性。

  如果外部监视工具需要 org.osgi.framework.bootdelegation 属性，请指定。此值是软件包的逗号分隔列表。
- 使用预定义的属性进行配置密码加密。有关更多信息，请参阅 通过密码加密进行保护时存在的限制（见第 1063 页）。
- 使用定制属性来定义 Web 应用程序的缺省端口。

  可以共享 server.xml，并在各种允许特定于机器或环境的定制作业的开发环境中使用 XML 配置文件。例如：

  1. 在 bootstrap.properties 文件中指定属性 default.http.port 和 default.https.port：

     ```
     default.http.port = 9081
     default.https.port = 9444
     ```

     注：如果未指定这些属性，那么缺省 HTTP 端口是 9080，而 HTTPS 端口是 9443。要覆盖缺省 HTTP 端点定义，请在服务器配置中将 `httpEndpoint` 元素的 `id` 属性设为 `defaultHttpEndpoint`。
  2. 在 server.xml 配置文件中使用以下属性：

     ```
     <httpEndpoint id="defaultHttpEndpoint"
     host="*"
     httpPort="${default.http.port}"
     httpsPort="${default.https.port}" />
     ```

     注：`host="*"` 表示“侦听所有适配器”。缺省情况下，服务器仅侦听地址 `127.0.0.1/localhost`。您还可以使用 host 属性来指定单个 IP 地址，然后系统仅侦听指定的适配器。
- 使用定制属性设置命令端口。

  设置命令端口以使服务器脚本能够与正在运行的 xigemaAS 服务器进行通信和请求某些操作，例如，停止 xigemaAS 服务器或者发出 Java™ 转储。

  缺省情况下，xigemaAS 服务器获取将由命令侦听器使用的临时端口。 您可使用 command.port 属性覆盖 xigemaAS 服务器的缺省行为。

-1

禁用命令端口。

0

在运行时选择临时端口。

1-65535

用户指定的端口。

0

在运行时选择临时端口。

注：建议您不要禁用命令端口。如果您禁用命令端口，那么将无法使用服务器脚本来请求某些操作，例如，停止 xigemaAS 服务器或发出 Java™ 转储。

- 使用定制属性配置服务器启动等待时间。

  您可以通过将 server.start.wait.time 属性添加到 boostrap.properties 文件，将服务器启动等待时间增大到超出产品缺省设置。指定 server.start.wait.time，以秒计。

  1. 在 bootstrap.properties 文件中指定 server.start.wait.time 属性。

     以下示例将服务器启动时间设置为 25 秒。

     ```
     server.start.wait.time = 25
     ```

     此设置表示服务器启动时，服务器的报告机制尝试报告启动的已完成阶段。如果服务器的报告机制在 25 秒内无法执行其功能，那么会发生错误 22。

     如果没有将 server.start.wait.time 属性添加到 bootstrap.properties 文件，那么会在内部将缺省服务器启动等待时间设置为 30 秒。
- 要应用更改，请重新启动服务器。

2.3.4 设置应用服务器的缺省主机名

可以将 defaultHostName 变量添加到 server.xml 文件，以设置用来确定 xigemaAS 服务器的缺省主机名。

此变量供服务配置使用。设置此值对于多宿主系统（例如，具有多个 NIC 和多个映射主机名的系统）而言特别重要。

👉 注:

- 此变量当前由 <httpEndpoint> host 属性和 <hostAuthInfo> rpcHost 属性使用
- 缺省主机名应该是系统的标准主机名。
- 值不应该包含任何通配符（例如 *）。
- 变量缺省值为 localhost。

1. 要设置据以确定 xigemaAS 服务器的缺省主机名，请将 defaultHostName 变量添加到 server.xml 文件。

   例如：

   ```
   <variable name="defaultHostName" value="localhost" />
   ```

2.3.5 设置缺省端口号

xigemaAS 概要文件的某些部件使用缺省 TCP/IP 端口号。可通过在服务器配置中指定另一端口号来覆盖缺省端口号。

运行时环境端口号

对于命令端口，xigemaAS 服务器获取要由命令侦听器使用的临时端口。可在 bootstrap.properties 文件中配置此端口。有关更多信息，请参阅设置应用服务器的引导属性（见第 1102 页）。

功能部件端口号

下表列示 xigemaAS 概要文件功能部件的缺省端口号及有关如何覆盖服务器配置中的缺省端口的示例。

表 15: xigemaAS 功能部件的缺省端口号

一个包含 2 列的表，此表列示 xigemaAS 的部件、其缺省端口及配置示例。

功能部件	缺省端口和配置示例
此功能部件侦听临时端口，不能配置端口号。	

servlet-3.0	• HTTP 端口：9080
	• HTTPS 端口：9443
	`<httpEndpoint id="defaultHttpEndpoint"`
	`httpPort="9082"`
	`httpsPort="9445" />`

wasJmsServer-1.0	• 入局非安全端口：7276
	• 入局安全端口：7286
	`<wasJmsEndpoint id="InboundJmsEndpoint"`
	`wasJmsPort="7278"`
	`wasJmsSSLPort="7288">`
	`</wasJmsEndpoint>`

2.3.6 使用虚拟主机

如果要隔离应用程序与处理它们的端点，那么可使用虚拟主机。

虚拟主机

虚拟主机是使单一主机能够模拟多台主机的配置实体。

虚拟主机维护它所处理的多用途因特网邮件扩展 (MIME) 类型列表。可将虚拟主机与一个或多个 Web 模块相关联，但只能将每个 Web 模块与一个虚拟主机相关联。即使多个虚拟主机共享一台物理机器，与一个虚拟主机相关联的资源也不能与另一虚拟主机所关联的资源共享数据。

每个虚拟主机有一个逻辑名和一个它已识别的一个或多个 DNS 别名组成的列表。DNS 别名是用于请求 servlet 的 TCP/IP 主机名和端口号（例如，`yourHostName:80`）的别名。如果未指定端口号，那么将采用 `80`。

虚拟主机配置使用通配符条目和端口作为它的虚拟主机条目。

- 缺省别名为 `*:80`，它使用非安全外部端口。
- 格式为 `*:9080` 的别名使用非安全内部端口。
- 格式为 `*:9443` 的别名使用安全内部端口。
- 格式为 `*:443` 的别名使用安全外部端口。

针对 Servlet、JavaServer Pages 文件或相关资源的客户机请求包含对该资源唯一的 DNS 别名和统一资源指示符 (URI)。接收到针对 servlet、JavaServer Pages 文件或相关资源的客户机请求时，DNS 别名将与所有已知虚拟主机组的列表进行比较，以查找正确的虚拟主机。该 URI 将与所有已知 URI 组的列表进行比较，以查找

正确的 URI 组。如果找到正确的虚拟主机组和 URI 组，那么会将请求发送至相应的服务器组以进行处理，并且会将响应返回给浏览器。如果找不到匹配的虚拟主机组或 URI 组，就会将错误返回给浏览器。

虚拟主机与特定节点或机器无关。它是一种配置，而不是真的对象，所以您可以创建它，但不能将它启动或停止。缺省虚拟主机 default_host 是在第一次启动应用程序服务器时自动配置的。除非您特别想将同一节点或物理机器上的资源互相隔离，否则除了缺省主机，您可能不需要任何虚拟主机。

缺省虚拟主机的 DNS 别名将配置为 `*:80` 和 `*:9080`，其中端口 80 是 HTTP 服务器端口，而端口 9080 是缺省服务器的 HTTP 传输的端口。缺省虚拟主机包括公共别名，例如机器的 IP 地址、短主机名和标准主机名称。在这些别名中，其中一个别名组成用于访问资源（例如 servlet）的路径的第一部分。例如，将在 `http://localhost:80/myServlet` 请求中使用别名 `localhost:80`。

在请求资源时，产品尝试将此请求映射到已定义的虚拟主机别名。虚拟主机的 `http://host:port/` 部分是不区分大小写的，但其后的 URL 是区分大小写的。URL 的匹配必须是字母数字精确匹配。不同的端口号将视为不同的别名。

例如，请求 `http://www.myhost.com/myservlet` 成功映射至 `http://WWW.MYHOST.COM/myservlet`，但不映射至 `http://WWW.MYHOST.COM/MYSERVLET` 或 `Www.Myhost.Com/Myservlet`。在后面两种情况下，这些映射将会因为区分大小写而失败。请求 `http://www.myhost.com/myservlet` 不能成功映射至 `http://myhost/myservlet` 或 `http://myhost:9876/myservlet`。这些映射因包含的字母数字不正确而失败。

您可以根据端口使用别名的通配符条目，并指定特定端口上的所有有效主机名和地址组合映射至特定虚拟主机。

如果您用来请求资源的别名无法映射至定义的虚拟主机的别名，那么会在用于发出请求的浏览器中接收到 404 错误。同时将出现一条消息，表明无法找到虚拟主机。

虚拟主机出现两组关联。应用程序部署将应用程序与虚拟主机相关联。虚拟主机定义将机器和 HTTP 传输的网络地址或应用程序服务器的 Web 服务器端口分配与虚拟主机相关联。查看 snoop Servlet 的 Web 客户机请求的流，例如，发生以下操作：

1. Web 客户机请求 snoop servlet 的 Web 地址为：`http://www.some_host.some_company.com:9080/snoop`
2. some_host 机器将 9080 端口分配给独立应用程序服务器 server1。
3. server1 查找虚拟主机分配以确定分配给别名 `some_host.some_company.com:9080` 的虚拟主机。
4. 应用程序服务器发现此 DNS 字符串不存在显式别名。但是，在端口号 9080 上为主机名分配的通配符 * 确实存在，所以此通配符是匹配项。定义该匹配的虚拟主机是 default_host。
5. 应用程序服务器查看部署在 default_host 上的应用程序，并找出 snoop servlet。
6. 应用程序服务器为 Web 客户机提供应用程序，并且请求方可以使用 snoop Servlet。

表 16: 虚拟主机的别名

下表为一个虚拟主机的多种别名和端口举例。

虚拟主机	别名	端口号
default_host	*	9080
default_host	localhost	9080
default_host	my_machine	9080
default_host	my_machine.my_company.com	9080

虚拟主机	别名	端口号
default_host	localhost	80

对于一个虚拟主机，您可以有任何数量的别名。您甚至可以有重叠别名，如：

应用程序服务器使用在 Web 客户机地址中指定的显式地址来寻找匹配。但是，它可能在匹配显式地址之前将匹配项解析为与模式匹配的任何其他别名。首先在别名列表中定义别名并不能保证产品寻找匹配别名时的搜索顺序。

带有重叠别名的虚拟主机。假设由于您碰巧为 admin_host 定义了端口 9080 而不是端口 9060，因此为两台虚拟主机定义了重叠的别名：

如果您对两台不同的虚拟主机使用同一别名，那么可能发生问题。例如，假设您在 default_host 上安装了缺省应用程序和 snoop servlet。您还有另一台名为 admin_host 的虚拟主机。然而，您尚未在 admin_host 上安装缺省应用程序或 snoop servlet。

表 17: 带有重叠别名的虚拟主机。

一个包含 3 列的表，列示两个主机，然后依次是每个虚拟主机的两个别名，每个别名的一个端口号。

虚拟主机	别名	端口号
default_host	*	9080
default_host	localhost	9080
admin_host	*	9060
admin_host	my_machine.com	9080

假设 Web 客户机请求得到 `http://my_machine.com:9080/snoop`。

如果应用程序服务器与针对 `*:9080` 的请求相匹配，那么会从 default_host 中提供应用程序。如果应用程序服务器与针对 `my.machine.com:9080` 的请求相匹配，那么表示找不到该应用程序。在发出该请求的浏览器中发生 404 错误。同时将出现一条消息，表明无法找到虚拟主机。

该问题产生的原因是无法在具有匹配别名的第一个虚拟主机中找到请求的应用程序。编码别名的正确方法是让入局请求中的别名只匹配所有虚拟主机定义中的一台虚拟主机。如果 URL 可以匹配多台虚拟主机，那么您将看到刚才描述的问题。

虚拟主机的用法

如果要隔离应用程序与处理它们的端点，那么可使用虚拟主机。

单个应用程序服务器通常响应来自多个不同主机和端口配置的请求。发生此情况可能有多种原因，例如，它在具有不同名称的多个网络接口上运行，或者它路由至/自 HTTP Server、代理或负载均衡器。在此情况下，您可能想要控制可从特定主机联系的应用程序。虚拟主机提供此功能。它针对所配置的主机别名列表与所请求主机名和端口号（通过 HTTP 主机头确定）匹配。

在 xigemaAS 中，缺省配置已经够用。缺省虚拟主机 (default_host) 与来自任何入局主机和端口组合的请求匹配，并将它们转发至缺省应用程序容器。

以下列表演示您配置虚拟主机时的关键配置元素。

- virtualHost 配置元素标识值。

- hostAlias 子元素配置。
- allowFromEndpoint 子元素配置（如果已使用）。
- WAR 的 ibm-web-bnd.xml 或 ibm-web-bnd.xmi 文件中的虚拟主机配置。
- httpEndpoint 的主机属性值。
- httpEndpoint 的标识属性值。

使两个应用程序相互隔离

以下示例演示虚拟主机的某个较常用用法，以帮助您了解某种必需配置。此示例显示如何配置在不同端口上运行的两个应用程序。此示例还进一步演示了仅在 localhost 接口上可用的一个应用程序。

```
<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" />
<httpEndpoint id="alternateEndpoint" host="*" httpPort="9081" />

<virtualHost id="application-1">
    <hostAlias>your_host_name:9080</hostAlias>
</virtualHost>

<virtualHost id="application-2">
    <hostAlias>localhost:9081</hostAlias>
</virtualHost>

<enterpriseApplication location="myApp.ear" name="App1"/>
<webApplication location="myApp2.war" name="App2" />
```

defaultHttpEndpoint 展示端口 9080 上的所有接口，alternateEndpoint 展示端口 9081 上的所有接口。

如果 App1 的 WAR 文件带有指定 <virtual-host name="application-1"/> 的 ibm-web-bnd.xml 文件，那么只能在 your_host_name:9080/app1_context_root 中访问此应用程序。

如果 App2（它是 WAR）具有指定 <virtual-host name="application-2"/> 的 ibm-web-bnd.xml 文件，那么只能在 localhost:9081/app2_context_root 中访问此应用程序。

如果所部署的第三个应用程序未指定特定虚拟主机，那么在此配置中，仅当它是包含 HOST 头（该头指定另一端口）的代理请求时，该应用程序才是可访问的。例如，如果在端口 80 上对代理发出了请求，那么该端口不会列示在任何 hostAlias 规范中，所以该请求应路由至 default_host 虚拟主机。

根据所请求主机或端口隔离应用程序

还将对 JMX 通信使用 xigemaAS 中的缺省虚拟主机。如果您希望将 JMX 通信与应用程序流量隔离，那么您需要完成以下步骤。

1. 决定虚拟主机名并更新应用程序以引用新（非缺省）主机。将 virtual-host 元素添加至 WAR 的 ibm-web-bnd.xml 或 ibm-web-bnd.xmi 文件。
2. 将 virtualHost 元素添加至 server.xml 文件。该名称必须与应用程序中指定的内容匹配，并且必须定义路由至新虚拟主机的 hostAliases。

   注：主机名和要匹配的端口是用户最初请求的项，它们可能与 xigemaAS 要使用的主机和端口匹配，也可能不匹配。以下示例演示已添加至 server.xml 文件的虚拟主机元素。

   ```
   <virtualHost id="proxiedRequests">
       <hostAlias>external.host.name:80</hostAlias>
       <hostAlias>external.host.name:443</hostAlias>
   </virtualHost>
   ```

   如果请求来自代理，那么此配置将对代理的主机和端口发送的任何请求单独路由至“proxiedRequests”虚拟主机。

根据发起端点限制访问

如果要将访问范围限制为使用 defaultHttpEndpoint 的缺省/系统应用程序，那么需要执行更多步骤。

1. 定义另一 httpEndpoint。以下示例演示另一 httpEndpoint。

```
<httpEndpoint id="localHostOnly" host="localhost" httpPort="9081"
 httpsPort="9444"/>
```

此 HTTP 端点指定 host="localhost"，意味着仅在 localhost 接口上展示端口 9081 和 9444。

2. 更新 virtualHost 定义以指定 allowFromEndpointRef 属性。如果指定了此属性，那么 virtualHost 仅接受来自指定端点的请求。例如：

```
<virtualHost id="default_host" allowFromEndpointRef="localHostOnly">
    <hostAlias>*:9081</hostAlias>
    <hostAlias>*:9444</hostAlias>
</virtualHost>

</virtualHost id="proxiedRequests">
    <hostAlias>*:9080</hostAlias>
    <hostAlias>*:9443</hostAlias>
    <hostAlias>external.host.name:80</hostAlias>
    <hostAlias>external.host.name:443</hostAlias>
</virtualHost>
```

通过此配置：

- default_host 虚拟主机现在仅接受发送至 localhost:9081 和 localhost:9444 并且源自 ocalHostOnly 端点的请求。指向端口 9081 和 9444 的任何其他请求被拒绝。例如，来自 defaultHttpEndpoing 并具有引用 localhost:9081 的 Host 头的请求被拒绝。
- proxiedRequests 虚拟主机现在接受向端口 9080 或 9443（它们是 defaultHttpEndpoint 使用的缺省端口）发出的任何请求，以及其 Host 头引用代理中的外部主机名和端口 80 或 443 的请求。

2.3.7 准备和运行应用程序客户机

了解如何准备服务器和客户机以从xigemaAS 应用程序客户机容器运行应用程序客户机。

成功运行应用程序客户机需要更新 `server.xml` 和 `client.xml` 文件。

1. 按如下所示准备服务器：

1. 将客户机模块 (`.jar`) 及其他模块（例如，EJB 模块 (`.jar`)）打包到应用程序 EAR 文件中。
2. 将该 EAR 文件放到 apps 目录（例如，wlp/usr/servers/*your_server*/apps）中。
3. 通过添加 `appClientSupport-1.0` 功能部件及其他必需功能部件来更新 `server.xml` 配置文件。

   重要：如果应用程序客户机是独立应用程序，那么此步骤不是必需的。

4. 通过使用您的应用程序信息配置 `<application/>` 来更新 `server.xml` 配置文件；例如：

```
<?xml version="1.0" encoding="UTF-8"?>
 <server description="new server">
  <!- Enable features ->
  <featureManager>
   <feature>javaee-7.0</feature>
  </featureManager>
  <application id="techsample" name="techSample" type="ear" location="TechnologySamples.ear"/
>
```

```
 </server>
```

2. 按如下所示准备客户机：
5. 将该 EAR 文件放到 apps 目录（例如，wlp/usr/clients/*your_client*/apps）中。
6. 通过使用您的应用程序信息配置 <application/> 来更新 *client.xml* 配置文件；例如：

```
<?xml version="1.0" encoding="UTF-8"?>
 <client description="new client">
  <!- Enable features ->
  <featureManager>
   <feature>javaeeClient-7.0</feature>
  </featureManager>
  <application id="techsample" name="techSample" type="ear" location="TechnologySamples.ear"/
>
 </client>
```

7. 可选： 将 *appClientSecurity-1.0* 功能部件 添加至 client.xml 文件。阅读有关手动创建 *xigemaAS* 应用程序客户机（见第 1110 页）的信息。
8. 启动服务器。
9. 通过输入 client run *your_client* 来运行客户机。
   如果客户机应用程序使用命令行自变量，请使用以下格式：

```
client run {your_client} -- arg1 arg2 ... argn
```

3. 如果服务器和客户机在不同计算机上运行，那么还要执行一些其他步骤。缺省情况下，服务器和客户机使用 localhost:2809。必须配置 IIOP 以在服务器与客户机之间建立连接，如下所示：
10. 停止服务器。
11. 使用 IIOP 配置更新 server.xml 文件；例如：

```
<iiopEndpoint id="defaultIiopEndpoint" host="user.host.vsettan.com"
 iiopPort="2814" />
```

12. 使用 IIOP 配置更新 client.xml 文件；例如：

```
<orb id="defaultOrb" nameService="corbaname::user.host.vsettan.com:2814" /
>
```

13. 启动服务器。

手动创建 xigemaAS 应用程序客户机

可从命令行创建 xigemaAS 应用程序客户机。

仅在 client.xml 文件中启用 Java™ EE 应用程序客户机 7.0 功能部件。

1. 打开命令行，然后将目录切换至 wlp/bin 目录。
   在以下示例中，*path_to_xigemaas* 指定您在操作系统上安装xigemaAS 的位置。

   Windows™ 系统上的示例：C:\Users\mo> cd *path_to_xigemaas*\wlp\bin

   Linux™ 上的示例：mo@*machine01*:~> cd *path_to_xigemaas*/wlp/bin
2. 运行以下命令以创建客户机，其中 *client_name* 是您要给予客户机的名称。如果未指定客户机名称，那么会使用 defaultClient。

   Windows™ 系统上的示例：C:\wlp\bin> client create *client_name*

Linux™ 上的示例：mo@*machine01*:~> client create *client_name*

```
client create client_name
```

如果已成功创建客户机，那么会收到以下消息：

```
已创建客户机 client_name。
```

可在 wlp/usr/clients/*client_name* 目录中找到 client.xml 文件。此文件包含 javaeeClient-7.0 功能部件。

⚠ 注意：如果缺省客户机存在，那么您将收到错误。如果缺省客户机不存在，那么系统会创建 defaultClient。

3. 通过准备其中带有客户机模块 (.jar) 的应用程序 (.ear) 文件来运行客户机应用程序。
   在客户机模块的 MANIFEST.MF 中指定 main 类，例如：

```
Manifest-Version: 1.0
Main-Class: com.ibm.ws.addressbook.ContactServiceClient_XMLInject
```

4. 将该 EAR 文件放置到 wlp/usr/clients/*client_name*/apps 目录下。
5. 更新 client.xml 文件以配置应用程序，例如：

```
<client>
 <featureManager>
  <feature>javaeeClient-7.0</feature>
 <featureManager>
 <application id="CLIENT_APP" name="CLIENT_APP" type="ear"
 location="clientApp.ear"/>
</client>
```

如果所指定客户机已存在，那么不会创建客户机，并且您会接收到异常消息：

```
CWWKE0005E: 未能启动运行时环境。
CWWKE0904E: 无法创建名为 client_name 的客户机，因为
客户机目录 C:\wlp\usr\clients\client_name 已存在。
```

可通过将 appSecurityClient-1.0 功能部件添加至 client.xml 文件来对应用程序客户机启用安全性（SSL、CSIv2 和 JAAS）：

```
<featureManager>
 <feature>javaeeClient-7.0</feature>
 <feature>appSecurityClient-1.0</feature>
</featureManager>
```

有关在应用程序客户机上配置安全性的更多信息，请参阅为 *xigemaAS* 应用程序客户机容器及其应用程序配置安全性（见第 1395 页）。

创建带有多个客户机模块的 xigemaAS 应用程序客户机

可在同一 EAR 文件中创建带有多个客户机模块的 xigemaAS 应用程序客户机。

可在应用程序 EAR 文件中指定多个客户机应用程序（打包在客户机模块中）。如果想要打包同一 EAR 文件中的多个客户机应用程序，那么必须在 <enterpriseApplication/> 中使用 defaultClientModule 属性。

1. 通过在 client.xml 和 server.xml 文件中更新 defaultClientModule 属性来指定要运行的客户机。

```
<client>
  <featureManager>
   <feature>javaeeClient-7.0</feature>
  <featureManager>
  <enterpriseApplication id="MultipleAppClientModules" name="MultipleAppClientModules"
 type="ear"
   defaultClientModule="HelloAppClient.jar" location="MultipleAppClientModules.ear"/>
</client>
```

👉 重要：可一次运行一个应用程序。

2.3.8 用于设置 xigemaAS 的平台即服务环境注意事项

平台即服务 (PaaS) 环境（例如，IBM® Bluemix™、Pivotal Cloud Foundry 和 OpenShift Enterprise）提供对应用程序实例的管理和监视，但还具有一些限制。由于 PaaS 环境的内部特征，一些 xigemaAS 功能部件冗余或行为不同，因而不受支持。

xigemaAS 服务器管理限制

在 PaaS 环境中，不支持以下管理功能部件：adminCenter-1.0

- adminCenter-1.0
- clusterMember-1.0
- collectiveController-1.0
- collectiveMember-1.0
- dynamicRouting-1.0
- healthAnalyzer-1.0
- healthManager-1.0
- scalingController-1.0
- scalingMember-1.0

文件系统限制

多数 PaaS 环境不会向其应用程序提供持久的本地文件系统。对于 xigemaAS，这会影响服务器中的应用程序和组件，这些应用程序和组件在本地写入数据并预期在服务器 JVM 重新启动期间持久存储数据。

事务中涉及多个资源管理器时，xigemaAS 事务管理器会向本地文件系统写入日志文件。如果在 JVM 发生故障并重新启动后日志不可用，那么无法自动完成事务，必须手动解决以解锁数据，并使数据在资源管理器中保持一致。为避免出现此场景，xigemaAS buildpack 或 cartridge 阻止写入事务日志记录，并向应用程序发起异常以阻止列出第二个资源。因此，尽管您仍可将事务与单个 XA 资源配合使用，也无法在事务中列出第二个事务资源。此外，由于 Web Service 原子事务始终写入日志记录，因此无法使用。

如果 PaaS 环境提供持久存储器，那么可通过从 JVM 配置除去以下 Java™ 属性，修改xigemaAS buildpack 或 cartridge 来启用两阶段事务：

```
-Dcom.ibm.tx.jta.disable2PC=true
```

以下功能部件依赖于持久本地存储器：

- wsAtomicTransaction-1.2

- 使用事务的其他功能部件，依赖于应用程序行为

网络限制

通常，PaaS 路由器不支持因特网 ORB 间协议 (IIOP) 流量，因此无法使用针对 Enterprise JavaBeans™ (EJB) 组件的远程请求。以下功能部件依赖于 IIOP 传输：

- `appClientSupport-1.0`
- `appSecurityClient-1.0`
- `ejbRemote-3.2`

在一些情况（例如，SSL 在路由器处终止）下，xigemaAS 依赖于 HTTP 头描述原始客户机请求的各个方面。在 PaaS 环境中使用 SSL 时，这些头必须由 PaaS 路由器设置。在 IBM® Bluemix™ 上，已设置这些头，以便您可使用 `ssl-1.0` 功能部件和依赖于它的任何功能部件，而不进行更改。

以下功能部件需要路由器设置 HTTP 头：

- `ssl-1.0`
- 依赖于 `ssl-1.0` 的其他功能部件，如*Secure Socket Layer*（见第 1021 页） 的 «**“用于启用此功能部件的功能部件”**» 部分中所列出

2.4 管理 xigemaAS

服务器配置由 server.xml 文件、bootstrap.properties 文件以及这两个主要配置文件随附的任何可选文件组成。没有用于 xigemaAS 的管理控制台，但您可使用管理中心以通过智能手机、平板或计算机上的 Web 浏览器管理 xigemaAS 服务器和应用程序及其他资源。

根据异常来配置 xigemaAS 。运行时环境会从一组内置配置缺省设置进行操作，并且您只需要指定用来覆盖那些缺省设置的配置。如果要这样做，请在运行时编辑 server.xml 文件或者 server.xml 中包括的另一个 XML 文件。

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。 它们是使服务器可组合的主机制。您在服务器配置中指定的功能部件列表会提供功能性服务器。

第一次安装并启动服务器时，有功能部件管理器及缺省服务器配置可供使用：

- 缺省情况下，服务器包含 `jsp-2.2` 功能部件来支持 servlet 和 JSP 应用程序。可以使用功能部件管理器来添加所需要的功能部件。
- 根据异常来配置服务器。指定所需要的功能部件时，那些功能部件的缺省配置会提供一个设计成涵盖最常见需求的丰富环境，因此您只需要指定对缺省配置的更改。

2.4.1 手动管理 xigemaAS

可以从命令提示符管理 xigemaAS，使用管理中心来配置 xigemaAS，以及捕获 xigemaAS 的状态。可以将 xigemaAS 服务器配置及其中运行的应用程序打包，以分发给同事，或者安装在其他系统上。如果可用，那么可以使用 Equinox OSGi 控制台来辅助调试。

server.xml 文件是服务器的主要配置文件。可以在文本编辑器中编辑此文件及其随附的文件。您也可以更改 server.xml 文件的位置。但是，对于大多数配置，您不需要这样做。

bootstrap.properties 文件用于指定在处理主要配置之前必须提供的属性。如果更新 bootstrap.properties 文件，那么必须重新启动服务器才能使更改生效。

可以在 server.xml 文件中配置的所有元素，以及该文件随附的文件，在 *xigemaAS* 功能部件（见第 906 页） 中作了描述。但是，唯一的必需元素是服务器定义：

```
<server/>
```

除了此服务器定义之外，您仅指定缺省配置值的覆盖项和新增项。例如，要更改事务超时值，您指定：

```
<transactions timeout="30" />
```

某些属性可以具有多个值。例如，使用值列表来定义要由服务器提供的功能部件：

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

另请参阅 添加和移除 *xigemaAS* 功能部件（见第 1129 页）。

如果可以在某一位置配置多个同一资源类型的实例（例如应用程序或数据源），那么只需提供对于该资源唯一的属性。可以根据需要，让其他属性使用缺省值，或者覆盖这些缺省值。因此，server.xml 文件的内容可以很简洁。例如，这里是运行 Web 应用程序的完整服务器配置：

```
<server>
     <featureManager>
         <feature>servlet_3.0</feature>
     </featureManager>
     <application name="snoop" location="/mywebapps/snoop" id="snoop" type="war"/>
</server>
```

有关特定服务器配置方面的详细信息，请参阅子主题。

定制 xigemaAS 环境

可以通过使用某些特定变量来定制 xigemaAS 环境，以支持在只读文件系统中放置产品二进制文件和共享资源。

可在 server.env 文件中配置以下列表中特定于 xigemaAS 的环境变量，以定制 xigemaAS 环境。`${wlp.install.dir}` 配置变量具有推断位置，它始终设置为其中包含启动脚本的目录的父目录。

- `WLP_USER_DIR`

  可以使用此环境变量来指定 `${wlp.user.dir}` 的替代位置。此变量必须是一个绝对路径。如果指定了此变量，那么运行时环境会在所指定的目录中查找共享资源和服务器定义。`${server.config.dir}` 等价于 `${wlp.user.dir}`/servers/serverName。如果未指定此环境变量，那么 `${wlp.user.dir}` 设置为 `${wlp.install.dir}/usr`。
- `WLP_OUTPUT_DIR`

  可以使用此环境变量来指定服务器生成输出（例如日志、workarea 目录和生成文件）的替代位置。logs 目录中的文件可能包括 console.log、messages.log 和所生成的任何 FFDC 文件。所生成的文件可以包括使用 `server dump` 或 `server javadump` 命令创建的服务器转储。此变量必须是一个绝对路径。如果指定此环境变量，那么会将 `${server.output.dir}` 设置为 `WLP_OUTPUT_DIR`/serverName 的等价项。如果未指定此环境变量，那么 `${server.output.dir}` 与 `${server.config.dir}` 相同。

  当使用此服务器命令时，服务器进程将使用输出目录作为其当前工作目录。

- WLP_DEBUG_ADDRESS

  以调试方式运行服务器时，可以使用此环境变量来指定替代端口。缺省值为 7777。
  在服务器命令中以调试方式运行xigemaAS 时，设置了以下值：JAVA_DEBUG="-Dwas.debug.mode=true -Dcom.ibm.websphere.ras.inject.at.transform=true -agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=${WLP_DEBUG_ADDRESS}"。但是，如果从 ws-server.jar 可执行 JAR 文件或嵌入式 xigemaAS 服务器 SPI 运行 xigemaAS，那么必须使用相同设置以为 xigemaAS 启用调试方式。

可在 server.env 文件中指定 WLP_OUTPUT_DIR、WLP_USER_DIR 和 WLP_DEBUG_ADDRESS 环境变量。还可以在 jvm.options 文件中指定 JVM 选项。仅当使用服务器管理脚本时，server.env 和 jvm.options 文件才起作用。如果使用 ws-server.jar 可执行 JAR 文件来启动服务器，那么不支持这些文件。

- 使用 server.env 文件来指定环境变量。

  可以使用安装级和服务器级 server.env 文件来指定环境变量，例如 JAVA_HOME、WLP_USER_DIR 和 WLP_OUTPUT_DIR。例如：

```
# Use a specific Java binary
  JAVA_HOME=/opt/xigemaas/java-i386-60/jre
# JAVA_HOME=c:\Java
```

  注：

  - server.env 文件仅支持 key=value 对。
  - 将忽略空行和以 # 字符开头的行。
  - 没有转义字符；所有字符都是文字，包括反斜杠、前导空格和拖尾空格。
  - 等号“=”周围不应有空格。
  - 不支持 shell 和变量扩展。
  - 只能在 ${wlp.install.dir}/etc/server.env 文件中指定 WLP_USER_DIR，因为此变量的用途是指定其余配置所在的位置。找到以及合并其余配置之后，其他位置中的任何进一步配置都不符合预期，也不受支持。

  服务器管理脚本会在下列两个位置搜索 server.env 文件：${wlp.install.dir}/etc/server.env 和 ${server.config.dir}/server.env。如果两个文件都存在，那么会合并这两个文件的内容；服务器级别文件中的值优先于运行时级别文件中的值。

  还可以在 shell 环境中指定这些环境变量，但 server.env 文件优先于这些变量。

- 使用 jvm.options 文件来定制 JVM 选项。

  可通过运行时级别和服务器级别来使用 jvm.options 文件，以指定更多服务器启动选项，例如 -X 参数。通过服务器管理脚本来执行启动、运行和调试操作时应用这些选项。确保每行只指定一个选项。例如：

```
# Set the maximum heap size to 1024m.
-Xmx1024m

# Set a system property.
-Dcom.Vsettan.example.system.property=ExampleValue

# Enable verbose output for class loading.
-verbose:class

# Enable verbose garbage collection.
-verbose:gc

# Specify an alternate verbose garbage collection log on IBM Java Virtual Machines only.
-Xverbosegclog:verbosegc.log
```

```
# Specify additional verbose garbage collection options on HotSpot Java Virtual Machines
 only.
-Xloggc:verbosegc.log
-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps
-XX:+PrintHeapAtGC
```

服务器管理脚本会在下列两个位置搜索 jvm.options：`${wlp.install.dir}`/etc/jvm.options 和 `${server.config.dir}`/jvm.options。如果这两个文件都存在，那么会使用 `${server.config.dir}`/jvm.options 文件中的选项。

注：

- 将忽略空行和以 # 字符开头的行。
- 没有转义字符；所有字符都是文字，包括反斜杠、前导空格和拖尾空格。
- 等号“=”周围不应有空格。
- 不支持 shell 和变量扩展。

如果使用服务器脚本启动 xigemaAS 服务器，那么当前会话中的所有操作系统环境变量都可用。如果通过使用 MBean 或管理中心启动服务器，那么只有对该系统上正在运行的远程命令可用的环境变量是可用的变量。

从命令行管理 xigemaAS

可以使用 `server` 命令和 ws-server.jar 可执行 JAR 文件来创建、启动或停止服务器，检查服务器是否正在运行，或者调试服务器。

`wlp/bin` 目录包含一个称为 `server` 的脚本来帮助控制服务器进程。此脚本的语法如下所示：

```
server <action> [server] [options]
```

有关 options 的可用值，请参阅 服务器命令选项（见第 1118 页）。

此脚本支持下列操作：

create

此命令用来创建新服务器。

run

此命令用来在前台启动服务器。

debug

此命令在调试器连接至调试端口之后在控制台前台运行指定的服务器。缺省端口为 7777。可以使用 `WLP_DEBUG_ADDRESS` 变量来指定替代端口。

dump

此命令用来创建服务器的快照并将结果保存到归档文件以便进一步调优和诊断。

javadump

用于创建服务器 *Java*™ 虚拟机 *(JVM)* 的快照并将结果保存到文件中的命令。每种转储类型都将创建一个文件，但是并非所有转储类型都受所有虚拟机支持。转储文件的缺省目录为 `${server.output.dir}`。要设置不同缺省目录，必须使用 IBM® JVM 并设置以下环境变量：

- IBM_HEAPDUMPDIR
- IBM_COREDIR

- IBM_JAVACOREDIR

package

此命令用来将服务器打包。

start

此命令用来启动服务器作为后台进程。

stop

此命令用来停止正在运行的服务器。

status

此命令用来检查指定的服务器是否处于运行状态。

version

此命令用来显示当前服务器和 Java™ 运行时环境的版本信息。

help

此命令用来获取命令行脚本帮助，包括其他选项的详细信息。

👉 注：如果未在命令行上指定服务器，那么会对缺省服务器实例 `defaultServer`（如果存在）执行操作。

您也可以通过使用 `${wlp.install.dir}/bin/tools` 目录中的可执行 JAR 文件 ws-server.jar 来执行相似的操作。

要在 Windows™ 系统上运行 `server` 脚本，请输入以下命令：

```
server.bat create server_name
server.bat package server_name
server.bat run server_name
server.bat help server_name
```

要在其他系统上运行 `server` 脚本，请输入以下命令：

```
server create server_name
server package server_name
server run server_name
server help server_name
```

要在不使用 `server` 脚本的情况下运行可执行 JAR 文件 ws-server.jar，请输入以下命令：

```
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar server_name --create
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar server_name
java -javaagent:bin/tools/ws-javaagent.jar -jar bin/tools/ws-server.jar --help
```

`--help` 选项为可执行 JAR 文件 ws-server.jar（例如 `--stop`、`--version`、`--clean` 和 `--include`）提供有关其他命令行参数的信息。

服务器命令选项

server 命令支持启动、停止、创建、打包和转储xigemaAS概要文件服务器。此主题描述了可以与 server 命令以及等价的可执行 JAR 文件 ws-server.jar 一起使用的所有可用选项和退出码。

语法

命令语法如下所示：

```
server action serverName [options]
```

其中 action 的值表示您可以在 xigemaAS 概要文件服务器上执行的操作。请参阅可以从命令提示符使用的 *xigemaAS*概要文件管理操作。

👉 注：如果未在命令行上指定服务器，那么会对缺省服务器实例 defaultServer（如果存在）执行操作。

选项

下列选项可用于 server 命令：

--archive="*path_to_the_target_archive_file*"

为 *package* 或 *dump*操作指定目标文件。此路径可以是相对路径（相对于 xigemaAS概要文件的安装根目录），也可以是绝对路径。缺省归档目标是具有服务器名称的压缩文件，将存储在安装根目录中。如果值包含空格，请使用引号将值引起来。可以将此选项用于 *package* 和 *dump* 操作。

如果您为归档文件名指定 .jar 扩展名，那么 server 命令将创建一个新的自抽取归档文件，可以使用 java 命令从此归档文件来安装 xigemaAS 概要文件服务器；有关更多信息，请参阅通过解压 *JAR* 文件来安装 *xigemaAS*（见第 1088 页）。

--clean

清除与所指定的服务器实例相关的所有持久高速缓存的信息，其中包括 OSGi 解析器元数据和持久 OSGi 捆绑软件数据。如果您使用此选项，那么服务器在下一次启动时将需要重新计算任何已高速缓存的数据，这可能会比重新启动耗用更多时间，重新启动时可以复用已高速缓存的数据。

👉 注：对于正常操作，不需要此选项。如果提供临时修订，或者存在与已高速缓存的数据有关的可疑问题，那么 xigemaAS 服务可能会请求您使用此选项。如果您正在开发产品扩展，并且正在更新 OSGi 清单或者打算清除持久 OSGi 捆绑软件数据，那么可能也需要此选项。

--include=*package_option*

指定要打包的文件，其中 *package_option* 可以采用下列其中一个值：

- all 指定要将 xigemaAS 安装目录中的所有文件打包。如果 ${WLP_USER_DIR} 和 ${WLP_OUTPUT_DIR} 是在 *server.env* 文件中定义，那么会将其中的文件打包。此值仅适用于 *package* 操作。
- usr 指定要将 ${WLP_USER_DIR} 目录中的文件打包。此值仅适用于 *package* 操作。
- minify 指定仅打包运行时环境中运行服务器所需的那些部分及 ${WLP_USER_DIR} 目录中的文件，以最大程度地减小所生成归档的大小。此值仅适用于 *package* 操作。

--include=*diagnose_option,diagnose_option,...*

指定要捕获的诊断信息类型。--include 的值是用逗号定界的列表，该列表可以包含下列任何值：

- `heap` 用于帮助诊断超出内存消耗和内存泄漏，这会显示内存中的实时对象及其之间的引用。在 IBM® J9 虚拟机上，产生的文件名称为 heapdump.*date.time.processID.sequenceNumber*.phd。在 HotSpot 虚拟机上，产生的文件名称为 java.*date.time.processID.sequenceNumber*.hprof。此值适用于 *dump* 和 *javadump* 操作。
- `system` 也用于帮助诊断超出内存消耗和内存泄漏，但它们还可用来查找虚拟机中的缺陷。这些转储仅在 IBM® J9 虚拟机上受支持。产生的文件的名称为 core.*date.time.processID.sequenceNumber*.dmp。此值适用于 *dump* 和 *javadump* 操作。
- `thread` 用于帮助诊断挂起的线程和死锁，有时可用于诊断 CPU 过度使用问题。这些转储总是可以使用 `server javadump` 命令来创建。在 IBM® J9 虚拟机上，产生的文件名称为 javacore.*date.time.processID.sequenceNumber*.txt。在 HotSpot 虚拟机上，产生的文件名称为 javadump.*date.time.processID.sequenceNumber*.txt。此值也适用于 *dump* 操作。

  注：仅当正在 Java™ SDK 上运行服务器时，才支持线程转储类型。如果服务器使用 JRE 启动，那么系统将报告错误来指示服务器不支持该转储类型。此限制仅适用于 HotSpot 虚拟机；线程 Java™ 转储类型受任何 IBM® JVM（JRE 或 SDK）支持。

--os=*os_value,os_value,...*

指定您希望所打包的服务器支持的操作系统。请提供以逗号分隔的列表。缺省值为 `any`，表明可将服务器部署到该源支持的任何操作系统。

要指定某个操作系统不受支持，请在它前面添加减号 ("-")。有关操作系统值的列表，请参阅位于以下 URL 的 OSGi Alliance Web 站点：*http://www.osgi.org/Specifications/Reference#os*。

此选项仅适用于 *package* 操作，并且只能与 `--include=minify` 选项一起使用。如果排除操作系统，那么稍后对归档重复 `minify` 操作时，不能包含操作系统。

服务器进程

使用在 server.env 文件中指定的环境变量来创建服务器进程。缺省情况下，添加了下列 JVM 选项：

- 跟踪、监视和其他服务器功能需要使用 -javaagent:wlp/bin/tools/ws-javaagent.jar 选项。
- -Xshareclasses 和相关选项允许在受支持的 IBM® J9 虚拟机上进行共享类高速缓存。高速缓存目录设置为 *WLP_OUTPUT_DIR*/.classCache。
- 针对 Java™ 8 之前的版本，-XX:MaxPermSize 选项将增大永久生成 HotSpot 虚拟机的大小。可将 WLP_SKIP_MAXPERMSIZE 环境变量设置为 `true` 以避免使用此缺省选项，这将避免出现如下警告：

```
Java HotSpot(TM) Client VM warning: ignoring option MaxPermSize=256m;
 support was removed in 8.0
```

可以使用 jvm.options 文件来覆盖这些缺省 JVM 选项或者添加其他 JVM 选项。有关 server.env 和 jvm.options 文件的更多信息，请参阅定制 *xigemaAS* 环境（见第 1114 页）。

缺省情况下，`server` 命令将设置 umask 值，以在运行该操作之前拒绝"其他"用户的所有许可权。但是，您可以将 WLP_SKIP_UMASK 环境变量设置为 `true` 以避免设置 umask 值。

服务器进程的当前工作目录设置为服务器输出目录。

启动服务器时，`server` 命令创建进程标识 (PID) 文件，停止服务器时，该命令删除 PID 文件。缺省情况下，PID 文件设置为 WLP_OUTPUT_DIR/.pid/serverName.pid。可通过设置 PID_FILE 环境变量更改 PID 文件的绝对路径，也可通过设置 PID_DIR 环境变量更改 PID 目录的绝对路径。

当使用 run 和 debug 操作时，服务器进程产生的标准输出和错误将输出到前台控制台；当使用 start 操作时，缺省情况下会将标准输出和错误重定向到 *WLP_OUTPUT_DIR*/serverName/logs/console.log 文件。可以通过设置 LOG_FILE 环境变量来更改日志名称，可以通过设置 LOG_DIR 环境变量来更改日志目录。有关日志记录配置的更多信息，请参阅 日志记录和跟踪（见第 1649 页）。

stop 操作会阻止新的应用程序请求进入服务器，这允许现有请求有时间完成。此时间过去后，余下服务器组件停止，服务器进程退出。未在允许时间内完成的应用程序请求将失败，但它们的确切行为取决于它们在服务器组件停止时的活动。

退出码

下列退出码适用于 server 命令以及等价的可执行 JAR 文件 ws-server.jar：

0

正常。0 指示已成功完成所请求的操作。对于服务器状态，0 指示服务器处于运行状态。

1

对于服务器状态，1 指示服务器未处于运行状态。对于其他操作，它指示调用了多余的操作。例如，启动已启动的服务器，或者停止已停止的服务器。如果使用了无效的 Java™ 选项，那么 JVM 也可能会返回此代码。

2

服务器不存在。

3

对运行中服务器调用了不受支持的操作。例如，调用打包操作时，服务器处于运行状态。

4

对已停止的服务器调用了不受支持的操作。例如，调用转储操作时，服务器未处于运行状态。

5

未知的服务器状态。例如，缺少 workarea 目录，或者连接 API 不起作用。

>=20

返回码大于或等于 20 时表明执行所请求的操作时发生错误。系统将显示消息并将它们及有关该错误的更多信息捕获到日志文件中。

用法

以下示例说明了正确的语法：

```
server run
server start myserver --clean
server package myserver --archive="archivefile.zip" --include=all
server dump myserver --archive="c:\mybackup\myserver.zip" --include=thread
server javadump myserver
server javadump myserver --include=heap,system
```

应用程序客户机命令

client 命令支持创建、运行、调试、打包和帮助操作。

语法

wlp/bin 目录包含名为 client 和 client.bat 的脚本以帮助运行客户机应用程序。这些脚本的语法如下所示：

```
client <action> client_name [options]
```

其中 action 的值表示您可对应用程序客户机执行的操作。

注意：如果未在命令行上指定应用程序客户机名称，那么将对缺省应用程序客户机实例 defaultClient（如果存在）执行该任务。

操作

client 脚本支持以下操作：

create

用于创建新应用程序客户机的命令。

run

用于在前台启动应用程序客户机的命令。

debug

在调试器连接至调试端口之后，用于在控制台前台运行指定应用程序客户机的命令。缺省端口为 7778。

package

用于打包应用程序客户机的命令。

help

用于获取命令行脚本帮助（包括其他选项的详细信息）的命令。

选项

client 脚本支持以下选项：

--archive="path to the target archive file"

指定打包操作要生成的归档目标。可将目标指定为绝对路径或相对路径。如果省略此选项，那么将在客户机输出目录中创建归档文件。目标文件名扩展名可能影响所生成归档的格式。打包操作的缺省归档格式为 zip。jar 归档格式生成类似原始安装程序归档的自解压 JAR 文件。

--clean

清除与此客户机实例相关的所有缓存信息。

--include=value,value,...

以逗号定界的值列表。有效值根据操作不同而变化。

template="templateName"

指定创建新客户机时要使用的模板的名称。

--autoAcceptSigner

自动接受签署者证书并存储在客户机信任库中，而不显示要求检查服务器证书的提示。

示例

以下示例显示可在 Windows™ 系统上运行的应用程序客户机命令操作：

```
client.bat create client_name
client.bat run client_name
client.bat help
```

以下示例显示可在其他系统上运行的应用程序客户机命令操作：

```
client create client_name
client run client_name
client help
```

从命令行运行应用程序客户机

可使用客户机运行任务来运行应用程序客户机。

必须先创建客户机并在 `client.xml` 文件中为客户机应用程序添加配置，才能运行应用程序客户机。有关如何创建应用程序客户机的示例，请参阅手动创建 *xigemaAS* 应用程序客户机（见第 1110 页）。

`wlp/bin` 目录包含名为 `client` 和 `client.bat` 的脚本以运行客户机应用程序。这些脚本的语法如下所示：

```
client action <client_name> [options]
```

- 使用以下命令来运行客户机：

  ```
  client run client_name
  ```

  ```
  client.bat run client_name
  ```

  其中 *client_name* 是客户机的名称。

  注意：如果缺省客户机存在，那么它会运行 `defaultClient`。如果缺省客户机不存在，那么系统会创建 `defaultClient` 然后运行（并且可能失败，因为未配置应用程序）。

运行 ddlGen 实用程序

如果服务器配置中有需要访问数据库的功能部件，那么您可生成数据定义语言 (DDL)。

运行 ddlGen 实用程序前，必须启动服务器。

此实用程序对服务器中配置的需要访问数据库的每个功能部件生成数据定义语言 (DDL)。通过在运行 ddlGe 实用程序的命令行中导出环境变量 *WLP_USER_DIR*，可更改 ddlGen 实用程序用于搜索服务器的路径。

1. 在 server.xml 文件中，在 featureManager 标记下添加 `localConnector-1.0` 功能部件。

   ```
   <featureManager>
        <feature>localConnector-1.0</feature>
   </featureManager>
   ```

2. 在命令行中，运行 `wlp/bin/ddlGen {generate|help} <server_name>` 命令，其中 `<server_name>` 是您要为其生成 DDL 的服务器的名称。

下表显示可能返回的非零代码：

表 18: ddlGen 实用程序的返回码和说明

返回码和说明

返回码	说明
0	成功。此 DDL 生成至 ${server.output.dir}/ddl。
20	所提供操作无效。
21	找不到服务器。消息 CWWKD0100E 显示此实用程序在其中查找服务器的文件系统目录。可通过在运行该实用程序的命令行中导出变量 WLP_USER_DIR 以更改此位置。
22	localConnector 功能部件不在服务器配置中，或者服务器未启动。
23	找不到生成 DDL 的 MBean。
24	生成 DDL 的 MBean 报告错误。服务器日志包含有关该错误的更多详细信息。
255	发生意外错误。

从命令行生成 xigemaAS 配置模式

语法

命令语法如下所示：

```
java [JVM options] -jar ws-schemagen.jar [options] outputFile
```

选项

提供了下列选项：

--encoding=charset

其中 *charset* 是您在创建输出文件时要使用的字符集。

--ignorePidsFile=fileName

其中 *fileName* 是文件名，该文件包含要忽略的 PID 列表。

--locale=language

其中 *language* 指定您在创建输出文件时要使用的语言。此字符串由 ISO-639 两字母小写语言代码（还可以选择后面跟着下划线）和 ISO-3166 大写两字母国家或地区代码组成。

用法示例

以下示例生成所安装产品的模式，并使用巴西葡萄牙语将其存储在称为 schema.xsl 的文件中：

```
java -jar ws-schemagen.jar schema.xsl
```

以下示例生成所安装产品的模式，并使用巴西葡萄牙语将其存储在称为 schema.xsl 的文件中：

```
java -jar ws-schemagen.jar schema.xsl --locale=pt_BR
```

以下示例显示帮助信息：

```
java -jar ws-schemagen.jar --help
```

从命令行生成 xigemaAS 服务器转储

在命令行中，可以使用 `server dump` 或 `server javadump` 命令来捕获 xigemaAS 服务器的状态信息。

可以使用 `server dump` 命令对 xigemaAS 服务器进行问题诊断，因为结果文件包含服务器配置、日志信息以及 workarea 目录中所部署应用程序的详细信息。可以将该命令应用到正在运行或已停止的服务器。

对于正在运行的服务器，也包含下列信息：

- 服务器中每个 OSGi 捆绑软件的状态
- 服务器中每个 OSGi 捆绑软件的连线信息
- 服务组件运行时 (SCR) 环境所管理的组件列表
- SCR 中每个组件的详细信息
- 每个 OSGi 捆绑软件的配置管理数据
- 所注册 OSGi 服务的相关信息
- 运行时环境设置，例如 Java™ 虚拟机 (JVM)、堆大小、操作系统、线程信息和网络状态

`server javadump` 命令可用来诊断 JVM 级别的问题，例如，挂起的线程、死锁、过多处理、超出内存消耗、内存泄漏以及虚拟机缺陷。只能在正在运行的服务器上使用该命令。每种转储类型都将创建一个文件，但是并非所有转储类型都受所有虚拟机支持。请参阅服务器命令选项（见第 1118 页）。转储文件的缺省目录为 `${server.output.dir}`。要设置不同的缺省目录，您必须使用 IBM® JVM 并设置下列环境变量：

- IBM_HEAPDUMPDIR
- IBM_COREDIR
- IBM_JAVACOREDIR

1. 打开命令行，然后将目录切换至 `wlp/bin` 目录。
2. 使用下列其中一个命令行工具来捕获状态信息。如果未指定服务器名称，那么会使用 defaultServer。
   - 要创建服务器状态的快照，请使用 `server dump` 命令。
     - ```
       server dump server_name --archive=package_file_name.dump.zip --include=heap
       ```

       其中 *package_file_name*.dump.zip 是您选择的文件名。此文件名可以包含完整路径名。如果省略完整路径，那么会在缺省目录 *${server.output.dir}* 中创建一个名为 *package_file_name*.dump.zip 的压缩文件。

     `--include` 参数是可选的。可以请求其他内存转储类型。例如，`--include=heap` 选项会请求堆转储；`--include=thread,heap,system` 选项会请求线程转储、堆转储和系统转储。
   - 要创建 JVM 状态的快照，请使用 `server javadump` 命令。
     - ```
       server javadump server_name --include=heap
       ```

     `--include` 参数是可选的。可以请求其他内存转储类型。例如，`--include=heap` 选项会请求堆转储；`--include=heap,system` 选项会请求堆转储和系统转储。输出文件在缺省目录 *${server.output.dir}* 中创建。要设置不同缺省目录，必须使用 IBM® JVM 并设置 IBM_HEAPDUMPDIR、IBM_COREDIR 和 IBM_JAVACOREDIR 环境变量。

注：通过将 UTF-8 编码用于条目名称来创建结果文件，因此用来打开该文件的工具必须能够将 UTF-8 编码用于条目名称。Java™ SDK 中的 `jar` 命令使用此格式。

如果指定的服务器不存在，那么命令会失败。如果指定的服务器存在，那么创建的结果文件包含服务器的状态信息。

从命令行打包 xigemaAS 服务器

可从命令行创建压缩文件，该文件包含 xigemaAS 运行时环境、共享资源目录中的文件、特定服务器以及服务器中嵌入的应用程序。您也可以选择将运行时二进制文件从压缩文件中排除。

xigemaAS 服务器属于轻量级，因此您很容易将服务器安装打包在压缩文件中。可以存储此包，将包分发给同事，使用包来将安装部署到不同位置或另一台机器，甚至将安装嵌入产品分发中。

👉 注：通过将 UTF-8 编码用于条目名称来创建结果文件，因此用来打开该文件的工具必须能够将 UTF-8 编码用于条目名称。Java™ SDK 中的 `jar` 命令使用此格式。

1. 要从命令行打包 xigemaAS 服务器，请完成以下步骤：
1. 打开命令行，然后将目录切换至 `wlp/bin` 目录。
2. 停止服务器。
3. 运行 `package` 命令以创建软件包。

   可打包xigemaAS 服务器或运行时。

   - 打包 xigemaAS 服务器。

     缺省归档格式为 .zip（在所有平台上）。您还可生成 .jar 归档。

     如果未指定服务器名称，那么会使用 defaultServer。如果未指定 --archive 参数，那么将 *server_name* 的值用于 *package_file_name*，而且会在 *${server.output.dir}* 目录中创建压缩文件。

     为环境选择正确命令。

     - 使用此命令生成 .zip 归档。

       ```
       server package server_name --archive=package_file_name.zip --include=all
       ```

       其中 *package_file_name*`.zip` 是您选择的文件名。此文件名可以包含完整路径名。如果省略完整路径，那么会在 *${server.output.dir}* 目录中创建一个称为 *package_file_name*.zip 的压缩文件。
     - 使用此命令生成 .jar 归档。.jar 归档的优势在于：`bin` 目录中的脚本保持其许可权，以便在安装软件包时可执行这些脚本。

       ```
       server package server_name --archive=package_file_name.jar --include=all
       ```

       其中，*package_file_name*`.jar` 是所选的文件名。

       有关针对此归档文件的解压选项的更多信息，请参阅*JAR 文件解压选项*（见第 1089 页）。

     您也可以将 `--include` 选项与此命令配合使用。例如，`--include=all` 选项会对 `${WLP_USER_DIR}` 目录中的运行时二进制文件及相关文件进行打包；`--include=usr` 选项仅对 `${WLP_USER_DIR}` 目录中的相关文件进行打包，从而有效地将运行时二进制文件从压缩文件中排除。

     `--include=usr` 选项不适用于 .jar 归档格式。

     如果您使用 `--include=minify` 选项，那么 `server` 命令将仅打包在运行服务器时所需要的运行时环境的那些部件以及 `${WLP_USER_DIR}` 目录中的文件。此选项将显著减小最终获得归档的大小。

     `minify` 操作所保留的运行时环境的部件取决于您正在打包的服务器中配置的功能部件。只会保留在运行该服务器时所需要的那些功能部件，其余功能部件都将移除。因此，随后您将无法启用已移除的

功能部件。例如，如果仅保留了 servlet-3.0 功能部件，那么您随后将无法启用 jpa-2.0 功能部件。

如果配置已更改，那么您可以重复执行 minify 操作以进一步减小此归档的大小。但是，minify 操作不存在逆向操作，因此，如果您稍后需要已移除的一个或多个功能部件，那么必须从已完成的 xigemaAS 服务器重新开始。

在 minify 操作正在运行时，服务器暂时已启动，并且您将看到相关联的消息。正因为如此，对于无法启动的服务器，您将无法使用 --include=minify 选项，但是可以使用 --include=all 或 --include=usr 选项来将此服务器打包。

可以通过将 --os 选项与 --include=minify 选项配合使用来指定您希望所打包的服务器支持的操作系统。

要打包仅支持 Linux™ 的服务器，请使用下列命令：

```
server package --archive="linux.zip" --include=minify --os=Linux
```

- 打包 xigemaAS 运行时。

  创建包含 wlp 目录但不包含 usr 目录的运行时归档。服务器软件包的命名约定为 *package_name*.zip；例如，CustomerPortalApp.zip。要创建运行时归档，运行不带服务器名称但带 --include=wlp 选项的 package 命令：

  ```
  server package --include=wlp
  ```

  要指定软件包文件名和目标位置，请添加 --archive=*package_path_name* 选项；例如：

  ```
  server package --include=wlp --archive=c:\temp\myPackage.zip
  ```

  如果未使用 --archive 选项指定有效软件包名称或目标位置，那么该命令将在 *$WLP_OUTPUT_DIR* 位置（缺省情况下为 *${wlp.install.dir}*/usr/servers 目录）创建 wlp.zip 运行时归档。运行该命令前，目标位置必须存在。因此，如果目标位置为 c:\temp，那么 C:\temp 目录必须存在并且必须具有写许可权，该命令才能将归档写至 C:\temp 目录。

从 JAR 文件运行 xigemaAS 服务器

可以从 Java 归档 (JAR) 文件启动 xigemaAS 服务器。此方法提供一种简洁而又可移动的方式来启动 xigemaAS 服务器。可使用 xigemaAS 服务器命令创建 JAR 文件，然后使用 Java -jar 命令将其作为可执行 JAR 文件运行。

创建可运行的 JAR 文件

可以指定 minify 以获得尽可能小的归档。必须指定 JAR 类型归档才能获得可运行的 JAR 文件。缺省归档类型为 .zip。例如：

```
server package <server name> --include=[minify,]runnable --archive=<jar file
 name>.jar
```

运行 JAR 文件

通过将标准 Java 命令与 -jar 选项配合使用来运行 JAR 文件，例如：

```
java -jar <jar file name>.jar
```

操作

运行 JAR 文件时，它将解压到临时位置，然后，由 xigemaAS 服务器 run 命令启动的服务器将在前台运行。所有输出将写到 stdout 或 stderr。缺省情况下，文件将解压到临时位置：

- 对于 Windows：%HOMEPATH%/wlpExtract/<jar file name>_nnnnnnnnnnnnnnnnnn
- 对于所有其他平台：$HOME/wlpExtract/<jar file name>_nnnnnnnnnnnnnnnnnn

可以使用 WLP_JAR_EXTRACT_ROOT 或 WLP_JAR_EXTRACT_DIR 环境变量来控制输出位置。

停止服务器

要停止 xigemaAS 服务器，请按 **Ctlr-C**。当 xigemaAS 服务器停止时，解压目录将自动删除。如果您以任何其他方式停止活动 shell，那么不会自动清除解压目录，您必须手动将其清除。

以调试方式运行

如果在启动 xigemaAS 服务器之前设置了环境变量 WLP_JAR_DEBUG，那么可以用调试方式运行 xigemaAS 服务器。

控制输出

缺省情况下，服务器输出将写到解压目录，当服务器停止时会删除该目录。如果要保存输出，请在启动服务器之前使用 WLP_OUTPUT_DIR 环境变量指定持久输出位置。

两阶段落实事务

缺省情况下，两阶段落实事务已禁用，因为事务日志位于扩展目录中，并将在 xigemaAS 服务器停止时删除。因此，无法进行事务恢复。

要启用两阶段落实，请将事务日志配置为位于文件系统或 RDBMS 内的持久位置中，并设置 WLP_JAR_ENABLE_2PC 环境变量。

要配置事务日志，请使用 server.xml 配置中事务元素上的 transactionLogDirectory 或 dataSourceRef 属性。

在 CYGWIN 下运行

在 CYGWIN shell 中运行 xigemaAS 服务器 JAR 文件具有两个需求：

1. 指定 WLP_JAR_CYGWIN 环境变量。

   当此变量在 CYGWIN 环境中运行时，会导致 xigemaAS 服务器 JAR 运行器来执行 UNIX 样式的文件和进程处理。
2. 在 Bash shell 下运行，而非 mintty。

   仅当在 Bash shell 下运行时，才会自动删除解压文件。您可以在 mintty 下运行，但必须手动删除解压文件。Mintty 不会自动转发触发 Java 关闭挂钩所需的必要信号。

环境变量引用

表 19: 环境变量名称及其定义

第一列包含环境变量的列表。第二列包含每个环境变量的描述。

环境变量名称	描述
WLP_JAR_EXTRACT_ROOT	将 JAR 文件解压到目录 ${WLP_JAR_EXTRACT_ROOT}/<jar file name>_nnnnnnnnnnnnnnnnn
WLP_JAR_EXTRACT_DIR	将 JAR 文件解压到目录 ${WLP_JAR_EXTRACT_DIR}。
WLP_OUTPUT_DIR	将 xigemaAS 服务器输出文件写到目录 ${WLP_OUTPUT_DIR}。
WLP_JAR_DEBUG	通过使用 server debug <server name> 而非 server run <server name> 来运行 xigemaAS 服务器。
WLP_JAR_ENABLE_2PC	如果设置为值 true，那么可在可运行 JAR 文件运行时启用 2PC。
WLP_JAR_CYGWIN	如果在 CYGWIN 下运行 JAR 文件，请设置为值 true。

从命令行启动和停止服务器

可以使用 server 任务来启动或停止服务器。

wlp/bin 目录包含一个称为 server 的脚本来帮助控制服务器进程。此脚本的语法如下所示：

```
server action serverName [options]
```

有关 *[options]* 的可用值，请参阅 服务器命令选项（见第 1118 页）。

注：如果未在命令行上指定服务器，那么会对缺省服务器实例 defaultServer（如果存在）执行操作。

- 使用以下命令来启动服务器：

  ```
  server start serverName
  ```

  其中 *serverName* 是服务器的名称。
- 使用以下命令来停止服务器：

  ```
  server stop serverName
  ```

  其中 *serverName* 是服务器的名称。

  注：正常服务器停止包括服务器关闭前的停顿阶段。此停顿阶段为 30 秒，允许服务执行关闭前工作，例如，停止入站侦听器，同时允许现有请求完成。对 stop 命令应用 --force 选项将跳过此停顿阶段。如果已调用 server stop，那么 --force 选项不起作用。如果使用 --force 选项，那么您可能会在 messages.log 文件中见到意外异常，它们在服务器接收到 server stop 命令后发生。

要在 Windows™ 系统上使用 `server` 脚本来启动或停止服务器，请输入以下命令：

```
server.bat start serverName
```

```
server.bat stop serverName
```

要在其他系统上使用 `server` 脚本来启动或停止服务器，请输入以下命令：

```
server start serverName
```

```
server stop serverName
```

使用 OSGi 控制台

Eclipse Equinox 当前提供可用来辅助调试的 OSGi 控制台。缺省情况下，此控制台不可用。通过使用 `osgiConsole-1.0` 功能部件并指定要连接至的端口，可在 xigemaAS 概要文件内运行的 OSGi 框架中启用此控制台。

xigemaAS 概要文件使用 OSGi 核心规范的 Eclipse Equinox 实现。Equinox 当前提供 OSGi 控制台。要启用此控制台，可以先通过在 bootstrap.properties 文件中设置 osgi.console 属性来给此控制台分配特定端口。然后，可以使用 Telnet 来连接至该端口上的控制台，并浏览 OSGi 框架。

- 将 `osgiConsole-1.0` xigemaAS 功能部件添加到 server.xml 文件。

  ```
  <feature>osgiConsole-1.0</feature>
  ```

- 给 OSGi 控制台分配特定端口。

  要设置 OSGi 控制台端口，请指定 osgi.console 属性。将此属性设置为 bootstrap.properties 文件中的引导属性。请参阅设置应用服务器的引导属性（见第 1102 页）。

  ```
  osgi.console=5471
  ```

  未设置 osgi.console 属性时，会禁用 OSGi 控制台。

- 使用 Telnet 来连接至 OSGi 控制台端口。

  ```
  telnet localhost 5471
  ```

- 使用控制台来浏览框架。

  可用的命令随所使用的 OSGi 框架而不同。命令行帮助提供了足够的入门信息。

添加和移除 xigemaAS 功能部件

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。要添加或移除 xigemaAS 功能部件，请在 server.xml 配置文件的 `<feature>` 子元素中添加或删除 XML 片段。添加或移除 xigemaAS功能部件时，会动态地应用更改。

可按本主题中所述添加和移除 xigemaAS 功能部件。

有关主要 xigemaAS 功能部件（包括启用这些 xigemaAS 功能部件的 XML 片段）的列表，请参阅 *xigemaAS 功能部件（见第 906 页）*。

1. 要添加或移除 xigemaAS 功能部件，请完成下列步骤：

1. 打开 server.xml 配置文件进行编辑。

   其中，*path_to_xigemaas* 是 xigemaAS 在操作系统上的安装位置，*server_name* 是服务器的名称。

   可以使用文本编辑器执行此操作。缺省情况下，配置根文档文件的路径和文件名是 *path_to_xigemaas*/wlp/usr/servers/*server_name*/server.xml。但是，可以更改路径。请参阅 定制 *xigemaAS* 环境（见第 1114 页）。
2. 在配置文件中添加或移除功能部件。

   功能部件集含括在 `<featureManager>` 元素中，而每项功能部件含括在 `<feature>` 子元素中。例如：

   ```
   <server>
     <featureManager>
       <feature>servlet-3.0</feature>
       <feature>localConnector-1.0</feature>
     </featureManager>
   </server>
   ```

   功能部件名称匹配不区分大小写；以下示例也是有效的服务器配置：

   ```
   <featureManager>
           <feature>Servlet-3.0</feature>
           <feature>localConnector-1.0</feature>
   </featureManager>
   ```
3. 保存对配置文件所作的更改。

此时，会应用更改。如果服务器处于运行状态，那么会动态地应用更改。

在配置文件中使用 include 元素、变量和 `Ref` 标记

可以将所有配置设置保留在单个 server.xml 文件中，也可以使用 include 元素来合并不同文件中的配置设置。可以在配置中使用变量来避免利用硬编码值，因为在不同环境中复用配置时，硬编码值可能不合适。可以使用 `Ref` 标记来引用（并因此复用）配置中其他位置定义的现有代码块。

在配置文件中使用 include 元素

可以将所有配置保留在单个 server.xml 文件中，也可以使用 include 元素来合并单独文件中的配置以创建对您最有用的结构。

维护复杂配置的较简单方式是将其分割成一组文件。例如：

- 您可能想要添加一个含有特定于本地主机的变量的文件，以便您的主要配置可用在多个主机上。
- 您可能想要将特定应用程序的所有配置保留在单独文件中，该文件可以随应用程序本身进行版本化。

这是用于添加配置文件的语法。如果您想要跳过找不到的包含文件，那么可以将 optional 属性设为 `true`：

```
<include optional="true" location="pathname/filename"/>
或 <include optional="true" location="url"/>
```

以下列表显示了可能的位置；按如下顺序进行搜索。

1. 在相对于父文件指定的位置中
2. 在服务器配置目录中
3. 在指定为绝对路径的位置中
4. 在 Web 服务器上

要确保包含配置的行为符合预期，需要了解所包含配置文件的下列处理规则：

- 对于单例服务（例如日志记录或应用程序监视），以条目在文件中的出现顺序来处理条目，后来的条目添加到或覆盖先前的条目。这对于配置实例（例如应用程序或数据源）也成立，其中配置实例具有相同的标识。
- include 语句可以放在 <server /> 元素中的任何位置。
- 每个所包含的文件必须包含 <server /> 元素，该元素与父配置文件中的那个元素匹配。
- 所包含的文件可以嵌套其他所包含的文件。
- 每个所包含的文件在逻辑上合并到主配置中 <include /> 语句在父文件中出现的位置。

在以下示例中，主服务器配置文件 server.xml 包含 blogDS.xml 配置文件的内容，该文件位于共享配置目录中。blogDS.xml 文件包含数据源定义。此定义已放入单独的配置文件中，这样它可以包括在若干个不同的 server.xml 文件，并因此用于多个服务器实例。

下面是 server.xml 文件中的示例代码：

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>jdbc-4.0</feature>
  </featureManager>
  <application id="blog" location="blog.war" name="blog" type="war"/>
  <include optional="true" location="${shared.config.dir}/blogDS.xml"/>
</server>
```

下面是 blogDS.xml 文件中的示例代码：

```
<server>
  <dataSource id="blogDS" jndiName="jdbc/blogDS" jdbcDriverRef="derbyEmbedded">
    <properties createDatabase="create" databaseName="C:/xigemaAS/basics/derby/data/blogDB" />
  </dataSource>
  <jdbcDriver id="derbyEmbedded"/>
    <library>
      <fileset dir="C:/xigemaAS/basics/derby" includes="derby.jar" />
    </library>
  </jdbcDriver>
</server>
```

在配置文件中使用变量

可以在配置中使用变量来避免利用硬编码值，因为在不同环境中复用配置时，硬编码值可能不合适。

可以通过在下列任一位置设置属性来定义变量：

- 在服务器配置文件或随附的文件中
- 在 bootstrap.properties 文件中

可引用以下预定义变量：

- 目录属性，请参阅 目录位置和属性（见第 1101 页）
- JVM 系统属性
- 进程环境变量

如果在多个位置指定了同一变量，那么优先顺序如下所示：

- bootstrap.properties 中的变量覆盖进程环境变量
- server.xml 或所包含 XML 文件中的变量覆盖 bootstrap.properties 中的变量及进程环境变量

👉 注：特定于特定服务器的变量（例如，端口号）是在 bootstrap.properties 文件中指定，这允许在多个服务器之间共享 server.xml，同时保持这些值在每个服务器中不同。在一组服务器之间共享变量（例如，特定主机的数据库配置）最好是在要包括到父配置文件中的 xml 文件内指定。

👉 注：变量名称必须以字母字符开头，并且只能包含以下字符：字母字符、数字字符以及“_”和“.”字符。

- 在配置文件中指定变量。

  变量定义语法为 *variable_name=value*。如果该值包含路径，那么它将在配置处理期间规范化（通过将双正斜杠和双反斜杠替换为单一正斜杠，除非该值以双正斜杠或双反斜杠开头，在此情况下斜杠保留不变。）

  👉 注：如果需要将变量值设置为包含双正斜杠（有时用于 JDBC 驱动程序连接 URL 时），请在双斜杠处将该值拆分为两个部分。通过放置双正斜杠充当初始字符，规范化将被阻止。例如，要存储值 `"jdbc:db2://host_name.com"`，请使用两个变量：

  ```
  URL_PART_1="jdbc:db2:"
  URL_PART_2="//host_name.com"
  ```

  配置文件中所定义变量的作用域会限定为使用这些变量的配置元素。例如，下列代码段创建称为 *updateTrigger_var* 的变量以用在 applicationMonitor 配置元素中：

  ```
  <applicationMonitor updateTrigger_var="mbean" />
  ```

  要创建特定配置实例（例如应用程序或资源条目）中使用的变量，还必须指定实例标识。例如：<httpEndpoint id="defaultHttpEndpoint" HTTP_default_var="8889" />

- 在 bootstrap.properties 文件中指定变量。

  bootstrap.properties 文件中所定义变量的作用域不会限定为特定配置元素。变量以键值对的形式输入。例如：

  ```
  HTTP_default_var=8006
  ```

- 使用配置中所定义的变量。

  变量替换语法为 `${variable_name}`。可通过指定 `${variable_name1}${variable_name2}` 并置多个变量值。例如，要使用 *HTTP_default_var* 变量，请将下列代码段添加到配置文件：

  ```
  <httpEndpoint id="defaultHttpEndpoint"
  httpPort="${HTTP_default_var}">
  </httpEndpoint>
  ```

- 在配置中使用 `variable` 元素

  可以使用 `variable` 元素在服务器配置中全局定义变量。如果在包含的文件中定义了相同的变量，那么该变量会由 server.xml 文件中的变量覆盖。例如，要使用 `variable` 元素，请将下列代码段添加到配置文件：

  ```
  <variable name="HTTP_default_var" value="8889" />
  ```

- 在配置中使用进程环境变量

  如果您使用 env. 配置变量前缀，那么进程环境变量可用，例如：

  ```
  <fileset dir="${env.LIBRARY_DIR}" includes="*.jar"/>
  ```

有关指定环境变量的更多信息，请参阅定制 *xigemaAS* 环境（见第 1114 页）。

- 在配置中使用变量表达式

  对于配置变量，可使用 `${<operand><operator><operand>}` 格式的受限变量表达式语法。此变量的描述如下所示：

  operand

  操作数可以是长整数字面值或包含长整数值的变量的名称。变量名称必须以字母字符开头，并且只能包含以下字符：字母字符、数字字符以及“_”和“.”字符。

  operator

  可用运算符如下所示：

  - `+`，表示加法
  - `-`，表示减法
  - `*`，表示乘法
  - `/`，表示除法

  如果无法解析表达式，使用了非整数值或发生了算术错误，那么该行为未定义。

  例如，如果定义了 *HTTP_port_base* 变量，那么可使用变量表达式来定义多个 httpEndpoint：

```
<httpEndpoint id="defaultHttpEndpoint" httpPort="${HTTP_port_base+0}"/>
<httpEndpoint id="httpEndpoint2" httpPort="${HTTP_port_base+1}"/>
```

- 覆盖配置中的可继承属性

  可以覆盖配置中可继承属性的缺省值。例如，onError 属性是其中一个可继承属性。要为 onError 属性全局定义变量名称，可以在 bootstrap.properties 中设置或在 server.xml 文件中使用 `variable` 元素来设置。如果在两个文件中指定相同的变量名称，那么会使用 server.xml 文件中的值。如果两个文件中均未显式地设置该属性，那么该属性会使用缺省值。如果给可继承属性设置了无效值，那么属性值回退到 bootstrap.properties 或 server.xml 文件中定义的全局值，或者回退到缺省值（如果未在全局级别定义）。

  另一个示例是xigemaAS概要文件中的日志记录属性。请参阅日志记录和跟踪（见第 1649 页）。

在配置文件中使用 `Ref` 标记

可以定义公共配置元素，然后通过在配置中的其他位置对该定义进行引用（使用 `Ref` 标记）来复用该定义。`Ref` 标记可以用在包含该元素定义的相同配置文件中，或者用在随附的配置文件中。

使用不同的方法来指定必需配置元素之间的关系。例如，下列数据源定义都有效。第一个不使用 `Ref` 标记，第二个使用直接元素定义和 `Ref` 标记的组合，而第三个仅使用 `Ref` 标记。

示例 1：不使用任何 `Ref` 标记。

```
<dataSource id="blogDS" jndiName="jdbc/blogDS">
  <properties createDatabase="create" databaseName="C:/xigemaas/basics/derby/data/blogDB"/>
  <jdbcDriver>
    <library>
      <fileset dir="C:/xigemaas/basics/derby" includes="derby.jar"/>
    </library>
  </jdbcDriver>
  <connectionManager maxPoolSize="10"/>
</dataSource>
```

示例 2：将直接元素定义和 Ref 标记组合。

```
<dataSource id="blogDS" jndiName="jdbc/blogDS" connectionManagerRef="derbyPool">
  <properties createDatabase="create" databaseName="C:/xigemaas/basics/derby/data/blogDB"/>
  <jdbcDriver libaryRef="derbyLib"/>
</dataSource>

<connectionManager id="derbyPool" maxPoolSize="10"/>

<library id="derbyLib"/>
  <fileset dir="C:/xigemaas/basics/derby" includes="derby.jar"/>
</library>
```

示例 3：仅使用 Ref 标记（properties 元素除外，只允许它进行嵌套）。

```
<dataSource id="blogDS" jndiName="jdbc/blogDS"
            connectionManagerRef="derbyPool" jdbcDriverRef="derbyEmbedded">
  <properties createDatabase="create" databaseName="C:/xigemaas/basics/derby/data/blogDB"/>
</dataSource>

<connectionManager id="derbyPool" maxPoolSize="10"/>

<jdbcDriver id="derbyEmbedded" libraryRef="derbyLib"/>

<library id="derbyLib" filesetRef="derbyFileset"/>

<fileset id="derbyFileset" dir="C:/xigemaas/basics/derby" includes="derby.jar"/>
```

使用配置 dropins 文件夹指定服务器配置

可在 configDropins 目录中指定附加配置文件而不在 server.xml 文件中指定包含元素。

1. 在 usr/servers/*server_name* 目录下创建 configDropins 目录。
   - usr/servers/*server_name*/configDropins/overrides

     如果要添加配置文件以替换服务器的 server.xml 文件中的任何内容，请创建 configDropins/overrides 目录。例如，要更改 server.xml 中定义的端口，请使用 configDropins/overrides 目录。
   - usr/servers/*server_name*/configDropins/defaults

     如果希望 server.xml 文件成为主配置，但希望对 server.xml 未定义的元素指定缺省值，请创建 configDropins/defaults 目录。例如，如果希望开发者能够提供配置，但又希望 server.xml 成为主配置并且不希望 server.xml 更改，请使用 configDropins/defaults 目录。
2. 将服务器配置文件放置在 configDropins/overrides 或 configDropins/defaults 目录中。

   系统会监视两个目录以获取更新，从而在您添加、移除或更新配置文件时动态更新运行时配置。

   如果存在任何冲突，那么以下规则确定优先顺序：
   - configDropins/overrides 目录中指定的配置优先于 server.xml 文件中的配置。server.xml 文件中指定的配置优先于 configDropins/defaults 目录中指定的配置。
   - configDropins/defaults 和 configDropins/overrides 目录中的文件内的配置优先于功能部件指定的任何缺省配置。
   - dropins 目录中的配置文件按字母顺序进行处理。较新的配置覆盖较旧的配置。例如，如果 configDropins/defaults 包含 a.xml、b.xml 和 c.xml，那么 c.xml 中的配置优先于 b.xml 中的配置，b.xml 中的配置优先于 a.xml 中的配置。

注：为维护平台间的一致性，在按字母顺序排序之前，文件名已转换为小写。这意味着，如果在同一 dropins 目录中指定了两个名称相同但大小写不同（例如，extraConfig.xml 和 ExtraConfig.xml）的两个文件，那么排序行为无法确定。

3. 可选： 关闭配置监视。请参阅控制动态更新（见第 1135 页）。

引用配置文件的标识变量

xigemaAS 运行时有时需要引用 server.xml 文件中的配置元素。此操作可能以若干方式进行，例如，在消息文本中或文件名中。

xigemaAS 运行时使用 xpath 样式语法来引用配置元素。最先显示元素类型，后跟括在方括号中的配置元素标识。如果该配置元素嵌套在另一配置元素内，那么内部配置元素之前会加上正斜杠，以分隔内部和外部元素。

- 例如，以下 following databaseStore 配置元素被引用为 databaseStore[DBTaskStore]，因为 databaseStore 未嵌套并且具有标识值 DBTaskStore。

```
<server>
  <databaseStore id="DBTaskStore">
     ...
  </databaseStore>
</server>
```

- 以下数据源配置元素被引用为 `databaseStore[DBTaskStore]/dataSource[DataSource0]`，因为该数据源嵌套在 databaseStore 下，databaseStore 具有标识值 `DBTaskStore`，数据源具有标识值 `DataSource0`。

```
<server>
  <databaseStore id="DBTaskStore">
    <dataSource id="DataSource0">
      ...
    </dataSource>
  </databaseStore>
</server>
```

- 在某些情况下，配置元素没有标识。在此情况下，将生成标识。例如，以下数据源配置元素可引用为 `databaseStore[default-0]/dataSource[DataSource0]`，因为 databaseStore 未定义标识。

```
<server>
  <databaseStore>
    <dataSource id="DataSource0">
      ...
    </dataSource>
  </databaseStore>
</server>
```

控制动态更新

可以通过配置来控制三种动态更新：更改服务器配置；添加和移除应用程序；更新已安装的应用程序。对于所有已部署的应用程序，您都可以配置是否启用应用程序监视，以及配置检查应用程序更新的频率。对于“dropins”目录，您也可以配置目录的名称和位置，以及选择是否要部署该目录中的应用程序。

缺省情况下，会监视已部署的应用程序是否有更新，而且更新会动态地应用到正在运行的应用程序。这适用于通过配置条目来部署的应用程序，以及那些从“dropins”目录部署的应用程序。要更改这些缺省行为，可以

在 server.xml 配置文件中设置 config 和 applicationMonitor 元素。可以使用文本编辑器来执行此操作，也可以使用开发者工具并在服务器配置设计视图中选择**配置管理服务**或**应用程序监视器**。

应用程序监视的缺省设置如下所示：

```
<applicationMonitor updateTrigger="polled" pollingRate="500ms"
                    dropins="dropins" dropinsEnabled="true"/>
```

用于配置监视的缺省设置如下所示：

```
<config updateTrigger="polled" monitorInterval="500ms"/>
```

👉 注：

- updateTrigger 属性具有三个可能的值：

  polled

  运行时环境会使用 monitorInterval 属性所指定的时间间隔在 server.xml 文件中扫描更改。

  mbean

  只有在通过对 MBean 的调用提示查找更新时，运行时环境才会这样做。此方式供开发者工具用来更新 server.xml 文件，除非您覆盖此方式。

  disabled

  不会动态应用更新。

- 指定 pollingRate 属性或 monitorInterval 属性时，在数字后面包含时间单位：
  - ms（毫秒）
  - s（秒）
  - m（分钟）
  - h（小时）
- dropins 属性指定用作“dropins”目录的目录名称。
- dropinsEnabled 属性是布尔属性，用来确定“dropins”目录中的应用程序是否已部署。

- 配置服务器配置的动态更改。

  对 server.xml 文件或任何随附文件的更改，是由运行时环境检测并应用到活动配置。要禁用此行为，可以在 server.xml 文件中设置 config 元素：

  ```
  <config updateTrigger="disabled"/>
  ```

  还可通过在 server.xml 文件中设置 config 元素以借助所提供 MBean 来控制对服务器配置的动态更新：

  ```
  <config updateTrigger="mbean"/>
  ```

  然后，可使用 FileNotificationMbean 通知服务器您要以动态方式重新处理的配置文件。
- 配置应用程序的动态添加和移除。

  如在 *xigemaAS* 中部署应用程序（见第 1494 页）中所述，可以通过两种机制，对服务器运行时环境执行动态添加和移除应用程序操作：

  - 在 server.xml 文件中添加或移除应用程序条目。

如果如上一步所述禁用服务器配置动态更改，那么添加或移除应用程序条目不会影响正在运行的服务器。仅在下一次重新启动服务器时才应用更改。如果您使用开发者工具来更新应用程序条目，那么会立即获得更改。

- 将应用程序文件移入和移出“dropins”目录。

  要控制此行为，可以在 `server.xml` 文件中设置 applicationMonitor 元素。例如，要禁止从“dropins”位置动态安装应用程序，请创建如下所示的条目：

```
<applicationMonitor dropinsEnabled="false" />
```

- 配置已安装应用程序的动态更新。

  缺省情况下，如果已部署的应用程序中添加、移除或修改任何文件，或者将整个应用程序替换为更新的版本，那么会自动停止先前版本，并启动新版本。此过程适用于任何已部署的应用程序，而不论应用程序是位于“dropins”目录，还是位于 server.xml 文件中所定义的位置。要控制此行为，可以在 `server.xml` 文件中设置 applicationMonitor 元素。例如，要禁用所有应用程序的动态更新，请创建如下所示的条目：

```
<applicationMonitor updateTrigger="disabled"/>
```

- 配置“dropins”目录的名称和位置。

  缺省情况下，“dropins”目录是 `${server.config.dir}/dropins`。要更改此项，可以在 `server.xml` 文件中设置 applicationMonitor 元素。对于位置，可以使用任何已知变量、bootstrap.properties 文件中的属性、绝对路径，或者相对于服务器目录的路径。例如，下列两项设置指向同一位置：

```
<applicationMonitor dropins="${server.config.dir}/applications" />
<applicationMonitor dropins="applications" />
```

  限制：对于 Web Service 应用程序，如果服务客户机和服务提供程序不在同一应用程序中，且服务提供程序中的 WSDL 文件已更改，那么需要手动重新启动 Web Service 客户机应用程序以避免发生 WSDL 定义高速缓存问题。

为 Java™ EE 应用程序配置类加载器和库

缺省情况下，每个应用程序均可以访问随附的 API 集及其自己的内部类和库。可以覆盖缺省设置，并配置每个应用程序的类装入操作。

每个 Java™ EE 应用程序在运行中 xigemaAS 概要文件服务器中均有其自己的类加载器。xigemaAS 概要文件对所有 Java™ EE 应用程序都采用一些缺省设置，这样这些应用程序就可以访问支持的规范 API（例如，如果启用了 servlet 功能部件，那么为 servlet API）。缺省情况下，每个应用程序均可以访问这些随附的 API 并访问其自己的内部类和库。如果必须覆盖缺省设置，并为应用程序配置类载入操作，请完成下列其中一项或多项任务。

注：如果使用配置来覆盖缺省设置，那么无法通过将应用程序拖放到“dropins”目录来部署应用程序。

将 Java™ 库与 Java™ EE 应用程序一起使用

将 Java™ 库与应用程序一起使用的一种方法是将它们包含到应用程序本身。这并非总是符合预期或者适当的，尤其是应用程序已经打包且不含库时。

在以下示例中，称为 `Alexandria` 的库由两个文件组成：

- alexandria-scrolls.jar 和

- commons-lang.jar

称为 Scholar 的应用程序正在称为 Academy 的服务器上运行，需要访问此库。

1. 在 ${WLP_USER_DIR} 目录下的 servers/Academy 目录中创建 mylib/Alexandria 目录。

   例如：wlp/usr/servers/Academy/mylib/Alexandria。
2. 将 alexandria-scrolls.jar 和 commons-lang.jar 文件复制到新文件夹。
3. 为应用程序配置类装入操作，以便装入 Alexandria 库。

   在 server.xml 文件或随附的文件中，添加下列代码：

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
  <classloader>
    <privateLibrary>
      <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar"
 scanInterval="5s" />
    </privateLibrary>
  </classloader>
</application>
```

   注：<privateLibrary> 元素也可以接受具有以逗号分隔的 <fileset> 元素标识列表的 filesetRef 属性。

在多个 Java™ EE 应用程序之间共享库

可以在多个 Java™ EE 应用程序之间共享库。所有应用程序都可以在运行时使用相同的类，或者每个应用程序可以使用从同一个位置装入的那些类的单独副本。

在以下示例中，称为 Alexandria 的库由两个文件组成：

- alexandria-scrolls.jar 和
- commons-lang.jar

称为 Scholar 的应用程序以及称为 Student 的应用程序正在称为 Academy 的服务器上运行，并且都需要访问此库。

1. 在 ${WLP_USER_DIR} 目录下的 servers/Academy 目录中创建 mylib/Alexandria 目录。

   例如：wlp/usr/servers/Academy/mylib/Alexandria。
2. 将 alexandria-scrolls.jar 和 commons-lang.jar 文件复制到新文件夹。
3. 为应用程序配置类装入操作，以便装入 Alexandria 库。

   在 server.xml 文件或随附的文件中，通过添加下列代码来定义库：

```
<library id="Alexandria">
  <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

   注：<library> 元素也可以接受具有以逗号分隔的 <fileset> 元素标识列表的 filesetRef 属性。
4. 从应用程序引用库，以便这两个应用程序共享库的单一副本。

   在 server.xml 文件或随附的文件中，添加下列代码：

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
  <classloader commonLibraryRef="Alexandria" />
</application>
<application id="student" name="Student" type="ear" location="student.ear">
  <classloader commonLibraryRef="Alexandria" />
```

```
</application>
```

注：<commonLibraryRef> 元素可以接受以逗号分隔的库标识列表。

5. 可选： 将另一个应用程序配置成从相同的 JAR 文件装入其自己的类集。

   例如，如果另一个称为 Spy 的应用程序需要其自己的类副本，那么可以使用磁盘上的相同物理文件。在 server.xml 文件或随附的文件中，添加下列代码：

```
<application id="spy" name="Spy" type="war" location="spy.war">
  <classloader privateLibraryRef="Alexandria" />
</application>
```

注：<privateLibraryRef> 元素可以接受以逗号分隔的库标识列表。

为所有 Java™ EE 应用程序提供全局库

可以提供任何 Java™ EE 应用程序都可以使用的全局库。如果要这样做，请将这些库的 JAR 文件放入全局库目录中，然后指定在每个应用程序的类加载器配置中使用全局库。但是，其他应用程序（例如，OSGi 应用程序）无法使用全局库。

在使用环境变量 WLP_USER_DIR 指定的用户目录下，您可以在下列位置中放置全局库：

- ${shared.config.dir}/lib/global
- ${server.config.dir}/lib/global

如果启动应用程序时这些位置中存在文件，并且该应用程序未配置 <classloader> 元素，那么应用程序会使用这些库。如果存在类加载器配置，那么不会使用这些库，除非显式引用全局库。

注意：如果您使用全局库，那么也会建议您为每个应用程序配置一个 <classloader> 元素。servlet 规范要求应用程序在其类加载器父链中共享全局库类加载器。这打破了每个应用程序之间类加载器可能彼此分离的情况。因此，应用程序更可能对 xigemaAS 中装入的类产生长时间的影响，应用程序可能对彼此产生长时间的影响，而且应用程序之间更可能发生类空间一致性问题，尤其是对运行中服务器添加和移除功能部件时。上述所有注意事项都不适用于在其配置中指定 <classloader> 元素的应用程序，因为应用程序维持这种分离。

在以下示例中，称为 Scholar 应用程序会配置成使用称为 Alexandria 的公共库，并且也配置成使用全局库。

在 server.xml 文件或随附的文件中，通过添加下列代码来对应用程序启用全局库：

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
  <classloader apiTypeVisibility="spec" commonLibraryRef="Alexandria, global" />
</application>
```

全局库的设置也可以显式地配置成具有特殊标识 global 的库元素。例如：

```
<library id="global">
  <fileset dir="/path/to/folder" includes="*.jar" />
</library>
```

从 Java™ EE 应用程序访问第三方 API

缺省情况下，Java™ EE 应用程序无法访问 xigemaAS 概要文件中提供的第三方 API。要启用此访问权，必须在 server.xml 文件或随附的文件中配置该应用程序。

在以下示例中，名为 Scholar 的应用程序需要访问 xigemaAS 概要文件中提供的第三方 API。

应用程序还使用称为 Alexandria 的公共库。此库位于 ${server.config.dir}/mylib/Alexandria 目录中。

注：升级后，第三方 API 可能无法保持兼容。有关更多信息，请参阅 *xigemaAS 外部支持*（见第 36 页）。

1. 为应用程序配置类装入操作，以便应用程序可以访问第三方 API。

   classloader 元素的 apiTypeVisibility 属性的缺省值为 spec,ibm-api,api。其中，spec 表示同时可用于编译和运行时的公共规范 API，ibm-api 表示 xigemaAS 中可用的 API，api 表示同时可用于编译和运行时的公共 API。在 classloader 元素的 apiTypeVisibility 属性中包含 third-party 可使第三方 API 可用。

   在 server.xml 文件或随附的文件中，通过添加下列代码来配置 API 类型可见性：

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
  <classloader apiTypeVisibility="spec, ibm-api, third-party" commonLibraryRef="Alexandria" /
>
</application>
```

2. 可选： 如果应用程序使用任何公共库，请将这些库设为使用相同的 API 类型可见性设置。

   在 server.xml 文件或随附的文件中，添加下列代码：

```
<library id="Alexandria" apiTypeVisibility="spec, ibm-api, third-party">
  <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

移除 Java™ EE 应用程序对第三方 API 的访问权

缺省情况下，Java™ EE 应用程序无法访问 xigemaAS 概要文件中提供的第三方 API。您还可以在 server.xml 文件或随附的文件中显式移除访问权。

在以下示例中，称为 Scholar 的应用程序先前已配置成访问第三方 API，如从 *Java EE 应用程序访问第三方 API*（见第 1139 页）中所述。您想要移除此访问权，以及在应用程序现在使用缺省访问设置的配置中显式进行此操作。

应用程序还使用称为 Alexandria 的公共库。此库位于 ${server.config.dir}/mylib/Alexandria 目录中。

1. 为应用程序配置类装入操作，以显示应用程序不再能访问第三方 API。

   在 server.xml 文件或随附的文件中，将 third-party 从随附的 apiTypeVisibility 属性值集中移除：

```
<application id="scholar" name="Scholar" type="ear" location="scholar.ear">
  <classloader apiTypeVisibility="spec, ibm-api" commonLibraryRef="Alexandria" />
</application>
```

2. 可选： 如果应用程序使用任何公共库，请将这些库设为使用相同的 API 类型可见性设置。

   在 server.xml 文件或随附的文件中，添加下列代码：

```
<library id="Alexandria" apiTypeVisibility="spec, ibm-api">
  <fileset dir="${server.config.dir}/mylib/Alexandria" includes="*.jar" scanInterval="5s" />
</library>
```

使用替代版本来覆盖随附的 API

如果 xigemaAS 概要文件也提供应用程序提供的类（所使用库提供的类），那么缺省情况下，会使用 xigemaAS 概要文件中的类。要更改此设置以便应用程序使用这些类的替代版本，必须在 `server.xml` 文件或随附的文件中配置应用程序。

如果 Web 应用程序包含的类也存在于服务器运行时环境中，那么您可能想要控制应用程序使用的其中的每个类副本。例如，如果应用程序和服务器运行时环境中存在不同版本的类，那么必须确保使用应用程序中打包的版本。

缺省情况下，xigemaAS 概要文件运行时环境中的类供所有 Java™ EE 应用程序使用。可以使用类加载器配置 delegation 属性来覆盖此行为。此配置特定于某个应用程序，或者特定于选择供应用程序使用的共享库。

在以下示例中，称为 `Scholar` 的应用程序需要使用它所提供的类（或者它所使用的库提供的类），而不是使用 xigemaAS 中提供的类副本。

- 如果类已打包在应用程序中，请使用 server.xml 配置文件或随附的文件中的 `classloader` 元素来覆盖缺省 `parentFirst` 授权行为：

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
  <classloader delegation="parentLast" />
</application>
```

  这将告知应用程序类加载器只有在应用程序及其关联库中查找类之后才查找 xigemaAS 概要文件类。
- 如果类已打包在共享库中，请将 delegation 属性添加到 `classloader` 元素来配置类加载器的使用，如下所示：

```
<application id="" name="Scholar" type="ear" location="scholar.ear">
  <classloader delegation="parentLast" commonLibraryRef="mySharedLib"/>
</application>

<library id="mySharedLib">
  <fileset dir="${server.config.dir}/myLib" includes="*.jar" />
</library>
```

还可以将 privateLibraryRef 属性用于应用程序中的专用库。请参阅在多个 *Java EE* 应用程序之间共享库（见第 1138 页）。

为 OSGi 应用程序配置库

每个 OSGi 应用程序都可以访问一组提供的 API 及其自己的内部类。还可以将共享库配置为提供从共享库对额外包的访问。

每个 OSGi 应用程序在运行的 xigemaAS 服务器中都有其自己的一组 OSGi 捆绑软件。每个 OSGi 捆绑软件都指定它所需要的包以及它为供其他 OSGi 捆绑软件使用而提供的包。OSGi 应用程序中的捆绑软件可以访问同一 OSGi 应用程序中其他捆绑软件提供的所有包。此外，OSGi 应用程序中的 OSGi 捆绑软件可以访问 xigemaAS 服务器提供的 API 包。共享库还可以用于提供 API 包以供 OSGi 应用程序使用。

可以在多个 OSGi 应用程序间共享库。所有应用程序（包括 Java EE 应用程序）都可以在运行时使用共享库提供的相同类。

1. 在 `${WLP_USER_DIR}` 目录下的 servers/defaultServer 目录中创建 mylib/osgi 目录。

   例如：wlp/usr/servers/defaultServer/mylib/osgi。
2. 将 `osgi-lib.jar` 和 `commons-lang.jar` 文件复制到新文件夹中。

3. 为应用程序配置共享库，以使该库装入。在 server.xml 文件或包含的文件中，通过添加下列代码来定义库：

```
<library id="mylib">
    <fileset dir="${server.config.dir}/mylib/osgi" includes="*.jar"
 scanInterval="5s">
</library>
```

注：library 元素也可以接受具有以逗号分隔的 fileset 元素标识列表的 filesetRef 属性。

4. 将该库作为 OSGi 库引用，以便使 OSGi 应用程序可以访问由该库提供的包以及共享该库的单个副本。在 server.xml 文件或包含的文件中，添加以下代码：

```
<osgiLibrary libraryRef="myLib"/>
```

5. 可选：配置包列表，以便可以从 OSGi 应用程序访问这些包。当使用 osgiLibrary 元素配置了共享库时，OSGi 应用程序可以访问该共享库中包含的包。还可以列出包，以便对 OSGi 应用程序可以访问哪些包提供更多的控制。包语法使用 OSGi Export-Package 头语法来定义每个包。要列出 server.xml 文件或包含的文件中的包，请添加以下代码：

```
<osgiLibrary libraryRef="myLib">
    <package>org.example.osgi.lib.pkg1; version=1.0</package>
    <package>org.example.osgi.lib.pkg2; version=1.1</package>
</osgiLibrary>
```

注：未使用任何包元素时，将对库进行扫描以查找该库提供的包。发现的每个包都将获得缺省版本 0.0.0。

为 xigemaAS 配置 JPA

Java Persistence API (JPA) 2.0 for xigemaAS基于 Apache OpenJPA 2.2.x 开放式源代码项目。

Apache OpenJPA 是符合 JPA 1.0 和 2.0 规范的实现。通过将 OpenJPA 用于基本实现，xigemaAS采用扩展来为 xigemaAS客户提供其他功能部件和实用程序。由于 JPA for xigemaAS根据 OpenJPA 构建，因此所有 OpenJPA 功能、扩展和配置均不受 xigemaAS扩展的影响。您不必对 OpenJPA 应用程序进行任何更改，即可在 xigemaAS中使用这些应用程序。

Java™ Persistence API (JPA) 2.1 for xigemaAS基于 EclipseLink 开放式源代码项目。EclipseLink 是所有版本的 JPA 规范的引用实现。此产品的 JPA 的提供程序为 org.eclipse.persistence.jpa.PersistenceProvider。

配置 JPA 日志记录

记录支持对应用程序的运行时行为进行查看、跟踪和故障诊断。每个 JPA 功能部件提供不同级别的日志记录，以供您指定所需的日志记录详细程度。

使用 jpa-2.0 或 jpa-2.1 功能部件时，可配置日志记录以帮助进行故障诊断。请熟悉这两个功能部件的日志记录功能。

jpa-2.0

有许多受支持的 jpa-2.0 跟踪规范可通过 xigemaAS 配置进行配置。这些跟踪字符串可与任何其他跟踪规范一起使用。

容器管理的 **JPA** 应用程序

- JPA=all

  启用所有 JPA 容器跟踪和所有 OpenJPA 跟踪
- openjpa=all

  启用所有 OpenJPA 跟踪
- 特定于 OpenJPA 的日志通道

  `openjpa.jdbc.SQL=all`

```
<server>
...
<logging traceSpecification="openjpa.jdbc.SQL=all"
  traceFileName="trace.log"
  maxFileSize="20"
  maxFiles="10"
  traceFormat="BASIC" />
</server>
```

应用程序管理的 **JPA** 应用程序

运行应用程序管理的 JPA 应用程序时，日志记录和跟踪由 OpenJPA 运行时控制。所有 JPA 跟踪和日志记录必须通过 OpenJPA 持久性属性进行配置。

```
<persistence version="2.0">
<persistence-unit>
  <properties>
     <property name="openjpa.Log" value="openjpa.jdbc.SQL=trace"/>
  </properties>
</persistence-unit>
</persistence>
```

值得注意的 **OpenJPA** 日志记录持久性属性

`openjpa.ConnectionFactoryProperties=PrintParameters=true`-- 如果为 `true`，那么 SQL 绑定参数将包括在异常和日志中。

`jpa-2.1`

如果启用了 `jpa-2.1` 功能部件，那么所有 JPA 日志记录和跟踪是通过 xigemaAS 记录器路由的。

受支持的跟踪字符串

- JPA=all

  启用所有 JPA 容器跟踪和所有 EclipseLink 类别
- eclipselink=all

  启用所有 EclipseLink 跟踪
- 特定于 EclipseLink 的日志类别
  - sql、transaction、event、connection、query、cache、propagation、sequencing、ejb、dms、 metadata、weaver、properties 和 server

- 即：eclipselink.sql=All -- 启用 EclipseLink SQL 跟踪

```
<server>
...
<logging  traceSpecification="eclipselink.sql=all"
  traceFileName="trace.log"
  maxFileSize="20"
  maxFiles="10"
  traceFormat="BASIC" />
</server>
```

值得注意的 **EclipseLink** 日志记录持久性属性

`eclipselink.logging.parameters` -- 如果为 `true`，那么 SQL 绑定参数将包括在异常和日志中。

1. 在 persistence.xml 文件的持久性单元定义中，根据您需要的日志记录详细信息级别指定日志记录级别。

   指定 `eclipselink.logging.level` 属性，其中的值为日志记录级别。有关可用日志记录级别列表，请参阅 EclipseLink 日志记录 Wiki 页面。以下示例将开启所有可用日志记录。

```
<persistence-unit name="pu">
        <properties>
           <property name="eclipselink.logging.level" value="ALL"/>
            ...
        </properties>
</persistence-unit>
```

配置 JPA 2.1 模式生成器

在基于 OpenJPA 的 `jpa-2.0` 功能部件中，可生成数据定义语言 (DDL)，或直接与数据库交互以使用 SchemaMapper 工具根据 JPA 实体定义来定义表模式。在基于 EclipseLink 的 `jpa-2.1` 功能部件中，可使用添加至 JPA 2.1 规范的新模式生成器功能部件，此功能部件的功能类似 OpenJPA SchemaMapper。

如果需要类似 OpenJPA SchemaMapper 的功能，那么可配置 JPA 2.1 规范中的模式生成器功能部件。

1. 在 persistence.xml 文件的持久性单元定义中，指定数据库操作属性及可能值（`none`、`create`、`drop` 和 `drop-and-create`）。

   每个值对应针对数据库执行的操作。以下示例会使系统删除对应持久性单元中指定的实体的表并在原地创建新表。

```
<persistence-unit name="pu">
        <properties>
            <property name="javax.persistence.schema-
generation.database.action"
        value="drop-and-create" />
...
        </properties>
 </persistence-unit>
```

2. 指定脚本操作属性及可能值（`none`、`create`、`drop` 和 `drop-and-create`）。

   如果指定 `none` 以外的任何值，那么还必须指定目标属性。这意味着，如果脚本操作为 `create`（此操作为实体定义生成创建语句），那么必须指定对应创建目标属性及这些语句写至的目标文件。

```
<persistence-unit name="pu">
        <properties>
```

```
            <property name="javax.persistence.schema-
generation.scripts.action"
      value="drop-and-create" />
            <property name="javax.persistence.schema-
generation.scripts.create-target"
        value="createTargetFile.ddl"/>
            <property name="javax.persistence.schema-
generation.scripts.drop-target"
      value="sampleDrop.ddl"/>
...
        </properties>
 </persistence-unit>
```

禁用 EclipseLink 共享对象高速缓存

EclipseLink 共享对象高速缓存包含已读取并保存以用于持久性单元的所有对象的子集。EclipseLink 共享高速缓存不同于本地 EntityManager/L1/持久性上下文高速缓存。共享高速缓存在持久性单元期间存在，并且由持久性单元的所有 EntityManager 和用户共享。

如果要迁移现有应用程序或运行应用程序跨越多个 Java 虚拟机 (JVM) 的环境，那么您可禁用 EclipseLink 共享对象高速缓存。

选择下列其中一种方法来禁用 EclipseLink 共享对象高速缓存。

1.
   - 在 persistence.xml 文件中设置 `<shared-cache-mode>NONE</shared-cache-mode>` 属性。

     ```
     <persistence-unit name="pu">
             <shared-cache-mode>NONE</shared-cache-mode>
             <properties>
     ...
             </properties>
      </persistence-unit>
     ```

   - 将 persistence.xml 文件内的持久性单元定义中的 `eclipselink.cache.shared.default` 属性设置为 `false`。

     ```
     <persistence-unit name="pu">
             <properties>
                 <property name="eclipselink.cache.shared.default"
      value="false" />
     ...
             </properties>
      </persistence-unit>
     ```

为 xigemaAS 配置会话持久性

如果必须在服务器重新启动或意外的服务器故障期间保留会话数据，那么可以配置 xigemaAS 以将会话数据持久存储到数据库。此配置可让多个服务器共享相同的会话数据，这样在发生故障转移时可以恢复会话数据。

根据数据库类型（SQL 类型和 NoSQL 类型）的不同，采取的配置方式也有所差异。若您需要将会话数据存储至 SQL 数据库，请参考配置基于 *SQL* 数据库的会话持久性（见第 1146 页）；若您需要将会话数据存储至 NoSQL 数据库（以 redis 数据库为例），请参考配置基于 *NoSQL* 数据库 *Redis* 的会话持久性（见第 1147 页）。

为 xigemaAS 配置会话持久性

如果必须在服务器重新启动或意外的服务器故障期间保留会话数据，那么可以配置 xigemaAS 以将会话数据持久存储到数据库。此配置可让多个服务器共享相同的会话数据，这样在发生故障转移时可以恢复会话数据。

根据数据库类型（SQL 类型和 NoSQL 类型）的不同，采取的配置方式也有所差异。若您需要将会话数据存储至 SQL 数据库，请参考配置基于 *SQL* 数据库的会话持久性（见第 1146 页）；若您需要将会话数据存储至 NoSQL 数据库（以 redis 数据库为例），请参考配置基于 *NoSQL* 数据库 *Redis* 的会话持久性（见第 1147 页）。

配置基于 SQL 数据库的会话持久性

要在 xigemaAS 中配置一个或多个服务器以将会话数据持久存储到数据库，请完成下列步骤。

1. 定义可以在您的所有服务器之间复用的共享会话管理配置。作为最低要求，您必须完成下列步骤：
   a. 启用 `sessionDatabase-1.0` 功能部件。
   b. 定义数据源：

```
<dataSource id="SessionDS" ... />
```

   c. 请参阅会话数据库配置中的数据源。

```
<httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS" ... />
```

   d. 请参阅会话管理配置中的持久存储位置。

```
<httpSession storageRef="SessionDB" ... />
```

> 注：`httpSession` 元素的 `storageRef` 属性及 `httpSessionDatabase` 元素的 `id` 属性不是必需的。如果已启用 `sessionDatabase-1.0` 功能部件，并且 `httpSessionDatabase` 元素引用了有效的数据源，那么即使未设置 `storageRef` 属性，也会启用会话持久性。

有关 `httpSession` 和 `httpSessionDatabase` 元素的详细信息，请参阅*Redis Database Session Persistence*（见第 1017 页）。

例如，可以创建名称为 ${shared.config.dir}/httpSessionPersistence.xml 的文件，如下所示：

```
<server description="Demonstrates HTTP Session Persistence Configuration">

    <featureManager>
        <feature>sessionDatabase-1.0</feature>
        <feature>servlet-3.0</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="${httpPort}">
        <tcpOptions soReuseAddr="true"/>
    </httpEndpoint>

    <fileset id="DerbyFiles" includes="*.jar" dir="${shared.resource.dir}/derby/client"/>
    <library id="DerbyLib" filesetRef="DerbyFiles"/>
    <jdbcDriver id="DerbyDriver" libraryRef="DerbyLib"/>
    <dataSource id="SessionDS" jdbcDriverRef="DerbyDriver">
        <properties.derby.client user="user1" password="password1"
                                 databaseName="${shared.resource.dir}/databases/SessionDB"
                                 createDatabase="create"/>
    </dataSource>

    <httpSessionDatabase id="SessionDB" dataSourceRef="SessionDS"/>
    <httpSession storageRef="SessionDB" cloneId="${cloneId}"/>

    <application id="test" name="test" type="ear" location="${shared.app.dir}/test.ear"/>
```

```
</server>
```

注：配置多个服务器以将会话数据持久存储到同一个数据库时，这些服务器必须共享同一项会话管理配置。不支持任何其他配置。

HTTP Server 插件使用插入至 response/request 头的克隆标识维护请求之间的会话亲缘关系。虽然克隆标识通常不会更改，但在 xigemaAS 中，您首次启动服务器时将生成克隆标识，并且在使用 `--clean` 选项启动服务器时将重新生成克隆标识。为在生产环境中使用，手动分配克隆标识将确保该标识保持不变并且请求亲缘关系得到正确维护。此克隆标识对于每个服务器必须是唯一的，长度可为 8 到 9 个字母数字字符，在以下步骤 3 中指定。

2. 在每个服务器中包括共享会话管理配置。例如，为名称为 `s1` 和 `s2` 的服务器实例创建两个 server.xml 文件，如下所示：
   - ${wlp.user.dir}/servers/s1/server.xml
   - ${wlp.user.dir}/servers/s2/server.xml

```
<server description="Example Server">
    <include location="${shared.config.dir}/httpSessionPersistence.xml"/>
</server>
```

请参阅在配置文件中使用 *include* 元素（见第 1130 页）。

3. 在每个服务器的 *bootstrap.properties* 文件中指定唯一变量。
   - ${wlp.user.dir}/servers/s1/bootstrap.properties

```
httpPort=9081
cloneId=s1
```

   - ${wlp.user.dir}/servers/s2/bootstrap.properties

```
httpPort=9082
cloneId=s2
```

4. 启动服务器之前，请为会话持久性创建表。
   - 如果服务器是安装在其中一个分布式操作系统上，那么不需要执行其他操作。服务器会自动创建表。
   - 如果服务器正在为会话持久性使用 DB2®，那么可以增加页大小以优化将大量数据写入数据库的性能。
5. 使主管 xigemaAS 服务器的所有机器的系统时钟同步。如果系统时钟不同步，那么可能会使会话过早地失效。
6. 可选： 必要时，在 xigemaAS 概要文件中将 HTTP 会话和安全性集成在一起。缺省情况下，在启用了安全性的受保护资源中创建和访问会话之后，只有该会话的始发所有者可以访问该会话。

配置基于 NoSQL 数据库 Redis 的会话持久性

默认情况下，xigemaAS 只会将 HTTP Session （HTTP 会话）存储在内存中，并且只存储当前服务器上的会话信息。这就导致在分布式集群环境下，会出现 session 数据不一致的情况。此外，如果服务器宕机，也会导致 session 数据丢失。xigemaAS 提供了 `redisSession-1.0` 功能部件，为 NoSQL 数据库 Redis 提供了会话持久性支持。

请在配置基于 Redis 的会话持久性之前完成 Redis 单机或集群的安装，并正确配置 Redis 数据源。有关配置 Redis 数据源的相关信息，请参考在 *xigemaAS* 中配置 *Redis DB* 连接（见第 1207 页）。

要在 xigemaAS 概要文件中将会话数据持久存储到 Redis 数据库，请完成下列步骤。

1. 请在 server.xml 文件的 `featureManager` 元素下添加 `redisSession-1.0` 的元素声明。例如：

```
<featureManager>
    <feature>redisSession-1.0</feature>
</featureManager>
```

2. 在 server.xml 文件中添加 httpSessionRedis 配置。

```
<httpSessionRedis id="SessionRedis"  sessionDBRef="redisDBTest1"/>
```

   👉 注：`sessionDBRef` 表示引用的 Redis 数据源，其值是引用的 RedisDB 的 id。

3. 配置会话策略。

```
<sessionPolicy id="sessionPolicy1" writeFrequency="END_OF_SERVLET_SERVICE"
  writeContents="ONLY_UPDATED_ATTRIBUTES"/>
```

   👉 注：其中，`writeFrequency` 为 session 数据写入数据库中的时间点，默认值为 END_OF_SERVLET_SERVICE，即在 Servlet 执行完成之后将会话数据写入持久存储库。`writeContents`：指定应该写入持久存储库的会话数据量。默认值为 ONLY_UPDATED_ATTRIBUTES，即仅写入已更新的属性。

完成上述配置后，部署在服务器上的所有 Web 应用程序被访问产生的会话数据都会存入对应的 redis 数据库中。对于 Redis 集群，集群中一条 session 数据至少会存在两台 Redis 服务器中（主和备各存一份）。当其中一台宕机后，另一台 Redis 服务器可以正常接受和处理请求。

配置和部署基本 JCA 资源适配器

可以配置和部署基本 Java™ EE 连接器体系结构 (JCA) 连接工厂和资源适配器。

您可以安装资源适配器和配置它提供的资源的实例。此任务使用一个名为 `ExampleRA.rar` 的示例资源适配器，它提供三种类型的资源：一个连接工厂和两种类型的受管对象。

1. 在 server.xml 文件中启用 JCA 功能部件。

   可以在 [path_to_xigamaas\wlp\usr\servers\server_name] 位置找到 server.xml 文件

```
<server>
  <featureManager>
    <feature>jca-1.6</feature>
    <feature>servlet-3.0</feature>
  </featureManager>
</server>
```

2. 将资源适配器 RAR 文件 (ExampleRA.rar) 放入服务器的 dropins 文件夹。

   如果服务器正在运行，那么您在控制台日志中将看到与以下消息相似的消息，指出已安装该资源适配器：

```
[AUDIT ] J2CA7001I: Resource adapter ExampleRA installed in 1.306 seconds.
```

3. 检查部署描述符、注解以及资源适配器的其他文档，以确定适配器提供的资源类型以及每个适配器接受的配置属性。

   示例资源适配器 ExampleRA.rar 在 `ra.xml` 部署描述符中具有这些信息。ra.xml 文件位于 [path_to_ExampleRA\ExampleRA\META-INF.]。部署描述符确定可配置的三种类型的资源。

```
<connection-definition>
```

```
 <managedconnectionfactory-
class>com.ibm.example.jca.adapter.ManagedConnectionFactoryImpl</
managedconnectionfactory-class>
 <config-property>
  <config-property-name>tableName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>
 ...
</connection-definition>

    <adminobject>
      <adminobject-interface>javax.resource.cci.ConnectionSpec</
adminobject-interface>
      <adminobject-class>com.ibm.example.jca.adapter.ConnectionSpecImpl</
adminobject-class>
      <config-property>
        <config-property-name>readOnly</config-property-name>
        <config-property-type>java.lang.Boolean</config-property-type>
        <config-property-value>false</config-property-value>
      </config-property>
    </adminobject>

    <adminobject>
      <adminobject-interface>javax.resource.cci.InteractionSpec</
adminobject-interface>
      <adminobject-class>com.ibm.example.jca.adapter.InteractionSpecImpl</
adminobject-class>
      <config-property>
        <description>Function name. Supported values are: ADD, FIND,
 REMOVE</description>
        <config-property-name>functionName</config-property-name>
        <config-property-type>java.lang.String</config-property-type>
      </config-property>
    </adminobject>
```

4. 在 server.xml 文件，配置可用资源类型的实例。

```
<server>
  <featureManager>
    <feature>jca-1.6</feature>
    <feature>servlet-3.0</feature>
  </featureManager>

  <connectionFactory jndiName="eis/conFactory">
    <properties.ExampleRA tableName="TABLE1"/>
  </connectionFactory>

  <adminObject jndiName="eis/conSpec">
    <properties.ExampleRA.ConnectionSpec/>
  </adminObject>

  <adminObject jndiName="eis/iSpec_ADD">
    <properties.ExampleRA.InteractionSpec functionName="ADD"/>
  </adminObject>

  <adminObject jndiName="eis/iSpec_FIND">
    <properties.ExampleRA.InteractionSpec functionName="FIND"/>
  </adminObject>
```

```
</server>
```

5. 使用资源注入来访问 Servlet 中的资源；例如：

```
    @Resource(lookup = "eis/conFactory")
    private ConnectionFactory conFactory;

    @Resource(lookup = "eis/conSpec")
    private ConnectionSpec conSpec;

    @Resource(lookup = "eis/iSpec_ADD")
    private InteractionSpec iSpec_ADD;

    @Resource(lookup = "eis/iSpec_FIND")
    private InteractionSpec iSpec_FIND;

    ...

        MappedRecord input =
conFactory.getRecordFactory().createMappedRecord("input");
        input.put("city", "Rochester");
        input.put("state", "Minnesota");
        input.put("population", 106769);

        Connection con = conFactory.getConnection(conSpec);
        try {
            Interaction interaction = con.createInteraction();
            interaction.execute(iSpec_ADD, input);
            interaction.close();
        } finally {
            con.close();
        }
```

注：如果您要从命名空间中查找资源，而不是使用注入，那么必须在 server.xml 文件中启用 JNDI 功能部件。

JCA 配置元素概述

Java Platform, Enteprise Edition 连接器体系结构 (JCA) 功能部件提供了配置元素来定义连接工厂、受管对象和激活规范的实例，并使这些实例与已安装的资源适配器相关联。每个 JCA 配置元素都由两个基本部分（顶级元素和子元素）组成，这两部分是所配置的实例必需的。

顶级元素用于配置 xigemaAS 概要文件服务器提供的一般功能，例如，JNDI 名称、连接管理和容器认证。子元素用于将实例与安装的资源适配器绑定，并使您可指定供应商定义的配置属性。

通用 JCA 预定义顶级配置元素：

- connectionFactory
- adminObject
- activationSpec

如果已启用 JMS 功能部件，那么还有用于 JMS 的预定义的通用配置元素：

- jmsConnectionFactory
- jmsQueueConnectionFactory
- jmsTopicConnectionFactory
- jmsDestination

- jmsQueue
- jmsTopic
- jmsActivationSpec

子元素是在安装资源适配器时从资源适配器部署描述符和注解生成的。您不会在可用服务器配置元素的静态文档中看到可用子元素的任何文档。

请按照下列规则来生成服务器配置子元素的名称：

- 如果资源适配器刚好提供了所列示的其中一个类别（例如，connectionFactory 或 adminObject）的一个接口，那么子元素为：`properties.<rar_identifier>`
- 如果接口名称唯一，并且没有软件包名称，那么子元素为：`properties.<rar_identifier>.<InterfaceName>`
- 如果实现名称唯一，并且没有软件包名称，那么子元素为：`properties.<rar_identifier>.<ImplementationName>`
- 在其他情况下，子元素名称为 `properties.<rar_identifier>.<fully.qualified.InterfaceName>` 或者 `properties.<rar_identifier>.<fully.qualified.ImplementationName>`

以下示例说明了由一个标识为 MyAdapter 的资源适配器提供的每个类别的一个接口的情况：

```
<connectionFactory jndiName="eis/cf1" containerAuthDataRef="auth1">
 <properties.MyAdapter portNumber="1234" someVendorProperty="100"/>
</connectionFactory>

<connectionFactory jndiName="eis/cf2" containerAuthDataRef="auth2">
 <properties.MyAdapter portNumber="1234" someVendorProperty="200"/>
</connectionFactory>
```

```
<jmsConnectionFactory jndiName="jms/cf">
 <properties.MyAdapter serverName="localhost" anotherProperty="40"/>
</jmsConnectionFactory>
```

```
<jmsQueueConnectionFactory jndiName="jms/qcf">
 <properties.MyAdapter serverName="localhost" vendorProp1="1"/>
</jmsQueueConnectionFactory>
```

```
<jmsTopicConnectionFactory jndiName="jms/tcf">
 <properties.MyAdapter serverName="localhost" prop1="A" prop2="B"/>
</jmsTopicConnectionFactory>
```

```
<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter functionName="find" executionTimeout="5000"/>
</adminObject>
```

```
<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter name="DEST1"/>
</jmsDestination>
```

```
<jmsQueue jndiName="jms/queue1">
 <properties.MyAdapter queueName="QUEUE1"/>
```

```
</jmsQueue>
```

```
<jmsTopic id="topic1" jndiName="jms/topic1">
 <properties.MyAdapter topicName="TOPIC1"/>
</jmsTopic>
```

```
<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter prop1="a" prop2="b" prop3="c"/>
</activationSpec>
```

```
<jmsActivationSpec id="app1/module1/MyJMSMessageDrivenBean">
 <properties.MyAdapter destinationRef="topic1"/>
</jmsActivationSpec>
```

配置资源适配器

您可以配置符合 Java™ EE 连接器体系结构 (JCA) 规范 V1.6、V1.5 或 V1.0 的资源适配器。

您可以安装和配置资源适配器和各种连接工厂、受管对象以及 JCA 规范中定义的激活规范。

1. 更新 server.xml 文件以在 featureManager 标记下添加 `jca-1.6` 功能部件。

   server.xml 文件位于 [*path_to_xigemaas*\wlp\usr\servers\server_name]

   ```
   <featureManager>
    <feature>jca-1.6</feature>
   </featureManager>
   ```

2. （可选）根据系统的需要来启用下列其他功能部件：

   - 如果您的资源适配器提供 JMS 规范接口，请启用 `jms-1.1` 功能部件。

     ```
     <feature>jms-1.1</feature>
     ```

   - 如果您要从应用程序查找连接工厂和受管对象，请启用 `jndi-1.0` 功能部件。

     ```
     <feature>jndi-1.0</feature>
     ```

   - 如果资源适配器提供了消息驱动的 Bean 的激活规范，请启用 `mdb-3.1` 功能部件。

     ```
     <feature>mdb-3.1</feature>
     ```

   - 如果您的资源适配器支持入站安全性，请启用 `jcaInboundSecurity-1.0` 功能部件。

     ```
     <feature>jcaInboundSecurity-1.0</feature>
     ```

   - 如果资源适配器支持 Bean 验证，并且您希望验证 Bean，那么可以启用 `beanValidation-1.0` 功能部件。

     ```
     <feature>beanValidation-1.0</feature>
     ```

3. 在服务器中配置一个或多个资源适配器。可以使用下列其中一种方法来配置资源适配器。

   - 通过编辑 server.xml 文件配置独立资源适配器。

     ```
     <resourceAdapter location="C:/adapters/MyAdapter.rar"/>
     ```

- 通过编辑 server.xml 文件来配置嵌入式资源适配器以安装嵌入了一个或多个资源适配器模块的应用程序。以下示例假定 app1.ear 文件包含一个或多个嵌入的 RAR 文件：

```
<application location="C:/applications/app1.ear"/>
```

- 允许服务器通过删除服务器 drop-ins 文件夹中的 RAR 文件来自动配置独立资源适配器。

```
wlp/usr/servers/your-server-name/dropins/MyDropinAdapter.rar
```

- 允许服务器通过删除服务器 dropins 文件夹中的 EAR 文件来自动配置包含一个或多个嵌入式资源适配器的应用程序。以下示例假定 app2.ear 文件包含一个或多个嵌入的 RAR 文件：

```
wlp/usr/servers/your-server-name/dropins/app2.ear
```

4. 启动应用程序服务器。启动服务器后，会在 console.log 文件中显示类似以下内容的消息：

```
[AUDIT ] J2CA7001I: Resource adapter MyAdapter installed in 0.495 seconds.
[AUDIT ] J2CA7001I: Resource adapter MyDropinAdapter installed in 0.311
 seconds.
[AUDIT ] J2CA7001I: Resource adapter app1.MyEmbeddedAdapter installed in
 0.247 seconds.
[AUDIT ] J2CA7001I: Resource adapter app2.anotherEmbeddedAdapter installed
 in 0.518 seconds.
```

需要资源适配器的唯一标识，以将所配置的连接工厂、受管对象和激活规范的实例标识为与已安装的资源适配器相关联。对于独立资源适配器，将模块名称用作标识。对于嵌入在应用程序中的资源适配器，将应用程序名称与模块名称的组合用作标识（应用程序名称与模块名称之间用句点字符定界）。

- 要使用包含资源适配器标识 MyAdapter 的 properties.MyAdapter 子元素指定独立资源适配器的属性：

```
<resourceAdapter location="C:/adapters/MyAdapter.rar">
  <properties.MyAdapter logFile="${server.output.dir}/logs/myAdapter.log"/>
</resourceAdapter>
```

- 要使用一个包括资源适配器标识 MyAdapter 的 properties.MyAdapter 子元素使连接工厂与独立资源适配器相关联：

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
<connectionFactory jndiName="eis/cf">
 <properties.MyAdapter serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- 要使连接工厂与资源适配器 MyEmbeddedAdapter（在 app1 应用程序中已启用）相关联，请使用 properties.app1.MyEmbeddedAdapter 子元素：

```
<application location="C:/applications/app1.ear"/>
<connectionFactory jndiName="eis/cf">
 <properties.app1.MyEmbeddedAdapter serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- 在某些情况下，模块名称不足以唯一地充当标识。例如，如果您安装了同一资源适配器的两个不同版本，可能就会发生这种情况。此外，模块名称可能是唯一的，但是将它用于配置不可取，因为它太长或者包含非字母数字字符。您可以通过指定 id 属性来覆盖资源适配器标识。

以下示例说明如何覆盖独立资源适配器的标识：

```
<resourceAdapter id="MyAdapterV1" location="C:/adapters/version-1.0/
MyAdapter.rar"/>
<resourceAdapter id="MyAdapterV2" location="C:/adapters/version-2.0/
MyAdapter.rar"/>
<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapterV1 serverName="localhost" portNumber="1234"/>
</connectionFactory>
<connectionFactory jndiName="eis/cf2">
 <properties.MyAdapterV2 serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- 以下示例说明如何覆盖嵌入在应用程序中的资源适配器的标识。该示例将标识更改为 MyEmbeddedRA：

```
<application location="C:/applications/app1.ear">
 <resourceAdapter id="MyEmbeddedAdapter" alias="MyEmbeddedRA"/>
</application>
<connectionFactory jndiName="eis/cf">
 <properties.app1.MyEmbeddedRA serverName="localhost" portNumber="1234"/>
</connectionFactory>
```

- 要计算嵌入式资源适配器的模块名称，资源适配器部署描述符 (ra.xml) 中的 <module-name> 条目优先于模块名称。例如，在 ra.xml 中提供以下定义：

```
<connector ...>
 <module-name>MyRARModule</module-name>
</connector>
```

模块名称将设置为“MyRARModule”。

如果连接器部署描述符中缺少模块名称，那么将使用引用应用程序部署描述符 (application.xml) 中的资源适配器模块的 URI 的短格式。例如，在 application.xml 中提供以下模块定义：

```
<module>
 <connector>connectors/MyRARModule.rar</connector>
</module>
```

模块名称将计算为“MyRARModule”。

如果应用程序中嵌入了多个资源适配器，并定义相同 <module-name> 值，那么 application.xml 中列出的第一个资源适配器会使用此模块名称。根据完整格式的 URI（所有正斜杠 / 字符转换为句点 .）计算具有此相同冲突名称的另一个连接器的模块名称。. 例如，如果两个连接器都嵌入在应用程序，并且它们在 ra.xml 中都包含以下定义：

```
<connector ...>
 <module-name>MyRARModule</module-name>
</connector>
```

并且下列定义存在于 application.xml 中：

```
<module>
 <connector>subfolder1/connector1.rar</connector>
</module>
<module>
 <connector>subfolder2/connector2.rar</connector>
```

```
</module>
```

第一个连接器的模块名称将为“MyRARModule”，第二个连接器的模块名称将为“subfolder2.connector2.rar”

配置 JCA 连接工厂

您可以配置符合 Java EE 连接器体系结构 (JCA) 规范的连接工厂。

您可以配置安装的资源适配器提供的连接工厂类型的一个或多个连接工厂实例。

👉 注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器，请参阅有关配置资源适配器的文档以获取更多详细信息。以下步骤中提供了配置基本场景的端到端示例。

1. 更新 server.xml 文件以在 featureManager 标记下添加 `jca-1.6` 功能部件。

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects. -->
  ...
</featureManager>
```

2. 安装资源适配器。

   例如，按如下所示更新 server.xml 文件：

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. 配置一个或多个连接工厂实例。

   当您配置连接工厂实例时，必须提供 `properties` 子元素（即使您不想覆盖任何配置属性也是如此），以便将 connectionFactory 元素与特定资源适配器所提供的连接工厂接口相关联。在以下示例中，`MyAdapter` 资源适配器仅提供一种类型的连接工厂：

```
<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapter portNumber="1234" someVendorProperty="100"/>
</connectionFactory>

<connectionFactory jndiName="eis/cf2" containerAuthDataRef="auth2">
 <connectionManager maxPoolSize="20" connectionTimeout="0"/>
 <properties.MyAdapter portNumber="1234" someVendorProperty="200"/>
</connectionFactory>
<authData id="auth2" user="user2" password="{xor}Lz4sLCgwLTtt"/>
```

4. （可选）如果需要，请标识可用的连接工厂属性子元素名称。

   - 如果资源适配器刚好提供了一个连接工厂接口（排除任何 JMS 连接工厂），那么子元素为：`properties.<rar_identifier>`
   - 如果接口名称唯一，并且没有软件包名称，那么子元素名称为：`properties.<rar_identifier>.<InterfaceName>`
   - 如果实现名称唯一，并且没有软件包名称，那么子元素名称为：`properties.<rar_identifier>.<ImplementationName>`
   - 在其他情况下，子元素名称为：`properties.<rar_identifier>.<fully.qualified.InterfaceName>`

使用以下示例来了解如何配置具有两个连接工厂且接口类名唯一的资源适配器。

在 ra.xml 文件的以下片段中，MyAdapter 资源适配器提供了具有唯一接口类名的两个连接工厂：

```
<connection-definition>
  <config-property>
  <config-property-name>ServerName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</
connectionfactory-impl-class>
</connection-defintion>
```

```
<connection-definition>
 <config-property>
  <config-property-name>ServerName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.sql.DataSource</connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.DataSourceImpl</
connectionfactory-impl-class>
</connection-defintion>
```

以下是此方案的服务器配置的示例：

```
<connectionFactory jndiName="eis/cf">
 <properties.MyAdapter.ConnectionFactory serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="jdbc/ds">
 <properties.MyAdapter.DataSource serverName="localhost"/>
</connectionFactory>
```

使用以下示例来了解如何配置具有两个连接工厂且实现类名唯一的资源适配器。

在 ra.xml 文件的以下片段中，MyAdapter 资源适配器提供了具有唯一实现类名的两个连接工厂：

```
<connection-definition>
  <config-property>
  <config-property-name>ServerName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</
connectionfactory-impl-class>
</connection-defintion>

<connection-definition>
  <config-property>
  <config-property-name>ServerName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>com.vendor.adapter.ConnectionFactory</
connectionfactory-interface>
```

```
 <connectionfactory-impl-class>com.vendor.adapter.MyConnectionFactoryImpl</
connectionfactory-impl-class>
</connection-defintion>
```

以下是此方案的服务器配置的示例：

```
<connectionFactory jndiName="eis/cf1">
 <properties.MyAdapter.ConnectionFactoryImpl serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/cf2">
 <properties.MyAdapter.MyConnectionFactoryImpl serverName="localhost"/>
</connectionFactory>
```

使用以下示例来了解如何配置具有两个连接工厂，而简单接口类名和实现类名都不唯一的资源适配器。

在 ra.xml 文件的以下片段中，MyAdapter 资源适配器提供了两个连接工厂，而简单接口类名和实现类名都不唯一：

```
<connection-definition>
 <config-property>
  <config-property-name>ServerName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.ConnectionFactoryImpl</
connectionfactory-impl-class>
</connection-defintion>

<connection-definition>
  <config-property>
  <config-property-name>HostName</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
 <connectionfactory-interface>com.vendor.adapter.custom.ConnectionFactory</
connectionfactory-interface>
 <connectionfactory-impl-class>com.vendor.adapter.custom.ConnectionFactoryImpl</
connectionfactory-impl-class>
</connection-defintion>
```

以下是此方案的服务器配置的示例：

```
<connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.javax.resource.cci.ConnectionFactory
 serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.com.vendor.adapter.custom.ConnectionFactory
 hostName="localhost"/>
</connectionFactory>
```

可以覆盖配置元素名称的后缀。请参阅有关定制 JCA 配置元素的信息，以了解如何覆盖配置元素名称的后缀。

配置 JCA 受管对象

您可以配置符合 Java EE 连接器体系结构 (JCA) 规范的受管对象。

您可以配置安装的资源适配器提供的受管对象的一个或多个实例。

👉 注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器，请参阅有关配置资源适配器的文档以获取更多详细信息。以下步骤中提供了配置基本场景的端到端示例。

1. 更新 server.xml 文件以在 featureManager 标记下添加 `jca-1.6` 功能部件。

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects. -->
  ...
</featureManager>
```

2. 安装资源适配器。

   例如，按如下所示更新 server.xml 文件：

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. 配置一个或多个受管对象实例。

   当您配置受管对象实例时，必须提供 `properties` 子元素（即使您不想覆盖任何配置属性也是如此），以将 adminObject 元素与特定资源适配器所提供的受管对象类型相关联。在以下示例中，`MyAdapter` 资源适配器仅提供一种类型的受管对象：

```
<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter functionName="find" executionTimeout="5000"/>
</adminObject>
```

4. （可选）如果需要，请标识可用的受管对象属性子元素名称。

   - 如果资源适配器仅提供一个受管对象接口，且不包含任何 JMS 目标、队列和主题，那么子元素名称为：`properties.<rar_identifier>`
   - 如果接口名称唯一，并且没有软件包名称，那么子元素名称为：`properties.<rar_identifier>.<InterfaceName>`
   - 如果实现名称唯一，并且没有软件包名称，那么子元素名称为：`properties.<rar_identifier>.<ImplementationName>`
   - 如果接口名称与实现名称的组合唯一，并且没有软件包名称，那么子元素名称为：`properties.<rar_identifier>.<InterfaceName>-<ImplementationName>`
   - 在其他情况下，子元素名称为：`properties.<rar_identifier>.<fully.qualified.InterfaceName>-<fully.qualified.ImplementationName>`

使用以下示例来了解如何配置具有两个受管对象且接口类名唯一的资源适配器。

在 ra.xml 文件的以下片段中，`MyAdapter` 资源适配器提供了具有唯一接口类名的两个受管对象：

```
<adminobject>
<adminobject-interface>javax.resource.cci.ConnectionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.ConnectionSpecImpl</adminobject-class>
<config-property>
```

```
 <config-property-name>isolationLevel</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.InteractionSpecImpl</adminobject-class>
<config-property>
 <config-property-name>FunctionName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>
```

以下是此方案的服务器配置的示例：

```
<adminObject jndiName="eis/connectionSpec">
 <properties.MyAdapter.ConnectionSpec isolationLevel="4"/>
</adminObject>

<adminObject jndiName="eis/interactionSpec">
 <properties.MyAdapter.InteractionSpec functionName="find"/>
</adminObject>
```

使用以下示例来了解如何配置具有两个受管对象且实现类名唯一的资源适配器。

在 ra.xml 文件中的以下片段中，MyAdapter 资源适配器提供了具有唯一实现类名的两种受管对象：

```
<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.FinderInteractionSpec</adminobject-class>
<config-property>
 <config-property-name>ResultSetType</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.UpdaterInteractionSpec</adminobject-class>
<config-property>
 <config-property-name>ExecutionTimeout</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
</config-property>
...
</adminobject>
```

以下是此方案的服务器配置的示例：

```
<adminObject jndiName="eis/finder">
 <properties.MyAdapter.FinderInteractionSpec resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/updater">
 <properties.MyAdapter.UpdaterInteractionSpec executionTimeout="3000"/>
```

```
</adminObject>
```

使用以下示例来了解如何配置具有两个受管对象，而简单接口类名和实现类名都不唯一的资源适配器。

在 ra.xml 文件的以下片段中，MyAdapter 资源适配器提供了两个受管对象，而简单接口类名和实现类名都不唯一：

```
<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.finder.InteractionSpecImpl</adminobject-
class>
<config-property>
 <config-property-name>ResultSetType</config-property-name>
 <config-property-type>java.lang.Integer</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.resource.cci.InteractionSpec</adminobject-interface>
<adminobject-class>com.vendor.adapter.updater.InteractionSpecImpl</adminobject-
class>
<config-property>
 <config-property-name>ExecutionTimeout</config-property-name>
 <config-property-type>java.lang.Long</config-property-type>
</config-property>
...
</adminobject>
```

以下是此方案的服务器配置的示例：

```
<adminObject jndiName="eis/finder">
 <properties.MyAdapter.javax.resource.cci.InteractionSpec-
com.vendor.adapter.finder.InteractionSpecImpl resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/updater">
 <properties.MyAdapter.javax.resource.cci.InteractionSpec-
com.vendor.adapter.updater.InteractionSpecImpl executionTimeout="3000"/>
</adminObject>
```

可以覆盖配置元素名称的后缀。请参阅有关定制 JCA 配置元素的信息，以了解如何覆盖配置元素名称的后缀。

配置 JCA 激活规范

您可以配置符合 Java EE 连接器体系结构 (JCA) 规范的激活规范。

您可以配置由已安装的资源适配器提供的激活规范的一个或多个实例。

👉 注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器，请参阅有关配置资源适配器的文档以了解更多详细信息。下列步骤中提供了有关配置基本方案的端到端示例。

1. 更新 server.xml 文件以在 featureManager 标记下添加 jca-1.6 功能部件。

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects. -->
```

```
  ...
</featureManager>
```

2. 安装资源适配器。

   例如，按如下所示更新 server.xml 文件：

```
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>
```

3. 配置一个或多个激活规范。

   当您配置激活规范时，必须提供 properties 子元素（即使您不想覆盖任何配置属性也是如此），以将 activationSpec 元素与特定资源适配器所提供的消息侦听器类型相关联。在以下示例中，MyAdapter 资源适配器仅提供一种类型的消息侦听器：

```
<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter messageFilter="ALL"/>
</activationSpec>
```

4. 如果需要，请标识可用的激活规范属性子元素名称。

   - 如果资源适配器刚好提供了一个消息侦听器接口（排除任何 JMS 连接工厂），那么子元素名称为：properties.<rar_identifier>
   - 如果消息侦听器接口名称唯一，并且没有软件包名称，那么子元素名称为：properties.<rar_identifier>.<MessageListenerInterfaceName>
   - 如果激活规范实现名称唯一，并且没有软件包名称，那么子元素名称为：properties.<rar_identifier>.<ActivationSpecificationImplementationName>
   - 如果激活规范实现名称唯一，并且没有软件包名称，那么子元素名称为：properties.<rar_identifier>.<ActivationSpecificationImplementationName>
   - 在其他情况下，子元素名称为：properties.<rar_identifier>.<fully.qualified.MessageListenterInterfaceName>

使用以下示例来了解如何配置具有两种消息侦听器类型且接口类名唯一的资源适配器。

在 ra.xml 文件中的以下片段中，MyAdapter 资源适配器提供了具有唯一接口类名的两种消息侦听器类型：

```
<messagelistener>
<messagelistener-type>javax.resource.cci.MessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.CCIActivationSpec</activationspec-
class>
 <config-property>
  <config-property-name>maxSize</config-property-name>
  <config-property-type>java.lang.Long</config-property-type>
 </config-property>
 ...
</activationspec>
...
</messagelistener>

<messagelistener>
<messagelistener-type>com.vendor.adapter.MyMessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.MyActivationSpec</activationspec-class>
 <config-property>
  <config-property-name>messageFilter</config-property-name>
```

```
   <config-property-type>java.lang.String</config-property-type>
  </config-property>
  ...
</activationspec>
...
</messagelistener>
```

以下是此场景的服务器配置的示例：

```
<activationSpec id="app1/module1/CCIMessageDrivenBean">
 <properties.MyAdapter.MessageListener maxSize="1024"/>
</activationSpec>

<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter.MyMessageListener messageFilter="ALL"/>
</activationSpec>
```

使用以下示例以了解如何配置资源适配器，该资源适配器的两种消息侦听器类型具有唯一实现类名

在 ra.xml 文件中的以下片段中，MyAdapter 资源适配器提供了具有唯一实现类名的两种消息侦听器类型：

```
<messagelistener>
<messagelistener-type>javax.resource.cci.MessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.CCIActivationSpec</activationspec-
class>
 <config-property>
  <config-property-name>maxSize</config-property-name>
  <config-property-type>java.lang.Long</config-property-type>
 </config-property>
...
</activationspec>
...
</messagelistener>

<messagelistener>
<messagelistener-type>com.vendor.adapter.MessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.MyActivationSpec</activationspec-class>
 <config-property>
  <config-property-name>messageFilter</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
...
</activationspec>
...
</messagelistener>
```

以下是此场景的服务器配置的示例：

```
<activationSpec id="app1/module1/CCIMessageDrivenBean">
 <properties.MyAdapter.CCIActivationSpec maxSize="1024"/>
</activationSpec>

<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter.MyActivationSpec messageFilter="ALL"/>
</activationSpec>
```

使用以下示例以了解如何配置资源适配器，该资源适配器的两种消息侦听器类型的简单接口类名或实现类名均不唯一。

在 ra.xml 文件的以下片段中，MyAdapter 资源适配器提供了两种消息侦听器类型，这两种消息侦听器类型的简单接口类名或实现类名均不唯一：

```
<messagelistener>
<messagelistener-type>javax.resource.cci.MessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.cci.ActivationSpec</activationspec-
class>
 <config-property>
  <config-property-name>maxSize</config-property-name>
  <config-property-type>java.lang.Long</config-property-type>
 </config-property>
...
</activationspec>
...
</messagelistener>

<messagelistener>
<messagelistener-type>com.vendor.adapter.MessageListener</messagelistener-type>
<activationspec>
 <activationspec-class>com.vendor.adapter.ActivationSpec</activationspec-class>
 <config-property>
  <config-property-name>messageFilter</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
 </config-property>
...
</activationspec>
...
</messagelistener>
```

以下是此场景的服务器配置的示例：

```
<activationSpec id="app1/module1/CCIMessageDrivenBean">
 <properties.MyAdapter.javax.resource.cci.MessageListener maxSize="1024"/>
</activationSpec>

<activationSpec id="app1/module1/MyMessageDrivenBean">
 <properties.MyAdapter.com.vendor.adapter.MessageListener messageFilter="ALL"/>
</activationSpec>
```

可以覆盖配置元素名称的后缀。请参阅有关定制 JCA 配置元素的信息以了解如何覆盖配置元素名称的后缀。

配置 JMS 连接工厂

您可以配置资源适配器提供的符合 Java EE 连接器体系结构 (JCA) 规范的 JMS 连接工厂。

您可以配置由已安装的资源适配器提供的 JMS 连接工厂类型的一个或多个 JMS 连接工厂实例。

为下列类型的 JMS 连接工厂提供了配置元素：

- javax.jms.ConnectionFactory：jmsConnectionFactory
- javax.jms.QueueConnectionFactory：jmsQueueConnectionFactory
- javax.jms.TopicConnectionFactory：jmsTopicConnectionFactory

注:

要添加 xigemaAS 的 JCA 支持，必须使用 某个文本编辑器来编辑 server.xml 文件。

1. 配置一个或多个 JMS 连接工厂实例。

   当您配置连接工厂实例时，必须提供 properties 子元素（即使您不想覆盖任何配置属性也是如此），以将 jmsConnectionFactory、jmsQueueConnectionFactory 或 jmsTopicConnectionFactory 元素与特定资源适配器所提供的连接工厂接口相关联。properties 子元素始终遵循 JMS 连接工厂的模式 properties.<rar_identifier>。

   在以下示例中，MyAdapter 资源适配器仅提供一种类型的连接工厂：

```
<jmsConnectionFactory jndiName="jms/cf" containerAuthDataRef="auth1">
 <properties.MyAdapter serverName="localhost" anotherProperty="40"/>
</jmsConnectionFactory>
<authData id="auth1" user="user1" password="{xor}Lz4sLCgwLTtu"/>

<jmsQueueConnectionFactory jndiName="jms/qcf">
 <connectionManager maxPoolSize="20" connectionTimeout="0"/>
 <properties.MyAdapter serverName="localhost" vendorProp1="1"/>
</jmsQueueConnectionFactory>

<jmsTopicConnectionFactory jndiName="jms/tcf">
 <properties.MyAdapter serverName="localhost" prop1="A" prop2="B"/>
</jmsTopicConnectionFactory>
```

注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器，并启用了 jms-1.1 功能部件。请参阅配置资源适配器（见第 1152 页）主题以获取更多详细信息。

配置 JMS 目标

您可以配置资源适配器提供的符合 Java EE 连接器体系结构 (JCA) 规范的 JMS 目标。

您可以配置由已安装的资源适配器提供的 JMS 目标、队列或主题类型的一个或多个实例。

为下列类型的 JMS 目标提供了配置元素：

- javax.jms.Destination: jmsDestination
- javax.jms.Queue: jmsQueue
- javax.jms.Topic: jmsTopic

注:

要添加 xigemaAS 的 JCA 支持，必须使用 某个文本编辑器来编辑 server.xml 文件。

1. 配置一个或多个 JMS 目标、队列或主题实例。

   当您配置目标实例时，必须提供 properties 子元素（即使您不想覆盖任何配置属性也是如此），以将 jmsDestination、jmsQueue 或 jmsTopic 元素与特定资源适配器所提供的 JMS 目标接口相关联。在以下示例中，MyAdapter 资源适配器仅提供一种类型的 JMS 目标、一种类型的 JMS 队列和一种类型的 JMS 主题：

```
<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter name="DEST1"/>
</jmsDestination>

<jmsQueue jndiName="jms/queue1">
 <properties.MyAdapter queueName="QUEUE1"/>
</jmsQueue>
```

```
<jmsTopic id="topic1" jndiName="jms/topic1">
 <properties.MyAdapter topicName="TOPIC1"/>
</jmsTopic>
```

2. （可选）如果需要，请标识可用的目标、队列和主题属性子元素名称。

   注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器。请参阅配置资源适配器（见第 1152 页）主题以获取更多详细信息。

   - 如果资源适配器仅提供一种类型的具有 javax.jms.Destination 接口的受管对象，那么子元素名称为：properties.<rar_identifier>
   - 如果实现名称是唯一的，且不具有软件包名称，那么子元素名称为：properties.<rar_identifier>.<ImplementationName>
   - 在其他情况下，子元素名称为：properties.<rar_identifier>.<fully.qualified.InterfaceName>
   - 如果资源适配器仅提供一种类型的具有 javax.jms.Queue 接口的受管对象，那么子元素名称为：properties.<rar_identifier>
   - 如果实现名称是唯一的，且不具有软件包名称，那么子元素名称为：properties.<rar_identifier>.<ImplementationName>
   - 在其他情况下，子元素名称为：properties.<rar_identifier>.<fully.qualified.InterfaceName>
   - 如果资源适配器仅提供一种类型的具有 javax.jms.Topic 接口的受管对象，那么子元素名称为：properties.<rar_identifier>
   - 如果实现名称是唯一的，且不具有软件包名称，那么子元素名称为：properties.<rar_identifier>.<ImplementationName>
   - 在其他情况下，子元素名称为：properties.<rar_identifier>.<fully.qualified.InterfaceName>

使用以下示例以了解如何配置具有两种 JMS 目标的资源适配器，这两种 JMS 目标具有唯一实现类名。

在 ra.xml 文件中的以下片段中，MyAdapter 资源适配器提供了具有唯一实现类名的两种 JMS 目标：

```
<adminobject>
<adminobject-interface>javax.jms.Destination</adminobject-interface>
<adminobject-class>com.vendor.adapter.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>queueName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
 ...
</adminobject>

<adminobject>
<adminobject-interface>javax.jms.Destination</adminobject-interface>
<adminobject-class>com.vendor.adapter.TopicImpl</adminobject-class>
<config-property>
 <config-property-name>topicName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>
```

以下是此场景的服务器配置的示例：

```
<jmsDestination jndiName="jms/destination1">
 <properties.MyAdapter.QueueImpl queueName="D1"/>
</adminObject>

<jmsDestination jndiName="jms/destination2">
 <properties.MyAdapter.TopicImpl topicName="D2"/>
</jmsDestination>
```

使用以下示例以了解如何配置具有两种受管对象的资源适配器，这两种受管对象不具有唯一实现类名。

在 ra.xml 文件中的以下片段中，MyAdapter 资源适配器提供了具有不唯一实现类名的两种受管对象：

```
<adminobject>
<adminobject-interface>javax.jms.Queue</adminobject-interface>
<adminobject-class>com.vendor.adapter.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>queueName</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
...
</adminobject>

<adminobject>
<adminobject-interface>javax.jms.Queue</adminobject-interface>
<adminobject-class>com.vendor.adapter.advanced.QueueImpl</adminobject-class>
<config-property>
 <config-property-name>name</config-property-name>
 <config-property-type>java.lang.String</config-property-type>
</config-property>
 ...
</adminobject>
```

以下是此场景的服务器配置的示例：

```
<jmsQueue jndiName="jms/myQueue">
 <properties.MyAdapter.com.vendor.adapter.QueueImpl queueName="Q1"/>
</jmsQueue>

<jmsQueue jndiName="jms/myAdvancedQueue">
 <properties.MyAdapter.com.vendor.adapter.advanced.QueueImpl name="Q1"/>
</jmsQueue>
```

在某些情况下，可能不需要太长的配置元素名称。请参阅有关定制 JCA 配置元素的信息，以了解如何覆盖配置元素名称的后缀。

配置 JMS 激活规范

您可配置资源适配器提供的符合 Java EE 连接器体系结构 (JCA) 规范的 JMS 激活规范。

您可以配置由已安装的资源适配器提供的 JMS 消息侦听器的一个或多个 JMS 激活规范实例。

👉 注:

要添加 xigemaAS 的 JCA 支持，必须使用 某个文本编辑器来编辑 server.xml 文件。

1. 配置一个或多个 JMS 激活规范实例。

当您配置激活规范实例时，必须提供 properties 子元素（即使您不想覆盖任何配置属性也是如此），以将 jmsActivationSpec 元素与特定资源适配器所提供的 JMS 消息侦听器相关联。properties 子元素始终遵循 JMS 激活规范的模式属性 <rar_identifier>。以下示例将配置 JMS 激活规范的两个实例：

```
<jmsActivationSpec id="app1/module1/MyJMSMessageDrivenBean">
 <properties.MyAdapter destinationRef="topic1"/>
</jmsActivationSpec>

<jmsActivationSpec id="app1/module1/AnotherJMSMessageDrivenBean">
 <containerAuthData user="user1" password="{xor}Lz4sLCgwLTtu"/>
 <properties.MyAdapter destinationRef="queue1"/>
</jmsActivationSpec>
```

👉 注：本主题假定在服务器中已配置具有唯一标识 MyAdapter 的资源适配器，并启用了 jms-1.1 和 mdb-3.1 功能部件。请参阅配置资源适配器（见第 1152 页）主题以获取更多详细信息。

定制 JCA 配置元素

您可以定制如何在安装资源适配器时生成 JCA 属性子元素。

当您安装独立资源适配器或者嵌入在应用程序中的资源适配器时，可以在 <resourceAdapter> 元素下添加一个或多个 <customize> 子元素，以选择用于所指定接口或实现类的子元素属性的后缀。定制子元素使您能够避免使用冗长的属性子元素名称，这样的名称可能其他时候需要用于必须具有唯一名称的配置元素。

👉 注：

要添加 xigemaAS 的 JCA 支持，必须使用 某个文本编辑器来编辑 server.xml 文件。

1. 对于独立资源适配器，从您要定制现有配置开始。

   例如，如果资源适配器 MyAdapter 提供了两个连接工厂，而简单接口类名和实现类名都不唯一：

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects.  -->
 ...
</featureManager>
<resourceAdapter location="C:/adapters/MyAdapter.rar"/>

<connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.javax.resource.cci.ConnectionFactory
 serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.com.vendor.adapter.custom.ConnectionFactory
 hostName="localhost"/>
</connectionFactory>
```

2. 将 customize 子元素添加至 resourceAdapter，以选择这两个连接工厂接口的后缀。

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects.  -->
 ...
</featureManager>
```

```
<resourceAdapter location="C:/adapters/MyAdapter.rar">
 <customize interface="javax.resource.cci.ConnectionFactory" suffix="cci"/
>
 <customize interface="com.vendor.adapter.custom.ConnectionFactory"
 suffix="custom"/>
</resourceAdapter>

<connectionFactory jndiName="eis/cci-cf">
 <properties.MyAdapter.cci serverName="localhost"/>
</connectionFactory>

<connectionFactory jndiName="eis/custom-cf">
 <properties.MyAdapter.custom hostName="localhost"/>
</connectionFactory>
```

3. 对于嵌入在应用程序中的资源适配器，从您要定制现有配置开始。

   例如，假设您有一个应用程序 app1，其中包含名为 MyAdapter 的嵌入式资源适配器，如下所示：

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects.  -->
 ...
</featureManager>

<application name="app1" type="ear" location="C:/applications/app1.ear"/>

<adminObject jndiName="eis/interactionSpec-find">
 <properties.app1.MyAdapter.javax.resource.cci.InteractionSpec-
com.vendor.adapter.finder.InteractionSpecImpl resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/interactionSpec-update">
 <properties.app1.MyAdapter.com.vendor.adapter.InteractionSpec-
com.vendor.adapter.updater.InteractionSpecImpl executionTimeout="3000"/>
</adminObject>
```

4. 为应用程序中的资源适配器归档 (RAR) 模块指定 resourceAdapter 元素。将 id 属性指定为 RAR 模块的模块名称。添加 customize 子元素以选择基于接口或实现类的受管对象的后缀。

   在此示例中，仅指定了实现类，足以标识受管对象：

```
<featureManager>
 <feature>jca-1.6</feature>
 <feature>jndi-1.0</feature>  <!-- Add the jndi feature to enable look up
 of connection factories and administered objects.  -->
 ...
</featureManager>

<application name="app1" type="ear" location="C:/applications/app1.ear">
 <resourceAdapter id="MyAdapter">
  <customize
 implementation="com.vendor.adapter.finder.InteractionSpecImpl"
 suffix="finder"/>
  <customize
 implementation="com.vendor.adapter.updater.InteractionSpecImpl"
 suffix="updater"/>
 </resourceAdapter>
 </application>
<adminObject jndiName="eis/interactionSpec-find">
```

```
 <properties.app1.MyAdapter.finder resultSetType="1003"/>
</adminObject>

<adminObject jndiName="eis/interactionSpec-update">
 <properties.app1.MyAdapter.updater executionTimeout="3000"/>
</adminObject>
```

从 Java™ EE 应用程序访问独立资源适配器

可以从 Java™ EE 应用程序访问独立资源适配器。

多个 Java EE 应用程序可以共享独立资源适配器类和资源。缺省情况下，Java EE 应用程序可以访问 JCA 规范 API，但是不能访问独立资源适配器的供应商类和资源。启用此访问的先决条件是，在服务器配置中必须配置了资源适配器和应用程序。

在以下示例中，一个名为 Scholar 的应用程序和一个名为 Student 的应用程序正在名为 Academy 的服务器上运行。这两个应用程序都需要访问名为 Socrates16 的资源适配器，在位于 C:/adapters/version-1.6 目录中的 socrates.rar 文件中提供了该资源适配器。

1. 配置独立资源适配器。

   在 server.xml 文件中，通过添加以下代码来配置独立资源适配器：

   ```
    <resourceAdapter id="Socrates16" location="C:/adapters/version-1.6/
   socrates.rar" />
   ```

2. 从应用程序引用该资源适配器，以便这两个应用程序都可以访问资源适配器模块中提供的类和资源。

   在 server.xml 文件中，通过添加以下代码将 classProviderRef 属性设置为应用程序的类装入配置中的资源适配器的标识：

   ```
   <application id="scholar" name="Scholar" type="ear"
    location="scholar.ear">
    <classloader classProviderRef="Socrates16" />
   </application>

   <application id="student" name="Student" type="ear"
    location="student.ear">
    <classloader classProviderRef="Socrates16" />
   </application>
   ```

3. 可选：配置独立资源适配器的类装入以访问第三方 API。

   缺省情况下，资源适配器和 Java 应用程序都无法访问第三方 API。每当应用程序的类装入配置需要访问第三方 API，并且该应用程序需要访问独立资源适配器，请将该资源适配器的类装入配置为还要访问第三方 API。

   在 server.xml 文件中，通过添加以下代码来配置资源适配器的类装入配置的 apiTypeVisibility 属性，以访问第三方 API：

   ```
   <resourceAdapter id="Socrates16" location="C:/adapters/version-1.6/
   socrates.rar">
    <classloader apiTypeVisibility="spec, ibm-api, api, third-party" />
   <resourceAdapter/>

   <application id="scholar" name="Scholar" type="ear"
    location="scholar.ear">
   ```

```
 <classloader classProviderRef="Socrates16" apiTypeVisibility="spec, ibm-
api, api, third-party" />
 </application>
<application id="student" name="Student" type="ear"location="student.ear">
 <classloader classProviderRef="Socrates16" apiTypeVisibility="spec, ibm-
api, api, third-party" />
</application>
```

配置 ManagedExecutorService 实例

可配置 ManagedExecutorService 实例以使用所指定线程上下文来运行异步任务。Java™ EE 应用程序最好避免直接管理它们自己的线程；因此，ManagedExecutorService 扩展 JSE ExecutorService 来提供一种方法以在应用程序服务器环境内启动异步任务。还可配置 ManagedExecutorService 以将与 Java™ EE 应用程序相关的各种线程上下文传播至异步任务的线程。

ManagedExecutorService 是在 `<concurrent-1.0>` 功能部件下提供的，并且在 `server.xml` 文件中启用，如下所示：

```
<featureManager>
 <feature>concurrent-1.0</feature>
</featureManager>
```

ManagedExecutorService 所执行任务的线程的上下文传播由上下文服务管理。服务器创建上下文服务的缺省实例 (`DefaultContextService`)，并将其配置为至少传播 `classloaderContext`、`jeeMetadataContext` 和 `securityContext`。如果已创建 ManagedExecutorService 但未引用特定上下文服务实例或直接在其中配置上下文服务实例，那么会使用此缺省上下文服务实例。有关更多信息，请参阅“配置线程上下文服务实例”主题。

缺省受管执行程序实例 (`DefaultManagedExecutorService`) 以 `java:comp/DefaultManagedExecutorService` 形式提供，并使用缺省上下文服务实例来捕获和传播线程上下文。

- `server.xml` 文件中的示例配置：
- 在 JNDI 中以名称 `concurrent/execSvc` 注册，且使用缺省上下文服务实例的受管执行程序服务实例：

  ```
  <managedExecutorService jndiName="concurrent/execSvc"/>
  ```

- 上下文服务配置为仅传播 `jeeMetadataContext` 的受管执行程序服务实例：

  ```
  <managedExecutorService jndiName="concurrent/execSvc1">
  <contextService>
          <jeeMetadataContext/>
  </contextService>
  </managedExecutorService>
  ```

- 具有 `classloaderContext` 和 `securityContext` 的受管执行程序服务实例：

  ```
  <managedExecutorService jndiName="concurrent/execSvc2">
  <contextService>
   <classloaderContext/>
   <securityContext/>
  </contextService>
  </managedExecutorService>
  ```

- 由多个受管执行程序服务实例共享的线程上下文服务：

  ```
  <contextService id="contextSvc1">
   <jeeMetadataContext/>
  </contextService>
  ```

```
<managedExecutorService jndiName="concurrent/execSvc3" contextServiceRef="contextSvc1"/>

<managedExecutorService jndiName="concurrent/execSvc4" contextServiceRef="contextSvc1"/>
```

- 从前一示例继承并且由受管执行程序服务实例使用的线程上下文服务：

```
<contextService id="contextSvc2" baseContextRef="contextSvc1">
 <classloaderContext/>
</contextService>

<managedExecutorService jndiName="concurrent/execSvc5" contextServiceRef="contextSvc2"/>
```

受管执行程序服务实例可注入到应用程序组件中（通过使用 `@Resource`），或使用资源环境引用 (`resource-env-ref`) 进行查找。不管如何获取实例，都可按可互换方式将其用作 `javax.enterprise.concurrent.ManagedExecutorService` 或其 `java.util.concurrent.ExecutorSerivce` 超类。

- 查找缺省受管执行程序的示例：

```
ManagedExecutorService executor =
     (ManagedExecutorService) new InitialContext().lookup(
         "java:comp/DefaultManagedExecutorService");
executor.submit(doSomethingInParallel);
```

- 使用 `@Resource` 以注入为 `java.util.concurrent.ExecutorService` 的示例：

```
@Resource(lookup="concurrent/execSvc1")
ExecutorService execSvc1;

...

// submit task to run
Future<Integer> future1 = execSvc1.submit(new Callable<Integer>() {
  public Integer call() throws Exception {
      // java:comp lookup is possible because <jeeMetadataContext> is configured
    DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
    ... make updates to the database
    return updateCount;
 }
});
Future<Integer> future2 = execSvc1.submit(anotherTaskThatUpdatesADatabase);

numUpdatesCompleted = future1.get() + future2.get();
```

- 使用 `@Resource` 以注入为 `javax.enterprise.concurrent.ManagedExecutorService` 的示例：

```
@Resource(lookup="concurrent/execSvc1")
ManagedExecutorService execSvc1;

...

// submit task to run
Future<Integer> future1 = execSvc1.submit(new Callable<Integer>() {
  public Integer call() throws Exception {
      // java:comp lookup is possible because <jeeMetadataContext> is configured
    DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
    ... make updates to the database
    return updateCount;
 }
});
Future<Integer> future2 = execSvc1.submit(anotherTaskThatUpdatesADatabase);

numUpdatesCompleted = future1.get() + future2.get();
```

- web.xml 文件中的 java.util.concurrent.ExecutorService 的 <resource-env-ref> 示例：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/execSvc2</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- web.xml 文件中的 javax.enterprise.concurrent.ManagedExecutorService 的 <resource-env-ref> 示例：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/execSvc2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- 使用资源环境引用的示例查找：

```
ExecutorService execSvc2 =
    (ExecutorService) new InitialContext().lookup("java:comp/env/concurrent/execSvc2");

futures = execSvc2.invokeAll(Arrays.asList(task1, task2, task3));
```

- 使用资源环境引用并强制转型为 ManagedExecutorService 的查找示例：

```
ManagedExecutorService execSvc2 =
    (ManagedExecutorService) new InitialContext().lookup("java:comp/env/concurrent/execSvc2");

futures = execSvc2.invokeAll(Arrays.asList(task1, task2, task3));
```

配置线程上下文服务实例

可配置 ContextService 实例以捕获受管线程上下文并应用它以对任何线程调用指定接口方法。

Java™ EE 应用程序最好避免直接管理它们自己的线程；因此，ContextService 提供了一种方法以对非受管线程和受管线程建立先前捕获的线程上下文，以覆盖已就位的任何线程上下文。

缺省线程上下文服务实例 (DefaultContextService) 由服务器创建，配置为至少捕获并传播 classloaderContext、jeeMetadataContext 和 securityContext。可配置线程上下文传播以包含以下类型的线程上下文：

classloaderContext

使任务的提交者的线程上下文类装入器可用于该任务。如果上下文类装入器已序列化，那么类装入器必须是来自应用程序的线程上下文类装入器。当前不支持针对 Web 应用程序捆绑软件的类装入器序列化。

jeeMetadataContext

使提交任务的应用程序组件的名称空间对该任务可用。

securityContext

必须在 server.xml 文件中启用 appSecurity-2.0 功能部件以使用此类型的线程上下文。使任务提交者的调用者主体集和调用主体集对该任务可用，并且此操作通过使用提交者的 WSPrincipal（使用 JAAS 登录）登录来完成。有关提交者主体集中的哪些信息不在安全上下文中的详细信息，请参阅*concurrent-1.0 功能部件限制*。

👉 重要：附加线程上下文提供者可以由堆栈产品中的功能部件提供。可选 baseContextRef 属性允许上下文服务实例从另一上下文服务实例的上下文配置继承。

- 在 server.xml 文件中启用线程上下文服务。此线程上下文服务在 <concurrent-1.0> 功能部件下提供。

```
<featureManager>
  <feature>concurrent-1.0</feature>
</featureManager>
```

在 server.xml 文件中配置线程上下文服务实例：

- 在 JNDI 中以名称 concurrent/threadContextSvc1 注册且仅捕获并传播 jeeMetadataContext 的线程上下文服务：

```
<contextService id="threadContextSvc1" jndiName="concurrent/${id}">
 <jeeMetadataContext/>
</contextService>
```

- 带有 classloaderContext 和 securityContext 的线程上下文服务：

```
<contextService jndiName="concurrent/threadContextSvc2">
 <classloaderContext/>
 <securityContext/>
</securityContext/>
```

- 从 threadContextSvc1 继承 jeeMetadataContext 并添加 securityContext 的线程上下文服务：

```
<contextService jndiName="concurrent/threadContextSvc3"
baseContextRef="threadContextSvc1">
 <securityContext>
</contextService>
```

查找缺省上下文服务的示例：

```
ContextService threadContextSvc =
    (ContextService) new InitialContext().lookup(
        "java:comp/DefaultContextService");
myContextualAsyncCallback = threadContextSvc.createContextualProxy(
    myAsyncCallback, MyAsyncCallback.class);
doSomethingAsync(arg1, arg2, myContextualAsyncCallback);
```

将线程上下文服务实例注入到应用程序组件中（通过使用 @Resource）或使用资源环境引用 (resource-env-ref) 进行查找的示例。

- 使用 @Resource 的示例：

```
@Resource(lookup="concurrent/threadContextSvc1")
ContextService threadContextSvc1;

...

Callable<Integer>  processSalesOrderCompletion = new Callable<Integer>() {
  public Integer call() throws Exception {
      DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
   ...update various database tables
   return isSuccessful;
 }
};
// capture thread context of current application component
execProps = Collections.singletonMap(ManagedTask.TRANSACTION,
ManagedTask.USE_TRANSACTION_OF_EXECUTION_THREAD);
processSalesOrderCompletion = (Callable<Boolean>)
 threadContextSvc1.createContextualProxy(processSaleCompletion, execProps,
Callable.class);
```

```
//later from a different application component
tran.begin();
...
successful = processSalesOrderCompletion.call();
if (successful)
  tran.commit();
else
  tran.rollback();
```

- 在 web.xml 文件中指定 resource-env-ref 的示例：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/threadContextSvc3</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ContextService</resource-
 env-ref-type>
</resource-env-ref>
```

- 使用资源环境引用的示例查找：

```
ContextService threadContextSvc3 =
(ContextService) new InitialContext().lookup("java:comp/env/concurrent/threadContextSvc3");
Runnable updateAndGetNextFromDatabase = threadContextSvc3.createContextualProxy
(new Runnable() {
  public void run() {
     DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
    ... update the database and get next item to process
  }
}, Runnable.class);
barrier = new CyclicBarrier(3, updateAndGetNextFromDatabase);
...
```

配置受管调度执行程序

可配置 ManagedScheduledExecutorService 实例来安排异步任务以与从中安排该任务的线程的线程上下文一起运行。Java™ EE 应用程序最好避免直接管理它们自己的线程；因此，ManagedScheduledExecutorService 扩展 JSE ExecutorService 来提供一种方法以在应用程序服务器环境内安排异步任务。您还可配置 ManagedScheduledExecutorService 以捕获与 Java™ EE 应用程序相关的线程上下文并将其传播至计划任务的线程。

重要：在 xigemaAS 中，受管计划执行程序没有它们自己的线程池。提交至受管计划执行程序实例的任务在公共 xigemaAS 执行程序线程池上运行。

受管计划执行程序 <concurrent-1.0> 功能部件在 server.xml 文件中按如下方式启用：

```
<featureManager>
  <feature>concurrent-1.0</feature>
</featureManager>
```

线程上下文捕获和传播由上下文服务管理。由服务器创建的上下文服务的缺省实例 (DefaultContextService)，配置为至少传播 classloaderContext、jeeMetadataContext 和 securityContext。如果已创建 ManagedScheduledExecutorService 但未引用特定上下文服务实例或直接在其中配置上下文服务实例，那么会使用此缺省上下文服务实例。有关上下文服务实例的更多信息，请参阅配置线程上下文服务实例（见第 1172 页）主题。

缺省受管调度执行程序实例 (DefaultManagedScheduledExecutorService) 以 java:comp/DefaultManagedScheduledExecutorService 形式提供，并使用缺省上下文服务实例来捕获和传播线程上下文。

- server.xml 文件中的示例配置：

- 在 JNDI 中以名称 concurrent/scheduledExecutor 注册且使用缺省上下文服务实例的受管调度执行程序：

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor"/>
```

- 带有配置为仅捕获并传播 classloaderContext 的上下文服务的受管调度执行程序：

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor1">
   <contextService>
      <classloaderContext/>
   </contextService>
</managedScheduledExecutorService>
```

- 带有 jeeMetadataContext 和 securityContext 的受管调度执行程序：

```
<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor2">
   <contextService>
      <jeeMetadataContext/>
      <securityContext/>
   </contextService>
</managedScheduledExecutorService>
```

- 由多个受管调度执行程序共享的线程上下文服务：

```
<contextService id="contextSvc1">
   <jeeMetadataContext/>
</contextService>

<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor3"
contextServiceRef="contextSvc1"/>

<managedScheduledExecutorService jndiName="concurrent/scheduledExecutor4"
 contextServiceRef="contextSvc1"/>
```

将受管调度执行程序注入至应用程序组件（通过使用 @Resource）或使用资源环境引用 (resource-env-ref) 进行查找。不管如何获取该实例，都可按可互换方式将其用作 javax.enterprise.concurrent.ManagedScheduledExecutorService 或以下任何超类：java.util.concurrent.ScheduledExecutorSerivce、java.util.concurrent.ExecutorService 和 javax.enterprise.concurrent.ManagedExecutorService

- 查找缺省受管调度执行程序的示例：

```
ManagedScheduledExecutorService executor =
  (ManagedScheduledExecutorService) new InitialContext().lookup(
         "java:comp/DefaultManagedScheduledExecutorService");
executor.schedule(beginSalePrices, 12, TimeUnit.HOURS);
executor.schedule(restoreNormalPrices, 60, TimeUnit.HOURS);
```

- 使用 @Resource 以注入为 java.util.concurrent.ScheduledExecutorService 的示例：

```
@Resource(lookup="concurrent/scheduledExecutor2")
ScheduledExecutorService executor;
...

// schedule a task to run every half hour from now
Runnable updateSalesReport = new Runnable() {
public void run() throws Exception {
   // java:comp lookup is possible because <jeeMetadataContext> is configured
   DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
... query and update various database tables
}
};
ScheduledFuture<?> future = executor.scheduleAtFixedRate(updateSalesReport, 0, 30, TimeUnit.MINUTES);
```

- 使用 @Resource 以注入为 javax.enterprise.concurrent.ManagedScheduledExecutorService 的示例：

```
@Resource(lookup="concurrent/scheduledExecutor2")
ManagedScheduledExecutorService executor;

... usage is same as previous example
```

- web.xml 文件中的 java.util.concurrent.ScheduledExecutorService 的示例 <resource-env-ref>：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/scheduledExecutor1</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ScheduledExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- web.xml 文件中的 javax.enterprise.concurrent.ManagedScheduledExecutorService 的示例 <resource-env-ref>：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/scheduledExecutor2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedScheduledExecutorService</resource-env-ref-type>
</resource-env-ref>
```

- 使用资源环境引用的示例查找：

```
ManagedScheduledExecutorService executor =
   (ManagedScheduledExecutorService) new InitialContext().lookup("java:comp/env/concurrent/scheduledExecutor2");
executor.schedule(payrollTask, fridaysAtMidnightTrigger);
```

配置受管线程工厂

可配置 ManagedThreadFactory 实例以创建新线程，这些线程与从中查找或注入受管线程工厂的线程的线程上下文一起运行。Java™ EE 应用程序最好避免直接管理它们自己的线程；因此，ManagedThreadFactory 扩展 JSE ThreadFactory 来提供一种方法以在应用程序服务器环境内创建受管线程。您还可配置 ManagedThreadFactory 以捕获与 Java™ EE 应用程序相关的线程上下文并将其传播至新线程。

受管线程工厂在 <concurrent-1.0> 功能部件下，并且在 server.xml 文件中按如下方式启用：

```
<featureManager>
  <feature>concurrent-1.0</feature>
</featureManager>
```

线程上下文捕获和传播由上下文服务管理。由服务器创建的上下文服务的缺省实例 (DefaultContextService)，配置为至少传播 classloaderContext、jeeMetadataContext 和 securityContext。如果 ManagedThreadFactory 未指定上下文服务，那么会使用此缺省上下文服务实例。有关上下文服务实例的更多信息，配置线程上下文服务实例（见第 1172 页）主题。

ManagedThreadFactory 的缺省实例 (DefaultManagedThreadFactory) 以 java:comp/DefaultManagedThreadFactory 形式提供，并使用缺省上下文服务实例来捕获和传播线程上下文。

- server.xml 文件中的示例配置：
- 在 JNDI 中以名称 concurrent/threadFactory 注册且使用缺省上下文服务实例的受管线程工厂：

```
<managedThreadFactory jndiName="concurrent/threadFactory" maxPriority="5"/>
```

- 带有配置为仅捕获并传播 securityContext 的上下文服务的受管线程工厂：

```
<managedThreadFactory jndiName="concurrent/threadFactory1">
  <contextService>
  <securityContext/>
 </contextService>
</managedThreadFactory>
```

- 带有 classloaderContext 和 jeeMetadataContext 的受管线程工厂：

```
<managedThreadFactory jndiName="concurrent/threadFactory2">
  <contextService>
  <classloaderContext/>
  <jeeMetadataContext/>
 </contextService>
</managedThreadFactory>
```

- 由多个受管线程工厂共享的线程上下文服务：

```
<contextService id="contextSvc1">
 <jeeMetadataContext/>
</contextService>


<managedThreadFactory jndiName="concurrent/threadFactory3"
contextServiceRef="contextSvc1"/>


<managedThreadFactory jndiName="concurrent/threadFactory4"
contextServiceRef="contextSvc1"/>
```

受管线程工厂可注入到应用程序组件中（通过使用 @Resource）或使用资源环境引用 (resource-env-ref) 进行查找。不管如何获取该实例，都可按可互换方式将其用作 javax.enterprise.concurrent.ManagedThreadFactory 或 java.util.concurrent.ThreadFactory。

- 查找缺省受管线程工厂的示例：

```
ManagedThreadFactory threadFactory =
     (ManagedThreadFactory) new InitialContext().lookup(
         "java:comp/DefaultManagedThreadFactory");
// Create an executor that always runs tasks with the thread context of the
 managed thread factory
ExecutorService executor = new ThreadPoolExecutor(
    coreThreads, maxThreads, keepAliveTime, TimeUnit.MINUTES,
    new ArrayBlockingQueue<Runnable>(workRequestQueueSize),
    threadFactory, new ThreadPoolExecutor.AbortPolicy());
```

- 使用 @Resource 以注入为 java.util.concurrent.ThreadFactory 的示例：

```
@Resource(lookup="concurrent/threadFactory2")
ThreadFactory threadFactory
...

// create a new thread
Thread dailySalesAnalysisTask = threadFactory.newThread(new Runnable() {
  public void run() {
        // java:comp lookup is possible because <jeeMetadataContext> is configured
     DataSource ds = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/ds1");
    ... analyze the data
  }
});
dailySalesAnalysisTask.start();
```

- 使用 @Resource 以注入为 javax.enterprise.concurrent.ManagedThreadFactory 的示例：

```
@Resource(lookup="concurrent/threadFactory2")
ManagedThreadFactory threadFactory;

... usage is same as previous example
```

- web.xml 文件中的 java.util.concurrent.ThreadFactory 的示例 <resource-env-ref>：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/threadFactory1</resource-env-ref-name>
 <resource-env-ref-type>java.util.concurrent.ThreadFactory</resource-env-ref-type>
</resource-env-ref>
```

- web.xml 文件中的 javax.enterprise.concurrent.ManagedThreadFactory 的示例 <resource-env-ref>：

```
<resource-env-ref>
 <resource-env-ref-name>concurrent/threadFactory2</resource-env-ref-name>
 <resource-env-ref-type>javax.enterprise.concurrent.ManagedThreadFactory</resource-
env-ref-type>
</resource-env-ref>
```

- 使用资源环境引用的示例查找：

```
ManagedThreadFactory threadFactory =
  (ManagedThreadFactory) new InitialContext().lookup("java:comp/env/concurrent/threadFactory");
// Create a scheduled executor that always runs tasks with the thread context of the managed thread factory
ScheduledExecutorService executor = Executors.newScheduledThreadPool(5, threadFactory);
... use executor to schedule tasks from any thread
```

使用 JMX 来连接至 xigemaAS

使用此信息来访问 xigemaAS 上的 Java™ 管理扩展 (JMX) 连接器。还可以使用 SSL 以远程方式访问受保护的 JMX 连接器。

xigemaAS 上支持两种 JMX 连接器，每种连接器通过不同的xigemaAS 功能部件来启用：localConnector-1.0 和 restConnector-1.0。

- 通过 xigemaAS 功能部件 localConnector-1.0 来启用本地连接器。使用中的由 SDK 实现的策略会保护通过本地连接器进行的访问。当前，SDK 要求客户机和 xigemaAS 在同一个主机上以相同的用户标识运行。
- 通过 xigemaAS 功能部件 restConnector-1.0 来启用 REST 连接器。单个管理员角色会保护通过 REST 连接器进行的远程访问。此外，需要 SSL 才能保持通信机密。restConnector-1.0 功能部件已经包含 ssl-1.0 功能部件。

👉 注：xigemaAS 上部署的应用程序可以不受限制地访问其 MBeanServer 目录。

配置与 xigemaAS 的本地 JMX 连接

您可以访问 xigemaAS 上的本地 Java™ 管理扩展 (JMX) 连接器。通过 xigemaAS 功能部件 localConnector-1.0 来启用本地连接器。

通过 xigemaAS 功能部件 localConnector-1.0 来启用本地连接器。使用中的由 SDK 实现的策略会保护通过本地连接器进行的访问。当前，SDK 要求客户机和 xigemaAS 在同一个主机上以相同的用户标识运行。

👉 注：xigemaAS 上部署的应用程序可以不受限制地访问其 MBeanServer 目录。

下列部分描述如何配置和访问 xigemaAS 上的本地连接器。

1. 在 server.xml 文件中通过使用以下代码来启用本地连接器。

```
<featureManager>
     <feature>localConnector-1.0</feature>
</featureManager>
```

2. 使用安装在同一主机上的 JConsole 工具或 JMX 客户机来访问本地连接器。
   - 对于 JConsole 工具，请从连接面板中选择本地进程 `ws-server.jar defaultServer`，然后单击**连接**。
   - 对于 JMX 客户机，请参阅在 *xigemaAS* 上使用 *JMX MBean*（见第 1182 页）。

配置与 xigemaAS 的安全 JMX 连接

您可以在 xigemaAS 上使用 SSL 来访问受保护的 Java™ 管理扩展 (JMX) 连接器。由 xigemaAS 功能部件 `restConnector-1.0` 来启用安全 JMX 连接。

通过 xigemaAS 功能部件 `restConnector-1.0` 来启用 REST 连接器。单个管理员角色会保护通过 REST 连接器进行的远程访问。此外，需要 SSL 才能保持通信机密。`restConnector-1.0` 功能部件已经包含 `ssl-1.0` 功能部件。

👉 注：xigemaAS 上部署的应用程序可以不受限制地访问其 MBeanServer 目录。

下列部分描述如何在 xigemaAS 上配置和访问 REST 连接器。

1. 在 server.xml 文件中使用下列代码来启用 REST 连接器。

```
<featureManager>
     <feature>restConnector-2.0</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 *SSL* 证书。

   确保证书 subjectDN 的 CN 值是运行该服务器的机器的主机名，并且信任库的 jConsole 连接中包含该服务器的证书。
3. 在 server.xml 文件中将用户或组配置到管理员角色。
   - 映射 *xigemaAS* 的管理员角色（见第 1180 页）
4. 访问 REST 连接器。

   可从 Java 客户机或直接通过 HTTPS 调用来访问 xigemaAS REST 连接器。Java 客户机使用该连接器的客户端（在 wlp/clients/restConnector.jar 中）并实现 `javax.management.MBeanServerConnection` 接口。HTTPS 调用使用该连接器的服务器端。至于服务器端的 HTTPS 调用，可进行 HTTPS 调用的任何编程语言（例如，C++、JavaScript、curl、Ruby 和 Perl）都可使用 REST API。REST API 包含管理端点（JMX）及文件传输。
   - 从 *JMX* 客户机应用程序或者通过使用 Java SDK 中提供的 jConsole 工具来访问 REST 连接器。使用 `-J` 标志将系统属性作为 Java 选项来传递，以及将类路径设为包含连接器类文件。连接器类文件打包在 clients/restConnector.jar 文件中。
     - 使用 SSL 证书的下列属性：

```
-J-Djavax.net.ssl.trustStore=<location of your client trust store>
-J-Djavax.net.ssl.trustStorePassword=<password for the trust store>
-J-Djavax.net.ssl.trustStoreType=<type of trust store>
```

       以下示例显示了使用 SSL 配置的 jConsole 工具：

```
jconsole -J-Djava.class.path=%JAVA_HOME%/lib/jconsole.jar;
```

```
                            %JAVA_HOME%/lib/tools.jar;
                            %WLP_HOME%/clients/restConnector.jar
          -J-Djavax.net.ssl.trustStore=key.jks
          -J-Djavax.net.ssl.trustStorePassword=xigemaas
          -J-Djavax.net.ssl.trustStoreType=jks
```

在 jConsole 启动后，选择**远程进程**，然后输入 JMX 服务 URL：`service:jmx:rest://<host>:<port>/xigemaJMXConnectorREST`。端口号是 HTTPS 端口。您还必须提供用户名和密码。

- 使用 HTTPS 调用直接访问 REST 连接器。

  要使用 HTTPS 调用访问 REST 连接器，您需要 xigemaAS 9.0.0.1 或更高版本。

  1. 使用浏览器打开 https://*<host>*:*<port>*/xigemaJMXConnectorREST/api，然后输入您在步骤 3 中指定的管理凭证。
  2. 检查可用 REST API。每项具有其行为、输入、输出、查询参数和头的描述。

映射 xigemaAS 的管理员角色

可以将 `quickStartSecurity` 元素或任何受支持的用户注册表用于 xigemaAS 上的管理员角色映射。

通过 REST 连接器来访问的所有 JMX 方法和 MBean 目前都受名称为“管理员”的单一角色保护。要快速开始，请使用 `quickStartSecurity` 元素来配置具有管理员角色的单用户，并设置缺省 SSL 配置。

您也可以使用任何受支持的用户注册表。如果已配置另一个用户注册表，那么不能使用 `quickStartSecurity` 元素。在这种情况下，必须将用户或角色从注册表映射到管理员角色。

- 将 `quickStartSecurity` 元素用于单用户映射。

  下面举例说明了最低必需配置：

```
  <featureManager>
     <feature>restConnector-1.0</feature>
   </featureManager>
   <quickStartSecurity userName="bob"  userPassword="bobpassword" />
   <keyStore id="defaultKeyStore" password="keystorePassword"/>
```

- 或者，将基本注册表用于管理员角色映射。

  下面举例说明了基本注册表，其中给用户“bob”或组“group1”指定了管理员角色：

```
<basicRegistry>
     <user name="bob" password="bobpassword"/>
     <user name="joe" password="joepassword"/>
     <group name="group1" ...>
     </group>
 </basicRegistry>

 <administrator-role>
     <user>bob</user>
     <group>group1</group>
 </administrator-role>
```

- 或者，将*LDAP* 注册表用于管理员角色映射（您将需要在 server.xml 文件中添加 `ldapRegistry-3.0` 功能部件）。

  下面举例说明了 LDAP 注册表，其中给用户“bob”指定了管理员角色。

```
 <ldapRegistry id="basic" host="" port="">
     <tds.properties ... />
 </ldapRegistry>

 <administrator-role>
     <user>cn=bob,o=vsettan,c=cn</user>
```

```
 </administrator-role>
```

为 xigemaAS 开发 JMX Java™ 客户机

可以开发 Java™ 管理扩展 (JMX) 客户机应用程序来访问 xigemaAS 的受保护 REST 连接器。

使用 JMX 远程客户机应用程序，可以通过 JMX 编程来管理 xigemaAS。

- 如下所示开发样本 JMX 客户机。REST 连接器支持标准 JMX API。

```
import javax.management.remote.JMXServiceURL;
import javax.management.MBeanServerConnection;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import java.util.HashMap;


public class Test {

  public static void main(String[] args) {
    System.setProperty("javax.net.ssl.trustStore", <truststore location>);
    System.setProperty("javax.net.ssl.trustStorePassword", <truststore password>);

    //If the type of the trustStore is not jks, which is default,
    //set the type by using the following line.
    System.setProperty("javax.net.ssl.trustStoreType", <truststore type>);

        try {
      HashMap<String, Object> environment = new HashMap<String, Object>();
      environment.put("jmx.remote.protocol.provider.pkgs",
 "com.ibm.ws.jmx.connector.client");
      environment.put(JMXConnector.CREDENTIALS, new String[] { "bob", "bobpassword" });

      JMXServiceURL url = new JMXServiceURL("service:jmx:rest://
<host>:<port>/xigemaJMXConnectorREST");
      JMXConnector connector = JMXConnectorFactory.newJMXConnector(url, environment);
      connector.connect();
      MBeanServerConnection mbsc = connector.getMBeanServerConnection();
    } catch(Throwable t) {
        ...
    }
  }
}
```

- 可选： 对 SSL 证书禁用主机名验证。随 xigemaAS 安装的证书可能不含实际运行服务器的主机名。如果要禁用 SSL 证书的主机名验证，可以将系统属性 `com.ibm.ws.jmx.connector.client.disableURLHostnameVerification` 设为 true，这会对所有连接禁用主机名验证。要以每个连接为基础禁用主机名验证，请在创建 JMX 连接时将该属性作为新的 `environment` 传递：

```
HashMap<String, Object> environment = new HashMap<String, Object>();
environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
environment.put("com.ibm.ws.jmx.connector.client.disableURLHostnameVerification",
 Boolean.TRUE);
environment.put(JMXConnector.CREDENTIALS, new String[] { "bob", "bobpassword" });
...
```

- 可选： 使用环境映射来配置 JMX REST 连接器设置。

```
...
HashMap<String, Object> environment = new HashMap<String, Object>();
environment.put("com.ibm.ws.jmx.connector.client.rest.maxServerWaitTime", 0);
environment.put("com.ibm.ws.jmx.connector.client.rest.notificationDeliveryInterval", 65000);
...
```

- 可选： xigemaAS REST 连接器允许您指定定制 SSL 套接字工厂，可以用来获取套接字。如果显示了 `javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.j: PKIX path building failed: java.security.cert.CertPathBuilderException: unable to find`

`valid certification path to requested target` 异常，那么可以根据您自己的密钥库来创建您自己的 SSLContext，然后从该上下文将 SocketFactory 与 REST 连接器配合使用。

```
KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
InputStream inputStream = new FileInputStream("myTrustStore.jks");
trustStore.load(inputStream, "password".toCharArray());
TrustManagerFactory trustManagerFactory =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
trustManagerFactory.init(trustStore);
TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();
SSLContext sslContext = SSLContext.getInstance("SSL");
sslContext.init(null, trustManagers, null);

Map<String, Object> environment = new HashMap<String, Object>();
environment.put(ConnectorSettings.CUSTOM_SSLSOCKETFACTORY, sslContext.getSocketFactory());
environment.put(ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION, true);
environment.put("jmx.remote.protocol.provider.pkgs", "com.ibm.ws.jmx.connector.client");
environment.put(JMXConnector.CREDENTIALS, new String[] { "admin", "password" });
JMXServiceURL url = new JMXServiceURL("REST", "myhost", 9443, "/xigemaJMXConnectorREST");
jmxConn = JMXConnectorFactory.connect(url, environment);
```

在 xigemaAS 上使用 JMX MBean

可以在 xigemaAS 上访问 Java™ 管理扩展 (JMX) 管理 Bean (MBean) 的属性以及调用其操作。此外，您可以从正在 xigemaAS 上运行的应用程序注册您自己的 MBean。

xigemaAS 上与 MBean 交互的主要接口为如下所示：

- `javax.management.MBeanServer`，适用于 xigemaAS 上运行的应用程序代码。
- `javax.management.MBeanServerConnection`，适用于在单独 Java™ 虚拟机中运行的外部代码。

可以使用其中的任一接口的实例来访问 MBean 的属性和调用 MBean 的操作。

- 对于 xigemaAS 上运行的应用程序代码，可以通过下列代码来使用 `javax.management.MBeanServer` 实例：

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
...
```

- 对于单独 Java™ 虚拟机中运行的外部代码，可以使用 `javax.management.MBeanServerConnection` 实例。请参阅为 *xigemaAS* 开发 *JMX Java* 客户机（见第 1181 页）。

所提供 MBean 的列表

xigemaAS 提供了您可以用来处理和监视服务器的 MBean 以及相应管理接口的列表。

对于该列表中的每个 MBean 或 MXBean：

- 名称是用来唯一标识 MBean 或 MXBean 的 `javax.management.ObjectName` 值。如果存在 MBean 或 MXBean 的多个实例，那么 `ObjectName` 值可以包含通配符 (*)，在本主题的“注释”条目中对此进行了描述。
- 管理接口条目指定可以用来构造 MBean 或 MXBean 的代理对象（如访问 *MBean* 属性和操作的示例（见第 1186 页）中所述）的 Java™ 接口的名称。有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

WebSphere:feature=channelfw,type=endpoint,name=*

- 管理接口：com.ibm.websphere.endpoint.EndPointInfoMBean
- 注释：为系统中的每个端点都提供了一个实例，其中 * 是唯一的端点名称。

WebSphere:feature=restConnector,type=FileService,name=FileService

- 管理接口：com.ibm.websphere.filetransfer.FileServiceMXBean
- 注释：此 MXBean 可让您在 xigemaAS 所在的主机上执行各种与文件相关的操作。

  您可以在下列位置查找此 MXBean 的类和 API 文档：

  ```
  xigemaas_home/dev/api/vsettan/com.vsettan.xigema.appserver.api.restConnector_version.jar
  xigemaas_home/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.restConnector_version-javadoc.zip
  ```

  公开的操作包括能够查询所给定文件或目录的特定元数据（最近一次进行修改的日期和大小等等），还能够查询所给定目录的所有子文件（和相应的元数据）。还支持创建和扩展归档，这对于压缩 xigemaAS 日志文件或者在部署应用程序之前将应用程序解压很有用。

  此 MXBean 包含两个属性：读取列表和写入列表。它们表示在使用 xigemaAS 所提供的 FileService 或 FileTransfer 功能时用户可以读/写的位置的列表。通过 MXBean，只能读取这些属性，但是可以通过 server.xml 文件中的下列元素来配置或定制这些属性：

  ```
  <remoteFileAccess>
            <readDir>${server.output.dir}/logs</readDir>
     <readDir>${server.output.dir}/apps</readDir>
     <writeDir>${server.output.dir}/dropins</writeDir>
  </remoteFileAccess>
  ```

  如果未指定 readDir 元素，那么缺省值为下列各项的组合：${wlp.install.dir}、${wlp.user.dir} 和 ${server.output.dir}。如果未指定 writeDir 元素，那么缺省值为空集合。

  server.xml 文件中必须包括 restConnector-1.0 功能部件，以便载入此 MXBean 以及支持其配置元素

  允许将 xigemaAS 定义的变量与所有采用用于表示文件路径的字符串的服务器端参数配合使用。在 *xigemaas_home*/README.TXT 文件中定义了这类变量。

WebSphere:feature=restConnector,type=FileTransfer,name=FileTransfer

- 管理接口：com.ibm.websphere.filetransfer.FileTransferMBean
- 注释：此 MBean 允许您在 xigemaAS 所在的主机上执行各种文件传输操作。

  您可以在下列位置查找此 MXBean 的类和 API 文档：

  ```
  xigemaas_home/dev/api/vsettan/com.vsettan.xigema.appserver.api.restConnector_version.jar
  xigemaas_home/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.restConnector_version-javadoc.zip
  ```

  此 MBean 在正运行其相应 xigemaAS 进程的同一 JVM 中的 PlatformMBeanServer 上已注册，但是只能使用 xigemaAS JMX REST 连接器来访问此 MBean。连接可以是本地连接或远程连接，但是必须使用 REST 连接器。

已公开的操作包括能够下载、上载和删除文件。服务器上的每个读取和写入请求都绑定至可配置的读取和写入列表，可通过 `FileServiceMXBean` 来访问这些列表。如果通过 xigemaAS JMX REST 连接器连接了 JConsole，那么还可以从内置 Java™ JConsole 来完全访问和操作 `FileTransferMBean`。

允许将 xigemaAS 定义的变量与所有采用用于表示文件路径的字符串的服务器端参数配合使用。在 *xigemaas_home*/README.TXT 文件中定义了这类变量。

WebSphere:name=com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean

- 管理接口：`com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean`
- 注释：ServerXMLConfigurationMBean 提供接口以检索服务器已知的所有服务器配置文件的文件路径。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean

- 管理接口：`com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean`
- 注释：RuntimeUpdateNotificationMBean 为服务器运行时更新提供通知。附加至通知的用户数据对象为 java.util.Map。此 MBean 发出的运行时更新通知的通知类型为 com.ibm.websphere.runtime.update.notification。

WebSphere:name=com.ibm.ws.config.mbeans.FeatureListMBean

- 管理接口：`com.ibm.websphere.config.mbeans.FeatureListMBean`
- 注释：FeatureListMBean 展示单个方法以对运行时安装的所有功能部件生成 XML 报告。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.ws.config.serverSchemaGenerator

- 管理接口：`com.ibm.websphere.config.mbeans.ServerSchemaGenerator`
- 注释：ServerSchemaGenerator MBean 展示用于从已安装映像生成模式（最常用方法）或从当前运行时生成模式的方法。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig

- 管理接口：`com.ibm.websphere.webcontainer.GeneratePluginConfigMBean`
- 注释：请参阅为 配置 *Web* 服务器插件。

xigemaAS:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean

- 管理接口：com.ibm.websphere.filemonitor.FileNotificationMBean

xigemaAS:service=com.ibm.websphere.application.ApplicationMBean,name=*

- 管理接口：com.ibm.websphere.application.ApplicationMBean
- 注释：系统中的每个应用程序可以使用一个实例，其中 * 是唯一的应用程序名称。

xigemaAS:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,jndiName=*

- 管理接口：com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean
- 注释：为系统中的每个连接管理器提供了一个实例，其中 * 是连接管理器的 jndiName。如果连接管理器没有 jndiName，并且连接管理器嵌套在 server.xml 中的 DataSource 元素下，那么系统可使用 DataSource 的 jndiName。

xigemaAS:type=JvmStats

- 管理接口：com.ibm.websphere.monitor.jmx.JvmMXBean
- 注释：启用 monitor-1.0 功能部件后可用。请参阅 *JVM* 监视（见第 1613 页）。

xigemaAS:type=ServletStats,name=*

- 管理接口：com.ibm.websphere.webcontainer.ServletStatsMXBean
- 注释：启用 monitor-1.0 功能部件后，所服务的每个 servlet 都可以使用一个实例，其中 * 的格式为 <AppName>.<ServletName>。请参阅 *Web* 应用程序监视（见第 1614 页）。

xigemaAS:type=ThreadPoolStats,name=Default Executor

- 管理接口：com.ibm.websphere.monitor.jmx.ThreadPoolMXBean
- 注释：启用 monitor-1.0 功能部件后可用。请参阅 *ThreadPool* 监视（见第 1615 页）。

WebSphere:feature=jaxws,type=WebServiceStats,service=*,port=*

- 管理接口：org.apache.cxf.management.counters.ResponseTimeCounterMBean
- 注释：启用 monitor-1.0 功能部件后可用。WebServiceStats 可为 Performance.Counter.Server 或者 Performance.Counter.client，其中 service=* 是服务端点的限定名，port=* 是服务端点的端口名。请参阅 *JAX-WS* 监视（见第 1615 页）。

👉 注：这是一个动态模型 MBean。

WebSphere:feature=wasJmsServer, type=MessagingEngine,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.MessagingEngineMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件时可用。每个 xigemaAS 可有一个可用的消息传递引擎实例。name=* 是 MBean 的名称，其中 * 是消息传递引擎 MBean 的唯一名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Queue,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.QueueMBean

- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是队列 MBean 的名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Topic,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.TopicMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是主题 MBean 的名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Subscriber,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.SubscriberMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是订户 MBean 的名称。

  注：SubscriberMBean 是现有 TopicMBean 的订户。请参阅 *JMS* 消息传递（见第 1064 页）。

访问 MBean 属性和操作的示例

您可使用 xigemaAS 访问 Java™ 管理扩展 (JMX) 管理 Bean (MBean) 的属性以及调用其操作。

在您获得 MBeanServer 实例（对于 xigemaAS 上运行的应用程序）或 MBeanServerConnection 实例（对于外部客户机）之后，可以访问 xigemaAS 所提供 MBean 的属性或者调用这些 MBean 的操作。请参阅在 *xigemaAS* 上使用 *JMX MBean*（见第 1182 页）。

以下代码示例假定变量 mbs 是 MBeanServer 或 MBeanServerConnection 实例。可以使用所提供的方法，以类似于 Java™ 反射的方式来访问属性和调用操作。此外，每个 MBean 的管理接口具有用于属性的 getter 方法及用于操作的方法。可以使用这些接口，通过其中一个 javax.managementJMX.newMBeanProxy 方法或者其中一个 javax.management.JMX.newMXBeanProxy 方法（用于 MXBean）来获取代理对象。管理接口的名称以 “MXBean” 结尾。对于管理接口的名称，请参阅 所提供 *MBean* 的列表（见第 1182 页）。

示例 1：检查应用程序“myApp”的状态

```
import javax.management.ObjectName;
import javax.management.JMX;
import com.ibm.websphere.application.ApplicationMBean;
...

ObjectName myAppMBean = new ObjectName(
"xigemaAS:service=com.ibm.websphere.application.ApplicationMBean,name=myApp");
if (mbs.isRegistered(myAppMBean)) {
  String state = (String) mbs.getAttribute(myAppMBean, "State");
  // alternatively, obtain a proxy object
  ApplicationMBean app = JMX.newMBeanProxy(mbs, myAppMBean, ApplicationMBean.class);
  state = app.getState();
}
```

示例 2：从应用程序“myApp”获取 servlet“Example Servlet”的响应时间统计信息

```
import javax.management.ObjectName;
import javax.management.openmbean.CompositeData;
import javax.management.JMX;
import com.ibm.websphere.webcontainer.ServletStatsMXBean;

...
```

```
ObjectName servletMBean = new ObjectName("xigemaAS:type=ServletStats,name=myApp.Example
 Servlet");
if (mbs.isRegistered(servletMBean)) {
  CompositeData responseTimeDetails = (CompositeData) mbs.getAttribute(servletMBean,
 "ResponseTimeDetails");
  CompositeData responseTimeReading = (CompositeData) responseTimeDetails.get("reading");
  Double mean = (Double) responseTimeReading.get("mean");
  Double standardDeviation = (Double) responseTimeReading.get("standardDeviation");
  // alternatively, obtain a proxy object
  ServletStatsMXBean servletStats = JMX.newMXBeanProxy(mbs, servletMBean,
 ServletStatsMXBean.class);
  StatisticsMeter meter = servletStats.getResponseTimeDetails();
  StatisticsReading reading = meter.getReading();
  mean = reading.getMean();
  standardDeviation = reading.getStandardDeviation();
}
```

示例 3：创建 Web 服务器插件配置文件

```
import com.ibm.websphere.webcontainer.GeneratePluginConfigMBean;

...

ObjectName pluginMBean = new
 ObjectName("WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig");
if (mbs.isRegistered(pluginMBean)) {
  mbs.invoke(pluginMBean, "generatePluginConfig", new Object[] {
    "installRoot", "serverName"}, new String[] {
    String.class.getName(), String.class.getName()
  });
  // alternatively, use a proxy object
  GeneratePluginConfigMBean plugin = JMX.newMBeanProxy(mbs, name,
 GeneratePluginConfigMBean.class);
  plugin.generatePluginConfig("installRoot", "serverName");
}
```

示例 4：查询 Web Service 端点的状态

```
import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;

...

// Init mbs as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-
Bus,
 type=Bus.Service.Endpoint,service=\"{http://vsettan.com.cn/}TestEndpointService\",
    port=\"TestEndpoint\",instance.id=1816106538");
MBeanInfo beanInfo = mbsc.getMBeanInfo(objName);

// Go through attributes to find the interested one
for (MBeanAttributeInfo attr : beanInfo.getAttributes()) {
  if (attr.getName().equals("State")) {
          String status = String.valueOf(mbs.getAttribute(objName, attr.getName()));
      break; }
}
```

示例 5：关闭 CXF 服务器总线

```
import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
```

```
import javax.management.MBeanOperationInfo;

...

// Init mbsc as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-
Bus,
    type=Bus,instance.id=1618108530");
MBeanInfo beanInfo = mbsc.getMBeanInfo(objName);

// Go through operation to find the interested one and invoke
for (MBeanOperationInfo operation : beanInfo.getOperations()) {
  if (operation.getName().equals("shutdown")) {
    mbs.invoke(objName, operation.getName(), new Object[] { true }, new String[]
 { boolean.class.getName() });
      break; }
}
```

注册 MBean 的示例

应用程序可在 xigemaAS 上注册其自己的 MBean 实例。之后，该 MBean 实例可供其他应用程序或外部管理员使用。

任何应用程序都可以通过使用 `MBeanServer` 实例来注册 MBean。假定应用程序包含称为 `org.example.Example` 的类，这个类实现接口 `org.example.ExampleMBean` 来定义一些属性和操作。如以下示例中所示，应用程序可能只需实例化 `Example` 类，然后使用唯一的 `ObjectName` 来注册这个类。如果所选择的 `ObjectName` 已在使用中，那么会报告 `javax.management.InstanceAlreadyExistsException`。

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.Example;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
Object mbean = new Example();
ObjectName name = new ObjectName("org.example.MyApplication:name=Example");
mbs.registerMBean(mbean, name);
```

此外，应用程序还可注册 MBean 来扩展 `java.lang.ClassLoader` 并提供对任何数量的 MBean 实现类的访问权。然后，可以使用任何其他 JMX 客户机（本地或远程）来创建并注册应用程序所随附的 MBean。例如，假设应用程序具有可执行下列任务的 MBean 类 `org.example.ApplicationClassLoader`：

- 实现任何空接口 `org.example.ApplicationClassLoaderMBean`
- 扩展 `java.lang.Classloader`，并且
- 提供对 `org.example.Example` MBean 实现类的访问权

应用程序可以注册 `ApplicationClassLoader` 的实例，以使 `Example` MBean 可供其他 JMX 客户机使用，如下所示：

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.ApplicationClassLoader;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
Object classLoader = new ApplicationClassLoader();
ObjectName name = new ObjectName("org.example.MyApplication:name=ClassLoader");
```

```
mbs.registerMBean(classLoader, name);
```

任何 JMX 客户机都可以创建 Example 实例。以下示例假定变量 mbs 是 MBeanServer 或 MBeanServerConnection 实例。请参阅在 *xigemaAS* 上使用 *JMX MBean*（见第 1182 页）。

```
import javax.management.ObjectName;

...

ObjectName loaderName = new ObjectName("org.example.MyApplication:name=ClassLoader");
ObjectName exampleName = new ObjectName("org.example.MyApplication:name=Example");
mbs.createMBean("org.example.Example", exampleName, loaderName);
```

必要时，可以使用其他形式的 MBeanServer.createMBean 方法，通过使用非缺省构造函数来创建 MBean。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

建立 JMX MBean xigemaAS 服务器连接

可以使用基于 Jython 的脚本来建立 Java™ 管理扩展 (JMX) MBean xigemaAS 服务器连接。

必须先获取并安装所选 Jython 版本，才能执行此过程。如果没有 Jython 运行时，那么指令会失败。

1. 设置环境。

   您需要的文件位于 *xigemaas_home*/clients/jython。

   a. 将 lib/restConnector.py 文件复制到 *jython_home*/Lib。
   b. 设置 *xigemaas_home*/clients 中的 restConnector.jar 的类路径。

   ```
   set CLASSPATH=%CLASSPATH%;c:\wlp\clients\restConnector.jar
   ```

2. 运行实用程序。

   示例 **1**：使用 connector.connect(host,port,user,password) 获取简单连接

   ```
   from restConnector import JMXRESTConnector
   JMXRESTConnector.trustStore = "c:/key.jks"
   JMXRESTConnector.trustStorePassword = "xigemaAS"

   connector = JMXRESTConnector()
   connector.connect("foo.bar.com",9443,"theUser","thePassword")
   mconnection = connector.getMBeanServerConnection()
   # mconnection.invoke(...)
   connector.disconnect()
   ```

   示例 **2**：使用 connector.connect(host,port,map) 以及用户提供的属性来获取高级连接

   ```
   import java
   import javax
   import jarray
   import com.ibm.websphere.jmx.connector.rest
   import com.ibm.ws.jmx.connector.client.rest

   map=java.util.HashMap()
   map.put("jmx.remote.provider.pkgs","com.ibm.ws.jmx.connector.client")
   map.put(javax.management.remote.JMXConnector.CREDENTIALS,jarray.array(["theUser","thePassword"],java.lang.S
   map.put(com.ibm.ws.jmx.connector.client.rest.ClientProvider.READ_TIMEOUT,2*60*1000)
   map.put(com.ibm.websphere.jmx.connector.rest.ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION,
    True)
   ```

```
connector = JMXRESTConnector()
connector.connect("foo.bar.com",9443,map)
mconnection = connector.getMBeanServerConnection()
# mconnection.invoke(...)
connector.disconnect()
```

示例 **3**：注册通知侦听器

```
import java
import javax

from restConnector import JMXRESTConnector
from restConnector import BaseNotificationListener

class SampleNotificationListener(BaseNotificationListener):
  def __init__(self):
    pass

  def handleNotification(self,notification,handback):
    print "Notification received:"
    print "  Source: " + notification.getSource().toString()
    print "  Type: " + notification.getType()
    print "  Message: " + notification.getMessage()

# main starts here

JMXRESTConnector.trustStore = "c:/key.jks"
JMXRESTConnector.trustStorePassword = "xigemaAS"

connector=JMXRESTConnector()
connector.connect("foo.bar.com",9443,"theUser","thePassword")
mconnection=connector.getMBeanServerConnection()

listener=SampleNotificationListener()
handback=java.lang.Object()

notifier1=javax.management.ObjectName("web:name=Notifier1")
mconnection.addNotificationListener(notifier1,listener,None,handback)
```

JMXRESTConnector.trustStore

设置 SSL 密钥文件存储在的路径

JMXRESTConnector.trustStorePassword

设置密钥的密码

JMXRESTConnector.connect(*host*,*port*,*user*,*password*)

对服务器创建连接器

JMXRESTConnector.connect(*host*,*port*,*map*)

创建具有用户属性的连接器

JMXRESTConnector.getMBeanServerConnection

获取与 MBean 服务器的连接

JMXRESTConnector.disconnect()

关闭连接

与 MBean 服务器建立连接之后，您可以使用 `invoke(...)` 方法来调用 MBean 服务器。

文件传输

restConnector-1.0 功能部件包含 FileTransfer 和 FileService MBean。FileTransfer MBean 支持对正在运行的 xigemaAS 服务器执行删除、上载和下载操作。FileService MBean 可让您访问目录列表和文件元数据，并且也提供归档操作，例如 create 和 expand。

FileTransfer 和 FileService MBean 可用来在 xigemaAS 上执行远程操作，例如更新配置或安装应用程序。要以远程方式执行配置更新，可以为目标 xigemaAS 服务器上载已更新的 server.xml 文件。要安装应用程序，可以上载应用程序归档和已更新的 server.xml 文件，或者只需将应用程序归档上载到受监视的 dropins 文件夹。

FileTransfer MBean 包含可配置的读写列表，以便您能控制使用 FileTransfer MBean 时可以读取或写入的目录。

有关如何配置 restConnector-1.0 功能部件并控制 FileTransfer MBean 读写列表的信息，请参阅所提供 *MBean* 的列表（见第 1182 页）中有关 FileTransfer MBean 的部分。

FileTransfer 中的 uploadFile 方法包含称为“expandOnCompletion”的布尔值，可让用户通过单个 MBean 调用即可上载和展开归档。在目标路径上创建和归档同名的目录，然后会自动调用 FileService MBean 来展开归档。

示例：使用参数 {"C:/temp/myArchive.zip", "${server.output.dir}/myArchive.zip", true} 来调用 uploadFile 会在 ${server.output.dir}（包含正在创建的归档的已解压内容）下生成 myArchive.zip 目录。

在 xigemaAS 中配置二进制日志记录

使用此信息作为在 xigemaAS 中配置二进制日志记录的指南。

与缺省 xigemaAS 日志和跟踪框架相比，二进制日志记录提供处理速度更快的日志和跟踪功能以及更灵活的方式来使用日志和跟踪内容。

服务器配置由 bootstrap.properties 文件、server.xml 文件以及这些文件随附的任何可选文件组成。bootstrap.properties 文件指定在处理主要配置之前必须提供的属性，这些属性会保持最少。server.xml 文件是服务器的主要配置文件。

server.xml 文件及其关联文件使用适合于大多数文本编辑器的简单 XML 格式。

bootstrap.properties 文件指定服务器是否应该将二进制日志记录用作日志和跟踪框架，或者用作缺省日志和跟踪框架。

可以通过服务器配置或者 bootstrap.properties 文件来配置二进制日志记录。

- 服务器配置：要从您自己的代码获取日志记录（进行服务器配置处理之后载入），请使用服务器配置来配置二进制日志记录。
- bootstrap.properties 文件：可能需要设置日志记录属性以使其在服务器配置文件得到处理之前生效。例如，如果需要分析服务器启动或配置处理早期发生的问题。在这种情况下，您可以在 bootstrap.properties 文件中配置二进制日志记录。

可以在 bootstrap.properties 或者 server.xml 文件中设置日志记录属性。使用 server.xml 文件中的属性，或者使用 bootstrap.properties 文件中的等价属性。从服务器读取 bootstrap.properties 文件开始，使用 bootstrap.properties 文件中的任何设置，直到完成处理 server.xml 文件为止。如果bootstrap.properties 文件中的日志记录属性未在 server.xml 文件中进行替换或重置，那么将继续使用 bootstrap.properties 文件中的属性值。

如果启用了二进制日志记录，那么会忽略 maxFileSize、maxFiles、messageFileName、traceFileName 和 traceFormat 日志记录元素属性（因为二进制日志记录是在没有 trace.log 和 messages.log 文件的情况下运行）。traceSpecification、consoleLogLevel 和 logDirectory 属性继续用来设置跟踪规范、控制台日志的级别以及日志和跟踪文件的布置。

如果您在 server.xml 文件中设置日志记录或二进制日志记录属性，那么可以通过在 bootstrap.properties 文件中将相应的属性设置为同一值来避免在启动时与运行时之间更改配置。请注意，如果在 bootstrap.properties 文件中未设置任何日志记录属性或者二进制日志记录属性，那么服务器将使用缺省日志记录设置。

- 通过更新 bootstrap.properties 文件对服务器启用二进制日志记录。

  在 bootstrap.properties 文件中，单独添加下列文本行：

  ```
  websphere.log.provider=binaryLogging-1.0
  ```

- 使用下列参数来配置二进制日志记录。

  列出的所有子元素都是 server.xml 文件中日志记录元素的子元素。

  下表列出了可在 server.xml 文件中配置的属性，以及可在 bootstrap.properties 文件中设置的等价属性：

表 20: 可在 server.xml 中配置的二进制日志记录属性以及可在 bootstrap.properties 中设置的等价属性

日志记录子元素	属性	等价的 **bootstrap.properties** 属性
binaryLog	purgeMaxSize purgeMinTime fileSwitchTime bufferingEnabled outOfSpaceAction	com.ibm.hpel.log.purgeMaxSize com.ibm.hpel.log.purgeMinTime com.ibm.hpel.log.fileSwitchTime com.ibm.hpel.log.bufferingEnabled com.ibm.hpel.log.outOfSpaceAction
binaryTrace	purgeMaxSize purgeMinTime fileSwitchTime bufferingEnabled outOfSpaceAction	com.ibm.hpel.trace.purgeMaxSize com.ibm.hpel.trace.purgeMinTime com.ibm.hpel.trace.fileSwitchTime com.ibm.hpel.trace.bufferingEnabled com.ibm.hpel.trace.outOfSpaceAction

以下示例显示配置为启用二进制日志记录的 bootstrap.properties 文件：

```
websphere.log.provider=binaryLogging-1.0
```

以下示例显示了具有二进制日志记录子元素的 server.xml 文件。日志内容设为在 96 小时后过期，跟踪内容设为最大保留 1024MB：

```
<server description="new server">

    <logging>
        <binaryLog purgeMinTime="96"/>
        <binaryTrace purgeMaxSize="1024"/>
    </logging>

</server>
```

在重新启动服务器之后，将启用并配置二进制日志记录。

在 xigemaAS 上管理事务服务

如果应用程序使用两个或多个资源，那么 xigemaAS 事务管理器会在全局事务控制下协调对所有资源管理器的更新。在 xigemaAS 中，事务服务是在您指定需要使用事务的功能部件（例如，jpa-2.0、jdbc-4.0 和 wasJmsServer-1.0）时隐式激活的。

可控制数据库事务恢复何时发生，可指定用于确定数据库事务恢复方式的配置设置，还可选择是将事务日志存储为操作系统文件还是存储在关系数据库中。

配置事务服务的启动

第一次使用事务服务时或服务器启动时，可能会发生数据库事务恢复。

缺省情况下，如果是第一次使用事务服务而不是在服务器启动时使用事务服务，那么服务器发生故障后会进行事务恢复。通过指定事务服务属性（用于控制恢复发生时间、系统是否会等待恢复完成才允许事务性工作继续），可改变此行为。

- 要配置事务服务启动，请在 server.xml 文件的 `transaction` 元素中指定以下属性：
- recoverOnStartup

  此属性可采用以下值：

  - true：在服务器启动时恢复事务。
  - false：在第一次使用事务服务时恢复事务。
- waitForRecovery

  此属性可采用以下值：

  - true：服务器会等待事务恢复完成才允许事务性工作继续。
  - false：服务器允许事务性工作继续而不等待事务恢复完成。

通过以下事务元素配置，事务恢复在服务器启动时发生，并且服务器会等待事务恢复完成才允许事务性工作继续。

```
<transaction
  recoverOnStartup="true"
  waitForRecovery="true"
/>
```

如何恢复数据库事务

xigemaAS 概要文件事务管理器恢复不确定数据库事务时，它会使用唯一标识或 JNDI 名称来定位当前 dataSource 元素，然后确定要用于恢复的用户标识和密码。

通过在 server.xml 配置文件中指定 dataSource 元素的属性来配置数据源。可对数据源指定唯一标识或 jndiName 属性，如下所示：

```
<dataSource id="ds1" jndiName="jdbc/ds1"... />
```

对数据源参与的事务暂挂恢复时，不得更改 id 或 jndiName 属性的值。如果更改 dataSource 元素的任何其他属性，那么恢复时会保留这些更改。因此，（例如）可添加 recoveryAuthDataRef 属性以指定要用于恢复的数据库用户标识和密码。

用于恢复的数据库用户标识和密码是根据以下优先顺序确定的：

1. 如果 dataSource 元素定义了 recoveryAuthDataRef 属性，那么会使用来自 authData 元素的用户标识和密码。例如：

   ```
   <authData id="recoveryAuth" user="dbuser1" password="{xor}Oz0vKDtu"/>
   <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2"
               recoveryAuthDataRef="recoveryAuth" .../>
   ```

2. 如果使用容器管理的认证，那么会使用来自容器管理的认证别名的用户标识和密码。例如：

   - 在 Vsettan-web-bnd.xml 文件中，您具有下列代码：

     ```
     <resource-ref name="jdbc/ds1ref" binding-name="jdbc/ds1">
         <authentication-alias name="user1Auth"/>
      </resource-ref>
     ```

   - 在 server.xml 文件中，必须定义以下代码：

     ```
     <authData id="user1Auth" user="dbuser1" password="{xor}Oz0vKDtu"/>
     <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" .../>
     ```

3. 使用来自 dataSource 元素的用户标识和密码。例如：

   ```
   <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" ...>
        <properties.db2.jcc databaseName="testdb" user="dbuser1" password="{xor}Oz0vKDtu"/>
   </dataSource>
   ```

4. 如果上述条件都不满足，并且尝试在没有任何用户标识和密码的情况下进行恢复，那么其行为由 JDBC 驱动程序和数据库确定。

👉 注：如果事务恢复由应用程序定义的数据源（例如，@DataSourceDefinition 注解或部署描述符中的 <data-source> 元素）执行，那么必须确保进行恢复时关联应用程序正在运行。不能使用 server.xml 文件中的配置设置来恢复应用程序定义的数据源。

将事务日志存储在关系数据库中

可选择将 xigemaAS 概要文件事务日志存储在关系数据库而不是操作系统文件中。在 xigemaAS 中提供此功能部件是为了实现兼容性及评估和测试用途。

xigemaAS事务服务将涉及两个或多个资源或分布在多个服务器上的每个全局事务的信息写至一个事务日志。这些事务由应用程序或部署了这些应用程序的容器启动或停止。事务服务保留事务日志以确保事务的完整性。信息在分布式事务的准备阶段写至事务日志，以便带有生效事务的 xigemaAS 在发生故障后重新启动时，事务服务能够使用日志来重演所有不确定事务。这允许整个系统返回至一致状态。

在 xigemaAS 的前发行版中，事务日志存储为操作系统文件。在 xigemaAS V9.0.0及更高版本中，它们保留缺省配置，但您可选择将事务日志存储到关系数据库管理系统 (RDBMS) 中。此配置选项的目标是 HA 环境中工作的客户。在 xigemaAS 的前发行版中，HA 事务支持要求使用共享文件系统来托管事务日志，例如，NFSv4 安装的网络连接存储器 (NAS) 或存储区域网络 (SAN)。此新功能部件允许客户（特别是投资了 HA 数据库技术的客户）将其 HA 数据库用作事务日志的共享存储库，以替代共享文件系统。

缺省情况下，xigemaAS概要文件事务日志存储在操作系统文件中。可使用 xigemaAS 概要文件支持的任何数据库类型。

1. 要将 xigemaAS概要文件事务日志配置为存储在 RDBMS 中，请完成以下步骤：
1. 在xigemaAS 概要文件 server.xml 文件中配置专用非事务性数据源。

   server.xml 文件中的以下示例片段显示如何将 xigemaAS 概要文件配置为将其事务日志存储在 DB2® 数据库中：

```
<transaction>
  <dataSource transactional="false">
    <jdbcDriver libraryRef="DB2JCC4LIB"/>
    <properties.db2.jcc currentSchema="CBIVP"
      databaseName="SAMPLE" driverType="4"
      portNumber="50000" serverName="localhost"
      user="db2admin" password="{xor}Oz1tPjsyNjE=" />
  </dataSource>  </transaction>

<library id="DB2JCC4LIB">
  <fileset dir="C:/SQLLIB/java" includes="db2jcc4.jar db2jcc_license_cu.jar"/>
</library>
```

2. （可选）创建数据库表。

   服务器第一次启动时，xigemaAS 概要文件会尝试创建必要的数据库表。例如，如果因为许可权不足而无法进行，那么服务器无法启动。在这些环境下，必须手动创建两个必需的数据库表。

   以下 DDL 结构显示如何在 DB2® 上创建表：

```
CREATE TABLE WAS_TRAN_LOG(
  SERVER_NAME VARCHAR(128),
  SERVICE_ID SMALLINT,
  RU_ID BIGINT,
  RUSECTION_ID BIGINT,
  RUSECTION_DATA_INDEX SMALLINT,
  DATA LONG VARCHAR FOR BIT DATA)
```

```
CREATE TABLE WAS_PARTNER_LOG(
  SERVER_NAME VARCHAR(128),
  SERVICE_ID SMALLINT,
  RU_ID BIGINT,
  RUSECTION_ID BIGINT,
  RUSECTION_DATA_INDEX SMALLINT,
  DATA LONG VARCHAR FOR BIT DATA)
```

   以下 DDL 结构显示如何在 Oracle 上创建数据库表：

```
CREATE TABLE WAS_TRAN_LOG(
  SERVER_NAME VARCHAR(128),
  SERVICE_ID SMALLINT,
  RU_ID NUMBER(19),
  RUSECTION_ID NUMBER(19),
  RUSECTION_DATA_INDEX SMALLINT,
  DATA BLOB)
```

```
CREATE TABLE WAS_PARTNER_LOG(
  SERVER_NAME VARCHAR(128),
  SERVICE_ID SMALLINT,
```

```
  RU_ID NUMBER(19),
  RUSECTION_ID NUMBER(19),
  RUSECTION_DATA_INDEX SMALLINT,
  DATA BLOB)
```

在 xigemaAS 上管理数据访问资源

使用 xigemaAS 功能部件为 JDBC、MongoDB、redis DB 和其他对象管理数据访问资源。

在 xigemaAS 上管理数据访问应用程序

xigemaAS 向使用 xigemaAS 功能部件（例如 `jdbc-4.0`）和其他功能部件的数据访问应用程序提供支持。

在 xigemaAS 中配置数据库连接

可以为数据库连接配置与不同 JDBC 提供者相关联的数据源。JDBC 提供者提供与特定供应商数据库进行 JDBC 连接所需的驱动程序实现类。

要从应用程序访问数据库，必须使用数据源。数据源由 JDBC 驱动程序提供，具有下列种类：

- `javax.sql.DataSource`

  这是数据源的基本形式。它并不提供互操作性来增强连接池，而且不能作为具备两阶段功能的资源来参与涉及多个资源的事务。
- `javax.sql.ConnectionPoolDataSource`

  这种数据源已对连接池启用。它不能作为具备两阶段功能的资源来参与涉及多个资源的事务。
- `javax.sql.XADataSource`

  这种数据源已对连接池启用，而且能够作为具备两阶段功能的资源参与涉及多个资源的事务。

JDBC 驱动程序必须至少提供其中一种数据源，才能在 xigemaAS 中使用。对于常用的 JDBC 驱动程序，xigemaAS 已经注意到各种数据源类型的实现类名。只需告知 xigemaAS 在何处查找 JDBC 驱动程序。

1. 在 server.xml 文件中，定义指向 JDBC 驱动程序 JAR 文件位置或压缩文件位置的共享库。

   例如：

   ```
   <library id="DB2JCC4Lib">
       <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
   </library>
   ```

2. 定义使用 JDBC 驱动程序的数据源。如果您未指定这种数据源，那么 xigemaAS 会根据可用项按以下顺序选择数据源。

   - `javax.sql.ConnectionPoolDataSource`
   - `javax.sql.DataSource`
   - `javax.sql.XADataSource`

   下面是接受缺省数据源类型的示例：

   ```
   <dataSource id="db2" jndiName="jdbc/db2">
       <jdbcDriver libraryRef="DB2JCC4Lib"/>
       <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
   </dataSource>
   ```

   下面是使用 `javax.sql.XADataSource` 类型的示例：

   ```
   <dataSource id="db2xa" jndiName="jdbc/db2xa" type="javax.sql.XADataSource">
       <jdbcDriver libraryRef="DB2JCC4Lib"/>
       <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
   </dataSource>
   ```

如果启用了至少一个 Java EE 7 功能部件，那么缺省数据源可用。如果未指定类型，那么此数据源使用另一优先级确定类型。

- javax.sql.XADataSource
- javax.sql.ConnectionPoolDataSource
- javax.sql.DataSource

此数据源以 java:comp/DefaultDataSource 形式提供。不必为其指定 jndiName。要配置缺省数据源，请指定数据源并将其标识设置为 DefaultDataSource。以下是配置缺省数据源以指向 DB2 数据库的示例：

```
<dataSource id="DefaultDataSource">
 <jdbcDriver libraryRef="DB2JCC4Lib"/>
 <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost"
 portNumber="50000"/>
</dataSource>
```

3. 可选： 配置数据源的属性，例如 JDBC 供应商属性和连接池属性。

   例如：

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2" connectionSharing="MatchCurrentState"
           isolationLevel="TRANSACTION_READ_COMMITTED" statementCacheSize="20">
    <connectionManager maxPoolSize="20" minPoolSize="5"
                       connectionTimeout="10s" agedTimeout="30m"/>
    <jdbcDriver libraryRef="DB2JCC4Lib"/>
    <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"
                        currentLockTimeout="30s" user="user1" password="pwd1"/>
</dataSource>
```

   有关 dataSource 元素、connectionManager 元素及一些常用 JDBC 供应商的配置属性的完整列表，请参阅数据源 *(dataSource)*（见第 213 页）。

4. 可选： 根据以下示例来配置常用数据库的数据源。

对于 **DB2®**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/db2">
    <jdbcDriver libraryRef="DB2JCC4Lib"/>
    <properties.db2.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="50000"/>
</dataSource>

<library id="DB2JCC4Lib">
    <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

对于 **Derby Embedded**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/derbyEmbedded">
    <jdbcDriver libraryRef="DerbyLib"/>
    <properties.derby.embedded databaseName="C:/databases/SAMPLEDB" createDatabase="create"/>
</dataSource>

<library id="DerbyLib">
    <fileset dir="C:/db-derby-10.8.1.2-bin/lib"/>
</library>
```

对于 **Derby Network Client**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/derbyClient">
    <jdbcDriver libraryRef="DerbyLib"/>
    <properties.derby.client databaseName="C:/databases/SAMPLEDB" createDatabase="create"
                             serverName="localhost" portNumber="1527"/>
```

```
</dataSource>

<library id="DerbyLib">
    <fileset dir="C:/db-derby-10.8.1.2-bin/lib"/>
</library>
```

对于 **Informix® JCC**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/informixjcc">
    <jdbcDriver libraryRef="DB2JCC4Lib"/>
    <properties.informix.jcc databaseName="SAMPLEDB" serverName="localhost" portNumber="1526"
/>
</dataSource>

<library id="DB2JCC4Lib">
    <fileset dir="C:/Drivers/jcc/4.8" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

对于 **Informix® JDBC**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/informix">
    <jdbcDriver libraryRef="InformixLib"/>
    <properties.informix databaseName="SAMPLEDB" ifxIFXHOST="localhost"
                         serverName="ol_machinename" portNumber="1526"/>
</dataSource>

<library id="InformixLib">
    <fileset dir="C:/Drivers/informix" includes="ifxjdbc.jar ifxjdbcx.jar"/>
</library>
```

对于 **Microsoft™ SQL Server（Microsoft™ JDBC 驱动程序）**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/mssqlserver">
    <jdbcDriver libraryRef="MSJDBCLib"/>
    <properties.microsoft.sqlserver databaseName="SAMPLEDB"
                                    serverName="localhost" portNumber="1433"/>
</dataSource>

<library id="MSJDBCLib">
    <fileset dir="C:/sqljdbc_4.0/enu" includes="sqljdbc4.jar"/>
</library>
```

对于 **Microsoft™ SQL Server（DataDirect Connect for JDBC 驱动程序）**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/ddsqlserver">
    <jdbcDriver libraryRef="DataDirectLib"/>
    <properties.datadirect.sqlserver databaseName="SAMPLEDB"
                                     serverName="localhost" portNumber="1433"/>
</dataSource>

<library id="DataDirectLib">
    <fileset dir="C:/DataDirect/Connect-4.2/lib" includes="sqlserver.jar"/>
</library>
```

对于 **MySQL**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/mySQL">
    <jdbcDriver libraryRef="MySQLLib"/>
    <properties databaseName="SAMPLEDB" serverName="localhost" portNumber="3306"/>
</dataSource>

<library id="MySQLLib">
    <fileset dir="C:/mysql-connector-java-x.x.xx" includes="mysql-connector-java-x.x.xx.jar"/
>
</library>
```

对于 **Oracle**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/oracle">
    <jdbcDriver libraryRef="OracleLib"/>
    <properties.oracle URL="jdbc:oracle:thin:@//localhost:1521/SAMPLEDB"/>
</dataSource>

<library id="OracleLib">
    <fileset dir="C:/Oracle/lib" includes="ojdbc6.jar"/>
</library>
```

对于 **Sybase**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/sybase">
    <jdbcDriver libraryRef="SybaseLib"/>
    <properties.sybase databaseName="SAMPLEDB" serverName="localhost" portNumber="5000"/>
</dataSource>

<library id="SybaseLib">
    <fileset dir="C:/Drivers/sybase" includes="jconn4.jar"/>
</library>
```

对于 **solidDB®**

```
<dataSource id="DefaultDataSource" jndiName="jdbc/solidDB">
    <jdbcDriver libraryRef="solidLib"/>
    <properties databaseName="SAMPLEDB" URL="jdbc:solid://localhost:2315/"/>
</dataSource>

<library id="solidLib">
    <fileset dir="C:/Drivers/solidDB" includes="SolidDriver2.0.jar"/>
</library>
```

对于 **xigemaAS** 不识别的 **JDBC** 驱动程序

```
<dataSource id="DefaultDataSource" jndiName="jdbc/sample" type="javax.sql.XADataSource">
    <jdbcDriver libraryRef="SampleJDBCLib"
                javax.sql.XADataSource="com.ibm.sample.SampleXADataSource"/>
    <properties databaseName="SAMPLEDB" hostName="localhost" port="12345"/>
</dataSource>

<library id="SampleJDBCLib">
    <fileset dir="C:/Drivers/SampleJDBC" includes="sampleDriver.jar"/>
</library>
```

在示例中，JDBC 驱动程序位于 C:/Drivers/SampleJDBC/sampleDriver.jar，并提供名称为 `com.ibm.sample.SampleXADataSource` 的 `javax.sql.XADataSource` 实现。JDBC 驱动程序还提供特定于供应商的数据源属性，例如 databaseName、hostName 和 port。

配置缺省数据源

可以为数据库连接配置与不同 JDBC 提供程序相关联的缺省数据源。JDBC 提供程序提供与特定供应商数据库进行 JDBC 连接所需的驱动程序实现类。

要从应用程序访问数据库，必须配置数据源。

1. 在 server.xml 文件中，使用 `DefaultDataSource` 标识配置 `datasource` 元素。

```
<dataSource id="DefaultDataSource">
    <jdbcDriver libraryRef="MyJDBCLib"/>
    <properties.derby.embedded databaseName="myDB"
 createDatabase="create"/>
    <containerAuthData user="user1"  password="{xor}Oz0vKDtu" />
</dataSource>
```

```
<library id="MyJDBCLib">
    <file name="C:/derby/derby.jar"/>
</library>
```

注：此服务器必须正在 Java Enterprise Edition 7 平台级别运行。如果在 server.xml 文件中启用了一个或多个 Java Enterprise Edition 7 功能部件，那么将启用此平台级别。

2. 要在 Web 应用程序中使用 `DefaultDataSource`，那么可使用依赖性注入获取引用：

```
@Resource
DataSource defaultDataSource;
```

或通过 JNDI 查找获取：

```
DataSource defaultDataSource = (DataSource) new
 InitialContext().lookup("java:comp/DefaultDataSource");
```

如何应用数据源配置更新

如果在服务器处于运行状态时更改 dataSource 元素的属性，那么会在不同时间，以不同的方式应用不同属性的更新。

要配置数据源，请在 server.xml 配置文件中指定 dataSource 元素的属性。如果更改运行中服务器的这些属性，那么会在不同时间，以不同的方式应用更新，取决于更改的属性。下表针对 dataSource 元素的每个属性，描述了如何在运行时应用配置更改。

表 21: 如何在运行时应用数据源配置更新

表的第 1 列列出了 dataSource 元素的属性。第 2 列针对每个属性，描述了如何在运行时应用配置更新。

属性名称	如何应用配置更新
beginTranForResultSetScrollingAPIs	更新会立即生效。
beginTranForVendorAPIs	更新会立即生效。
commitOrRollbackOnCleanup	更新会立即生效。
connectionManagerRef	将破坏所有连接及连接池。数据源随后由新连接管理器进行管理。
connectionSharing	更新分别随事务中的第一个连接句柄应用。
isolationLevel	更新随新连接请求应用；当前连接会保留其隔离级别。
jdbcDriverRef	将破坏所有连接及连接池。随后会使用新的 JDBC 驱动程序。
jndiName	将破坏所有连接及连接池。随后会使用新的 JNDI 名称。
propertiesRef	如果数据源是 Derby Embedded，那么在新属性生效之前，将破坏所有连接及连接池。对于其他 JDBC 驱动程序，新属性会随新连接请求生效。
queryTimeout	更新会立即生效。
recoveryAuthDataRef	用于事务恢复的认证数据。将破坏所有连接及连接池。随后会使用新的恢复认证数据。
statementCacheSize	下次使用时会调整语句高速缓存的大小。

属性名称	如何应用配置更新
supplementalJDBCTrace	将破坏所有连接及连接池。随后会使用新设置。
syncQueryTimeoutWithTransactionTimeout	更新会立即生效。
transactional	更新会应用到连接池中的新连接及未使用的现有连接。
type	将破坏所有连接及连接池。随后会使用新设置。

应用程序定义的数据源

可以通过注解或借助部署描述符，在应用程序中定义数据源，如 Java™ EE 规范所定义。

> 注：对于应用程序定义的数据源，建议提供 commonLibraryRef 类加载器属性。privateLibraryRef 属性无法用于 java:global 命名空间，并且不鼓励用于其他作用域。如果多个应用程序声明了同一 java:global 命名空间以指定数据源，那么这些应用程序的 server.xml 文件必须都对同一共享库指定 commonLibraryRef 属性。

在应用程序中定义数据源时，必须使 JDBC 驱动程序可供应用程序使用。这通过在 server.xml 中为应用程序配置共享库来完成。

例如：

```
<application id="myApp" name="myApp" location="myApp.war" type="war">
     <classloader commonLibraryRef="DB2Lib"/>
</application>
<library id="DB2Lib">
     <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
</library>
```

然后，可以通过注解或借助部署描述符在应用程序中定义数据源。

- 如以下示例中所示使用注解：

```
@DataSourceDefinition(
        name          = "java:comp/env/jdbc/db2",
        className     = "com.ibm.db2.jcc.DB2DataSource",
        databaseName = "SAMPLEDB",
        serverName    = "localhost",
        portNumber    = 50000,
        properties    = { "driverType=4" },
        user          = "user1",
        password      = "pwd1"
        )

public class MyServlet extends HttpServlet {

    @Resource(lookup="java:comp/env/jdbc/db2")
    DataSource ds;
```

- 例如，如以下示例中所示在 web.xml 文件中使用部署描述符：

```
<data-source>
    <name>java:comp/env/jdbc/db2</name>
    <class-name>com.ibm.db2.jcc.DB2DataSource</class-name>
    <server-name>localhost</server-name>
    <port-number>50000</port-number>
    <database-name>SAMPLEDB</database-name>
    <user>user1</user>
    <password>pwd1</password>
    <property><name>driverType</name><value>4</value></property>
  </data-source>
```

通常，在 server.xml 文件中的 dataSource 或 connectionManager 上定义的属性也可以在应用程序定义的数据源上加以指定。此情况的两个例外是 connectionManagerRef 和 jdbcDriverRef，您不能指定它们，因为应用程序定义的数据源隐式定义了连接管理器和 JDBC 驱动程序。将应用程序定义的数据源用于两阶段落实时，可以指定 recoveryAuthDataRef 属性以选择用于事务恢复的认证数据。但是，一定要知道事务恢复只能在应用程序处于运行状态时才能进行。可以在应用程序定义的数据源中使用变量、经过编码的密码以及持续时间语法。

👉 注：持续时间语法不适用于注解中显式定义的属性，例如 loginTimeout 或 maxIdleTime。

下面举例说明了两个使用连接管理器属性的数据源、变量、经过编码的密码以及持续时间语法的数据源。

```
@DataSourceDefinitions(value = {
    @DataSourceDefinition(
        name          = "java:comp/env/jdbc/derby",
        className     = "org.apache.derby.jdbc.EmbeddedDataSource40",
        databaseName = "${shared.resource.dir}/data/SAMPLEDB",
        minPoolSize   = 1,
        maxPoolSize   = 10,
        maxIdleTime = 180,
        properties = { "agedTimeout=10m", "connectionTimeout=30s", "createDatabase=create" }
        ),
    @DataSourceDefinition(
        name          = "java:comp/env/jdbc/oracle",
        className     = "oracle.jdbc.pool.OracleDataSource",
        url           = "jdbc:oracle:thin:@//localhost:1521/SAMPLEDB",
        user          = "user1",
        password      = "{xor}Oz0vKDtt"
        )
})
```

为使用 DB2® 数据库的应用程序配置客户机重新路由

可使用客户机重新路由功能部件来配置企业应用程序，以便 DB2® 数据库从通信损失恢复，并且应用程序可在干扰降至最低的情况下继续工作。重新路由是持续运行支持的核心，但仅当具有已对应用程序服务器连接标识的备用位置时，重新路由才可行。

此任务假定：

- 应用程序连接至的 DB2® 数据源正在运行：
  - DB2® Database for Linux™, UNIX™, and Windows™ V9.7 或更高版本
- 您已对 DB2® 数据库配置冗余设置或将 DB2® 服务器故障转移至备用节点的功能。

可对 DB2® 使用客户机重新路由以提供有关备用服务器的信息，以防与主数据库服务器的连接失败。

如果客户端没有任何配置，那么 DB2® 的 Java™ 通用连接 (JCC) Java™ 数据库连接 (JDBC) 驱动程序支持客户机重新路由功能，只要已在 DB2® 服务器上启用此驱动程序，并且此驱动程序建立与 DB2® 服务器的初始连接。如果 JCC JDBC 驱动程序连接至具有已配置的一个或多个备用服务器的 DB2® 服务器，那么主服务器会将有关备用服务器的信息发送至 JCC JDBC 驱动程序。如果与主服务器的连接失败，那么 JCC JDBC 驱动程序能够将连接重新路由至备用服务器。但是，如果应用程序服务器进程崩溃，那么备用服务器信息丢失，并且客户机需要再次连接至主服务器。如果客户机无法进行与主服务器的初始连接，那么客户机不知道备用服务器并且无法重新路由。

要解决此问题，您可使用 clientRerouteAlternateServerName 和 clientRerouteAlternatePortNumber 数据源属性对应用程序服务器中的 DB2® 数据源配置备用服务器的名称和端口，以支持客户机重新路由，即使进行初始连接尝试时也是如此。如果 JCC JDBC 驱动程序无法连接至主 DB2® 服务器，但客户机重新路由所需的信息已存在，那么 JCC JDBC 驱动程序可将连接重新路由至备用服务器。

如果连接已重新路由并且 JCC JDBC 驱动程序已连接至备用 DB2® 服务器，那么备用服务器将有关它自己的备用服务器的信息发送至 JCC JDBC 驱动程序。于是 JCC JDBC 驱动程序具有在备用 DB2® 服务器不可用时再次重新路由连接所需的信息。现在，最初作为备用服务器的服务器将作为主服务器，并且建立新的备用服务器。但是，JCC JDBC 驱动程序不再保留主服务器和备用服务器的这一新状态。如果应用程序服务器失效并且已重新启动，那么 JCC JDBC 驱动程序必须从原始服务器配置启动，并尝试连接至最初被视为主服务器的服务器。

1. 在 server.xml 文件中使用以下属性定义 DB2® 数据源：
   - `clientRerouteAlternateServerName`
     - 有效值类型：
       - 域名；例如，www.Vsettan.com
       - IP 地址（IPv4 和 IPv6）；例如：23.72.11.219
     - 多值格式：
       - 用逗号分隔；例如：host1, host2, host3
       - 用空格分隔；例如，host1 host2 host3
     - 顺序意义：
       - 所提供主机名的顺序是 JCC JDBC 驱动程序尝试查找下一个要连接至的可用服务器时使用的顺序。
   - `clientRerouteAlternatePortNumber`
     - 有效值类型：
       - 表示端口号的整数；例如：50000
     - 多值格式：
       - 用逗号分隔；例如：port1, port2, port3
       - 用空格分隔；例如，port1 port2 port3
     - 顺序意义：
       - 所提供端口的顺序必须与其关联服务器的顺序匹配。
2. （可选）可添加下列属性中的一个或全部：
   - `retryIntervalForClientReroute`

     此属性定义 JCC JDBC 驱动程序在每次尝试建立连接之间等待的秒数。

     如果未分配值，那么使用缺省行为。
   - `maxRetriesForClientReroute`

     此属性定义 JCC JDBC 驱动程序重试多少次与服务器的连接才决定移至下一个服务器。仅当设置 `RetryIntervalForClientReroute` 属性后，才会使用此属性。

     如果未分配值，那么使用缺省行为。

```
<dataSource id="DefaultDataSource"jndiName="jdbc/db2">
     <properties.db2.jcc
        databaseName="sampleDatabase"
        driverType="4"
        serverName="host"
        portNumber="50000"
```

```
        clientRerouteAlternateServerName="host01, host02, host03"
        clientRerouteAlternatePortNumber="50000, 50005, 50000"
        retryIntervalForClientReroute="2"
        maxRetriesForClientReroute="3" />
 ...
</dataSource>
```

注：确保对端口和主机指定了相同数目的条目数。否则，系统将显示警告，并且不会启用客户机重新路由功能。

为数据库连接配置连接池

可以通过为数据源定义连接管理器来为数据源配置连接池。

以下示例代码使用 server.xml 文件中的 connectionManager 元素来为数据源定义连接池：

```
<dataSource id="DefaultDataSource" jndiName="jdbc/example" jdbcDriverRef="DB2" >
  <connectionManager maxPoolSize="10" minPoolSize="2" />
 <properties.db2.jcc databaseName="TESTDB"/>
</dataSource>
```

服务器对连接管理器元素上未定义的任何连接管理设置使用缺省值。如果根本没有为数据源定义连接管理器，那么服务器会对所有设置使用缺省值。

如果对连接使用线程本地存储器，那么可以提升多线程系统上应用程序的性能。请参阅调整 *xigemaAS*（见第 1633 页）。

可以定义多个数据源，并将每个数据源与不同的连接管理器相关联。然而，不能将多个数据源与单个连接管理器相关联。

如何应用连接池配置更新

如果在服务器处于运行状态时更改 connectionManager 元素的属性，那么会在不同时间，以不同的方式应用不同属性的更新。

要配置连接池，请在 server.xml 配置文件中指定 connectionManager 元素的属性。如果更改运行中服务器的这些属性，那么会在不同时间，以不同的方式应用更新，取决于更改的属性。下表针对 connectionManager 元素的每个属性，描述了如何在运行时应用配置更改。

表 22: 如何在运行时应用连接管理器配置更新

表的第 1 列列出了 connectionManager 元素的属性。第 2 列针对每个属性，描述了如何在运行时应用配置更新。

属性名称	如何应用配置更新
agedTimeout	更新会立即生效。
connectionTimeout	更新会立即生效。
maxIdleTime	更新会立即生效。
maxNumberOfMCsAllowableInThread	更新会立即生效。
maxPoolSize	更新会立即生效。
minPoolSize	更新会立即生效。

属性名称	如何应用配置更新
numConnectionsPerThreadLocal	更新会立即生效。
reapTime	更新会立即生效。
purgePolicy	更新会立即生效。

👉 注：属性 agedTimeout 和 maxIdleTime 会立即加以更新。但是，它们不会全部被使用，除非 reapTime 属性的值大于零。

因为连接管理器的更新会立即生效，所以如果更改活动连接，那么可能会发生错误；包括可能提前结束连接的潜在风险。

在 xigemaAS 中配置 MongoDB 连接

使用 MongoDB 的应用程序可以在 xigemaAS 上运行。应用程序使用您为服务器配置的 MongoDB Java™ 驱动程序和数据源来访问 MongoDB 实例。

👉 注：xigemaAS 为 MongoDB 提供配置支持。MongoDB（有着“humongous”的含义，表示巨大、庞大）是可伸缩的高性能开放式源代码 NoSQL 数据库。

仅 MongoDB Java 驱动程序 V2.10.0 到 V2.12.5 受支持。

要使应用程序能使用 MongoDB，您为 MongoDB Java™ 驱动程序配置共享库，并且在 xigemaAS 的 server.xml 文件中配置对该共享库的库引用。应用程序可以直接从应用程序或通过 server.xml 文件中的 `mongodb-2.0` 功能部件和 `mongoDB` 实例配置来访问 MongoDB。

1. 在应用程序和 xigemaAS 运行时可以访问的位置安装 MongoDB Java™ 驱动程序。

   例如，将 MongoDB 驱动程序 .jar 文件放入 *xigemaas_profile_root*/usr/servers/*server_name*/lib 目录。
2. 在 xigemaAS 服务器的 server.xml 文件中配置 MongoDB 驱动程序 .jar 文件的共享库。

```
<library id="MongoLib">
   <file name="${server.config.dir}/lib/mongo.jar" />
</library>
```

3. 使应用程序能够访问 MongoDB（从应用程序直接访问或者使用 `mongodb-2.0` 功能部件）。
   - 启用从应用程序直接访问 MongoDB。
     1. 在 server.xml 文件的应用程序元素中配置共享库的库引用。

```
<application ...>
   <classloader commonLibraryRef="MongoLib"/>
</application>
```

        现在，应用程序可以直接访问 MongoDB API。如果您希望应用程序使用运行时注入引擎，请继续执行后续步骤。
   - 在 server.xml 文件中配置 `mongodb-2.0` 功能部件、mongo 和 mongoDB 元素。
     1. 将 `mongodb-2.0` 功能部件添加到 server.xml 文件。

```
<featureManager>
   <feature>mongodb-2.0</feature>
   <feature>jndi-1.0</feature>
</featureManager>
```

        仅当您使用 JNDI 来查找资源时，才需要 JNDI 功能部件。如果您使用资源注入，那么不需要 JNDI 功能部件。

2. 配置一个具有对前一步骤所创建共享库的引用的 mongo 元素。

```
<mongo id="mongo" libraryRef="MongoLib" />
```

3. 配置 MongoDB 元素。

```
<mongoDB jndiName="mongo/testdb" mongoRef="mongo" databaseName="db-test" />
```

配置 JNDI 名称将使应用程序或 xigemaAS 运行时能够查找 MongoDB 实例。

4. 使应用程序能够访问 MongoDB。

以下示例说明了 JNDI 查找和资源注入：

```
public class TestServlet extends HttpServlet {
      @Resource(name = "mongo/testdb")
      protected DB db;
      ...
   protected void doGet(HttpServletRequest request,
         HttpServletResponse response) throws ServletException, IOException {
   // Alternatively use InitialContext lookup
      DB lookup = (DB) new InitialContext().lookup("java:comp/env/mongo/testdb");
...
```

5. 如果您正在使用 JNDI 查找功能，请将资源环境引用添加到应用程序的 web.xml 文件：

```
<resource-env-ref>
   <resource-env-ref-name>mongo/testdb</resource-env-ref-name>
   <resource-env-ref-type>com.mongodb.DB</resource-env-ref-type>
</resource-env-ref>
```

测试从应用程序使用 MongoDB 的情况。

在 xigemaAS 配置安全 MongoDB 连接

可以在 xigemaAS 中为 MongoDB 连接配置应用程序管理的安全性或容器管理的安全性。

启用应用程序来使用 MongoDB。请参阅创建使用 *MongoDB* 的 *xigemaAS* 应用程序。

可以使用应用程序管理的安全性或容器管理的安全性来保护 MongoDB 应用程序。对于这两种安全性，必须在显式启用认证来保护 MongoDB 连接的情况下运行 MongoDB 服务器。

- 为 MongoDB 配置应用程序管理的安全性。

  如果 mongo 配置元素未指定用户和密码属性，那么产品会假定应用程序使用应用程序管理的安全性或者未使用安全性。应用程序必须使用 MongoDB API 进行认证，才能启用应用程序管理的安全性；例如：

```
<mongo id="mongo1" libraryRef="MongoLib" />
<mongoDB jndiName="mongo/testdb" mongoRef="mongo1" databaseName="db-test-1"/>

{ ...
// Java snippet
@Resource(name = "mongo/testdb")
 protected DB db;

  private void auth() {
    if (!db.isAuthenticated())
      db.authenticate("user", "password".toCharArray());
 }
```

- 为 MongoDB 配置容器管理的安全性。

  mongo 配置元素必须指定用户和密码，才能使用容器管理的安全性。每个 mongo 配置只接受一个用户。所有 MongoDB 实例都使用指定的用户和密码。例如，以下示例中引用 mongo1 的所有 MongoDB 实例都使用 mongoUserName 和 pw：

```
<mongo id="mongo1" libraryRef="MongoLib" user="mongoUserName" password="pw"/>
<mongoDB jndiName="mongo/testdb" mongoRef="mongo1" databaseName="db-test-1"/>
```

```
<mongoDB jndiName="mongo/testdb2" mongoRef="mongo1" databaseName="db-test-2"/>
```

使用容器管理的安全性的应用程序不得调用 `com.mongodb.DB.authenticate(user, pass)`。

确保 MongoDB 服务器处于运行状态，然后从应用程序测试 MongoDB 安全性。

连接至分布式 MongoDB 实例集

访问存储在分布式 MongoDB 实例集中的数据与连接至单个 MongoDB 实例似乎是相同的过程。

启用应用程序来使用 MongoDB。请参阅创建使用 *MongoDB* 的 *xigemaAS* 应用程序。

当您在 server.xml 文件中配置 mongo 功能部件时，可以传递一系列作为副本集成员或者共享 mongo 服务器的 `hostNames` 和端口。

如果 `host:port` 组合是副本集成员，那么客户机将查找所有成员，缺省情况下将使用主成员。如果这些组合是共享 mongo 服务器，那么客户机会将所有请求发送至 ping 时间最短的最接近的成员。如果最接近的成员已关闭，那么客户机会自动故障转移到下一个服务器。

- 在 server.xml 文件中配置 `hostNames` 和端口。

```
<mongo id="mongo1" libraryRef="MongoLib" hostNames="localhost,localhost,localhost" ports="9991,9992,9993"/
>
```

您已配置共享 MongoDB 配置。

在 xigemaAS 中配置 Redis DB 连接

Redis 数据库是一个开源的、可基于内存也可持久化的 Key-Value 数据库，其工作效率远远高于其他关系型数据库。xigemaAS 提供对 redis 单节点和集群的数据源支持，用于实现和 redis 服务器的交互。应用程序可以通过 JNDI 查找或资源注入两种方式来访问 redis DB。

xigemaAS 对 redis 数据源的支持包括：

- 支持同时配置单机和集群两种方式；
- 支持同时配置多个 redis 数据源（JNDI 名称不能重复）；
- 支持多个数据源引用同一个连接池配置；
- 支持动态修改 redis 配置。

有关 redis 单节点和集群服务器搭建及配置的详细信息，请参阅 redis 官网 *redis.io*。

配置单节点 redis 请参阅安装单机版 *Redis*。配置 redis 集群的详细步骤请参阅使用 *create-cluster* 命令快速创建 *Redis* 集群及使用 *redis-trib.rb* 命令搭建集群。

要使应用程序能使用 redis 数据库，您需要为 redis Java™ 客户端驱动程序配置共享库，并且在 xigemaAS 概要文件的 server.xml 中配置对该共享库的库引用。应用程序可以直接从应用程序或通过 server.xml 文件中的 **`redisdb-1.0`** 功能部件和 reidsDB 实例配置来访问 redis DB。

1. 在应用程序和 xigemaAS 运行时可以访问的路径下安装 redis Java™ 客户端驱动程序。

   例：将 redis 驱动程序（jedis-2.8.0.jar 和 commons-pool2-2.0.jar）文件放入 ${shared.resource.dir}/lib 目录。

   
   重要：必须使用 Jedis 2.8 版本或更高版本！

2. 在 server.xml 文件中配置共享库。

```
<library id="redisLib">
    <file name="${shared.resource.dir}/lib/jedis-2.8.0.jar"/>
    <file name="${shared.resource.dir}/lib/commons-pool2-2.0.jar"/>
</library>
```

3. 在 <application> 元素下添加子元素 <classloader> 以引用已配置的共享库，使应用程序能够访问 redis DB。 例如:

```
<application context-root="test" type="war" location="redistest.war"
 name="redisdb">
    <classloader commonLibraryRef="redisLib" />
</application>
```

⚠ 警告：在对应用程序进行打包时，请勿将编译时用到的 **redis** 客户端文件（**jedis*.jar**）打包至导出的 **WAR** 包，否则会导致类型转换失败，发生运行时错误。

现在，应用程序已可以直接访问 Jedis API。如果您希望应用程序使用运行时注入引擎，请继续执行后续步骤。

4. 在 server.xml 文件中配置 `redisdb-1.0` 功能部件，以及 redisPool 和 redisDB 元素。

   a. 在 server.xml 配置文件中的 `<featureManager>` 元素内添加 `<feature>redisdb-1.0</feature>`。

```
<featureManager>
    <feature>redisdb-1.0</feature>
</featureManager>
```

   b. 配置一个引用了所创建共享库的 redis 数据源连接池（redisPool 元素）。

```
<redisPool id="poolId" libraryRef="redisLib" maxIdle="8"/>
```

   c. 在 <redisDB> 元素内配置数据源的 JNDI 名称、类型以及单节点可配置的密码等信息。类型（type）用来指定是单节点（standAlone）还是集群（cluster），默认值为 standAlone。 例如：

👉 提示：配置 JNDI 名称使应用程序或 xigemaAS 运行时能够查找到 redisDB 实例。

单节点配置示例：

```
<redisDB id="redisDBTest" jndiName="redis/test" type="standAlone"
 redisPoolRef="poolId" password="abc">
    <node id="1" host="172.16.173.22" port="6379"/>
</redisDB>
```

👉 注：若将 redis 配置为单节点（type="standAlone"），但配置了多个节点信息（<node.../>），那么此数据源将不会生效。

集群配置示例如下：

```
<redisDB jndiName="redisCluster/test" type="cluster"
 redisPoolRef="poolId">
    <node host = "172.16.150.236" port="7000"/>
    <node host = "172.16.150.236" port="7001"/>
    <node host = "172.16.150.236" port="7002"/>
    <node host = "172.16.150.237" port="7003"/>
    <node host = "172.16.150.237" port="7004"/>
    <node host = "172.16.150.237" port="7005"/>
</redisDB>
```

5. 单节点或集群 redis 配置成功后，应用代码就可以通过配置的 *jndiName* 访问 redisDB 了。以下示例说明了应用程序中的 JNDI 查询和资源注入：

```
public class TestServlet extends HttpServlet {
@Resource(name = "redis/test")
protected JedisPool db;

@Resource(name = "redisCluster/test")
protected JedisCluster cluster;
...
protected void doGet(HttpServletRequest request,HttpServletResponse
 response)
          throws ServletException, IOException {
// Alternatively use InitialContext lookup
JedisPool lookup = (JedisPool) new InitialContext().lookup("redis/test");
JedisCluster lookup = (JedisCluster) new
 InitialContext().lookup("redisCluster/test")
```

6. 如果您正在使用 JNDI 查询功能，请将资源环境引用添加到应用程序的 web.xml 文件。例如：

```
<resource-env-ref>
    <resource-env-ref-name>redis/test</resource-env-ref-name>
    <resource-env-ref-type>redis.clients.jedis.JedisPool</resource-env-
ref-type>
</resource-env-ref>
```

现在，您已成功配置 redis 数据源，可以测试从应用程序使用 redis DB了。

在 xigemaAS 中使用 Ektorp 客户机库配置 CouchDB 连接

在 xigemaAS 上运行的应用程序可以使用 CouchDB。要访问 CouchDB 实例，应用程序可以使用 Ektorp 客户机库为 NoSQL 数据库配置连接器。

xigemaAS 提供对 CouchDB 的配置支持。CouchDB 是可缩放的高性能开放式源代码 NoSQL 数据库。

```
<dependency>
    <groupId>org.ektorp</groupId>
    <artifactId>org.ektorp</artifactId>
    <version>1.4.1</version>
</dependency>
```

为允许应用程序使用 CouchDB，必须为 CouchDB Java™ 驱动程序配置共享库，并在 server.xml 文件中配置对共享库的库引用。应用程序可以直接从应用程序或通过 server.xml 文件中的 `couchdb-1.0` 功能部件和 CouchDB 实例配置来访问 CouchDB。

1. 在应用程序和 xigemaAS 运行时可访问的位置安装 CouchDB Java™ 驱动程序。

   例如，将 Ektorp 驱动程序文件及其依赖性放置在 xigemaAS_profile_root/usr/servers/server_name/lib 目录中。

2. 在 xigemaAS 服务器的 server.xml 文件中为 Ektorp 驱动程序文件配置共享库。

```
<library id="couchdb-lib">
    <fileset
    dir='${server.config.dir}/lib'
    includes='org.ektorp-1.4.1.jar
    commons-codec-1.6.jar
    commons-io-2.0.1.jar
```

```
    commons-logging-1.1.1.jar
    httpclient-4.2.5.jar
    httpclient-cache-4.2.5.jar
    httpcore-4.2.4.jar
    jackson-annotations-2.2.2.jar
    jackson-core-2.2.2.jar
    jackson-databind-2.2.2.jar
    slf4j-api-1.6.4.jar
    slf4j-simple-1.6.4.jar'/>
</library>
```

3. 使应用程序能够访问 CouchDB（从应用程序直接访问或者使用 `couchdb-1.0` 功能部件）。
   - 允许从应用程序直接访问 CouchDB。
     1. 在 server.xml 文件的应用程序元素中配置共享库的库引用。

        ```
        <application ...>
           <classloader commonLibraryRef="couchdb-lib"/>
        </application>
        ```

        现在，应用程序可以直接访问 CouchDB API。如果您希望应用程序使用运行时注入引擎，请继续执行后续步骤。
   - 在 server.xml 文件中配置 `couchdb-1.0` 功能部件及 couchdb 元素。
     1. 将 `couchdb-1.0` 功能部件添加至 server.xml 文件。

        ```
        <featureManager>
           <feature>couchdb-1.0</feature>
           <feature>jndi-1.0</feature>
        </featureManager>
        ```

        仅当您使用 JNDI 来查找资源时，才需要 JNDI 功能部件。如果您使用资源注入，那么不需要此功能部件。
     2. 配置具有对上一步骤中所创建共享库的引用的 couchdb 元素。

        ```
        <couchdb id="couchdb" jndiName="couchdb/connector"
              libraryRef="couchdb-lib" url="http://example.com:5984"
         username="username"
              password="password"/>
        ```

        配置 JNDI 名称将使应用程序或 xigemaAS 运行时能够查找 CouchDB 实例。
     3. 允许应用程序访问 CouchDB。

        以下示例说明了 JNDI 查找和资源注入：

        ```
        public class TestServlet extends HttpServlet {
              @Resource(name = "couchdb/connector")
              protected CouchDbInstance db;
              ...
           protected void doGet(HttpServletRequest request,
                 HttpServletResponse response) throws ServletException,
         IOException {
           // Alternatively use InitialContext lookup
              CouchDbInstance lookup = (CouchDbInstance) new
        InitialContext().lookup("java:comp/env/couchdb/connector");
        ...
        ```

4. 如果您正在使用 JNDI 查找功能，请将资源环境引用添加到应用程序的 web.xml 文件：

```
<resource-env-ref>
   <resource-env-ref-name>couchdb/connector</resource-env-ref-name>
   <resource-env-ref-type>org.ektorp.CouchDbInstance</resource-env-
ref-type>
</resource-env-ref>
```

可使用 couchdb-1.0 功能部署配置与在线 Cloudant® 服务的连接。在 couchdb 配置元素中指定现有 Cloudant® 帐户的 URL、用户标识和密码。例如：

```
<couchdb id='couchdb' jndiName='couchdb/connector' libraryRef='couchdb-
lib' url='https://mylink.cloudant.com/' username='myusername'
 password='mypassword'/>
```

请参阅通过密码加密进行保护时存在的限制（见第 1063 页）的文档以了解如何保护配置文件中的密码。

现在，您已配置应用程序以允许使用 CouchDB，您已准备好测试从应用程序使用 CouchDB。

在 xigemaAS 概要文件上管理 Web 应用程序

xigemaAS 概要文件向使用 xigemaAS 功能部件（例如 servlet-3.0、servlet-3.1 和 jsp-2.2）和其他功能部件的 Web 应用程序提供支持。

指定载入并初始化 servlet 的时间

缺省情况下，xigemaAS 会延迟 servlet 载入，直至收到对相关联 Web 应用程序的请求。要覆盖此缺省行为，可以将 Web 容器 deferServletLoad 属性指定为 false。

servlet 规范定义 load-on-startup servlet 属性，该属性是在 Web 应用程序的 web.xml 文件中加以指定。如果 servlet 的 load-on-startup 属性具有非负值，那么部署 Web 应用程序时必须载入并初始化 servlet。xigemaAS 概要文件会优化服务器启动时间和内存使用，方法是直至收到对 Web 应用程序的请求之后才启动 servlet。可以覆盖此延期，以便在安装 Web 应用程序时载入和初始化 servlet，而不是等待直至收到对该应用程序的第一个请求。

要配置服务器以在安装 Web 应用程序时载入 servlet，请将以下行添加到 server.xml 配置文件或随附的文件：

```
<webContainer deferServletLoad="false"/>
```

此设置适用于服务器中安装的所有 Web 应用程序。

配置 xigemaAS 以使用 Servlet 3.1

可配置 xigemaAS 以使用 Servlet 3.1 功能部件，它提供对 Servlet 3.1 规范的全面支持。

要配置 xigemaAS 服务器以运行已启用 Servlet 3.1 的应用程序，必须设置 <servlet-3.1> 功能部件。

1. 更新 server.xml 文件以添加 <servlet-3.1> 功能部件。

   例如：

```
<featureManager>
 <feature>servlet-3.1</feature>
</featureManager>
```

重要：

- `websocket-1.0` 和 `websocket-1.1` 功能部件需要 `servlet-3.1` 功能部件，因此，配置 `websocket-1.0` 或 `websocket-1.1` 功能部件会导致装入 `servlet-3.1` 功能部件。
- 可将 Java™ EE 6 功能部件（例如，`jsp-2.2` 和 `jsf-2.0`）与 `servlet-3.1` 功能部件配合使用。但是，不能使用 Java™ EE 6 功能部件来使用 Servlet 3.1 功能部件。
- 可针对每个服务器实例在 Servlet 3.0 与 Servlet 3.1 功能部件实现之间进行选择，但必须考虑所有行为更改。如果所需行为仅包含在 Servlet 3.1 功能部件中，那么您必须使用 Servlet 3.1 功能部件。如果 Servlet 3.1 功能部件中的行为更改对现有应用程序有负面影响，那么应使用 Servlet 3.0 功能部件以保留该应用程序的现有行为。
- 不能在同一 xigemaAS 概要文件服务器中同时使用 Servlet 3.0 和 Servlet 3.1 功能部件。如果同时配置两个功能部件，那么将产生错误。阅读 Servlet 3.1 行为更改主题以了解有关 Servlet 3.0 和 Servlet 3.1 中的更改。

Servlet-3.1 功能部件已启用，并在运行时装入到 xigemaAS 概要文件服务器中。

Servlet 3.1 行为更改

Servlet 3.1 实现包含的行为更改可能导致为 Servlet 3.0 编写的应用程序在您使用 Servlet 3.1 功能部件时的行为不同或失败。

可针对每个服务器实例在 Servlet 3.0 与 Servlet 3.1 功能部件实现之间进行选择并考虑行为更改。如果所需行为仅包含在 Servlet 3.1 功能部件中，那么您必须使用 Servlet 3.1 功能部件。如果 Servlet 3.1 功能部件中的行为更改对现有应用程序有负面影响，那么应使用 Servlet 3.0 功能部件以保留该应用程序的现有行为。不能在同一服务器中同时使用 Servlet 3.0 和 Servlet 3.1 功能部件。同时配置这两个功能部件是错误行为。如果您同时配置这两个功能部件，那么系统不会装入任一 servlet 功能部件。

引入行为更改出于以下三个原因：

- Servlet 3.1 规范中的说明需要的更改。
- 为让 Servlet 3.1 实现传递 Servlet 3.1 Technology Compatibility Kit (TCK) 而需要的更改。
- 用于改进 servlet 实现的更改。

以编程方式添加的 servlet、过滤器和侦听器

Servlet 3.1 规范中的说明现在指出：如果未在 `web.xml` 文件或 `web-fragment.xml` 文件中声明 ServletContextListener 或未使用 @WebListener 注释 ServletContextListener，那么 ServletContextListener 以编程方式配置 servlet、过滤器或侦听器是非法操作。因此，为执行这类程序配置而对 ServletContext 进行的任何调用会导致 UnsupportedOperationException。

异步处理启动后转发

在 Servlet 3.0 实现中，响应始终在 RequestDispatcher 接口的转发方法返回前关闭。但是，因为 Servlet 3.1 规范中的说明，如果请求置于异步方式，那么 Servlet 3.1 实现不会在 RequestDispatcher 接口的转发方法返回前关闭或清仓响应。此更改可能影响现有 3.0 应用程序，这些应用程序在从转发返回时添加响应输出，因为这类响应数据现在已发送，但在 Servlet 3.0 中这类响应数据未发送。

URL 模式冲突

在 Servlet 3.0 中，即使 URL 模式映射至多个 servlet，应用程序也应成功启动。但是，因为 Servlet 3.1 规范中的说明，该应用程序一定无法成功启动。在 xigemaAS Servlet 3.1 实现中，消息为输出，应用程序启动失败：

```
SRVE9016E: 无法对名为 [{1}] 的 servlet 插入映射 [{0}]。已对名为 [{2}] 的
 servlet 定义该 URL 模式。

说明：发生了应用程序错误。servlet 映射 URL 模式不应映射至多个 servlet。

用户操作：更改 servlet 映射的 URL 模式。
```

ServletContext.getMinorVersion()

在 Servlet 3.0 功能部件实现中，此 API 返回 `0`。

在 Servlet 3.1 功能部件中，此 API 现在返回 `1`。

ServletContext.getServerInfo()

在 Servlet 3.0 功能部件实现中，此 API 返回 `SMF WebContainer`。

在 Servlet 3.1 功能部件中，此 API 现在返回 `xigemaAS/9.0.0.`*<x>*，其中 *<x>* 是 xigemaAS 修订包编号。

ServletResponse.reset()

如果响应未落实，那么您可使用 ServletResponse.reset() 清除所有缓存响应数据、状态码和响应头。如果正在使用 Servlet 3.1 功能部件，那么此方法还会清除先前调用的 ServletResonse.getWriter() 或 ServletResponse.getOutputStream() 的任何记录。

X-Powered-By 头

在 Servlet 3.0 功能部件实现中，X-Powered-By 头设置为 `Servlet/3.0`。在 Servlet 3.1 功能部件实现中，X-Powered-By 头设置为 `Servlet/3.1`。

资源引用注入目标合并

在 Servlet 3.0 规范中，如果同名 `web.xml` 资源引用定义没有 `<injection-target>` 元素，那么 `web-fragment.xml` 文件中定义的资源引用的 `<injection-target>` 元素仅添加至父 `web.xml` 文件。在 Servlet 3.1 规范中，已说明 `web-fragment.xml` 描述符中的所有`<injection-target>` 元素将添加至同名资源引用的 `<injection-target>` 元素的父 `web.xml` 描述符列表。如果正在使用 Servlet 3.1 功能部件，那么此功能部件可能通过激活先前在 `web.xml` 文件中排除的注入目标来更改现有应用程序功能。

Web 描述符中对重复元素的容忍度

在 Servlet 3.1 规范中，已说明 `web.xml` 文件不能包含两个 `<absolute-ordering>` 元素。部署带有多个 `<absolute-ordering>` 元素的应用程序将失败。此外，`web-fragment.xml` 描述符不能包含两个 `<ordering>` 元素。部署带有多个 `<ordering>` 元素的应用程序将失败。以前此部署不会失败，但元素的功能可能无法确定。

metadata-complete 用例中的 Web 片段排序更改

如果 `web.xml` 描述符标记为 `metadata-complete="true"`，那么 `<absolute-ordering>` 元素的处理会更改。以前在 `metadata-complete="true"` 用例中，会使用所有 Web 片段归档。使用 Servlet-3.1 功能部件时，metadata-complete 用例中的 `<absolute-ordering>` 元素被视为完整。此更改导致处理时排除 `<absolute-ordering>` 元素中未列示的片段。

AsyncContext.dispatch()

如果正在使用 AsyncContext.dispatch()，例如，不带任何参数，那么请求被分派至原始 URL。如果正在使用 Servlet-3.0 功能部件，那么原始请求未附带查询字符串时，该查询字符串会提供给所分派资源。但是，如果正在使用 Servlet 3.1 功能部件，那么查询字符串提供给分派资源时，该查询字符串会提供给所分派资源。例如：

```
Request for /FirstResource?param=One
First Resource:
    getParameter("param") returns "One"
    forward request to /SecondResource?param=Two
SecondResource
     getParameter(param) returns "Two"
     ac.start()
     ac.dispacth() dispatches to /FirstResource
First Resource
     Servlet-3.0 feature : getParamter("param") returns "One"
     Servlet-3.1 feature : getParameter("param") returns "Two"

This change was required by the Servlet 3.1 TCK.
```

在 AsyncContext.dispatch() 或 AsyncContext.complete() 后获取请求或响应对象是不允许的，将导致抛出以下异常：

```
java.lang.IllegalStateException: SRVE9015E: Cannot obtain the
 request or response object after an AsyncContext.dispatch() or
 AsyncContext.complete().
    at
 com.ibm.ws.webcontainer31.async.AsyncContext31Impl.getRequest(AsyncContext31Impl.java:
    [...]
```

SessionCookieConfig.setComment()

根据 Java™ Servlet 3.1 规范，如果在 ServletContext 完成初始化后调用此 API，那么此 API 返回 illegalStateException，并且 Servlet 3.1 功能部件遵循此必需行为。但是，初始化上下文后，Servlet 3.0 功能部件不会阻止使用此 API，因此，依赖于 Servlet 3.0 功能部件行为的应用程序不会与 Servlet 3.1 功能部件配合工作。

sendRedirect(java.lang.String location) API

`sendRedirect(java.lang.String location)` API 接受相对 URL；但是，servlet 容器必须先将相对 URL 转换为绝对 URL，才能将响应发送至客户机。如果该位置是相对位置且没有前导"/" (`folder/default.jsp`)，那么该容器将其解释为相对于当前请求 URI。如果该位置是相对位置且带有前导"/"，那么容器会将其解释为相对于 servlet 容器根。

例如，如果应用程序提供的重定向位置为 folder/default.jsp 且没有前导"/"，并且入站请求 URL 为 http://host:port/context_root/folder 或 http://host:port/context_root/folder/，那么该请求将重定向至 http://host:port/context_root/folder/folder/default.jsp，它相对于当前请求 URI。

com.ibm.ws.webcontainer.redirectwithpathinfo 属性设置为 **true** 时，会在 Servlet 3.0 功能部件中发现此行为。此属性在 Servlet 3.1 功能部件中被忽略，按描述所言，此行为是缺省行为。

缺省错误页

xigemaAS 扩展功能允许指定带有 Web 扩展的缺省错误页，例如，ibm-web-ext.xml。

作为 Servlet 3.0 及更高版本的功能，缺省错误页是对指定错误页功能的修改。与常规（非缺省）错误页一样，缺省错误页是在 Web 模块描述符 (web.xml) 和 Web 片段描述符 (web-fragment.xml) 中指定的。

常规（非缺省）错误页指定 exception-type 或 error-code。缺省错误页省略 exception-type 和 error-code。如果 servlet 抛出异常或设置错误代码结果，并且没有与异常类型或错误代码相匹配的已配置错误页，那么系统使用缺省错误页。

Servlet 3.0 规范允许定义缺省错误页，Servlet 3.0 模式也支持此功能。根据 Servlet 3.1 规范，缺省错误页不包含 exception-type 或 error-code 元素。

错误页和缺省错误页的示例如下所示。

缺省错误页优先顺序规则

确定 web.xml、web-fragment.xml 和 ibm-web-ext.xml 文件中的缺省错误页的优先顺序时，有三个规则适用。

- 规则 1：web.xml 和 web-fragment.xml 文件。

  如果在 web.xml 文件中指定了缺省错误页，那么该错误页会覆盖（屏蔽）web-fragment.xml 文件中指定的任何缺省错误页。而且，如果除此之外还有多个 web-fragment.xml 文件指定了缺省错误页，也不会发生任何错误。
- 规则 2：web-fragment.xml 和 web-fragment.xml。

  如果未在 web.xml 文件中指定缺省错误页，并且两个或更多 web-fragment.xml 文件指定了不同缺省错误页，那么会发生错误。
- 规则 3：ibm-web-ext.xml 和 web.xml 或 web-fragment.xml 文件。

  ibm-web-ext.xml 文件与 web.xml 或 web-fragment.xml 文件之间的优先顺序规则依赖于 Web 容器功能部件级别。

如果 Web 容器功能部件级别为 3.0，那么 ibm-web-ext.xml 文件定义的缺省错误页优先于 web.xml 或 web-fragment.xml 文件中定义的缺省错误页。

👉 注：如果 Web 容器使用功能部件级别 3.0，那么您不能使用 Servlet 3.1 模式。请参阅有关对 servlet 3.0 模式使用缺省错误页的规则。

如果 Web 容器功能部件级别为 3.1 或更高，那么 web.xml 或 web-fragment.xml 文件指定的缺省错误页优先于 ibm-web-ext.xml 文件中指定的缺省错误页。

模式规则

确定缺省错误页是在 web.xml 文件中还是在 web-fragment.xml 文件中处理时，有两个规则适用。这些规则依赖于 Web 容器功能部件版本、所使用的 servlet 模式以及 Java™ 定制属性的设置。

- 规则 1：使用 Servlet 3.0 模式和使用 Web 容器功能部件 V3.0 的缺省错误页。

  如果 Web 容器功能部件版本为 3.0 且在使用 Servlet 3.0 模式的 `web.xml` 或 `web-fragment.xml` 文件中指定了缺省错误页，那么仅当 `com.ibm.ws.webcontainer.allowdefaulterrorpage` Java™ 系统属性设置为 **`true`** 时，才会处理这些缺省错误页。如果未设置 Java™ 系统属性或此属性未设置为 **`true`**，那么系统会忽略缺省错误页。将使用通过 `ibm-web-ext.xml` 文件指定的缺省错误页。
- 用例 2：使用 Web 容器功能部件 V3.1 的缺省错误页。

  如果 Web 容器功能部件版本为 3.1 或更高，那么始终处理 `web.xml` 或 `web-fragment.xml` 文件中指定的缺省错误页，无论使用何种版本的 servlet 模式以及不管是否设置 Java™ 定制属性都是如此。

  如果描述符使用 Servlet 3.1 模式，那么会出现这种情况，因为使用 Servlet 3.1 模式的描述符的处理需要 Web 容器功能部件 V3.1。

注意：直到 Servlet 3.0 才添加了 Web 片段。`web-fragment.xml` 文件没有来自 Servlet 2.5 的模式。

错误页和缺省错误页示例

`ibm-web-ext.xml` 文件中定义的缺省错误页：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">

    <default-error-page uri="/ExtErrorPage.html"/>
</web-ext>
```

error-code 错误页元素，它是在 `web.xml` 文件或 `web-fragment.xml` 文件中定义的：

```
<error-page>
 <error-code>404</error-code>
 <location>/ErrorCodeErrorPage.html</location>
</error-page>
```

exception-type 错误页元素，它是在 `web.xml` 文件或 `web-fragment.xml` 文件中定义的：

```
<error-page>
 <exception-type>javax.servlet.ServletException</exception-type>
 <location>/ExceptionTypeErrorPage.html</location>
</error-page>
```

缺省错误页元素，它是在 `web.xml` 文件或 `web-fragment.xml` 文件中定义的：

```
<error-page>
 <location>/DefaultErrorPage.html</location>
</error-page>
```

模式示例

使用 Servlet 2.5 模式的 web.xml 文件头示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.su
n.com/xml/ns/javaee/web-app_2_5.xsd"
      version="2.5">
```

使用 Servlet 3.0 模式的 web.xml 文件头示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.su
n.com/xml/ns/javaee/web-app_3_0.xsd"
      version="3.0">
```

使用 Servlet 3.1 模式的 web.xml 文件头示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jc
p.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
```

使用 Servlet 3.0 模式的 web-fragment.xml 文件头示例：

```
<?xml version="1.0" encoding="utf-8"?>
<web-fragment xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.su
n.com/xml/ns/javaee/web-fragment_3_0.xsd"
      version="3.0">
```

使用 Servlet 3.1 模式的 web-fragment.xml 文件头示例：

```
<?xml version="1.0" encoding="utf-8"?>
<web-fragment xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.
jcp.org/xml/ns/javaee/web-fragment_3_1.xsd"
      version="3.1">
```

Servlet 3.1 功能部件的功能

本产品支持部分 Servlet 3.1 功能。查看一些提供的功能的说明和描述。

将在规范中描述新的 Servlet 3.1 功能，此处未进行描述。但是，Servlet 3.1 功能部件的附加注意事项如下所示：

异步 I/O

Servlet 3.1 的新功能部件指定非阻塞读启动时，余下请求生命周期中的任何资源不能调用 API，它可能导致阻塞读。例如，对于资源设置读侦听器后的 POST 请求，针对 getParameter() 和 getPart() API 的任何后续调用将导致 IllegalStateException。

使用异步 servlet 时，必须考虑使用 AsyncContext.setTimeout API 设置超时，否则系统使用容器缺省值（例如，30 秒）。此超时在每次异步开始使用 ServletRequest 时重置。StartAsync API 被调用，并因为在上次异步启动后的超时时间段内未调用 AsyncContext.complete API 而到期。使用 Servlet 3.1 功能部件提供的异步 I/O 支持时，也应使用 AsyncContext.setTimeout API 设置超时值以允许异步 I/O 完成。完成取决于其他外部因素，例如，环境或网速。

升级处理

> 重要：请将 ServletOutputStream 类与 WriteListener 接口配合使用，并将 ServletInputStream 类与 ReadListener 接口配合使用。请勿将这些类与 ObjectInputStream 类或 ObjectOutputStream 类配合使用。这些类会绕开 ReadListener 和 WriteListener 接口的一些必需检查（主要是 isReady 检查），并且可能会导致意外的行为。

升级处理针对具有非阻塞读写功能的 Servlet 3.1 功能部件执行。非阻塞读或写操作异步执行时，对服务器等待操作完成的时间没有限制。可使用 `server.xml` 文件中的 Web 容器定制属性（例如，upgradereadtimeout 和 upgradewritetimeout）设置超时。请参阅超时为 5 秒的以下示例：

```
<webContainer upgradeReadTimeout="5000" />
<webContainer upgradeWriteTimeout="5000" />
```

异步 servlet 处理请求时，不得使用 Servlet 3.1 的升级功能来升级该请求。

支持 Servlet 3.1 功能部件以进行升级的应用程序要求请求上的连接在客户机与托管该升级的应用程序之间保持打开。如果升级处理完成时，该应用程序未从其处理程序或任何其他资源（例如，ReadListener 或 WriteListener）启动 WebConnection close()，那么 TCP 连接保持打开，直到服务器重新启动。

如果使用 Servlet 3.1 功能部件中的 UpgradeHandler 和 ReadListener，那么仅当客户机关闭与托管所升级应用程序的服务器的连接时，才会调用 ReadListener.onAllDataRead 方法。针对 onReadListener.onAllDataRead 的 Javadoc™ 返回以下消息：

读取当前请求的所有数据时已调用。

在升级情况下，服务器不知道数据结尾，因为所升级数据不是按定界 HTTP 请求主体的方式定界的。除客户机连接关闭时以外，无法确定数据结尾。

基于表单的认证

成功认证后，客户机将重定向至原始请求的资源。Servlet 3.1 规范指定：“为提高所重定向请求的 HTTP 方法的可预测性，容器应使用 303 (SC_SEE_OTHER) 状态码重定向，需要与 HTTP 1.0 用户代理进行互操作的情况（在此情况下，应使用 302 状态码）除外。”Servlet-3.1 功能部件保留与 HTTP 1.0 用户代理的互操作性并始终使用 302 状态码。有关配置 Servlet 3.1 以获取安全性的更多信息，请阅读“配置 xigemaAS 以使用 Servlet 3.1”主题。

大型发布数据

添加 ServletRequest.getContentLengthLong() API 要求支持接收长度超过 Integer.MAX_VALUE 并且无法被单字节数组或字符串完全容纳的发布数据。

获取的发布数据内容使用的 API 以字符串或 byte[] 形式返回内容时，此添加会有影响。例如，用于访问参数的 javax.servlet.ServletRequest 方法：

```
String    getParamter(String name)
String[]  getParameterValues()
Map<String,String> getParameterMap()
```

可能发生以下情况，您发送包含多个参数的发布数据，这些参数组合到一起时长度超过 Integer.MAX_VALUE。但是，每个参数名和参数值的长度必须小于 Integer.MAX_VALUE。

发送大量发布数据包括以下附加注意事项：

- 必须以长度小于 Integer.MAX-VALUE 的块的形式发送发布数据。
- 处理开始之前，必须完整读取 Web 容器处理的发布数据（例如，参数或部件）。大型发布数据可能导致大量内存需求，因为它可能需要发布数据大小的双倍内存，以便 Web 容器处理成功。

配置 xigemaAS 以使用 Expression Language 3.0

可配置 xigemaAS 以使用 Expression Language (EL) 3.0 功能部件，它提供对 EL 3.0 规范的全面支持。

要配置xigemaAS服务器以运行已启用 EL 3.0 的应用程序，必须设置 `<el-3.0>` 功能部件。

1. 更新 `server.xml` 文件以添加 `<el-3.0>` 功能部件。

   例如：

   ```
   <featureManager>
    <feature>el-3.0</feature>
   </featureManager>
   ```

   重要：

   - EL 3.0 功能部件不需要任何附加功能部件。可独立于 JavaServer Pages (JSP) 2.3 配置该功能部件。
   - JSP 2.3 功能部件需要 EL 3.0 功能部件。配置 JSP 2.3 功能部件时，EL 3.0 功能部件还会装入至服务器运行时。
   - 可将其他 Java™ EE 6 功能部件（例如，JSF 2.0 和 CDI 1.0）与 EL 3.0 功能部件配合使用。
   - 可针对每个服务器实例在 EL 3.0 与 EL 2.2（包含在 JSP 2.2 功能部件中）之间进行选择，但必须考虑所有行为更改。如果所需行为仅包含在 EL 3.0 功能部件中，那么您必须使用 EL 3.0 功能部件。如果 EL 3.0 功能部件中的行为更改对现有应用程序有负面影响，那么应使用 EL 2.2 功能部件（包含在 JSP 2.2 中）以保留该应用程序的现有行为。
   - 不能在同一 xigemaAS 服务器中同时使用 EL 3.0 功能部件和 JSP 2.2 功能部件（包含在 JSP 2.2 中）。如果同时配置两个功能部件，那么将产生错误：

     ```
     CWWKF0033E: 单例功能部件
     com.ibm.websphere.appserver.javax.el-2.2 和
      com.ibm.websphere.appserver.javax.el-3.0 不能同时装入。所配置功能部件
      jsp-2.2 和 el-3.0 包含导致冲突的一个或多个功能部件。
     ```

   请参阅*Expression Language (EL) 3.0 功能部件的功能*（见第 1220 页）以了解 EL 3.0 功能部件中的更改（与 EL 2.2 功能部件相比）。

EL 3.0 功能部件已启用，并在运行时装入到 xigemaAS 概要文件服务器中。

Expression Language (EL) 3.0 功能部件的功能

Expression Language (EL) 3.0 功能部件提供对 EL 3.0 规范的全面支持。

EL 3.0 规范中提供了对 EL 3.0 功能的描述，此处未做完整描述。但是，下面列示了一些关键增强功能：

- EL 3.0 现在可作为单独功能部件提供，您可独立于 JavaServer Pages (JSP) 2.3 对它进行配置。
- 添加了对 Lambda 表达式的支持（带有参数的值表达式）。有关更多信息，请参阅 EL 3.0 规范的 1.20 节。
- 添加了针对集合对象的操作。有关更多信息，请参阅 EL 3.0 规范的 2.0 章。
- 新增运算符：
  - 字符串并置。有关更多信息，请参阅 EL 3.0 规范的 1.8 节。
  - 赋值。有关更多信息，请参阅 EL 3.0 规范的 1.13 节。
  - 分号。有关更多信息，请参阅 EL 3.0 规范的 1.14 节。
  - 字段和方法。有关更多信息，请参阅 EL 3.0 规范的 1.22 节。

👉 重要：EL 3.0 功能部件中存在的更改可能损坏现有应用程序。强制 NULL 变为非基本类型（字符串除外）的缺省操作返回 NULL。例如，一个强制变为双精度值的空值现在返回空值，之前返回 0.0。以下代码示例描述此场景：

```
Integer number=null;
factory.coerceToType(number, java.lang.Double.class)
```

配置 xigemaAS 以使用 JavaServer Faces 2.2

可配置 xigemaAS 概要文件以使用 JavaServer Faces (JSF) 2.2 功能部件，它提供对 JSF 2.2 规范的全面支持。

xigemaAS JSF 实现基于 MyFaces 开放式源代码实现。要配置 xigemaAS 服务器以运行已启用 JSF 2.2 的应用程序，必须设置 `<jsf-2.2>` 功能部件。

1. 更新 `server.xml` 文件以添加 `<jsf-2.2>` 功能部件。

   例如：

   ```
   <featureManager>
    <feature>jsf-2.2</feature>
   </featureManager>
   ```

   👉 重要：

   使用 JavaServer Faces 2.2 时，应考虑以下几点：

   - 与 JSF 2.0 功能部件不同，JSF 2.2 功能部件未隐式装入 Bean Validation 功能部件。如果将应用程序从 JSF 2.0 迁移至 JSF 2.2，并且应用程序使用 bean 验证，那么您还必须启用 `beanValidation-1.1` 功能部件。
   - JSF 2.2 功能部件需要 `servlet-3.1`、`jsp-2.3`、`timedexit-1.0` 和 `el-3.0` 功能部件。在 `server.xml` 文件中启用 JSF 2.2 功能部件时，也会启用其中每个功能部件。
   - 不能同时运行 JSF 2.2 功能部件和 Java EE 6 功能部件；例如，`servlet-3.0`、`jsp-2.2` 和 `cdi-1.0` 功能部件。
   - 可针对每个服务器实例在 JSF 2.0 与 JSF 2.2 功能部件实现之间进行选择，但必须考虑所有行为更改。如果所需行为仅包含在 JSF 2.2 功能部件中，那么您必须使用 JSF 2.2 功能部件。如果 JSF 2.2 功能部件中的行为更改对现有应用程序有负面影响，那么应使用 JSF 2.0 功能部件以保留该应用程序的现有行为。

- 不能在同一 xigemaAS 服务器中同时使用 JSF 2.0 和 JSF 2.2 功能部件。如果同时配置两个功能部件，那么将产生错误：

  ```
  CWWKF0033E:
  不能同时装入单例功能部件 jsf-2.0 和 jsf-2.2。所配置功能部件 jsf-2.0 和
   jsf-2.2 包含的一个或多个功能部件导致该冲突。您的配置不受支持；请更新
   server.xml 以移除不兼容的功能部件。
  ```

- JSF 2.2 与之前发行版（例如，JSF 2.1 和 JSF 2.0）兼容；但是，应考虑以下例外：
  - 先前版本的 JSF 规范中导致异常被允许的错误现在传播至异常处理程序。请阅读 JSF 2.2 规范的概述章节中的“与先前版本的向后兼容性”。
  - 规范中对组合组件属性 ELResolver 和组合计算机元数据的更改。请阅读 JSF 2.2 规范的概述章节中的“与先前版本的向后兼容性”。

JSF 2.2 功能部件已启用，并在运行时装入到 xigemaAS 概要文件服务器中。

为确保 CDI 正确管理该类，请将生产者方法类定义为受管 bean（通过指定范围）或将 CDI bean-discovery-mode 设置为 `all`。可在 Web 归档的 `beans.xml` 文件中将 CDI bean-discovery-mode 设置为 `all`：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
 bean-discovery-mode="all">
 </beans>
```

JavaServer Faces 2.2 功能部件的功能

JavaServer Faces (JSF) 2.2 功能部件提供对 JSF 2.2 规范的全面支持。

JSF 2.2 规范中提供了对 JSF 2.2 功能的描述，此处未做完整描述。但是，下面列示了一些关键增强功能：

- Faces Flows 允许开发者按逻辑组合 JSF 视图以表示功能模块。每个模块都有一组严格定义的入口点和出口点。组合模块会创建功能流。功能流的示例是订单结帐流程。请阅读 JSF 2.2 规范的 7.5 节。
- 资源库合同允许资源库驻留在 web-app 根目录的 contracts 目录中，或 JAR 文件的 META-INF/contracts 条目名中。请阅读 JSF 2.2 规范的 10.1.3 节。
- JSF 2.2 允许应用程序使用无状态视图。请阅读 JSF 2.2 规范的 7.8.1.1 节。
- JSF 2.2 可通过传递元素和传递属性来处理 HTML5 属性。请阅读 JSF 2.2 规范的 10.1.4 节。

配置 xigemaAS 以使用 JavaServer Pages 2.3

可配置 xigemaAS 概要文件以使用 JavaServer Pages (JSP) 2.3 功能部件，它提供对 JSP 2.3 规范的全面支持。

要配置 xigemaAS 服务器以运行已启用 JSP 2.3 的应用程序，必须设置 `<jsp-2.3>` 功能部件。

1. 更新 `server.xml` 文件以添加 `<jsp-2.3>` 功能部件。

   例如：

   ```
   <featureManager>
    <feature>jsp-2.3</feature>
   </featureManager>
   ```

   👉 重要：

- `jsp-2.3` 功能部件需要 `servlet-3.1` 和 `el-3.0` 功能部件，配置这些功能部件后会导致装入这些功能部件。
- 不能将 `jsp-2.3` 与 `servlet-3.0` 功能部件配合使用。
- 可将其他 Java™ EE 6 功能部件（例如，JSF 2.0 和 CDI 1.0）与 JSP 2.3 功能部件配合使用。
- 可针对每个服务器实例在 JSP 2.2 与 JSP 2.3 功能部件实现之间进行选择，但必须考虑所有行为更改。如果所需行为仅包含在 JSP 2.3 功能部件中，那么您必须使用 JSP 2.3 功能部件。如果 JSP 2.3 功能部件中的行为更改对现有应用程序有负面影响，那么应使用 JSP 2.2 功能部件以保留该应用程序的现有行为。
- 不能在同一 xigemaAS 服务器中同时使用 JSP 2.2 和 JSP 2.3 功能部件。如果同时配置两个功能部件，那么将产生错误：

```
CWWKF0033E: 单例功能部件
jsp-2.3 和 jsp-2.2 不能同时装入。所配置功能部件 jsp-2.3 和 jsp-2.2 包含导致冲突的一个或多个功能部件。
```

请参阅*JavaServer Pages 2.3 功能部件的功能*（见第 1222 页）以了解 JSP 2.3 功能部件中的更改。

JSP 2.3 功能部件已启用，并在运行时装入到 xigemaAS 概要文件服务器中。

JavaServer Pages 2.3 功能部件的功能

JavaServer Pages (JSP) 2.3 功能部件提供对 JSP 2.3 规范的全面支持。

新 JSP 2.3 功能的描述是在 JSP 2.3 规范中提供的，此处未描述。但是，与 JSP 2.2 功能部件相比，新规范的添加内容不多：

- 对 Expression Language (EL) 3.0 的支持
- 对 JSP 提供 Servlet 3.1 API。
- JSP 2.2 与 JSP 2.3（直接阻止使用 JSP 2.2 运行的 JSP 成功使用 JSP 2.3）之间没有已知行为差异。但是，如果 JSP 使用 Expression Language (EL) 或 Servlet API 功能，那么必须考虑 Servlet 3.1 与 Servlet 3.0 之间的变化以及 EL 3.0 与 EL 2.2 之间的变化。

在 xigemaAS 上管理上下文和依赖性注入应用程序

xigemaAS 通过使用 xigemaAS 功能部件 `cdi-1.0` 和 `cdi-1.2` 在应用程序中提供对上下文和依赖性注入的支持。

配置 xigemaAS 以使用上下文和依赖性注入 1.2

可配置 xigemaAS 以使用上下文和依赖性注入 (CDI) 1.2 功能部件，它提供对上下文和依赖性注入 1.2 规范的全面支持。

要配置 xigemaAS 服务器以运行已启用 CDI 1.2 的应用程序，必须设置 `<cdi-1.2>` 功能部件。

1. 更新 server.xml 文件以添加 `<cdi-1.2>` 功能部件。例如：

```
<featureManger>
 <feature>cdi-1.2</feature>
</featureManger>
```

注：

- 可将其他 Java™ EE 7 功能部件（例如，jsp-2.3 和 jsf-2.2）与 cdi-1.2 功能部件配合使用。但是，不能将 Java™ EE 6 功能部件（例如，jsp-2.2 和 jsf-2.0）与 cdi-1.2 功能部件配合使用。

- 可针对每个服务器实例在 CDI 1.0 与 CDI 1.2 功能部件实现之间进行选择，但必须考虑行为更改。如果只有 CDI 1.2 功能部件包含该行为，那么必须使用 CDI 1.2 功能部件。如果 CDI 1.2 功能部件中的行为更改对现有应用程序可能产生负面影响，那么应使用 CDI 1.0 功能部件以保留该应用程序的现有行为。
- 不能在同一 xigemaAS 服务器中同时使用 CDI 1.0 和 CDI 1.2 功能部件。如果同时配置两个功能部件，那么将产生错误。阅读“CDI 1.2 行为更改”主题以了解有关从 CDI 1.0 到 CDI 1.2 的更改。

CDI 1.2 功能部件已启用，并在运行时装入到 xigemaAS 服务器中。

上下文和依赖性注入 1.2 概述

xigemaAS 概要文件通过使用 xigemaAS 功能部件 `cdi-1.0` 和 `cdi-1.2` 在应用程序中提供对上下文和依赖性注入的支持。

Contexts and Dependency Injection (CDI) 1.2 功能部件提供对 CDI 1.2 规范的全面支持。CDI 1.2 规范中提供了 CDI 1.2 功能的完整描述，请参阅用于 *Java™ EE* 平台的上下文和依赖性注入。

CDI 1.2 功能部件提供的服务集合包含严格定义的有状态对象生命周期，这些对象绑定至生命周期上下文和类型安全依赖性注入机制。

将上下文和依赖性注入 1.2 与 JavaServer Faces 应用程序配合使用

可将 CDI 1.2 功能部件与 JavaServer Faces (JSF) 2.2 功能部件配合使用，以允许 JSF 应用程序使用 CDI 1.2 功能部件中提供的精密上下文和依赖性注入模型。此服务是通过与 Unified Expression Language (EL) 的集成提供的，EL 允许直接在 JSF 或 JavaServer Pages (JSP) 页面中使用任何上下文对象。

将上下文和依赖性注入 1.2 与 Enterprise JavaBeans™ (EJB) 配合使用

可将 CDI 1.2 功能部件与 Enterprise JavaBeans™ (EJB) 3.2 功能部件配合使用，以使用上下文生命周期管理来增强 EJB 组件模型。CDI 1.2 功能部件提供的服务将 Java™ EE Web 层与 Java™ EE 企业服务集成到一起。尤其是，这允许 EJB 组件用作 JSF 管理的 bean，从而集成 EJB 和 JSF 的编程模型。

将上下文和依赖性注入 1.2 与 Servlet 3.1 配合使用

可将 CDI 1.2 功能部件与 Servlet 3.1 功能部件配合使用，以允许 servlet 应用程序使用 CDI 1.2 功能部件提供的服务。通过使用这两个功能部件允许上下文管理的 bean 通过使用字段、方法或构造函数注入来插入至 servlet 应用程序。CDI 1.2 功能部件还提供了 servlet 侦听器、过滤器和拦截器的自动注册。

上下文和依赖性注入 1.2 应用程序中的 Java™ 拦截器

CDI 1.2 功能部件针对拦截器扩展了 Java™ 模型。CDI 1.2 功能部件允许将拦截器与 bean 相关联。拦截器是使用类型安全拦截器绑定来绑定的。如果将 CDI 1.2 和 EJB 3.2 功能部件装入至 xigemaAS 服务器，那么此模型可扩展至 EJB bean。

上下文和依赖性注入 1.2 行为更改

上下文和依赖性注入 (CDI) 1.2 实现包含一些行为更改，它们可能导致从 CDI 1.0 迁移的应用程序在 CDI 1.2 上以不同方式运行或失败。

对于每个服务器实例，可在 CDI 1.0 与 CDI 1.2 功能部件实现之间进行选择并考虑行为更改。如果只有 CDI 1.2 功能部件包含所需行为，那么必须使用 CDI 1.2 功能部件。如果 CDI 1.2 功能部件中的行为更改对现有应用程序可能产生负面影响，那么将通过使用 CDI 1.0 功能部件保留该应用程序的现有行为。不能在同一服务

器中同时使用 CDI 1.0 和 CDI 1.2 功能部件，因为这些功能部件不兼容。如果同时配置这两个功能部件，那么服务器会产生配置错误。

CDI 1.0 功能部件基于 CDI 的 Apache OpenWebBeans 实现。CDI 1.2 功能部件基于 CDI 的 Weld 实现。引入行为更改是因为这两个实现存在差别。

对话标识 CID

在 CDI 1.0 实现中，CID 是全局唯一的。在 CDI 1.2 中，对于每个 HTTP 会话，它是唯一的。此行为符合 CDI 规范，并且是 Weld 选择的约定。要获取全局唯一 CID，必须在对话开始时通过调用 `Conversation.begin` 来指定 CID。

在 beans.xml 文件中引用模式

在 CDI 1.2 实现中，以下是 beans.xml 文件中引用的模式的示例：

```
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/
xml/ns/javaee/beans_1_1.xsd"
```

如果您使用无效模式，那么服务器将给出异常错误。可通过设置 `org.jboss.weld.xml.disableValidating=true` JVM 属性来关闭 beans.xml 文件的验证，这也会避免产生该错误。如果 beans.xml 文件指定修饰符或拦截器，那么必须使用有效模式，否则修饰符和拦截器不会正确实例化。

隐式 bean 归档

CDI 1.2 实现定义两种不同类型的 bean 归档：显式和隐式。

显式 bean 归档包含 beans.xml 文件：

- 版本号为 1.1（或更高版本），并且 bean-discovery-mode 为 `all`
- 没有版本号
- 为空文件

隐式 bean 归档是任何其他归档，它包含一个或多个 bean 类（这些类的 bean 定义注解是在规范的 2.5.1 节“bean 定义注解”中定义的）或者一个或多个会话 bean。请参阅以下规范：用于 *Java™ EE* 平台的上下文和依赖性注入。

将模式更新为 CDI 1.2 实现时，为使 bean 归档保持为显式，bean 发现方式必须设置为 `all`：

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
bean-discovery-mode="all"
  version="1.1">
```

👉 注：隐式 bean 归档仅发现具有 bean 定义注解的 bean。

这是一种新的 bean 归档，它可产生一个归档，此归档不会成为 CDI bean 归档，但它在 CDI 1.2 实现中会变为隐式 bean 归档。要停止此行为，可添加将 bean 发现方式设置为 none 的 beans.xml 文件，以阻止归档变为 bean 归档。另一解决方案是将以下属性添加至您的 xigemaAS 概要文件服务器的 server.xml 文件：

```
<cdi12 enableImplicitBeanArchives="false"/>
```

将此属性设置为 false 可阻止不带 beans.xml 文件的归档变为隐式 bean 归档。

在 xigemaAS 上管理会话启动协议 (SIP)

通过在 server.xml 文件中添加并配置元素，可在 xigemaAS 上配置 SIP。

在 xigemaAS 服务器中安装 sipServlet-1.1 功能部件。有关更多信息，请参阅添加和移除 *xigemaAS* 功能部件（见第 1129 页）。

此任务描述如何在 server.xml 文件中为 xigemaAS 手动配置 SIP。

有关您在 xigemaAS 上配置 SIP 时可使用的元素和属性的信息，请参阅*SipServlet-1.1* 的文档。

1. 在 server.xml 文件中，通过添加 sipContainer 元素来配置 SIP 容器。

   有关属性及其描述的完整列表，请参阅*sipContainer* 元素。

   ```
   <sipContainer invalidateSessionOnShutdown="true" msgArrivalTimeAttr="true"
    markInternalResponse="true"></sipContainer>
   ```

2. 通过添加 sipStack 元素来配置 SIP 堆栈。

   可配置自动响应、消息通知及连接复用之类的行为。

   有关属性及其描述的完整列表，请参阅*sipStack* 元素。

   ```
   <sipStack hideMessageBody="true" hideMessageReqUri="true"
       auto100OnInvite="false" auto482OnMergedRequests="true"
       forceConnectionReuse="false" acceptNonUtf8Bytes="true"
       pathMtu="1600" compactHeaders="API">
       <commaSeparatedHeaders>Comma separated headers</commaSeparatedHeaders>
       <hideMessageHeaders>iup</hideMessageHeaders>
       <sipQuotedParameters>Require</sipQuotedParameters>
   </sipStack>
   ```

   SIP 计时器提供了会话到期机制。通过在 sipStack 元素中指定计时器属性，可对 SIP 计时器设置 RFC 3261 中指定的缺省值以外的值。

   ```
   <sipStack timerT1="500" timerT2="4000" timerT4="5000" timerA="500"
    timerB="32000" timerD="32000"
   timerE="500" timerF="32000" timerG="500" timerH="32000" timerI="5000"
    timerJ="32000" timerK="5000"></sipStack>
   ```

3. 通过添加 sipApplicationRouter 元素来配置 SIP 应用程序路由器。

   配置 SIP 应用程序路由器时，您可以使用缺省应用程序路由器 (DAR)，也可以创建定制应用程序路由器。此 DAR 组件使用按 Java™ 属性文件建模的配置文本文件，以定义应用程序路由器向应用程序发送 SIP 请求的顺序。

   如果未配置应用程序路由器并且要手动部署应用程序，那么应用程序将按应用程序的部署顺序触发。如果通过 server.xml 文件部署多个应用程序，那么不能定义应用程序顺序。

- 要将 DAR 组件与 DAR 配置文件配合使用，请在 `sipApplicationRouter` 元素的 `sipDarConfiguration` 属性上指定 DAR .properties 文件的位置。

```
<sipApplicationRouter sipDarConfiguration="pathToDar.properties"></
sipApplicationRouter>
```

- 要使用定制应用程序路由器，请在 server.xml 文件中配置路由器：

1. 将 `bells-1.0` 功能部件添加至 server.xml 文件。

```
<featureManager>
     <feature>bells-1.0</feature>
</featureManager>
```

有关更多信息，请参阅*bells-1.0* 的文档。

2. 在 server.xml 文件中，将定制应用程序路由器 JAR 文件添加为共享库，并对 `javax.servlet.sip.ar.spi.SipApplicationRouterProvider` 服务添加 bell 配置。

```
<library id="carLib" name="carLib" description="Custom App Router
 Library">
     <file name="pathToJar.jar"></file>
</library>
<bell libraryRef="carLib">
 <service>javax.servlet.sip.ar.spi.SipApplicationRouterProvider</
service>
</bell>
```

有关共享库的更多信息，请参阅共享库（见第 1603 页）。

3. 在 `sipApplicationRouter` 元素上，将 `carProvider` 属性设置为定制应用程序路由器提供程序类的全名。

```
<sipApplicationRouter carProvider="example.example">></
sipApplicationRouter>
```

通过将 `sipNoRouteErrorCode` 属性添加至 `sipApplicationRouter` 元素，您可指定没有活动 servlet 可映射至入局初始请求时 SIP 容器发送的错误响应代码。

```
<sipApplicationRouter carProvider="example.example"
 sipNoRouteErrorCode="403"></sipApplicationRouter>
```

有关属性及其描述的完整列表，请参阅*sipApplicationRouter 元素*。

4. 通过添加 `sipEndpoint` 元素来配置 SIP 端点。

可配置定制端口、主机和绑定行为。您配置的任何 SIP 端点将覆盖缺省端点。

有关属性及其描述的完整列表，请参阅*sipEndpoint 元素*。

```
<sipEndpoint host="localhost" sipTCPPort="5060" bindRetries="60"
 bindRetriesDelay="5000"></sipEndpoint>
```

注：缺省情况下将禁用传输层安全性 (TLS)。要对 SIP 容器启用 TLS，请对 xigemaAS 服务器启用并配置 SSL 通信。有关更多信息，请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）。

5. 通过添加 `sipTasksDispatcher` 元素来配置 SIP 性能。

可配置能够并行运行的 SIP 任务数。SIP 应用程序任务是一个应用程序代码序列，它在 SIP 容器的单个线程上串行执行。例如，`SipServlet.doInvite()` 方法或 `SipTimerListener.timeout()` 方法的应用程序实现均被视为单个任务，只要从这些方法调用的应用程序代码在容器提供的受管 xigemaAS 线程上串行执行。实际的最大并行 SIP 任务数受最大受管 xigemaAS 线程数限制。

有关属性及其描述的完整列表，请参阅*sipTasksDispatcher 元素*。

```
<sipContainer>
     <sipTasksDispatcher concurrentContainerTasks="15"/>
</sipContainer>
```

6. 如果要使用域名系统 (DNS) 命名权限指针 (NAPTR) 记录解析 SIP URI，请配置域解析器。

   a. 添加并配置 `domainResolver` 元素。

   如果配置此元素，那么 SIP URI 通过 DNS 解析为要联系的下一个中继段的 IP 地址、端口和传输协议。

   👉 注：SIP 不支持在主客户机失效时使用服务器的 DNS 过程向备份客户机发送响应。

   有关属性及其描述的完整列表，请参阅*domainResolver 元素*。

```
<domainResolver dnsAutoResolve="true">
     <dnsServers>dns.server.com</dnsServers>
</domainResolver>
```

   b. 配置 DNS 服务器以使用 SIP 容器。

   以下示例是 BIND 数据库文件，用于在 DNS 服务器上配置 RFC 3263 支持。

```
; Copyright (C) 2004  Internet Systems Consortium, Inc. ("ISC")
; Copyright (C) 2001  Internet Software Consortium.
;
; Permission to use, copy, modify, and distribute this software for any
; purpose with or without fee is hereby granted, provided that the above
; copyright notice and this permission notice appear in all copies.
;
; THE SOFTWARE IS PROVIDED "AS IS" AND ISC DISCLAIMS ALL WARRANTIES WITH
; REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
; AND FITNESS.  IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT,
; INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
; LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE
; OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
; PERFORMANCE OF THIS SOFTWARE.

; $Id: include.db,v 1.2.206.1 2004/03/06 10:22:13 marka Exp $

; Test $INCLUDE current domain name and origin semantics

example.com. 43200 IN SOA ns.example.com. email.example.com. ( 2003032001 10800 3600
 604800 86400 )
;
example.com.             43200 IN NS      ns.example.com.
;
ns.example.com.          43200 IN A       10.0.0.20
sipserver1.example.com.  43200 IN A       10.0.0.21
sipserver2.example.com.  43200 IN A       10.0.0.22
sipserver3.example.com.  43200 IN A       10.0.0.23
;
router.example.com.        43200  IN CNAME sipserver3
;
sipserver1.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abcd
sipserver2.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abba
;
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver1.example.com.
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver2.example.com.
_sip._tcp.example.com.   43200 IN SRV 1 4 5060  sipserver1.example.com.
```

```
_sip._tcp.example.com.   43200 IN SRV 1 2 5060  sipserver2.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 1 5061  sipserver1.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 0 5061  sipserver2.example.com.
;
example.com. 43200 IN NAPTR 0 0 "s" "SIPS+D2T" "" _sips._tcp.example.com.
example.com. 43200 IN NAPTR 1 0 "s" "SIP+D2T"  "" _sip._tcp.example.com.
example.com. 43200 IN NAPTR 2 0 "s" "SIP+D2U"  "" _sip._udp.example.com.
```

有关在 SIP 应用程序中执行 DNS 查找的信息，请参阅在 *xigemaAS* 上的会话启动协议 *(SIP)* 应用程序中执行 *DNS* 查找（见第 1490 页）。

在 xigemaAS 上管理 JavaMail

通过在 server.xml 文件中添加并配置元素，可在 xigemaAS 上配置 JavaMail™。

如果有外部邮件服务器，那么可使用 JavaMail™ API 在运行于 xigemaAS 服务器上的应用程序中发送和接收电子邮件。API 通过提供公共存储器和传输协议（例如，POP3、IMAP 和 SMTP）以允许应用程序与外部邮件服务器交互。

xigemaAS 支持 JavaMail™ 1.5。有关 JavaMail™ 1.5 的更多信息，请参阅*JavaMail™ API 文档*。

有关您可用于在 xigemaAS 上配置 JavaMail™ 的元素和属性的信息，请参阅*JavaMail 1.5*（见第 995 页）。

1. 在 server.xml 文件中，添加 `javaMail-1.5` 功能部件。

   添加该功能部件后，可在运行于服务器上的任何应用程序中调用 JavaMail™ 库。

```
<featureManager>
        <feature>javaMail-1.5</feature>
</featureManager>
```

2. 可选： 如果要创建 javax.mail.Session 对象，添加并配置 `mailSession` 元素。

   配置邮件会话后，通过使用 Java™ 命名和目录接口 (JNDI) 创建并插入会话。

   注：如果使用标准 JNDI 上下文 `java:comp/env/mail/exampleMailSession`，将 `jndiName` 属性配置为 `jndiName="mail/exampleMailSession"`。

```
<mailSession  mailSessionID="examplePop3MailSession"
  jndiName="ExampleApp/POP3Servlet/exampleMailSession"
  description="POP3 javax.mail.Session"
  storeProtocol="pop3"
  transportProtocol="smtp"
  host="exampleserver.com"
  user="iamanexample@example.com"
  password="example"
  from="smtp@testserver.com">
  <property name="mail.pop3.host" value="pop3.example.com" />
  <property name="mail.pop3.port" value="3110" />
</mailSession>
```

2.4.2 配置更新

更新 server.xml 文件

可以从命令行更新 server.xml 文件。如果配置有任何问题，那么具有问题的特定配置元素更新不起作用，但其他成功的更新会加以实现。

要更改此行为，可以将 onError 变量从缺省值 WARN 更新为 FAIL。如果值设为 FAIL，那么任何配置更新问题都会导致整个更新失败。

如果更新 server.xml 文件中包含的任何文件或 configDropins 目录中的配置，那么会进行更新。

使用已更新的配置来重新启动服务器

xigemaAS 运行时环境会高速缓存当前使用的配置，以便在您重新启动服务器时，不会处理 server.xml 文件，除非发生了更改。如果更改了 server.xml 文件，那么会更新已高速缓存的配置以使用新值。如果在新配置中发现任何问题，那么仍使用已高速缓存的配置值。控制台日志会显示仍在使用旧值的每个配置元素的警告消息。例如：

```
CWWKG0076W: The previous configuration for httpEndpoint with id defaultHttpEndpoint is still in
 use.
```

可通过运行带有 --clean 选项的服务器脚本，在不使用已高速缓存配置的情况下，启动服务器。

2.4.3 在 server.xml 文件中添加来自外部 XML 文件的配置信息

可使用 include 元素以在 server.xml 文件中添加来自外部 XML 文件的配置信息。

如果外部 XML 文件中有配置信息，那么您可使用 include 元素以在 server.xml 文件中添加配置信息。例如，如果您有 XML 文件 simpleSecurity.xml，且该文件具有以下内容：

```
<server>
 <quickStartSecurity userPassword="thePassword"/>
</server>
```

您可在 server.xml 文件中使用以下方法以在 simpleSecurity.xml 文件中添加配置信息：

```
<server>
 <featureManager>
     <feature>servlet-3.0</feature>
 </featureManager>
 <quickStartSecurity userName="theUser"/>
 <include location="simpleSecurity.xml"/>
</server>
```

生效配置如下所示：

```
<server>
 <featureManager>
     <feature>servlet-3.0</feature>
 </featureManager>
 <quickStartSecurity userName="theUser"/>
 <quickStartSecurity userPassword="thePassword"/>
</server>
```

冲突处理

可在 server.xml 文件中配置 *onConflict* 属性以处理 server.xml 文件与外部文件之间的值冲突。此属性可配置为下列三个值中的一个：*Merge*、*Replace* 和 *Ignore*。

Merge

这些值将合并到一起。*Merge* 是 *onConflict* 属性的缺省值，*Merge* 相当于您获取的行为（如果您在 server.xml 文件中指定所有冲突元素）。在以上示例中，有两个 `quickStartSecurity` 元素，它们将有效合并为单个元素。生效配置如下所示：

```
<quickStartSecurity userName="theUser" userPassword="thePassword"/>
```

有关配置元素如何合并的更多信息，请参阅配置元素合并规则（见第 1230 页）。

Replace

所包含配置文件中的值将替换 server.xml 文件中的冲突值。在先前示例中，所包含的 `quickStartSecurity` 元素替换 server.xml 文件中的相应元素，所以生效配置如下所示：

```
<quickStartSecurity userPassword="thePassword"/>
```

Ignore

所包含文件中的值被忽略。在先前示例中，所包含文件中的 `quickStartSecurity` 元素被忽略，所以生效配置如下所示：

```
<quickStartSecurity userName="theUser"/>
```

2.4.4 配置元素合并规则

如果在服务器配置中多次指定某个配置元素，那么这些元素将合并。以下规则适用于配置合并：

- 单例元素始终合并。在以下示例中，服务器配置中该元素的所有实例为 `featureManager`，它们将合并为单个 `featureManager` 元素：

  ```
  <featureManager>
      <feature>servlet-3.0</feature>
  </featureManager>
  <featureManager>
      <feature>jdbc-4.0</feature>
  </featureManager>
  ```

  生效配置变为：

  ```
  <featureManager>
      <feature>servlet-3.0</feature>
      <feature>jdbc-4.0</feature>
  </featureManager>
  ```

- 如果在服务器配置顶级指定的出厂元素具有相同标识，那么它们将合并。在以下示例中，带有 `id="ds1"` 的 `dataSource` 元素将合并，带有 `id= "ds2"` 的 `dataSource` 元素将保持原样。例如，

  ```
  <dataSource id="ds1" jdbcDriverRef="myDriver"/>
  <dataSource id="ds1"  jndiName="jdbc/myDriver"/>
  <dataSource id="ds2" jdbcDriverRef="myDriver2"/>
  ```

生效配置变为：

```
<dataSource id="ds1" jdbcDriverRef="myDriver" jndiName="jdbc/myDriver"/>
<dataSource id="ds2" jdbcDriverDref="myDriver2"/>
```

- 如果出厂元素没有标识值，那么它被视为与没有标识值的相同类型的其他元素不同。没有标识值的多个出厂元素不会合并到一起。在以下示例中，dataSource 不会合并，所以生效配置与指定配置相同：

```
<dataSource jdbcDriverRef="myDriver"/>
<dataSource jndiName="jdbc/myDriver"/>
```

生效配置变为：

```
<dataSource jdbcDriverRef="myDriver"/>
<dataSource jndiName="jdbc/myDriver"/>
```

- 如果要合并的元素有冲突属性，那么所合并元素使用配置解析器遇到的最后一个值。在以下示例中，带有 id= "ds1" 的 dataSource 元素将合并，系统会使用 jdbcDriverRef="myDriver2" 而移除 jdbcDriverRef="myDriver"。

```
<dataSource id="ds1" jdbcDriverRef="myDriver"/>
<dataSource id="ds1" jdbcDriverRef="myDriver2"/>
```

生效配置变为：

```
<dataSource id="ds1" jdbcDriverRef="myDriver2"/>
```

- 如果出厂元素嵌套在另一元素下，那么它仅与同一生效父代下的其他元素合并。在以下示例中，带有 id= "ds1" 的 dataSource 元素将合并，properties.derby.embedded id="props1" 元素与父代也为 dataSource id="ds1" 的其他 properties.derby.embedded id="props1" 元素合并。

```
<dataSource id="ds1">
 <properties.derby.embedded id="props1" databaseName="myDB"/>
</dataSource>
<dataSource id="ds2">
 <properties.derby.embedded id="props1" user="myUser"/>
</dataSource>
<dataSource id="ds1">
 <properties.derby.embedded id="props1" createDatabase="create"/>
</dataSource>
```

生效配置变为：

```
<dataSource id="ds1">
 <properties.derby.embedded id="props1" databaseName="myDB"
 createDatabase="create"/>
</dataSource>
<dataSource id="ds2">
 <properties.derby.embedded id="props1" user="myUser"/>
</dataSource>
```

- 如果出厂元素嵌套在另一元素下，并且该出厂元素没有指定标识值，那么将根据嵌套元素的基数应用特殊规则。如果需要特定类型的多个嵌套元素，那么不会合并这些元素。但是，如果只需要一个嵌套元素，那么这些嵌套元素将合并到一起。例如，

```
<topLevel>
 <multipleNested enabled="true"/>
 <multipleNested value="1"/>
 <singleNested enabled="false"/>
 <singleNested value="2"/>
</topLevel>
```

生效配置变为：

```
<topLevel>
 <multipleNested enabled="true"/>
 <multipleNested value="1"/>
 <singleNested enabled="false" value="2"/>
</topLevel>
```

- 如果两个出厂元素嵌套在另一元素下，并且它们的标识值不匹配，那么合并行为取决于嵌套元素的基数。如果需要多个嵌套元素，那么不会合并这些元素。如果只需要一个嵌套元素，那么这些嵌套元素将合并到一起，尽管它们的标识值有冲突。例如，

```
<topLevel>
 <multipleNested id="1" enabled="true"/>
 <multipleNested id="2" value="1"/>
 <singleNested id="3" enabled="false"/>
 <singleNested id="4" value="2"/>
</topLevel>
```

生效配置变为：

```
<topLevel>
 <multipleNested id="1" enabled="true"/>
 <multipleNested id="2" value="1"/>
 <singleNested id="4" enabled="false" value="2"/>
</topLevel>
```

2.5 扩展 xigemaAS

可以通过使用产品扩展来扩展xigemaAS 概要文件的功能。可以编写您自己的 xigemaAS 功能部件，并将其安装到现有 xigemaAS 服务器，也可以将它们打包以交付给用户。

本节描述如何为产品扩展开发功能部件，如何将功能部件安装到内置“usr”产品扩展以及如何在应用程序服务器中使用这些功能部件。 xigemaAS 提供各种可用来扩展运行时环境的系统编程接口 (SPI)；您也可以使用更多高级功能，例如，以程序化方式从 Java™ 应用程序操作 xigemaAS 服务器。

有关为 xigemaAS 编写产品扩展的概述，请参阅产品扩展（见第 1028 页）。

有关如何扩展 xigemaAS 的完整详细信息，请参阅下列子主题：

2.5.1 为 xigemaAS 开发 xigemaAS 功能部件

xigemaAS 功能部件由一个功能部件清单文件以及一个或多个 OSGi 捆绑软件组成。该 OSGi 捆绑软件包含将功能部件安装到 xigemaAS 概要文件服务器时用于提供特定功能的类和服务。

手动开发功能部件；请参阅手动开发 *xigemaAS* 功能部件（见第 1233 页）。

有关开发 xigemaAS 功能部件的完整详细信息，请参阅下列子主题：

手动开发 xigemaAS 功能部件

可以手动创建 xigemaAS 功能部件并将其安装到 xigemaAS。

一个功能部件可以由单个 OSGi 捆绑软件和一个功能部件清单文件组成。此示例向应用程序提供一个库，以便外部包在缺省应用程序类路径上可见。通过将功能部件清单复制到 ${wlp.user.dir}/extension/lib/features 目录，并将 OSGi 捆绑软件复制到 ${wlp.user.dir}/extension/lib 目录，可以将该功能部件安装到 xigemaAS。然后，可以在 server.xml 文件中使用该功能部件。

有关功能部件清单文件格式的详细信息，请参阅*xigemaAS* 功能部件清单文件（见第 1234 页）。

此示例描述如何手动构造 xigemaAS 功能部件。

1. 要手动创建 xigemaAS 功能部件，请完成以下步骤：
1. 例如，创建 OSGi 捆绑软件（其中包含 Java™ 类以及具有相应 OSGi 头的捆绑软件清单文件），以导出您想要向应用程序公开的 Java™ 包。`Bundle-SymbolicName` 是唯一的必需头；此条目根据反向域名名称约定来指定捆绑软件的唯一标识。较好的实践是指定捆绑软件的版本，而在此示例中，导出了一些 Java™ 包以供应用程序使用：

```
Bundle-SymbolicName: com.usr.samplebundle
Bundle-Version: 1.0.1
Export-Package: com.usr.samplebundle.pkg1; version="1.0.0",
                com.usr.samplebundle.pkg2; version="1.0.1"
```

2. 使用 `jar` 命令来封装 Java™ 类和功能部件清单文件。例如：

```
jar cfm samplebundle.jar MANIFEST.Mf *.class
```

3. 创建功能部件清单文件 *feature-name*.mf，用来向运行时环境描述功能部件。
   a. 提供所需的清单头：
      - `Subsystem-SymbolicName`，用来指定功能部件的标识和可见性；
      - `Subsystem-Content`，用来查找构成功能部件的文件；
      - `IBM-Feature-Version`，用来标识运行时环境所需功能部件支持的版本。
   b. 最佳实践：添加可选清单头以指示子系统规范的适用版本 (`Subsystem-ManifestVersion`)、功能部件版本 (`Subsystem-Version`) 和功能部件短名称 (`IBM-ShortName`)。指定这些值将帮助您在将来对功能部件进行升级。
   c. 在 `IBM-API-Package` 头中，列出要在应用程序的缺省类加载器上提供的包。
   d. 可选： 当您创建 xigemaAS 功能部件时，将它安装到用户产品扩展，然后该功能部件中的包都可以供用户产品扩展中安装的任何其他功能部件访问。要使一个或多个 SPI 包可供其他产品扩展中的功能部件使用，请在 `IBM-SPI-Package` 头中列出这些包。

```
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: com.example.myfeature.sample-1.0; visibility:=public
Subsystem-Version: 1.0.0.qualifier
Subsystem-Type: osgi.subsystem.feature
```

```
Subsystem-Content: samplebundle; version="[1,1.0.100)"
IBM-Feature-Version: 2
IBM-API-Package: com.usr.samplebundle.pkg1; type="api",
                 com.usr.samplebundle.pkg2; type="api"
IBM-SPI-Package: com.sample.myservice.spi;IBM-ShortName: sample-1.0
```

4. 将捆绑软件复制到 ${wlp.user.dir}/extension/lib 目录。
5. 将功能部件清单复制到 ${wlp.user.dir}/extension/lib/features 目录。
6. 如果在功能部件清单文件中定义了 `Subsystem-Name` 和 `Subsystem-Description` 头，并且对值进行了本地化，请将 `Subsytem-Localization` 头中所指定的本地化文件复制到 ${wlp.user.dir}/extension/lib/features/l10n 目录。

将功能部件安装到 xigemaAS 之后，您可以将功能部件名称添加到 server.xml 文件中已配置的功能部件列表。例如：

```
<featureManager>
     <feature>usr:sample-1.0</feature>
</featureManager>
```

xigemaAS 功能部件清单文件

xigemaAS 功能部件包含一个功能部件清单文件及一个或多个 OSGi 捆绑软件的集合，这类捆绑软件提供与 xigemaAS 概要文件运行时环境中的特定功能相对应的类和服务。您可以找到有关功能部件清单格式以及清单文件中每个头的含义的简介。

xigemaAS 概要文件中的功能部件清单文件使用 OSGi 企业 R5 规范中的子系统服务元数据格式。功能部件由存储在 lib/features 目录中的功能部件清单文件（.mf 文件）定义，并且必须使用定制类型的子系统：`osgi.subsystem.feature`。有关 OSGi 清单语法的更多信息，请参阅 OSGi 核心规范第 1.3.2 节。

👉 注：在功能部件清单文件中，属性采用格式 *`name=value`*，但伪指令采用格式 *`name:=value`*。

定义了下列头：

表 23: 功能部件清单文件的头

此表显示 xigemaAS中的功能部件清单文件的头。第一列显示了头列表，第二列显示了每个头的描述，而第三列陈述是否需要头。

头	描述	是否必需
`Subsystem-ManifestVersion`	功能部件清单文件的版本格式。必须设置为 1。	否
`Subsystem-SymbolicName`	功能部件的符号名称，及任何属性或伪指令。	是
`Subsystem-Version`	功能部件的版本。有关如何定义此头的详细信息，请参阅 OSGi 核心规范部分 3.2.5。	否
`Subsystem-Type`	功能部件的子系统类型。所有功能部件当前都是同一种子系统类型：`osgi.subsystem.feature`。	是
`Subsystem-Content`	功能部件的子系统内容。这是运行此功能部件所需的捆绑软件和子系统的逗号分隔	是

头	描述	是否必需
	列表。如果要允许在 server.xml 文件中配置自动功能部件，那么必须具有包含所需功能部件的功能头，并且也必须在子系统内容中定义所需功能部件。	
Subsystem-Localization	功能部件的本地化文件的位置。	否
Subsystem-Name	功能部件的人类可读短名称。此值可以本地化。	否
Subsystem-Description	功能部件的描述。此值可以本地化。	否
IBM-Feature-Version	此子系统类型的版本。必须设置为 2。	是
IBM-Provision-Capability	功能头，用来描述是否可以自动地供应功能部件。	否
IBM-API-Package	由此功能部件、其他产品扩展中的功能部件和 xigemaAS 内核向应用程序提供的 API 包。	否
IBM-API-Service	由此功能部件提供给 OSGi 应用程序的 OSGi 服务。	否
IBM-SPI-Package	由此功能部件向其他产品扩展中的功能部件和 xigemaAS 内核提供的 SPI 包。	否
IBM-ShortName	功能部件的短名称。	否
IBM-AppliesTo	此功能部件所适用于的 xigemaAS 版本。	否
Subsystem-License	此功能部件的许可类型。	否
IBM-License-Agreement	许可协议文件的位置的前缀。	否
IBM-License-Information	许可信息文件的位置的前缀。	否
IBM-App-ForceRestart	指定对运行中服务器安装或卸载功能部件后将重新启动应用程序。	否

Subsystem-SymbolicName

此头的语法与捆绑软件的 Bundle-SymbolicName 语法匹配。它具有遵循包名称样式语法的符号名称，并且可以选择性地接受一组属性和伪指令。

支持下列属性：

- superseded。此属性指示此功能部件是否由功能的一个或多个功能部件或项目取代。它接受下列其中一个值：
  - true - 取代此功能部件。
  - false - 不取代此功能部件。

  此属性是可选的；缺省值为 false。

  有关更多信息，请参阅取代的功能部件。

- superseded-by。此属性指定取代此功能部件的逗号分隔功能部件列表（如果有），可选。

支持以下伪指令：

- visibility。此伪指令采用下列其中一个值：
  - `public` - 将功能部件认为是 API。此功能部件受开发者工具支持，用于 server.xml 文件中，并且在消息中输出。
  - `protected` - 将功能部件认为是 SPI。此功能部件不受开发者工具支持，不用于 server.xml 文件中，也不在消息中输出。提供此功能部件是为了扩展程序可以利用它来构建更高级别的功能部件。
  - `private` -（缺省值）此功能部件是产品的内部功能部件。不支持将此功能部件用于 server.xml 文件中，也不供扩展程序功能部件引用。随时都可以更改此功能部件，其中包括在两个修订包之间更改此功能部件。

例如：

```
Subsystem-SymbolicName: com.ibm.example.feature-1.0;
    visibility:=public; superseded=true; superseded-by="com.ibm.example.feature-2.0"
```

如果 superseded-by 列表中的功能部件名称用方括号 [] 括起来，那么此功能部件已经与取代功能部件分隔开。在以下示例中，功能部件 `appSecurity-1.0`（此功能部件包含功能部件 `servlet-3.0` 和 `ldapRegistry-3.0`）被功能部件 `appSecurity-2.0`（此功能部件未包含功能部件 `servlet-3.0` 和 `ldapRegistry-3.0`）取代：

```
IBM-ShortName: appSecurity-1.0
Subsystem-SymbolicName: com.ibm.websphere.appserver.appSecurity-1.0; visibility:=public;
superseded=true; superseded-by="appSecurity-2.0, [servlet-3.0], [ldapRegistry-3.0]"
```

有关更多信息，请参阅独立的功能部件。

- singleton. 此伪指令采用下列其中一个值：
  - *True*。此功能部件是单例功能部件。
  - *False*。此功能部件不是单例功能部件。

  singleton 伪指令是可选的。缺省值为 *False*。

  此伪指令用于声明特定功能部件是单例。单例意味着一次只能将给定功能部件的一个版本装入至运行时。缺省情况下，功能部件不是单例。如果该功能部件是单例，并且服务器配置需要给定功能部件的多个版本，那么运行时将尝试查找所有必需功能部件容许的通用版本。有关版本容许的更多信息，请参阅 Subsystem-Content 下的 ibm.tolerates 伪指令。

  如果功能部件是单例，那么符号名称值为“<singleton feature name >-<singleton version>”形式，其中 name 和 version 用连字符分隔。单例功能部件名称可包含连字符，但跟在最后一个连字符之后的字符将解释为单例版本。如果跟在最后一个连字符后的字符是无效版本，那么将使用单例版本 0.0.0 并且完整符号名称将用作单例名称。处理 Subsystem-Content 下的 ibm.tolerates 伪指令时将使用单例版本；例如：

  ```
  Subsystem-SymbolicName: com.ibm.example.feature-1.0;
      visibility:=public; singleton:=true
  ```

Subsystem-Content

此头针对运行时和安装定义功能部件的内容。它遵循的头语法和子系统规范相同，语法如下所示：

```
Subsystem-Content ::= content ( ',' content )*
        content ::= unique-name ( ';' parameter )*
```

```
        unique-name ::= unique-name  (see OSGi core spec section 1.3.2)
```

unique-name 使用 Bundle-SymbolicName 或 Subsystem-SymbolicName 头格式。支持下列属性：

- version - 查找捆绑软件时要匹配的版本范围。仅选择此范围中的捆绑软件。版本范围的典型示例是 [1,1.0.100)。
- type - 要供应的内容类型。可以指定任何值来指示内容类型；某些类型将促使在使用该功能部件的服务器的 OSGi 框架中安装并启动捆绑软件，而所有类型都会促使将内容包括在含有该功能部件的安装软件包中。预定义了下列值：
  - osgi.bundle - 这是缺省值，指示一个 OSGi 捆绑软件（应该同时供应给服务器的 OSGi 框架以及安装软件包）。
  - osgi.subsystem.feature - 此值指示应该将功能部件同时供应给服务器的 OSGi 框架以及安装软件包。这些功能部件需要使用 Subsystem-SymbolicName 头中指定的名称。
  - jar - 此值指示安装软件包中应包括 JAR 文件，并且此 JAR 文件是通过使用版本范围和/或位置值的组合选择的。
  - file - 此值指示 location 属性中所标识的文件应该包括在安装软件包中。

以下伪指令受支持：

- location - 捆绑软件的位置。对于捆绑软件或 JAR 类型，此值可以是目录的逗号分隔列表（用于标识 dev 目录中的规范和 API 捆绑软件），也可以是直接指向 JAR 文件的单个条目。如果 type 为 file，那么仅允许此值为单个条目，并且它必须直接指向该文件。

  例如：

```
Subsystem-Content: com.ibm.websphere.appserver.api.basics; version="[1,1.0.100)"; type=jar;
 location:="dev/api/xigemaas/,lib/",
                    com.ibm.websphere.appserver.spi.application;
          location:="dev/spi/xigemaas/com.ibm.websphere.appserver.spi.application_1.0.0.jar"; type="jar",
                    com.ibm.websphere.appserver.spi.application_1.0.0-javadoc.zip;
          location:="dev/spi/xigemaas/javadoc/com.ibm.websphere.appserver.spi.application_1.0.0-
javadoc.zip"; type="file"
```

- start-phase - 在系统启动期间捆绑软件应该启动的开始阶段。start-phase 伪指令可采用下列其中一个值：
  - SERVICE - 此值指示最早阶段。缺省情况下，它映射到开始阶段 9。
  - CONTAINER - 如果未提供任何 start-phase，那么这是缺省值。启动应用程序容器时，它指示容器阶段。缺省情况下，它映射到开始阶段 12。
  - APPLICATION - 启动应用程序时，此值指示最晚的阶段。

  捆绑软件也可以定义为刚好在这些阶段之前或之后启动，方法是添加晚于关键阶段的 _LATE 或者添加早于关键阶段的 _EARLY。因此，如果要在容器阶段之后立即运行，请使用 CONTAINER_LATE；如果要在 APPLICATION 阶段之前运行，那么使用 APPLICATION_EARLY。
- ibm.tolerates - 指定存在版本冲突时要在系统中提供的单例功能部件 *type=osgi.subsystem.feature* 的备用单例版本。

  unique-name 指定单例功能部件的首选版本的符号名称。如果已知包含功能部件使用给定单例功能部件的其他单例版本，那么可使用 ibm.tolerates 伪指令指定这些单例版本。其他功能部件定义的给定单例功能部件的必需版本值存在冲突时，这会给予定义功能部件更高兼容性。

  ibm.tolerates 伪指令中列示的单例版本仅在存在版本冲突时使用。ibm.tolerates 伪指令中列示的版本顺序不重要 - 可选择 ibm.tolerates 伪指令中列示的任何版本来满足依赖关系要求。

  给定单例功能部件的容许版本必须显式列示在 ibm.tolerates 伪指令中。使用逗号来分隔容许版本列表。不支持指定版本范围。

例如：

```
Subsystem-Content: com.ibm.websphere.appserver.example.featureA-1.1; ibm.tolerates:="1.2";
 type="osgi.subsystem.feature",
                    com.ibm.websphere.appserver.example.featureB-1.1; ibm.tolerates:="1.2, 1.4, 1.6";
 type="osgi.subsystem.feature"
```

注：

容许版本不可转移。这可以避免您的功能部件所依赖的功能部件被自动选为支持更高级别的功能部件而不经测试。

例如：用户功能部件 featureC-1.1 在其清单文件的 Subsystem-Content 头中包含 sipServlet-1.1。sipServlet-1.1 包含 servlet-3.0 并容许 servlet 3.1。如果 featureC-1.1 是在 servlet-3.1 存在之前编写的，然后它使用的功能部件 (sipServlet-1.1) 添加并容许 servlet-3.1，那么 featureC-1.1 应说明它是否也容许 servlet-3.1。

如果将 server.xml 文件配置为具有以下两个功能部件：

```
<feature>usr:featureC-1.1</feature> // includes: sipServlet-1.1
<feature>websocket-1.0</feature> // this feature requires servlet-3.1
```

那么您将见到系统显示类似如下的错误消息：

```
CWWKF0033E: 不能同时装入单例功能部件 servlet-3.0 和 servlet-3.1。
所配置功能部件 usr:featureC-1.1 和 websocket-1.0 包含的一个或多个功能部件导致该冲突。
您的配置不受支持；请更新 server.xml 以移除不兼容的功能部件。
```

报告此错误是因为 featureC-1.1 未被选为容许 servlet-3.1，所以必须具有 servlet-3.0，并且 websocket-1.0 不支持 servlet-3.0，所以必须具有 servlet-3.1。

此解决方案适用于同样直接依赖于 servlet-3.0 并容许 servlet-3.1 的 featureC-1.1。

Subsystem-Name

使用此头，为功能部件提供人类可读的短名称。可以指定文字串或属性名。如果指定属性名，那么可以对该值进行本地化。

例如

```
Subsystem-Name: %name
```

其中，name 的值在 Subsystem-Localization 头（在先前示例中为 loc.properties）指定的位置处的属性文件中按以下格式定义：

```
name=feature_name
```

Subsystem-Description

使用此头，为功能部件提供描述。可以指定文字串或属性名。如果指定属性名，那么可以对该值进行本地化。

例如

```
Subsystem-Description: %desc
```

其中，desc 的值在 Subsystem-Localization 头（在先前示例中为 loc.properties）指定的位置处的属性文件中按以下格式定义：

```
desc=feature_description
```

IBM-Provision-Capability

清单中具有 IBM-Provision-Capability 头的功能部件是自动供应的功能部件。此头描述自动供应此功能部件前必须供应的其他功能部件。列示其他功能部件时，使用该功能部件的 Subsystem-SymbolicName 头。如果所有功能部件都是在 server.xml 文件中配置的，那么 runtime 会检查所有自动供应的功能部件的功能是否得以满足，如果已满足，那么会自动供应这些功能部件。

IBM-Provision-Capability 头的格式使用标准 OSGi LDAP 过滤器。

IBM-API-Package

使用此头来指示对于应用程序可见的 API 包。它与 Export-Package 头语法匹配。这意味着它是一个以逗号分隔的 API 包列表，但是每个 API 包都可以具有一些属性。

支持下列属性：

- type - API 包的类型。type 属性接受下列其中一个值：
  - spec - 指示由标准主体（例如 javax.servlet 或 org.osgi.framework）提供的 API。
  - ibm-api - 指示由 xigemaAS 提供的附加 API。
  - api - 指示用户定义的 API。这是缺省值。
  - third-party - 指示一个可见但并非由 xigemaAS 控制的 API。通常，这些是开放式源代码包。
  - internal - 指示非 API 包，必须向应用程序提供这些非 API 包，才能使应用程序正常运行。如果 Java™ 代码经过字节码增强或交织以在运行时添加对内部代码的引用，那么可使用此选项。

例如：

```
IBM-API-Package: javax.servlet; type="spec",
                 com.ibm.websphere.servlet.session; type="ibm-api",
                 com.ibm.wsspi.webcontainer.annotation; type="internal"
```

IBM-API-Service

使用此头来指示功能部件中对于 OSGi 应用程序可见的服务。此功能部件还必须在 OSGi 服务注册表中注册服务。

它具有下列语法

```
IBM-API-Service ::= service ( ',' service )*
                    service ::= service-name ( ';' attribute )*
                    service-name ::= unique-name
```

service-name 是服务的 Java™ 类或接口名称。这些属性解释为服务的服务特性。

例如：

```
IBM-API-Service: com.ibm.example.service.FeatureServiceOne;
                 myServiceAttribute=myAttributeValue,
                 com.ibm.example.service.FeatureServiceTwo
```

如果 OSGi 应用程序想要使用 IBM-API-Service 头提供的服务，那么应用程序必须包括对该服务的 Blueprint 引用，才能在该应用程序中供应该服务。

例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <reference id="FeatureServiceOneRef"
         interface="com.ibm.example.service.FeatureServiceOne" />
</blueprint>
```

为使服务在 OSGi 应用程序中可供捆绑软件使用，接口包必须对该捆绑软件可用，这意味着所使用捆绑软件的清单文件中的 Import-Package 头必须指定该接口包。功能部件捆绑软件中的 Export-Package 头和功能部件清单文件的 IBM-API-Package 头也必须指定该接口包。必须使用 OSGi BundleContext 接口或任何其他机制（例如，声明式服务或 Blueprint）在 OSGi 服务注册表中注册功能部件提供的服务。有关更多信息，请参阅开发使用简单激活的 *OSGi* 捆绑软件（见第 1243 页）和使用 *OSGi* 声明式服务来编写高级功能部件（见第 1247 页）。

IBM-SPI-Package

创建您自己的 xigemaAS 功能部件时，将它安装到用户产品扩展。您的功能部件中的所有包都可以供安装到用户产品扩展中的任何其他功能部件访问。但是，如果您想要让安装到另一个产品扩展中的功能部件访问您的功能部件中的包，那么必须在 IBM-SPI-Package 头中列出该包的名称。

IBM-SPI-Package 头中列出的任何包必须由 xigemaAS 功能部件中的捆绑软件导出，方法是将包列在捆绑软件清单文件的 Export-Package 头中。

IBM-ShortName

此头是功能部件的短名称，用于在 server.xml 文件中指定功能部件。如果清单文件中没有 IBM-ShortName 头，那么缺省情况下会使用 Subsystem-SymbolicName。IBM-ShortName 头仅对公共功能部件有效。

IBM-AppliesTo

此头指定此功能部件适用于的 xigemaAS 版本。提供以逗号分隔的项目列表，每个项目的格式如下所示：

```
product_id; productVersion=product_version; productInstallType=product_install_type;
 productEdition=product_editions
```

如果您提供多个项目，那么每个项目的 *product_id* 的值必须是唯一的。

productVersion 的值可以是确切版本（例如，9.0.0.1），也可以是最低版本（由以加号 (+) 结尾的版本指示，例如，9.0.0.1+）。

productEdition 的值可以是单个版本，也可以是版本的逗号分隔列表（括在引号中）。

例如：

```
IBM-AppliesTo: com.ibm.websphere.appserver; productVersion=8.5.5.6; productInstallType=Archive;
 productEdition="BASE,DEVELOPERS,EXPRESS,ND"
```

Subsystem-License

此头定义此功能部件的许可类型。如果您为 Subsystem-License 头提供值，但没有为 IBM-License-Agreement 头和 IBM-License-Information 头提供值，那么安装期间会显示 Subsystem-License 头值，供用户验收。

如果已安装具有相同 Subsystem-License 头值的功能部件，那么在安装期间不会显示许可，而且不要求许可批准。如果 `Subsystem-Content` 头中的依赖性意味着正在安装的两个或多个功能部件具有相同的 Subsystem-License 头值，那么安装期间用户只需接受许可一次。

例如：

```
Subsystem-License: L-JTHS-93TMHH
```

```
Subsystem-License: http://www.apache.org/licenses/LICENSE-2.0.html
```

IBM-License-Agreement

此头指定许可协议文件的位置的前缀。提供子系统归档中 LA_*language* 文件的文件路径，文件路径由“_”字符之前的内容表示（语言代码是由安装工具附加）。如果尚未接受此许可，那么安装功能部件时用户必须接受许可。这些许可证文件会复制到 xigemaAS 安装目录。

例如：

```
IBM-License-Agreement: lafiles/LA
```

IBM-License-Information

此头指定许可信息文件的位置的前缀。提供子系统归档中 LI_*language* 文件的文件路径，文件路径由“_”字符之前的内容表示（语言代码是由安装工具附加）。如果尚未接受此许可，那么安装功能部件时用户必须接受许可。这些许可证文件会复制到 xigemaAS 安装目录。

例如：

```
IBM-License-Information: lafiles/LI
```

IBM-App-ForceRestart

对运行中服务器安装或移除功能部件后，此头会导致应用程序重新启动。此头可以采用下列其中一个值：

- `install` - 安装功能部件后重新启动应用程序。
- `uninstall` - 卸载功能部件后重新启动应用程序。
- `install,uninstall` - 安装或卸载功能部件后重新启动应用程序。

示例功能部件清单文件

以下示例显示了 `example-1.0` 功能部件的定义。公共可见性属性允许直接在服务器配置 (server.xml) 文件中指定此功能部件；此功能部件还将包括在开发者工具的**服务器配置**视图中所显示的功能部件下拉列表中，并且将可包括在其他产品扩展中的功能部件中。进行运行时安装时，如果已将此功能部件安装到 usr 产品扩展中，那么可以通过在 server.xml 文件中包括以下代码来将此功能部件配置到服务器中：

```
<featureManager>
 <feature>usr:example-1.0</feature>
</featureManager>
```

在服务器中配置此功能部件将导致在服务器运行时环境的 OSGi 框架中安装并启动所指定捆绑软件 com.ibm.example.bundle1。单个 API 包 (com.ibm.example.publicapi) 将对该服务器中的所有应用程序可见，配置为对 api 包类型不可见的 Java™ EE 应用程序除外。如果 OSGi 应用程序希望使用该包，那么它们必须显式导入该包。两个 SPI 包（com.ibm.example.spi.utils 和 com.acme.spi.spiservices）将对服务器中的所有功能部件代码可见，就像 API 包一样。

```
IBM-Feature-Version: 2
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: com.ibm.example-1.0; visibility:=public
Subsystem-Version: 1.0.0.qualifier
Subsystem-Type: osgi.subsystem.feature
Subsystem-Content: com.ibm.example.bundle1; version="1.0.0"
Subsystem-Localization: OSGI-INF/l10n/loc
Manifest-Version: 1.0
Subsystem-Name: %name
Subsystem-Description: %desc
IBM-API-Package: com.ibm.example.publicapi; type="api"
IBM-SPI-Package: com.ibm.example.spi.utils, com.ibm.example.spi.spiservices
IBM-ShortName: example-1.0
```

自动供应功能部件

自动供应允许一个功能部件在可供应之前具有对必须供应的功能部件的依赖性。

自动供应的功能部件对其他功能部件存在依赖性。因为依赖性，自动供应的功能部件的生命周期如下所示：

- 当提供了所有必需功能部件，自动供应该功能部件。
- 取消供应任何必需功能部件后，自动取消供应该功能部件。

1. 要将功能部件配置为自动供应的功能部件，请遵循以下步骤：
1. 确定运行时自动供应此功能部件之前必须供应的功能部件。
2. 将 IBM-Provision-Capability 添加至清单头。IBM-Provision-Capability 头的格式使用标准 OSGi LDAP 过滤器。
3. 将该功能部件部署至服务器。

供应必需功能部件后，自动供应该功能部件。

在以下示例中，如果供应了功能部件 requiredFeature1-1.0 和 requiredFeature2-1.0，那么将自动供应此功能部件。如果从 server.xml 文件中移除任一必需功能部件，那么将自动取消供应此功能部件。

```
IBM-Provision-Capability: osgi.identity; filter:="(&(type=osgi.subsystem.feature)
(osgi.identity=requiredFeature1-1.0))", osgi.identity;
filter:="(&(type=osgi.subsystem.feature)(osgi.identity=requiredFeature2-1.0))"
```

自动安装自动供应的功能部件

如果还安装了所有必需的功能部件，那么可自动安装自动供应的功能部件。

要将功能部件配置为自动安装，必须将 IBM-Install-Policy 头添加至功能部件清单。此头是可选的。如果指定了 IBM-Install-Policy 头，那么以下值有效：

- manual：不自动安装该功能部件。
- when-satisfied：如果已安装所有必需功能部件，那么将自动安装该功能部件。

如果未设置该头，那么不会自动安装该功能部件，相当于将 IBM-Install-Policy 头设置为 manual。

开发使用简单激活的 OSGi 捆绑软件

用于控制 OSGi 捆绑软件代码生命周期的最简单方法是在捆绑软件的其中一个类中实现 `org.osgi.framework.BundleActivator` 接口。当服务器启动和停止捆绑软件时，会调用 `BundleActivator` 接口的启动和停止方法。

Identify捆绑软件 activator 类。例如：`Bundle-Activator: com.example.bundle.Activator`。

示例

```
package com.example.bundle;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

public class Activator implements BundleActivator {
 public void start(BundleContext context) throws Exception {
  System.out.println("Sample bundle starting");
    // Insert bundle activation logic here
 }

 public void stop(BundleContext context) throws Exception {
  System.out.println("Sample bundle stopping");
    // Insert bundle deactivation logic here
 }
}
```

使用 ManagedService 接口来接收配置数据

xigemaAS 配置由 OSGi 配置管理服务管理，并且可以根据 OSGi 配置管理服务规范来访问此配置。配置属性集由持久存储的身份 (PID) 来标识，其中 PID 用作元素名称，且用于将 server.xml 文件中的元素与注册用来接收属性的组件相关联。

对于使用 `BundleActivator` 接口来管理其生命周期的 OSGi 捆绑软件，接收配置属性的直观方式是实现 `org.osgi.service.cm.ManagedService` 接口，该接口指定 PID 作为其属性。

示例

👉 切记:

1. 在 Eclipse 中，必须从**窗口 > 首选项 > 插件开发 > 目标平台**选择 SPI 目标运行时。
2. 将以下语句添加至 MANIFEST.MF 文件：

   ```
   Import-Package: org.osgi.service.cm;version="1.5.0"
   ```

3. 按 Ctrl + Shift + O 以更新捆绑软件激活器。

在此示例中，Activator 类实现 `ManagedService` 接口及 `BundleActivator` 接口，并使用 `updated` 方法来接收配置属性。可以提供缺省属性值来简化必须在用户配置中指定的内容。

```
public class Activator implements BundleActivator, ManagedService {

  public void start(BundleContext context) throws Exception {
    System.out.println("Sample bundle starting");
    // register to receive configuration
    ServiceRegistration<ManagedService> configRef = context.registerService(
      ManagedService.class,
      this,
      getDefaults()
      );
  }
```

```
  public void stop(BundleContext context) throws Exception {
    System.out.println("Sample bundle stopping");
    configRef.unregister();
  }

  Hashtable getDefaults() {
    Hashtable defaults = new Hashtable();
    defaults.put(org.osgi.framework.Constants.SERVICE_PID, "simpleBundle");
    return defaults;
  }

  public void updated(Dictionary<String, ?> properties) throws ConfigurationException {
    if (properties != null)
    {
      String configColor = (String) properties.get("color");
      String configFlavor = (String) properties.get("flavor");
    }
  }

}
```

可以选择在 server.xml 文件或者包含的文件中使用以下条目来提供捆绑软件的用户配置：

```
<simpleBundle color="red" flavor="raspberry" />
```

👉 注：用户配置中的元素名称 *simpleBundle* 与 `ManagedService` 注册中使用的 `org.osgi.framework.Constants.SERVICE_PID` 属性的值相匹配。

有关更高级的配置用法，请参阅使用 *OSGi* 元类型服务来描述配置（见第 1251 页）。

使用 OSGi 服务注册表

可以创建对象，并将其注册为 OSGi 服务以供部署到 xigemaAS 的第三方功能部件使用。

服务是 OSGi 灵活的轻量级组件模型。创建服务并使它们与 Java™ 代码连线在一起时，可以使用诸如 ServiceTrackers 等机制来帮助查找您想要的服务，并使用声明式服务 (DS) 和 Blueprint 以声明式方式来指定连线。xigemaAS 概要文件已针对使用 DS 来连线进行标准化，除了少数需要附加灵活性的情况。

注册 OSGi 服务

可以创建对象，并将其注册为 OSGi 服务以供第三方功能部件使用。

通过使用无格式的旧 Java 代码，可以创建对象，然后将其注册为使用 `BundleContext` 类的服务。因为代码必须运行，所以您通常在 `BundleActivator` 接口中注册对象。注册对象时，可以指定它所提供的接口并提供属性映射。会返回 `ServiceRegistration` 对象；如有必要，可以随时使用 `ServiceRegistration` 对象来更改属性。完成服务后，使用 `ServiceRegistration` 对象来注销服务。

要获取服务，可以查询 `BundleContext` 以查找实现必需接口的服务，而且可以选择性地提供 LDAP 语法过滤器来匹配服务属性。根据所调用的方法，可以检索最佳匹配项或所有匹配项。然后，可以使用返回的 `ServiceReference` 来提供属性以在代码中执行进一步匹配。可以使用 `ServiceReference` 来获取实际的服务对象。完成使用服务时，使用 `BundleContext` 来释放服务。

1. 通过在捆绑软件中添加以下代码来声明服务接口。

```
package com.ibm.foo.simple;

/**
 * Our multifunctional sample interface
 */
public interface Foo
{
```

```
}
```

2. 指定接口的实现代码。

```
package com.ibm.foo.simple;

/**
 * The implementation of the Foo interface
 */
public class FooImpl implements Foo
{
    public FooImpl()
    {
    }
    public FooImpl(String vendor)
    {
    }

    /**
     * used by the ServiceFactory implementation.
     */
    public void destroy() {

    }
}
```

3. 使用 `BundleContext` 直接在代码中注册服务、修改服务属性以及注销服务。

```
import java.util.Dictionary;

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceRegistration;

/**
 * Registers and unregsiters a Foo service directly,
 * and shows how to modify the service properties in code.
 */
public class FooController
{

    private final BundleContext bundleContext;
    private ServiceRegistration<Foo> sr;

    public FooController( BundleContext bundleContext )
    {
        this.bundleContext = bundleContext;
    }

    public void register(Dictionary<String, Object> serviceProperties) {
        Foo foo = new FooImpl();
        //typed service registration with one interface
        sr = bundleContext.registerService( Foo.class, foo, serviceProperties );
        //or
        //untyped service registration with one interface
        sr = (ServiceRegistration<Foo>)bundleContext.registerService(
                Foo.class.getName(), foo, serviceProperties );
        //or
        //untyped service registration with more than one interface (or class)
        sr = (ServiceRegistration<Foo>)bundleContext.registerService(new String[] {
                Foo.class.getName(), FooImpl.class.getName()}, foo, serviceProperties );
    }

    public void modifyFoo(Dictionary<String, Object> serviceProperties) {
        //with the service registration you can modify the service properties at any time
        sr.setProperties( serviceProperties );
    }

    public void unregisterFoo() {
        //when you are done unregister the service using the service registration
        sr.unregister();
    }

}
```

4. 从另一个类获取服务和返回服务：

```
package com.ibm.foo.simple;

import java.util.Collection;

import org.osgi.framework.BundleContext;
import org.osgi.framework.InvalidSyntaxException;
import org.osgi.framework.ServiceReference;

/**
 * A simple Foo client that directly obtains the Foo service and returns it when done.
 */
public class FooUser
{

    private final BundleContext bundleContext;

    public FooUser( BundleContext bundleContext )
    {
        this.bundleContext = bundleContext;
    }

    /**
     * assume there's only one Foo
     */
    public void useFooSimple() {
        ServiceReference<Foo> sr = bundleContext.getServiceReference( Foo.class );
        String[] propertyKeys = sr.getPropertyKeys();
        for (String key: propertyKeys) {
            Object prop = sr.getProperty( key );
            //think about whether this is the Foo we want....
        }
        Foo foo = bundleContext.getService( sr );
        try {
            //use foo
        } finally {
            //we're done
            bundleContext.ungetService( sr );
        }
    }

    /**
     * Use a filter to select a particular Foo. Note we get a collection back and have to
 pick one.
     * @throws InvalidSyntaxException
     */
    public void useFooFilter() throws InvalidSyntaxException {
        Collection<ServiceReference<Foo>> srs = bundleContext.getServiceReferences(
          Foo.class, "(&(service.vendor=xigemaAS)(id='myFoo')" );
        ServiceReference<Foo> sr = srs.iterator().next();
        String[] propertyKeys = sr.getPropertyKeys();
        for (String key: propertyKeys) {
            Object prop = sr.getProperty( key );
            //think about whether this is the Foo we want....
        }
        Foo foo = bundleContext.getService( sr );
        try {
            //use foo
        } finally {
            //we're done
            bundleContext.ungetService( sr );
        }
    }
}
```

使用 OSGi 服务

服务可随时异步地注册和注销。因此，调用服务的时间应该尽可能短。可以使用 ServiceTracker 类来并发地跟踪服务可用性。

如果您想要跟踪服务，那么可以通过使用捆绑软件上下文、所需接口以及要匹配的属性来创建 ServiceTracker 对象，然后打开跟踪程序。可以查询跟踪程序以获取最佳匹配项或所有匹配项。请确保您在使用服务之后不会占用该服务。不必告诉跟踪程序您已完成；跟踪程序会在内部高速缓存匹配的服务，并在服务注销后将其清除。当您已经完成了使用跟踪程序时，请使用 serviceTracker.close() 方法来关闭跟踪程序。

以下示例说明如何使用 ServiceTracker 对象来跟踪服务：

```
package com.ibm.foo.tracker;

import com.ibm.foo.simple.Foo;
import org.osgi.framework.BundleContext;
import org.osgi.util.tracker.ServiceTracker;

/**
 * Simplest use of a ServiceTracker to get a service
 */
public class TrackingFooUser
{

    private ServiceTracker<Foo,Foo> serviceTracker;

    public TrackingFooUser( BundleContext bundleContext )
    {
        serviceTracker = new ServiceTracker<Foo, Foo>( bundleContext, Foo.class, null );
        serviceTracker.open();
    }

    public void doFoo() {
        Foo foo = serviceTracker.getService();
        //use foo
        //no need to return it... just don't use it for long.
    }

    public void shutdown() {
        serviceTracker.close();
    }
}
```

使用 OSGi 声明式服务来编写高级功能部件

可以使用捆绑软件 activator 类以及接口（例如，ManagedService 和 ServiceTracker）的直接实现来控制简单的功能部件。随着捆绑软件之间的关系变得更复杂，使用 OSGi 声明式服务 (DS) 之类的工具将功能部件分解为各项服务会更好。DS（有时称为“服务组件运行时”，即 SCR）提供了 OSGi 服务的生命周期和注入管理。

将功能部件逻辑组织为一组声明式服务有许多优势：

- 可以延迟到使用服务（包括载入用于提供此服务的 Java™ 类）时才激活此服务；允许服务器快速启动，以及将资源使用量保持在最低程度。
- 将对服务的引用放入服务注册表中（即使尚未激活该服务），以便可以解析对于此服务的依赖性。
- 可以在运行时注入对于其他服务的依赖性，并且将根据这类依赖性对各种服务的激活进行排序。
- 必要时，可以根据服务属性的更改来取消激活服务和重新激活此服务。

许多在线资源（包括*OSGi 社区 Wiki*）提供了有关使用 OSGi 声明式服务的详细信息。

此任务对于如何向 DS 声明服务、如何获取对于其他服务的引用以及如何管理每项服务的配置属性进行了简单描述。

向 OSGi 声明式服务声明服务

可以使用单独的 XML 文件在捆绑软件中声明每项服务。

声明式服务 (DS) 支持对已声明的组件执行操作，每个组件由捆绑软件中的 XML 文件定义。将包含组件声明的捆绑软件添加至框架时，DS 将读取每个组件声明，并在服务注册表中注册所提供的服务。DS 随后会管理组件的生命周期：根据声明式属性和满足依赖性的组合来控制它的生命周期。

组件的 XML 描述可让 DS 解析服务依赖性，而不要求实例化组件，也不要求载入其实现类。这便于资源的延迟加载，从而有助于改进服务器的启动以及减少运行时内存占用量。

在捆绑软件的 MANIFEST.MF 文件中使用 `Service-Component` 头列示了用于描述组件的 XML 文件，这些 XML 文件通常位于该捆绑软件的 /OSGI-INF 目录中。

有许多工具可用来生成必需的 XML；下列示例显示了 XML 本身。

本主题描述了一个使用 XML 向 DS 声明其组件的简单 OSGi 捆绑软件。

1. 通过组件的实现类名来标识组件。

```
<component>
  <implementation class="com.example.bundle.HelloComponent"/>
</component>
```

2. 通过引用服务提供的接口的名称来声明该服务。这是捆绑软件启动时将由 DS 发布到服务注册表的名称。

```
<component>
  <implementation class="com.example.bundle.HelloComponent"/>
  <service>
     <provide interface="com.example.HelloService"/>
  </service>
</component>
```

3. 组件的名称。组件名称还充当服务“持久身份”（即，PID），它用来使配置属性与该服务相关联。在激活时以及每次更新属性时，具有相匹配的 PID 的配置属性将注入到组件中。

```
<component name="HelloService">
  <implementation class="com.example.bundle.HelloComponent"/>
  <service>
    <provide interface="com.example.HelloService"/>
  </service>
</component>
```

注：在 xigemaAS 中，用户可以将以下元素添加至 server.xml 配置文件，并且属性将注入到 `HelloComponent` 类中。

```
<HelloService firstKey="firstValue" secondKey="secondValue" />
```

4. 将此 XML 文件打包到捆绑软件中。

   例如，此 XML 文件位于 OSGI-INF/HelloService.xml，您向捆绑软件清单 MANIFEST.MF 文件添加一个头，以便 DS 可以找到该文件：

```
Service-Component: OSGI-INF/HelloService.xml
```

   如果有多个组件打包在同一捆绑软件中，那么必须以逗号分隔的列表形式输入相应的 XML 文件。例如：

```
Service-Component: OSGI-INF/HelloService.xml, OSGI-INF/GoodbyeService
```

5. HelloService 组件的 Java™ 实现如下所示：

```
package com.example.bundle;

import com.example;
import org.osgi.service.component.ComponentContext;

/*
 * This class must be public and have a public default constructor for it to be
 * usable by DS. This class is not required to be exported from the bundle.
 */
public class HelloComponent implements HelloService {
 /**
    * Optional: DS method to activate this component. If this method exists, it
    * will be invoked when the component is activated. Best practice: this
    * should be a protected method, not public or private
  *
  * @param properties
    *            : Map containing service & config properties
    *            populated/provided by config admin
  */
 protected void activate(ComponentContext cContext, Map<String, Object> properties)
 {WSSubject.setRunAsSubject(oldSubject);
  modified(properties);
 }

 /**
    * Optional: DS method to deactivate this component. If this method exists,
    * it will be invoked when the component is deactivated. Best practice: this
    * should be a protected method, not public or private
  *
    * @param reason
    *            int representation of reason the component is stopping
  */
 protected void deactivate(ComponentContext cContext, int reason) {
 }

 /**
    * Optional: DS method to modify the configuration properties. This may be
    * called by multiple threads: configuration admin updates may be processed
    * asynchronously. This is called by the activate method, and otherwise when
    * the configuration properties are modified while the component is
    * activated.
  *
  * @param properties
  */
 public synchronized void modified(Map<String, Object> properties) {
    // process configuration properties here
 }

 /**
    * Service method defined by com.example.HelloService interface
  */
  public void sayHello() {
  System.out.println("Hello");
 }
}
```

启用服务来接收配置数据

要启用服务来接收配置数据，可以使服务与持久存储的身份相关联，并编写服务代码来接收数据。还可以为此数据提供描述和缺省值，并提供若干语言的标签和描述。

涉及一些步骤以使服务能接收配置数据。只有“使服务与配置管理持久存储的身份相关联”以及“为服务编写代码来接收配置属性”是必须执行的操作；这对于嵌入式方案而言足够了。剩余的步骤会改进用户的配置体验。

使服务与持久存储的身份相关联

如 OSGi 配置管理规范中所述，通过使用持久存储的身份 (PID) 使配置属性集与其消费组件相关联。

OSGi 配置管理规范提供了许多关联机制，下列机制是 xigemaAS 中最常用的机制：

直接向 **OSGi** 配置管理 **(CA)** 服务注册 **`org.osgi.service.cm.ManagedService`** 或 **`org.osgi.service.cm.ManagedServiceFactory`** 的实现

这在低级内核捆绑软件中最常用，其中，通过 OSGi 声明式服务 (DS) 或 Blueprint 进行服务管理在捆绑软件启动时不可用。注册指定用来确定要接收的配置集的 PID。

将服务定义为 **DS**

这是功能部件捆绑软件中服务接收其配置的最常用方式。服务名称用作 PID 来关联配置数据。DS 从 CA 接收配置集，并将其传递到定义的服务。

可在项目 *.bnd 文件中使用下列条目来声明服务：

```
Service-Component: com.ibm.ws.transaction; \
      provide:='com.ibm.tx.config.ConfigurationProvider'; \
      immediate:='true'; \
      modified:='modified'; \
    implementation:=com.ibm.ws.transaction.services.JTMConfigurationProvider
```

这会生成以下 XML 代码，此代码也可以由开发者进行编码，而不是使用 bnd `Service-Component` 条目进行编码：

```
<component name="com.ibm.ws.transaction" xmlns="http://www.osgi.org/xmlns/scr/v1.1.0"
           immediate="true" modified="modified">
    <implementation class="com.ibm.ws.transaction.services.JTMConfigurationProvider" />
  <service>
      <provide interface="com.ibm.tx.config.ConfigurationProvider" />
    </service>
  <property name="service.vendor" value="IBM" />
</component>
```

此示例中的组件名称 `com.ibm.ws.transaction` 用作 PID 来关联配置数据。如果此组件没有提供任何元数据来描述其配置，那么可以在 server.xml 文件或随附的文件中，使用该 PID 通过定义以下形式的条目，为此组件指定配置属性：

```
<com.ibm.ws.transaction made.up.property.key="47">
```

编写服务代码以在激活期间及修改配置时接收配置属性。

为服务编写代码来接收配置属性

通过激活方法上提供的 `org.osgi.service.component.ComponentContext` 对象来提供配置属性。

您必须完成使服务与持久存储的身份相关联（见第 1250 页）中所描述的任务。

如果在激活之后更新了属性，那么用于注入的方法取决于服务在其 OSGi 声明式服务 (DS) 声明中提供的上下文。

通常，最好是通过在服务声明上使用 modified 属性来声明要专门用于注入所更新属性的方法。如果修改的方法不可用，那么 DS 会取消激活服务，然后使用新属性来重新激活服务。

取消激活某项服务之后再将其激活，这还可能导致重新启动从属服务，因此，除非明确要求取消激活某项服务，否则应该避免此操作。使用 modified 属性是接收配置更新的首选方法。

在先前任务使服务与持久存储的身份相关联（见第 1250 页）中，您给 DS 定义了服务。下面举例说明了该任务中所描述 DS 声明内的 activate 和 modified 方法。

```
private static Dictionary<String, Object> _props = null;

    protected void activate(ComponentContext cc) {
        _props = cc.getProperties();
    }

    protected void modified(Map<?, ?> newProperties) {
        if (newProperties instanceof Dictionary) {
            _props = (Dictionary<String, Object>) newProperties;
                } else {
            _props = new Hashtable(newProperties);
        }
    }
```

从配置属性获取值时，请使用下列机制来允许一些灵活性：

- 编写方法代码以要求同一个捆绑软件所随附的至少一个缺省属性，但允许用户覆盖，以免迁移用户配置。
- 忽略多余或无法识别的属性。

服务必须能够单独操作缺省配置。为了提供合理级别的功能，用户覆盖不得是必需的。

为配置元数据提供描述和缺省值

高级配置

高级配置包含有关为配置和 OSGi 元类型服务扩展提供描述和缺省值的信息。

可以用符合 OSGi 元类型服务规范的元数据来描述每项服务的配置属性。依照规范，产生的 XML 文件会打包到 OSGI-INF/metatype 目录中的捆绑软件。有关更多信息，请参阅“为配置提供描述和缺省值”。

xigemaAS 运行时和开发者工具识别对 OSGi 元类型规范的一些扩展，以便在用户界面中进行更复杂的配置和更好的展示。有关更多信息，请参阅 *OSGi 元类型服务扩展*。

使用 OSGi 元类型服务来描述配置

可以用符合 OSGi 元类型服务规范的元数据来描述每项服务的配置属性。元数据可以包含缺省值、可翻译的名称和描述以及允许对输入值进行验证的信息。依照规范，所产生 XML 文件会打包到 OSGI-INF/metatype 目录中包含您的服务的捆绑软件。

提供元数据来描述配置是可选的，但它的确具有以下优势：

- 缺省值可以与实现代码分离，并放入元类型 XML 文件，在该文件中可以很容易地找到缺省值；
- 可以为每个属性指定相应的数据类型和其他验证数据，从而允许使用配置解析器和开发者工具进行验证，并简化您编写用来处理这些属性的代码；
- 您的配置将包括在 XML 模式中，该模式描述可供开发者工具和其他实用程序使用的配置；
- 可以为每个属性提供可翻译的名称和描述，并且将在开发者工具中显示可翻译的名称和描述。

1. 在捆绑软件的 OSGI-INF/metatype 目录中创建 XML 文件，然后为 OSGi 元类型命名空间添加命名空间声明：

```
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0">
</metatype:MetaData>
```

2. 添加对象类定义 (OCD) 元素以包含属性集，该元素具有标识及（可选）名称和描述。此外，还提供 `Designate` 元素，以将 OCD 映射到代码和 server.xml 文件中使用的 PID。

```
<OCD name="b2c" description="bundle two config" id="b2c-id">
</OCD>

<Designate pid="testBundleTwo">
<Object ocdref="b2c-id" />
</Designate>
```

3. 在 OCD 中为每个配置属性添加属性定义 (AD) 元素。每个属性需要一个标识，该标识也用在 server.xml 文件中以及用来接收所注入配置的代码中。它可以选择性地具有可供开发者工具及其他图形工具使用的名称和描述。如果指定数据类型，那么允许运行时环境验证该类型的输入并简化代码的处理过程。如果指定有用的缺省值，那么允许用户提供的配置是最低配置，并包含已知配置中的所有配置缺省值：

```
<AD name="boolProperty" description="a boolean property" id="boolProp"
    type="Boolean" default="false" />
```

4. 然后，您具有以下 metatype.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0">

  <OCD name="b2c" description="bundle two config" id="testBundleTwo-2-id">
    <AD name="textProperty" description="a text property"
              id="textProp" type="String" default="default string" />
    <AD name="boolProperty" description="a boolean property"
              id="boolProp" type="Boolean" default="false" />
    <AD name="intProperty" description="an integer property"
              id="intProp" type="Integer" default="14" />
 </OCD>

  <Designate pid="testBundleTwo-2">
    <Object ocdref="testBundleTwo-2-id" />
 </Designate>

</metatype:MetaData>
```

5. 给服务编写代码以接收配置属性。如果没有元类型描述，那么所有属性都将作为 `String` 值提供，并且将加以处理，如下所示：

```
String textProp = (String) properties.get("textProp");
Boolean boolProp = Boolean.parseBoolean((String) properties.get("boolProp"));
int intProp = Integer.parseInt((String) properties.get("intProp"));

String textProp = (String) properties.get("textProp");
Boolean boolProp = (Boolean) properties.get("boolProp");
int intProp = (Integer) properties.get("intProp");
```

运行时环境将已经验证输入值是否为正确的类型。

为配置属性标签和描述提供翻译字符串。

单个配置实例与多个配置实例

您也可以使用 OSGi 元类型服务来配置多个配置实例。

如使用 *OSGi* 元类型服务来描述配置（见第 1251 页）中所述，可以使用 OSGi 元类型服务来支持给定服务的一组配置属性（由配置 PID 所标识）。例如，另一种常见情况是以 xigemaAS 概要文件支持应用程序和数据源的多个条目的方式，支持同一种配置类型的多个实例。要这样做，可以提供元类型定义来告知 xigemaAS 配置解析器及配置管理服务，它正在处理工厂配置。此外，收到该配置的类也需要实现 `org.osgi.service.cm.ManagedServiceFactory` 接口。

要支持 server.xml 文件中顶级配置元素的多个实例，如下所示：

```
<server>
    <teenager name="joy" age="15" />
    <teenager name="angela" age="18" />
</server>
```

必须通过将 factoryPid 属性添加至 Designate 元素，在元数据中定义配置。

👉 注：如果使用 ManagedServiceFactory 接口来接收配置，那么仍需要 pid 属性；如果使用声明式服务 (DS) 组件，那么此属性不是必需的。

```
<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
                   xmlns:ibm="http://www.vsettan.com.cn/xmlns/appservers/osgi/metatype/v1.0.0">

    <OCD id="teenager-ocd" name="teenager" >
       <AD id="name" name="name" type="String" />
       <AD id="age" name="age" type="Integer" />
    </OCD>

    <Designate factoryPid="teenager" pid="teenager">
        <Object ocdref="teenager-ocd" />
    </Designate>

</metatype:MetaData>
```

使用如下所示的工厂 pid 将 ManagedServiceFactory 实现注册为 ManagedServiceFactory 服务类型：

```
bundleContext.registerService(ManagedServiceFactory.class, new MgdSvcFactoryImpl(), new Hashtable();
defaults.put(org.osgi.framework.Constants.SERVICE_PID, "teenager"));
```

ManagedServiceFactory 实现会收到每个 *teenager* 配置实例的一个属性集，而各个属性集均由其自己的内部生成 PID（提供给 updated() 方法）唯一标识，如下所示：

```
public void updated(String pid, Dictionary<String, ?> properties)
      throws ConfigurationException {
      String name = (String) properties.get("name");
    Integer age = (Integer) properties.get("age");
 }
```

如果删除特定的配置实例（例如，由于从 server.xml 文件中删除了其中一个 *teenager* 元素），那么会通过 deleted() 方法来通知 ManagedServiceFactory 实现，并且会提供所删除实例的 pid。这让 ManagedServiceFactory 实现可以跟踪在任何给定时刻为有效的实例。

提供工厂配置的缺省实例

使用 OSGi 元类型服务时，您可以创建工厂配置的缺省实例。xigemaAS 的其中一个设计准则是要保持用户配置尽可能既小巧又简单。通过提供工厂配置的缺省实例，您不必将这些配置添加到 server.xml 文件。

要提供缺省配置实例，您需要将实例包括在 OSGi 捆绑软件内的 XML 文件中，并通过在捆绑软件清单文件中使用 IBM-Default-Config 头来引用该文件，如下所示：

```
IBM-Default-Config: OSGI-INF/wlp/defaultInstances.xml
```

XML 文件的格式和 server.xml 文件的格式相同，但您必须为每个实例指定唯一标识。例如，为提供单个配置实例与多个配置实例（见第 1252 页）主题上的示例中使用的 *teenager* 配置的缺省实例，defaultInstances.xml 文件必须具有以下设置：

```
<server>
  <teenager id="predefined-teen1" name="Susie" age="19" />
</server>
```

缺省实例不会通过配置模式来公开给用户，因此在开发工具中不可见；然而，您可以记载实例，以便用户可以覆盖 server.xml 文件中的个别属性，如下所示：

```
<teenager id="predefined-teen1" age="13" />
```

这行代码将覆盖缺省实例的 age 属性，但 name 属性仍保持有效。

OSGi 元类型服务的扩展

运行时环境可以识别 OSGi 元类型规范的一些扩展，这些扩展可以用来实现更复杂的配置以及在用户界面中获得更好的表示效果。

运行时元类型扩展

将此命名空间添加至 metatype.xml 文件以使用以下扩展：

```
xmlns:ibm="http://www.ibm.com/xmlns/appservers/osgi/metatype/v9.0.0"
```

ibm:alias

别名扩展用来为配置定义用户友好的名称，同时降低 server.xml 文件中配置元素的名称产生冲突的风险。

以下示例显示了 ibm:alias 扩展：

```
<OCD id="com.ibm.ws.jdbc.dataSource.properties"
      name="%properties" description="%properties.desc" ibm:alias="properties">
   <AD id="username".../>
</OCD>
```

在此示例中，`properties` 是配置的用户友好名称。别名和标识必须不同。

在 server.xml 文件中使用 ibm:alias 条目时，该条目必须以产品扩展名作为前缀。安装在缺省用户位置中的扩展的产品扩展名是 usr。对于通过使用 wlp/etc/extension 目录中的 *extension-name*.properties 文件来定义到 xigemaAS 概要文件安装的产品扩展，产品扩展名是为 *extension-name* 选择的名称。

对于先前示例中所示的元类型，如果将功能部件安装在缺省 usr 位置中，那么以下是有效 server.xml 条目的示例：

```
<usr_properties username="JANE"/>
```

```
<com.ibm.ws.jdbc.dataSource.properties username="JANE"/>
```

ibm:type

在元类型规范中定义标准属性类型。有数种 xigemaAS 扩展类型可供使用。有关更多信息，请参阅扩展类型（见第 1255 页）。

ibm:reference

reference 属性指定 PID 所引用的 OCD 类型。它仅与 ibm:pid 类型一起使用，且支持在 server.xml 文件中嵌套元素；请参阅嵌套配置元素（见第 1258 页）。

以下示例显示了 ibm:reference 扩展：

```
<AD id="fooRef" type="String" ibm:type="pid" ibm:reference="com.ibm.ws.foo".
../>
```

ibm:final

final 属性指示不能在配置中指定值。相反地，总是使用元类型中的缺省值。使用 name="internal" 来指示工具不显示此属性。

以下示例显示了 ibm:final 扩展：

```
<AD id="foo" name="internal" ibm:final="true" type="String" default=${someVa
riable}"/>
```

ibm:variable

variable 属性用来指定要用于缺省值的变量（如果未指定变量）。行为是以下列顺序选择：

- 在 server.xml 中指定的值
- 指定为系统属性的值（例如，在 bootstrap.properties 中指定）
- 元类型中的缺省值

以下示例显示了 ibm:variable：

```
<AD id="traceString" ibm:variable="trace.string" default=*.all=enabled".../>
```

ibm:unique

unique 属性指示配置值在使用相同唯一属性组的所有属性定义中必须唯一。支持以下唯一属性组：

- jndiName：针对用于通过将 osgi.jndi.service.name 属性与 JDNI 名称配合使用来注册服务的属性使用此组。有关更多信息，请参阅在 *xigemaAS* 功能部件中使用 *JNDI* 缺省命名空间开发（见第 1268 页）

缺省值语法

可以在缺省表达式中使用 ${prop-name} 语法，从其他配置属性构造字符串。

以下示例显示了缺省值语法：

```
<AD id="httpEndpoint.target"
        name="internal" description="internal use only"
        ibm:final="true" required="false" type="String"
      default-"(&amp;(virtualHost=${id}) (enabled=true))"/>
```

扩展类型

Duration

duration 类型用来表示时间。它以多个时间单位进行描述。例如，“1h30m”将是一个半小时。“1d5h10s”将是 1 天 5 小时 10 秒。这些单位已全球化，因此用户可以使用其当地语言中的缩写来输入值。

对于英语，以下列表显示可用单位：

- d - 天

- h - 小时
- m - 分钟
- s - 秒
- ms - 毫秒

缺省情况下，使用类型持续时间时，对用户指定的值进行求值（以毫秒计）。例如，“10s”将是字典中的长整型值 10000。此外，如果用户指定一个没有任何单位的值，那么将以毫秒为单位对此值进行求值。例如，值“10”将求值为 10 毫秒。但是，您也可以指定 duration 类型，以便它依照不同的单位进行求值。例如，对 ibm:type="duration(s)" 指定值“10”将求值为 10 秒，并作为 10 存储在字典中。

以下列表将显示可能类型：

- duration(h)
- duration(m)
- duration(s)
- duration(ms)
- duration

指定 duration 和指定 duration(ms) 没有差别。

👉 注：

最佳实践：总是在值中包含单位，并使用最易于阅读的单位来表示值。例如，针对 ibm:type="duration(s)" 不指定值“7200”，而是将值指定为“2h”。

以下示例显示了 duration 类型：

- `<AD id="timeout" type="String" ibm:type="duration(s)".../>`
- `<AD id="timeout" type="String" ibm:type="duration".../>`

Location

location 类型可让 UI 工具以更有帮助的方式来显示属性，而这些属性表示各种文件和目录位置。它不影响运行时环境进行的处理。字典对象总是为字符串。

以下示例显示可能类型：

location

对文件进行引用。引用可以是绝对文件、相对文件，也可以是文件的 URL。

location(file)

通过使用绝对或相对文件路径来引用文件。

location(dir)

通过使用绝对或相对文件路径来引用目录。

location(url)

在 URL 末尾对文件进行引用。

以下示例显示了 location 类型：

```
<AD id="location" name="%appmgr.location.name" description="%appmgr.location
.desc" type="String" required="true" ibm:type="location"/>
```

Password

这种 password 类型用于密码字段。如果使用，那么字典对象是 com.ibm.wsspi.kernel.service.utils.SerializableProtectedString 的实例。密码字段的值不会记录在跟踪文件中。有效编码选项为 `xor` 和 `aes`。

以下示例显示了 password 类型：

```
<AD id="password" type="String" ibm:type="password".../>
```

Hashed password

此 passwordHash 类型与 password 类型相似，用于散列密码字段。如果使用，那么字典对象是 com.ibm.wsspi.kernel.service.utils.SerializableProtectedString 的实例。散列密码字段的值不会记录在跟踪文件中。有效的编码选项为 `xor`、`aes` 和 `hash`。

使用 PasswordUtil.encode(String, String, Map) 方法并指定以下参数来对散列密码验证新密码：

1. 新密码。
2. 散列算法，通过调用 PasswordUtil.getCryptoAlgorithm 方法获取。此散列算法必须与该散列密码的算法匹配。
3. 属性对象，其中某个属性使用 PasswordUtil.PROPERTY_HASH_ENCODED 来表示密钥，并使用散列密码来表示值。

如果 PasswordUtil.encode 的返回值与散列密码相同，那么密码匹配。

以下示例显示 passwordHash 类型：

```
<AD id="hashedPassword" type="String" ibm:type="passwordHash".../>
```

Pid

pid 类型用来对配置中的另一个对象进行引用。它与 ibm:reference 属性一起使用，且支持在 server.xml 文件中嵌套元素；请参阅嵌套配置元素（见第 1258 页）。

以下示例显示了 pid 类型：

```
<AD id="fooRef" type="String" ibm:type="pid" ibm:reference="com.ibm.ws.foo".
../>
```

OnError

onError 类型会在字典中产生 onError 枚举类型的实例。可能的值为 WARN、FAIL 和 IGNORE。

以下示例显示了 onError 类型：

```
<AD id="errorBehavior" type="String" ibm:type="onError".../>
```

用户界面元类型扩展

将此命名空间添加至 metatype.xml 文件以使用以下扩展：

```
xmlns:ibmui="http://www.ibm.com/xmlns/appservers/osgi/metatype/ui/v1.0.0"
```

ibmui:localization

localization 扩展用来指定元类型本地化文件。元类型本地化文件用来查找其他 UI 扩展的标签和描述的翻译。在大多数情况下，ibmui:localization 扩展的值与 <Metadata> 元素上的 localization 属性匹配。

以下示例显示了 ibmui:localization 扩展：

```
<OCD id="com.ibm.ws.jdbc.dataSource.properties"
        name="%properties"
    description="%properties.desc"
      ibmui:localization="OSGI-INF/l10n/metatype">
   <AD id="username".../>
</OCD>
```

ibmui:extraProperties

extraProperties 扩展用来指示可以在此配置上设置任意的配置属性集。

以下示例显示了 ibmui:extraproperties 扩展：

```
<OCD id="com.ibm.ws.jdbc.dataSource.properties"
    name="%properties" description="%properties.desc" ibmui:extraProperties="true">
   <AD id="username".../>
</OCD>
```

在元类型本地化文件中查找与扩展相关联的标签和描述（如果已使用 ibmui:localization 扩展来指定了元类型本地化文件）。对于扩展标签，请先检查 `extraProperties.<ocd id>.name` 键，然后检查 `extraProperties.name` 键。对于扩展描述，请先检查 `extraProperties.<ocd id>.description` 键，然后检查 `extraProperties.description` 键。

ibmui:group

group 扩展用来指定属性属于组。在用户界面中，注解为同一个组的属性分为一组。

以下示例显示了 ibmui:group 扩展：

- `<AD id="username" ibmui:group="userInfo".../>`
- `<AD id="password" ibmui:group="userInfo".../>`
- `<AD id="port" ibmui:group="hostInfo".../>`

在元类型本地化文件中查找组标签和描述信息（如果已使用 ibmui:localization 扩展来指定了元类型本地化文件）。对于组标签，请先检查 `<group>.<ocd id>.name` 键，然后检查 `<group>.name` 键。对于组描述，请先检查 `<group>.<ocd id>.description` 键，然后检查 `<group>.description` 键。

嵌套配置元素

可以使用元类型扩展来定义配置，而该配置可以表示为 server.xml 文件中的嵌套 XML 元素。

示例

以下示例显示如何在 server.xml 文件中支持此用户配置：

```
<family mother="jane" father="john">
  <child name="susie" age="8" />
  <child name="danny" age="5" />
</family>
```

元类型 XML 使用 `ibm:type="pid"` 和 `ibm:reference`，如以下示例中所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<metatype:MetaData
  xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
  xmlns:ibm="http://www.ibm.com/xmlns/appservers/osgi/metatype/v1.0.0">

  <OCD id="family" name="family">
    <AD id="mother" name="mother" type="String" default="Ma" />
```

```
    <AD id="father" name="father" type="String" default="Pa" />
    <AD id="child" name="child" ibm:type="pid" ibm:reference="child-pid"
      required="false" type="String" cardinality="6" />
  </OCD>

  <Designate pid="family">
    <Object ocdref="family" />
  </Designate>

  <OCD id="child" name="child" >
    <AD id="name" name="name" type="String" />
    <AD id="age" name="age" type="Integer" />
  </OCD>

  <Designate factoryPid="child-pid">
    <Object ocdref="child" />
  </Designate>

</metatype:MetaData>
```

以下示例显示接收 `family` 属性的代码如何使用 `ConfigurationAdmin` 服务来获取 `child` 属性集：

```
public void updated(Dictionary<String, ?> properties)
     throws ConfigurationException {

  Set<String> pids = new HashSet<String>();
  String mother = "null";
  String father = "null";

  try {
    if (properties != null) {
      mother = (String) properties.get("mother");
      father = (String) properties.get("father");
      String[] children = (String[]) properties.get("child");
      if (children == null || children.length == 0) {
        return;
      }

      // Get the configuration admin service
      ConfigurationAdmin configAdmin = null;
      ServiceReference configurationAdminReference =
        bundleContext.getServiceReference(ConfigurationAdmin.class.getName());

      if (configurationAdminReference != null) {
        configAdmin = (ConfigurationAdmin)
          bundleContext.getService(configurationAdminReference);
      }

      for (String childPid : children) {
        pids.add(childPid);
        Configuration config = configAdmin.getConfiguration(childPid);
        String name = (String) config.getProperties().get("name";
        Integer age = (Integer) config.getProperties().get("age");
      }
    }
  }

  catch (Exception e) {
    e.printStackTrace();
  }
}
```

对配置元数据进行本地化

每个元数据条目的名称和描述属性都可以本地化，而翻译的字符串会打包到语言特定属性文件。

以下示例显示如何在元类型文件的头中指定已本地化文件的位置：

```
<metatype:MetaData xmlns:metatype="http://www.osgi.org/xmlns/metatype/v1.1.0"
    localization="OSGI-INF/I10n/metatype">
```

其中，OSGI-INF/I10n 是已翻译属性文件在捆绑软件中的位置，而 metatype 是缺省语言属性文件的前缀。例如，如果缺省值（通常是英文）位于文件 metatype.properties 中，那么每个语言环境会添加其自己的后缀：metatype_fr.properties, metatype_es.properties 等。

与元类型 XML 文件（总是位于 OSGI-INF/metatype 目录）不同，已翻译的文件可以位于捆绑软件中的任何位置，并使用 localization 属性来指定。最好不要将属性文件与元类型 XML 文件一起放在 OSGI-INF/metatype 目录中；元类型服务会尝试将该位置中的所有内容都解析为 XML 文件，并且虽然这不会造成失败，但仍会在控制台中生成不需要的异常。xigemaAS 概要文件约定是将它们放入 OSGI-INF/I10n 目录，但这不是必需的。

在元类型 XML 文件中，要显示某一值是已本地化的字符串，可以在值的开头使用百分比符号。例如，可在元类型 XML 文件中使用以下定义：

```
<AD name="%client.inactivity.timeout" description="%client.inactivity.timeout.desc"
    id="clientInactivityTimeout" required="false" type="Integer" default="60" />
```

并且可在属性文件中使用以下定义：

```
client.inactivity.timeout=Client inactivity timeout
client.inactivity.timeout.desc=The maximum duration, in seconds, between transactional requests
from a remote client. 任何超出此超时值的客户机不活动时间段都将导致在此应用程序服务器中回滚事务。
```

提供应用程序端点

要将 xigemaAS 功能部件作为 Web 应用程序来提供，可以在该功能部件中包含一个或多个 Web 应用程序捆绑软件 (WAB)。WAB 是含有 Web-ContextPath 清单头的 OSGi 捆绑软件。

- 要在功能部件的捆绑软件中启用 WAB，请向功能部件的 .mf 文件中的 Subsystem-Content: 头添加 com.ibm.wsspi.appserver.webBundle-1.0 功能部件：

```
Subsystem-Content:
 my.user.feature.bundle; version="[1,1.0.100)",
 com.ibm.wsspi.appserver.webBundle-1.0; type="osgi.subsystem.feature"
```

保护应用程序端点

可以通过执行下列步骤来保护功能部件的应用程序端点：

1. 在功能部件的 .mf 文件中，向 Subsystem-Content: 头中添加 com.ibm.wsspi.appserver.webBundleSecurity-1.0 功能部件。

   此添加操作将导致对任何受保护的 Servlet（在功能部件捆绑软件的 WEB-INF/web.xml 文件中指定）进行认证，并启用基于角色的授权。您还可以对 WEB-INF/web.xml 文件中所定义的任何角色指定用户、组和特殊主体。

```
Subsystem-Content:
 my.user.feature.bundle; version="[1,1.0.100)",
 com.ibm.wsspi.appserver.webBundleSecurity-1.0; type="osgi.subsystem.feature"
```

2. 要将角色映射至用户、组和特殊主体，请执行下列步骤：

   a. 将 IBM-Authorization-Roles 头添加至 OSGi 捆绑软件的 MANIFEST.MF 文件。

   此头必须指定一个名称，此名称是您在 server.xml 文件中指定的角色映射的标识。

   ```
   IBM-Authorization-Roles: my.feature.role.map
   ```

   b. 在 server.xml 文件中，添加 `authorization-roles` 元素以将角色名称映射至用户和组。

   `authorization-roles` 元素的 `id` 属性必须与 MANIFEST.MF 文件中的 IBM-Authorization-Roles 头具有相同的值。为您想要对其指定用户和组的每个角色添加一个 `<security-role>` 子元素。

   ```
   <authorization-roles id="my.feature.role.map">
     <security-role name="employee">
       <special-subject type="ALL_AUTHENTICATED_USERS"/>
     </security-role>
     <security-role name="manager">
       <user name="bob"/>
       <user name="mary"/>
       <group name="managers"/>
     </security-role>
   </authorization-roles>
   ```

xigemaAS SPI 实用程序

xigemaAS 提供了服务编程接口 (SPI) 来完成各种任务。

资源位置符号

通过使用表示符号位置的变量，可以增强 xigemaAS 用户配置的可移植性。使用这些变量有助于防止编写绝对路径的代码，否则，用户配置会很脆弱且可移植性不强。接收配置属性的功能部件代码可能必须处理包含此类变量的值。

xigemaAS概要文件的位置服务可用于将符号位置解析为物理资源。例如，符号位置 `${wlp.install.dir}/myFile` 可以映射到xigemaAS 概要文件的安装目录中的本地文件 myFile。大多数方法会返回打包了物理资源的 `WsResource` 对象，但您也可以使用 `resolveString` 方法将符号位置变换为可用于获取 `File` 对象的 `String`。

位置服务的名称是 `com.ibm.wsspi.kernel.service.location.WsLocationAdmin`，并且它是由xigemaAS内核提供，因此您不必在 server.xml 文件中指定功能部件以使该功能部件变为可用。

符号

`com.ibm.wsspi.kernel.service.location.WsLocationConstants` 类定义对目录位置进行引用的符号：

- `/`
- `server.config.dir`
- `server.output.dir`
- `server.workarea.dir`
- `shared.app.dir`
- `shared.config.dir`
- `shared.resource.dir`
- `wlp.install.dir`
- `wlp.server.name`
- `wlp.user.dir`
- `usr.extension.dir`

有关每个符号的意义，请参阅目录位置和属性（见第 1101 页）。

监视本地文件更改

xigemaAS 具有高度动态行为，用以响应配置、应用程序和其他资源的更改。此动态行为大体上以监视本地文件系统中是否发生更改为基础。执行此监视的服务是通过 `FileMonitor` SPI 提供给所有 xigemaAS 功能部件。文件监视器服务是由 xigemaAS 内核提供，因此您不必在 server.xml 文件中指定功能部件以使该功能部件变为可用。

`FileMonitor` SPI 提供不同的属性以指定监视的资源及监视频率。必须实现 `FileMonitor` 接口，并向服务注册表注册实现类。

```
...
import com.ibm.wsspi.kernel.filemonitor.FileMonitor;
...

public class MyFileMonitor implements FileMonitor {
  ...
    private final BundleContext bundleContex;
    ...

  public MyFileMonitor(BundleContext bundleContext) {
    this.bundleContext = BundleContext;
    ...
  }

  public ServiceRegistration<FileMonitor> monitorFiles(Collection<String> paths, long
 monitorInterval) {
    ...
    final Hashtable<String, Object> fileMonitorProps = new Hashtable<String, Object>();
    fileMonitorProps.put(FileMonitor.MONITOR_FILES, paths);
    fileMonitorProps.put(FileMonitor.MONITOR_INTERVAL, monitorInterval);
    ...
    return bundleContext.registerService(FileMonitor.class, this, fileMonitorProps);
  }
  ...
}
```

配置 xigemaAS 概要文件中功能部件的跟踪和日志记录

可以将 xigemaAS 概要文件的跟踪和日志记录机制用于 xigemaAS 功能部件。记录服务是 xigemaAS 内核的一部分，因此您不必在 server.xml 文件中指定功能部件，也可以使用该功能部件。

xigemaAS 提供了下列 SPI 来将跟踪和日志记录机制集成到您的定制功能部件代码：

`com.ibm.websphere.ras`

`com.ibm.websphere.ras` 包提供了用来记录消息和跟踪记录的类，以及一些扩展点。通常，功能部件代码可以使用 `java.util.logging` 包来记录跟踪和消息，以及通过 xigemaAS 日志记录配置来控制输出，但 xigemaAS 包的扩展功能有时很有用，而在禁用跟踪的情况下，跟踪保护就稍微有效一些。

`com.ibm.websphere.ras.annotations`

`com.ibm.websphere.ras.annotations` 包提供用来与其他包中的类一起使用的注解。例如，可以使用 `@Sensitive` 注解来防止带注解变量的内容出现在跟踪或消息输出中。

`com.ibm.ws.ffdc`

`com.ibm.ws.ffdc` 包提供的工具可以写入首次故障数据捕获 (FFDC) 记录以帮助调试非预期的异常。

com.ibm.wsspi.logging

com.ibm.wsspi.logging 包提供日志和 FFDC 记录的拦截点。

1. 下列步骤说明如何配置示例 xigemaAS 功能部件（称为 myfeature），以使用 xigemaAS 概要文件的跟踪和日志记录机制。
1. 对功能部件 myfeature 指定消息文件的位置，以及 com.ibm.websphere.ras.TraceComponent 类所需的组名。

```
import java.util.ResourceBundle;

public class myFeatureConstants {

    public static final String TR_RESOURCE_BUNDLE =
        "com.mycompany.myFeature.internal.resources.FeatureMessages";

    public static final String TR_GROUP = "myFeature";

    public static final ResourceBundle messages =
 ResourceBundle.getBundle(TR_RESOURCE_BUNDLE);

}
```

2. 在功能部件服务代码的实现类中，调用 com.ibm.websphere.ras.TraceComponent 类的 register() 方法，以向 xigemaAS 概要文件随附的跟踪管理器注册实现类。然后，可以配置跟踪管理器以跟踪功能部件的 DS 方法。

```
...
import com.ibm.websphere.ras.Tr;
import com.ibm.websphere.ras.TraceComponent;


public class myFeatureServiceImpl {

    private static final TraceComponent tc = Tr.register(myFeatureServiceImpl.class);


    protected void activate(ComponentContext cc, Map<String, Object> newProps) {
        if (tc.isDebugEnabled()) {
            Tr.debug(tc, "myFeatureComponentImpl activated"); }
...
```

3. 使用 TraceOptions 注解来指定跟踪组名和消息束名。

```
@TraceOptions(traceGroup = myFeatureConstants.TR_GROUP, messageBundle =
    myFeatureConstants.TR_RESOURCE_BUNDLE)
package com.mycompany.myFeature;

import com.ibm.websphere.ras.annotation.TraceOptions;
import com.mycompany.myfeature.internal.myFeatureConstants;
...
```

生成首次故障数据捕获 (FFDC) 记录

FFDC 记录包含代码捕捉到意外异常时所记录的异常堆栈及其他可选数据。com.ibm.ws.ffdc.FFDCFilter 类上的方法用来生成这些记录，并且有一些方法可能促使捕获各种数据。

FFDCFilter 类的典型用法如下所示：

```
try{
            // … do something
    } catch (Exception e) {
            FFDCFilter.processException(e, getClass().getName(), unique-probe-id);
```

```
            if (TraceComponent.isAnyTracingEnabled() && tc.isDebugEnabled()) {
                Tr.debug(tc, "Exception when doing something; " + e);
            }
            return;
    }
```

其中，源标识（本示例中的类名）和唯一探测器标识（通常为原代码行号）会组合，以提供生成结果记录的源代码中的确切位置。缺省情况下，记录会写入 ${server.output.dir}/logs/ffdc 目录。

持续发生异常的情况下，FFDC 记录所使用的文件空间受重复记录自动过滤功能限制。对于任何匹配源标识、探测器标识和异常名称，每天最多写入带独有消息的 10 个异常。

功能部件代码可以通过向 FFDC 类注册 `com.ibm.ws.ffdc.DiagnosticModule` 实现，将数据提供给 FFDC 记录。功能部件代码也可以通过向 FFDC 类注册 `com.ibm.wsspi.logging.IncidentForwarder` 实现来拦截 FFDC 记录。

添加 Web Service 全局处理程序

需要对所有 Web Service 端点注册 Web Service 处理程序的组件必须实现处理程序接口并在服务注册表中注册该实现。

全局处理程序服务由 jaxws-2.2、jaxrs-1.1、jaxrs-2.0 或 jaxrs-2.0 client 提供，所以您必须在 server.xml 文件中指定以下功能部件或功能部件组合：

- jaxws-2.2
- jaxrs-1.1
- jaxrs-2.0
- jaxrs-2.0 客户机
- jaxws-2.2 和 jaxrs-1.1
- jaxws-2.2 和 jaxrs-2.0
- jaxws-2.2 和 jaxrs-2.0 客户机

处理程序 SPI 提供不同属性以指定 ENGINE_TYPE、FLOW_TYPE 以及客户端 (IS_CLIENT_SIDE) 或处理程序生效的服务器端 (IS_SERVER_SIDE)。

必须实现处理程序接口并在服务注册表中注册实现类。

每个 xigemaAS SPI 的 Java API 文档以独立压缩文件的形式在 ${wlp.install.dir}/dev 目录的某个 Javadoc 子目录中提供。

在 xigemaAS 内公开 REST 端点

在 xigemaAS SPI 中使用 REST 处理程序框架以公开新的 REST 端点。

REST 处理程序框架供 xigemaAS 扩展程序在公开新的 REST 端点时使用。可在一个 OSGi 组件或一组组件中公开 REST 端点。

1. 创建 OSGi 组件，该组件在侦听附加至 /vsettan/api 并实现 `com.ibm.wsspi.rest.handler.RESTHandler` 接口的子根时注册自身；例如：

```
@Component(service = { RESTHandler.class },
           configurationPolicy = ConfigurationPolicy.IGNORE,
           immediate = true,
           property = { "service.vendor=xigemaAS",
                        RESTHandler.PROPERTY_REST_HANDLER_ROOT + "=/myTest/abc" })
public class RESTHANDLERTest1 implements RESTHandler {
...
```

2. 将该组件打包至作为已扩展用户功能的一部分的 OSGi 捆绑软件。

3. 确保您的功能部件包括 OSGi 子系统内容：

```
com.ibm.websphere.appserver.restHandler-1.0; type="osgi.subsystem.feature"
```

4. 在 server.xml 文件中配置 *SSL* 证书。
5. 在 server.xml 文件中将用户或组配置到管理员角色。
   - 映射 *xigemaAS* 的管理员角色（见第 1180 页）
6. 启动您的功能部件。

   启动该功能部件会启动 REST 处理程序框架并注册您的 OSGi 组件。该功能部件启动后，您可调用 `https://<host>:<https_port>/vsettan/api/myTest/abc`。

包含受保护的功能部件

您的功能部件可以通过在功能部件清单文件的 Subsystem-Content 头中列出一个或多个其他的功能部件来包含功能部件。可以包含与您自己的功能部件在同一个产品扩展中的任何功能部件；如果包含的功能部件位于不同的产品扩展中，或者位于 xigemaAS 概要文件中，那么它必须具有公共或受保护的可见性。

包含的功能部件必须由其 `Subsystem-SymbolicName` 所指定，并且具有类型 `"osgi.subsystem.feature"`；例如：

```
Subsystem-Content:
  com.ibm.wsspi.appserver.webBundle-1.0; type="osgi.subsystem.feature",
  com.ibm.websphere.appserver.json-1.0; type="osgi.subsystem.feature"
```

有关 xigemaAS 概要文件公共功能部件的信息，请参阅*xigemaAS* 功能部件（见第 906 页）。以下部分描述了 xigemaAS 概要文件的受保护功能部件。

xigemaAS 概要文件的受保护功能部件

应用程序管理器

此功能部件提供用于实现新应用程序容器的高级功能。

Subsystem-SymbolicName：com.ibm.websphere.appserver.appmanager-1.0。

提供的 API 和 SPI：

- dev/api/vsettan/com.vsettan.xigema.appserver.api.basics_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.application_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar

类加载器服务

此功能部件提供用于实现新应用程序容器的高级功能。

Subsystem-SymbolicName：com.ibm.websphere.appserver.classloading-1.0。

提供的 API 和 SPI：

- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.classloading_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar

容器服务

此功能部件提供用于实现新应用程序容器的高级功能。

Subsystem-SymbolicName：com.ibm.websphere.appserver.containerServices-1.0。

提供的 API 和 SPI：

- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.containerServices_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.anno_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.javaeedd_1.1.10.jar

事务管理器 **1.1**

此功能部件提供符合 JTA 1.1 的事务管理器。

Subsystem-SymbolicName：com.ibm.websphere.appserver.transaction-1.1。

提供的 API 和 SPI：

- dev/api/spec/com.vsettan.as.javaee.transaction.1.1_1.0.10.jar
- dev/api/vsettan/com.vsettan.xigema.appserver.api.transaction_1.1.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.containerServices_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.anno_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.javaeedd_1.1.10.jar

事务管理器 **1.2**

此功能部件提供符合 JTA 1.2 的事务管理器。

Subsystem-SymbolicName：com.ibm.websphere.appserver.transaction-1.2。

提供的 API 和 SPI：

- dev/api/spec/com.vsettan.as.javaee.transaction.1.2_1.0.10.jar
- dev/api/spec/com.vsettan.xigema.appserver.api.transaction_1.1.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.containerServices_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.anno_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.javaeedd_1.1.10.jar

Web 捆绑软件

此功能部件支持在功能部件中使用 Web 应用程序捆绑软件 (WAB)。如果功能部件提供应用程序端点，请包含此功能部件，如提供应用程序端点（见第 1260 页）中所述。

Subsystem-SymbolicName：com.ibm.wsspi.appserver.webBundle-1.0。

提供的 API 和 SPI：

- dev/api/spec/com.vsettan.as.javaee.servlet.3.0_1.0.10.jar
- dev/api/vsettan/com.vsettan.xigema.appserver.api.servlet_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.servlet_1.1.10.jar

Web 捆绑软件安全性

此功能部件支持将安全性应用到 Web 捆绑软件；请参阅保护应用程序端点（见第 1260 页）。

Subsystem-SymbolicName：com.ibm.wsspi.appserver.webBundleSecurity-1.0。

提供的 API 和 SPI：

- dev/api/spec/com.vsettan.as.javaee.servlet.3.0_1.0.10.jar
- dev/api/vsettan/com.vsettan.xigema.appserver.api.servlet_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.servlet_1.1.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.containerServices_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.anno_1.0.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.artifact_1.2.10.jar
- dev/spi/vsettan/com.vsettan.xigema.appserver.spi.javaeedd_1.1.10.jar

查找 OSGi 应用程序

可以使用 `org.apache.aries.blueprint` 包中的类来扩展 OSGi 应用程序编程模型；此第三方 SPI 是通过 `blueprint-1.0` 服务器功能部件提供。必须访问 OSGi 应用程序捆绑软件才能应用扩展。在 xigemaAS 概要文件中，OSGi 应用程序作为子系统来运行。要查找 OSGi 应用程序，您可以在用户功能部件中创建 `ServiceTracker`。

本主题描述了用户功能部件的开发者如何找到正在运行的 OSGi 应用程序。提供 OSGi 应用程序编程模型扩展的用户功能部件通常需要此任务。例如，新用户功能部件可提供此类扩展，方式是实现新的捆绑软件扩充器（通常称为容器），或者更简单一点，就是跟踪并调用从某些 OSGi 应用程序内部发布的服务。

此类用户功能部件必须使用正在运行的特定 OSGi 应用程序的 `BundleContext` 来创建新的 `BundleTracker` 和 `ServiceTracker` 实例。可以从与 OSGi 应用程序相关联的 `org.osgi.service.subsystem.Subsystem` 获取此 `BundleContext`。以下过程描述了如何获取该子系统服务。

1. 要通过在用户功能部件中创建 `ServiceTracker` 来查找 OSGi 应用程序，请完成下列步骤：
1. 构造以要查找的子系统为目标的 `org.osgi.framework.Filter`。
2. 创建 `org.osgi.util.tracker.ServiceTracker`，它使用步骤 1 中的过滤器来获取与您想要查找的 OSGi 应用程序相关联的 `org.osgi.service.subsystem.Subsystem` 服务。

   此 `Subsystem` 服务实例提供您处理 OSGi 应用程序所需的一切。

以下示例说明如何使用 `ServiceTracker` 在用户功能部件中查找符号名称为 my.app 的应用程序：

```
import org.osgi.framework.BundleContext;
import org.osgi.service.subsystem.Subsystem;
import org.osgi.util.tracker.ServiceTracker;
import org.osgi.util.tracker.ServiceTrackerCustomizer;
```

在以下代码摘录中，变量 `ctx` 是其中一个用户功能部件捆绑软件的 `BundleContext`：

```
String SERVICE_FILTER = "(&(objectClass=org.osgi.service.subsystem.Subsystem)
                          (subsystem.type=osgi.subsystem.application)
(subsystem.symbolicName=my.app))"

      org.osgi.framework.Filter filter = ctx.createFilter(SERVICE_FILTER);
```

可以将最后的“null”参数替换为用于实现 `ServiceTrackerCustomizer<Subsystem, Subsystem>` 的类的实例：

```
org.osgi.util.tracker.ServiceTracker<Subsystem, Subsystem> str = new ServiceTracker<Subsystem,
 Subsystem>(ctx, filter, null);
```

可以构造 SERVICE_FILTER 以利用如下所示的常量：

```
org.osgi.framework.Constants.OBJECTCLASS;
```

```
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_SYMBOLICNAME_PROPERTY;
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_TYPE_APPLICATION;
org.osgi.service.subsystem.SubsystemConstants.SUBSYSTEM_TYPE_PROPERTY;
```

在 xigemaAS 功能部件中使用 JNDI 缺省命名空间开发

您可以使对象在缺省 Java™ 命名和目录接口 (JNDI) 名称空间中可用。为此，您必须在 OSGi 服务注册表中使用 `osgi.jndi.service.name` 服务属性注册该对象。`osgi.jndi.service.name` 的值是必需的 JNDI 名称。同样，要在缺省 JNDI 名称空间中查找对象，可以使用 `osgi.jndi.service.name` 服务属性来搜索 OSGi 服务注册表。`osgi.jndi.service.name` 的值为 JNDI 名称。

与显式调用 `Context.bind` 或 `Context.lookup` 相比，使用服务注册表具有下列优势：

- 启用了 `jndi-1.0` 时，您的功能部件将正常工作，但是您的功能部件不需要显式依赖 JNDI。
- 移除了您的功能部件时，您不需要显式解除对象与 JNDI 的绑定，因为当捆绑软件停止时，OSGi 框架会自动注销服务。
- 您可以使用声明式服务或 `ServiceFactory`（而不使用 `Reference` 和 `ObjectFactory`）来轻松实现延迟初始化。

1. 使用 `osgi.jndi.service.name` 属性和 JNDI 名称来注册服务。

   有关注册服务的更多信息，请参阅注册 *OSGi* 服务（见第 1244 页）。

2. 更新 metatype.xml 以允许在服务器配置中指定 JNDI 名称。

   要允许用户为服务指定 JNDI 名称，您应该使用 `jndiName id`，以便在 xigemaAS 概要文件运行时与其他功能部件一致，例如：

   ```
   <AD id="jndiName" name="JNDI name" description="JNDI name for a widget."
    type="String" ibm:unique="jndiName"/>
   ```

   可以使用内部属性为 `osgi.jndi.service.name` 服务属性自动设置 `jndiName` 属性的值，例如：

   ```
   <AD id="osgi.jndi.service.name" name="internal" description="internal"
    type="String" default="${jndiName}"/>
   ```

   有关 OSGi 元类型的更多信息，请参阅高级配置（见第 1251 页）。

3. 如果您需要 Java EE 资源引用信息，请实现 `ResourceFactory` 接口。

   如果服务需要 Java EE 资源引用信息（例如，`res-auth`），那么您可以在 OSGi 服务注册表中使用 `jndiName` 和 `creates.objectClass` 属性来注册 `ResourceFactory`。会使用 `osgi.jndi.service.name` 属性来自动重新注册 `ResourceFactory` 服务。

   例如：

   ```
       import com.ibm.wsspi.resource.ResourceFactory;
       public class WidgetResourceFactory implements ResourceFactory { ... }

       Properties properties = new Properties();
       properties.put(ResourceFactory.JNDI_NAME, "widget/abc");
       properties.put(ResourceFactory.CREATES_OBJECT_CLASS,
   Widget.class.getName());
       bundleContext.registerService(ResourceFactory.class, new
   WidgetResourceFactory(), properties);
   ```

此外，可以使用声明式服务和元类型来自动注册服务。在这种情况下，您可以指定 `creates.objectClass` 属性作为声明式服务属性。您不需要指定 `jndiName` 属性，因为在步骤 *2*（见第 1268 页），会从用户配置中使用 metatype.xml 文件中的 `<AD id="jndiName">` 元素来自动设置该属性；您也不需要在 metatype.xml 文件中具有 `<AD id="osgi.jndi.service.name>` 元素，因为将自动重新注册 `ResourceFactory` 服务。

4. 使用 `osgi.jndi.service.name` 属性和 JNDI 名称来查找对象。

   例如：

   ```
   bundleContext.getServiceReference(DataSource.class,
    "(osgi.jndi.service.name=jdbc/myds)");
   ```

   此外，您可以使用 `jndiName` 和 `creates.objectClass` 属性来查找 `ResourceFactory`。

5. 更新 metatype.xml 以允许在 server.xml 中使用资源标识来指定资源。无论该资源是否具有 `jndiName`，都允许访问该资源。

   例如，

   ```
   <AD id="dataSourceRef" type="String" ibm:type="pid"
    ibm:reference="com.ibm.ws.jdbc.dataSource" cardinality="1"
    name="%dataSourceRef" description="%dataSourceRef.desc"/>
   ```

   如果您正在使用声明式服务，那么可以使用内部属性并通过过滤器来设置 `.target` 服务属性。例如，如果您的声明式服务组件具有名为 `dataSource` 的引用，那么可以使用以下属性定义来确保使用 dataSourceRef 配置属性所引用的 dataSource。

   ```
   <AD id="dataSource.target" type="String" default="(service.pid=
   ${dataSourceRef})" ibm:final="true" name="internal"
    description="internal"/>
   ```

将定制 TAI 作为 xigemaAS 功能部件来开发

要将定制 TAI 作为 xigemaAS 功能部件来开发，您可以实现 xigemaAS 概要文件服务器中提供的 `com.ibm.wsspi.security.tai.TrustAssociationInterceptor` 接口，并创建产品扩展。

有关定制 TAI 的概述，请参阅为 *xigemaAS* 开发定制 *TAI*（见第 1407 页）。

有关产品扩展的更多信息，请参阅产品扩展（见第 1028 页）。

注：如果有多个 TAI，那么可使用用户功能部件或共享库来配置全部 TAI。不要同时使用两个 TAI 配置。

1. 实现定制 TAI。有关更多信息，请参阅为 *xigemaAS* 开发定制 *TAI*（见第 1407 页）。
2. 将实现类转换为 OSGi 服务。

   可以采用下列其中一种方法来执行转换：

   - 将定制 TAI 类转换为声明式服务 (DS) 组件。有关更多信息，请参阅向 *OSGi* 声明式服务声明服务（见第 1248 页）。
   - 编写作为 DS 组件的新定制 TAI 类并将其委派给定制 TAI 类。
   - 使用 OSGi 核心 API 直接在服务注册表 (SR) 中注册定制 TAI 类。有关更多信息，请参阅使用 *OSGi* 服务注册表（见第 1244 页）。
3. 将定制 TAI 打包为 OSGi 捆绑软件并导出定制 TAI 服务。

4. 创建功能部件清单来包含 OSGi 捆绑软件。有关功能部件清单文件的更多信息，请参阅 *xigemaAS* 功能部件清单文件（见第 1234 页）。
5. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。例如：

```
<featureManager>
   ...
  <feature>usr:customTaiSample-1.0</feature>
</featureManager>
```

动态内容管理

通常，您会通过在功能部件清单文件的 Subsystem-Content 头中列示捆绑软件以将捆绑软件安装到运行时环境中。但是，您还可通过将用户编写捆绑软件作为用户编写功能部件的 Subsystem-Content 的一部分来动态添加和移除 OSGi 捆绑软件。用户编写捆绑软件包含 OSGi 捆绑软件上下文以安装和控制其他捆绑软件。

👉 注：在以下部分中，用户编写功能部件称为 UserFeatureA，用户编写捆绑软件称为 FeatureBundleA。

在 xigemaAS 中安装、启动、停止和卸载捆绑软件

安装捆绑软件

通过使用下列其中一个方法，可编写 FeatureBundleA 以获取 OSGi 捆绑软件上下文 org.osgi.framework.BundleContext：

- 实现 BundleActivator 接口 org.osgi.framework.BundleActivator。启动方法的 BundleContext 参数由 OSGi 框架传递，用户编写捆绑软件激活时此参数对该捆绑软件可用。有关 BundleActivator 接口的更多信息，请参阅开发使用简单激活的 *OSGi* 捆绑软件（见第 1243 页）。
- 实现可用规范（例如，OSGi 声明式服务或 Blueprint），此规范允许您通过另一方法或接口访问捆绑软件上下文。有关更多信息，请参阅使用 *OSGi* 声明式服务来编写高级功能部件（见第 1247 页）。

FeatureBundleA 获取捆绑软件上下文后，可使用 installBundle(String location) 或 installBundle(String location, InputStream stream) 方法来安装其他捆绑软件。

动态安装的捆绑软件会在缺省重新启动时恢复状态。在清除缓存的重新启动后，它们不会被保留，需要重新安装。有关更多详细信息，请参阅捆绑软件高速缓存。

启动捆绑软件

如果要启动已安装捆绑软件，应由安装捆绑软件 FeatureBundleA 对该捆绑软件调用 start 方法。

停止和卸载捆绑软件

如果已从服务器配置中移除用户编写功能部件 UserFeatureA，那么 FeatureBundleA 会停止并且还会卸载。卸载 FeatureBundleA 会导致卸载 FeatureBundleA 安装的所有捆绑软件（如果它们尚未卸载）。系统会对每个捆绑软件调用 org.osgi.framework.Bundle.uninstall() 方法，此方法会停止并卸载该捆绑软件。如果通过任何其他手段卸载 FeatureBundleA，那么此卸载过程也适用。

如果服务器停止时从服务器配置中移除 UserFeatureA，那么系统会在下一次服务器启动时移除 UserFeatureA 安装的捆绑软件。如果捆绑软件的启动级别仍为缺省值，那么系统会在捆绑软件重新启动前将其移除。如果捆绑软件的启动级别已修改，那么系统可能在捆绑软件重新启动后才会将其移除。

FeatureBundleA 根据 OSGi 核心规范使用 org.osgi.framework.Bundle 和 org.osgi.framework.BundleContext 接口执行其他生命周期管理任务。

xigemaAS 中的捆绑软件高速缓存、包可见性及编程模型支持

捆绑软件高速缓存

服务器关闭时，所有当前安装的捆绑软件会停止，OSGi 元数据会保存至捆绑软件高速缓存。缺省启动时，这些已安装捆绑软件返回至其先前状态。干净启动时，FeatureBundleA 安装的所有捆绑软件会删除其持久数据。因此，干净启动时不会恢复这些捆绑软件。FeatureBundleA 本身会恢复，因为它由功能部件管理器重新安装（只要 UserFeatureA 仍在服务器配置中）。如果要在干净启动后重新安装任何捆绑软件，应由 FeatureBundleA 执行重新安装。系统不会向您发出干净启动的通知，但您可使用 OSGi BundleContext getBundle(String location) 方法来检查是否安装了某个捆绑软件。

包可见性

动态安装且未列示在功能部件清单文件的 Subsystem-Content 头中的捆绑软件具有以下可见性：

- 动态安装的捆绑软件可导入当前配置的功能部件集提供的任何 API 和 SPI 包。
- 由同一产品扩展内任何其他捆绑软件导出且未声明为 API 或 SPI 的包对动态安装的捆绑软件不可见。
- 从动态安装的捆绑软件导出的包不能声明为 API 或 SPI。
- 导入从动态安装的捆绑软件导出的包时没有任何限制。

编程模型支持

动态安装的捆绑软件可使用 OSGi 企业规范的实现，只要相应运行时功能部件配置为启用这些捆绑软件。

2.5.2 xigemaAS 功能部件打包和安装

可以将 xigemaAS 功能部件打包并将其安装到 xigemaAS。

xigemaAS 功能部件可打包为子系统归档，如 OSGi 企业规范 (5.0) 中所定义。子系统归档是扩展名为 .esa 的压缩文件，内含该功能部件清单及构成该功能部件的资源文件。xigemaAS 开发者工具将使用子系统归档格式来导入和导出 xigemaAS 功能部件。

- 可以手动将 xigemaAS 功能部件安装至 xigemaAS。
  - 功能部件清单文件必须位于 ${wlp.install.dir}/lib/features 目录中，而且相关的捆绑软件必须位于 ${wlp.install.dir}/lib 目录中，才能将功能部件安装到 xigemaAS 概要文件内核；
  - 功能部件清单文件必须位于 ${wlp.user.dir}/extension/lib/features 目录中，而且相关的捆绑软件必须位于 ${wlp.user.dir}/extension/lib 目录中，才能将功能部件安装到用户配置；
  - 功能部件清单文件和相关的捆绑软件必须位于产品扩展目录中，才能将功能部件安装到产品扩展。产品扩展是在 ${wlp.user.dir}/etc/extension/lib/features 目录中使用 <extension-name>.properties 文件注册的。有关更多信息，请参阅产品扩展（见第 1028 页）。

为功能部件扩展提供产品信息

可以为功能部件扩展提供版本产品信息。

要为功能部件扩展提供版本产品信息，可以在扩展的 lib/versions 目录中为安装提供唯一命名的产品信息属性文件。文件扩展名必须为 .properties。

可以在产品信息属性文件中指定以下属性：

```
com.vsettan.xigema.productId=yourProductID
com.vsettan.xigema.productOwner=TheProductOwner
com.vsettan.xigema.productVersion=yourProductVersion
com.vsettan.xigema.productName=yourProductName
com.vsettan.xigema.productInstallType=yourProductInstallType
com.vsettan.xigema.productEdition=yourProductEdition
com.vsettan.xigema.productQualifier=yourProductQualifier
```

如果要覆盖特定的功能部件扩展信息，请在产品信息属性文件中包含以下属性：

```
com.vsettan.xigema.productReplaces=theProductIdToReplace
```

以下示例显示了使用 *version* 选项的 *productInfo* 脚本的输出，输出中显示了产品信息属性文件中所指定的产品名称条目和版本条目：

```
com.vsettan.xigema.productId=com.vsettan.xigema.appserver
com.vsettan.xigema.productOwner=Vsettan
com.vsettan.xigema.productVersion=9.1.x.x
com.vsettan.xigema.productName=xigemaAS
com.vsettan.xigema.productInstallType=Archive
com.vsettan.xigema.productEdition=N/A
com.vsettan.xigema.productLicenseType=N/A
```

```
Command:
productInfo version
```

```
Output:
Product : xigemaAS
Version:  9.1.x.x
LicenseType: Production
SN-KEY : TOP#A000001XXXXXX
Authorized : 华胜信泰
Project :  XX项目
```

2.5.3 在应用程序中嵌入 xigemaAS 服务器

可以使用 xigemaAS 所提供的系统编程接口 (SPI) 来配置、控制和监视应用程序中的 xigemaAS 服务器。

xigemaAS 提供了下列 SPI 来启动或停止 xigemaAS 服务器：

- `com.ibm.wsspi.kernel.embeddable.Server`
- `com.ibm.wsspi.kernel.embeddable.ServerBuilder`

使用 `Future` 对象来存储启动或停止操作的结果。嵌入式操作所使用的返回码与 `server` 命令所使用的返回码相同。有关返回码、服务器脚本所使用的 JVM 选项以及服务器脚本所使用的处理环境的更多信息，请参阅服务器命令选项（见第 1118 页）。

此外，当服务器正在启动、已启动或者已停止时，可以通过创建您自己的用于实现 `com.ibm.wsspi.kernel.embeddable.ServerEventListener` 接口的类来接收异步通知。

👉 注：要在应用程序中创建嵌入式服务器的实例，您必须执行下列步骤：

- 将 `ws-server.jar` 文件包含在类路径中。`ws-server.jar` 文件位于 xigemaAS 安装的 ${wlp.install.dir}/bin/tools 目录中。

- 指定目标服务器的名称。目标服务器必须存在。
- 可选：使用 -javaagent JVM 选项配置 ws-javaagent.jar 文件。ws-javaagent.jar 文件位于 xigemaAS 安装的 *${wlp.install.dir}*/bin/tools 目录中。建议您配置 ws-javaagent.jar 文件，但只有在使用的服务器功能（例如，监视或跟踪）需要此文件时，才必须进行此配置。如果联系 xigemaAS支持机构，那么您可能需要提供跟踪，如果是这样，那么必须使用 ws-javaagent.jar 文件启动服务器，即使通常不会使用它。

👉 注：在嵌入式环境中：

- 未检查环境变量，并且未读取 jvm.options 和 server.env 文件。
- 假定由调用者来管理 JVM 和环境。

1. 将 SPI 导入到调用者类，并定义在运行 xigemaAS 服务器时所需的参数。

```
import com.ibm.wsspi.kernel.embeddable.Server;
import com.ibm.wsspi.kernel.embeddable.ServerBuilder;
public class MyEmbeddedServer {
    serverName="defaultServer";
    userDir = new File("usr");
     File outputDir = new File("usr/servers/");
    ...
}
```

其中：

- serverName 是必需参数，并且必须与先前创建的服务器的名称相匹配。
- userDir（可选）用来设置用户目录的路径。缺省情况下，用户目录是 `${wlp.user.dir}`。
- outputDir（可选）用来设置输出目录的路径。缺省情况下，输出目录是 `${wlp.user.dir}/servers`。

2. 使用 `ServerBuilder` 类来初始化服务器。

```
    ServerBuilder sb = new ServerBuilder();
    Server xigemaasServer = sb.setName(serverName)
                             .setUserDir(userDir)
                             .setOutputDir(outputDir)
                             .build();
```

3. 调用 `Server.start()` 方法来启动服务器。将来调用 `get()` 以阻塞到启动操作完成为止。

使用下列其中一种方法来确定服务器是否已成功启动：

- 检查所返回的结果代码。
- 使用 `successful()` 方法。
- 如果服务器已启动，那么 `server.isRunning()` 方法将返回 true。

```
    Future<Result> startReturnCode = xigemaasServer.start();
    Result result = startReturnCode.get(); // block until operation complete, if necessary
    System.out.println("Start returned: success=" + result.successful() + ", rc=" +
result.getReturnCode() + ", ex=" + result.getException());
```

4. 调用 `Server.stop()` 方法以停止服务器。对 future 调用 `get()` 以进行阻止，直到停止操作完成。

使用以下某种方法来确定是否成功停止了服务器：

- 检查返回的结果代码。
- 检查所返回的结果代码。
- 使用 `successful()` 方法。

- 如果服务器已停止，那么 `server.isRunning()` 方法会返回 false。

```
    Future<Result> stopReturnCode = xigemaasServer.stop();
    Result result = stopReturnCode.get(); // block until operation complete, if necessary
    System.out.println("Stop returned: success=" + result.successful() + ", rc=" +
result.getReturnCode() + ", ex=" + result.getException());
```

5. 实现 ServerEventListener 接口。如果您实现 ServerEventListener 接口，那么在服务器启动或停止时会接收到通知。

```
// update the class declaration to indicate that it implements ServerEventListener
public class MyEmbeddedServer implements ServerEventListener {
    ...
    MyEmbeddedServer() throws ServerException {
        // set the listener via the server builder
        ServerBuilder sb = new ServerBuilder();
        Server xigemaasServer = sb.setName(serverName)
                                  .setServerEventListener(this)
                                  .build();
    }

    ...
    @Override
    public void serverEvent(ServerEvent event) {
        // provide an implementation of the serverEvent method
        System.out.println("serverEvent: " + event);
    }
}
```

2.5.4 从定制配置创建 xigemaAS 服务器

对于您需要的任何环境，可以从定制配置来创建 xigemaAS 服务器。

xigemaAS 服务器脚本创建命令提供了一个 --template 选项。可以使用此选项来支持从定制配置创建服务器，定制配置位于产品扩展的 templates/servers/<template-name> 子目录中。定制服务器模板必须至少包含 server.xml 文件，还可以包含任何配置文件，例如：bootstrap.properties 或 jvm.options。

1. 可以采用以下方式来使用 --template 选项：
   server create --template=<extension-name>:<template-name>

2.6 保护 xigemaAS 及应用程序

此信息适用于 xigemaAS 上部署的所有类型的应用程序。

xigemaAS 中的安全性支持所有 Servlet 3.0 安全性功能部件及受保护的 Java™ JMX 连接。下列 xigemaAS 功能部件适用于 xigemaAS 中的安全性：

- 当提供了 `servlet-3.0` 功能部件时，`appSecurity-2.0` 将对 Web 应用程序启用安全性，当提供了 `ejbLite-3.1` 功能部件时，将对 EJB 启用安全性。
- `ssl-1.0` 使用 HTTPS 来启用 SSL 连接。
- `restConnector-1.0` 支持 JMX 客户机通过基于 REST 的连接器进行远程访问。
- `oauth-2.0` 使用 OAuth 2.0 协议来支持对资源授权。
- `ldapRegistry-3.0` 支持 LDAP 用户注册表。

要了解 xigemaAS 概要文件中安全性的工作原理，请参阅安全性（见第 1031 页）。

2.6.1 xigemaAS 安全性入门

可以使用 <quickStartSecurity> 元素来快速地对 xigemaAS 启用简单（单用户）安全性设置。

您可以遵循一些基本配置步骤来设置安全 xigemaAS 服务器和 Web 应用程序。xigemaAS 中的配置操作是动态的，这意味着不必重新启动服务器，配置更新即可生效。

1. 创建和启动服务器。
   - 在 Windows™ 系统上：

     ```
     server.bat create MyNewServer
     server.bat start MyNewServer
     ```

   - 在 Windows™ 系统之外的所有系统上：

     ```
     server create MyNewServer
     server start MyNewServer
     ```

2. 在 server.xml 文件中包括 appSecurity-2.0 和 servlet-3.0 功能部件。

   server.xml 文件位于 *myNewServer* 的服务器目录中，例如，wlp\usr\servers\myNewServer\server.xml。

   ```
   <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>servlet-3.0</feature>
   </featureManager>
   ```

3. 定义要授予其服务器管理活动 Administrator 角色的用户名和密码。

   ```
   <quickStartSecurity userName="Bob" userPassword="bobpwd" />
   ```

   
   注：选择对您有意义的用户名和密码。切勿对应用程序使用示例中的名称和密码。

4. 使用相关安全性约束来配置部署描述符以保护 Web 资源。例如，使用 <auth-constraint> 和 <role-name> 元素来定义可以访问 Web 资源的角色。

   以下示例 web.xml 文件显示对应用程序中所有 URI 的访问权是受 testing 角色保护。

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://
   java.sun.com/dtd/web-app_2_3.dtd">

   <web-app id="myWebApp">

    <!-- SERVLET DEFINITIONS -->
    <servlet id="Default">
       <servlet-name>myWebApp</servlet-name>
       <servlet-class>com.web.app.MyWebAppServlet</servlet-class>
       <load-on-startup/>
    </servlet>

    <!-- SERVLET MAPPINGS -->
    <servlet-mapping id="ServletMapping_Default">
       <servlet-name>myWebApp</servlet-name>
       <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <!-- SECURITY ROLES -->
    <security-role>
       <role-name>testing</role-name>
    </security-role>

    <!-- SECURITY CONSTRAINTS -->
    <security-constraint>
       <web-resource-collection>
   ```

```
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>testing</role-name>
    </auth-constraint>
  </security-constraint>

  <!-- AUTHENTICATION METHOD: Basic authentication -->
  <login-config>
    <auth-method>BASIC</auth-method>
  </login-config>

</web-app>
```

5. 在 server.xml 文件中配置应用程序。

   在以下示例中，用户 Bob 映射到应用程序的 `testing` 角色：

```
<application type="war" id="myWebApp" name="myWebApp"
             location="${server.config.dir}/apps/myWebApp.war">
    <application-bnd>
        <security-role name="testing">
            <user name="Bob" />
        </security-role>
    </application-bnd>
</application>
```

6. 使用用户名 Bob 来访问应用程序并登录。

   `myWebApp` 应用程序的缺省 URL 是 `http://localhost:9080/myWebApp`

您现在已保护您的应用程序。

安全性快速概述

为了解 xigemaAS 中安全性的基本工作流程，详细介绍了一些常见安全性术语以及示例。

安全性关键术语

认证

认证确认用户的身份。最常用的认证形式是用户名和密码，例如通过基本认证或者用于 Web 应用程序的表单登录。认证用户时，请求源在运行时表示为 `Subject` 对象。

权限

授权决定用户是否能够访问系统中所给定的角色。Java™ EE 模型使用主体、角色和角色映射来确定是否允许访问。

角色

在 Java™ EE 应用程序中定义了角色。系统预定义了一些角色（例如，管理员角色）；另一些角色由应用程序开发者定义。在 Java™ EE 中，通常根据主体 在应用程序中承担的角色来授权或拒绝主体 访问某一角色。

主体

主体（Subject）是常规术语，也是 Java™ 对象：`javax.security.auth.Subject`。通常，术语主体表示系统中的活动实体，例如系统上的用户，甚至是系统进程自身。

安全性工作流示例

以下示例说明了在用户请求访问资源时如何应用安全性。例如，用户 Bob 想要访问 servlet myWebApp。请参阅 *xigemaAS 安全性入门*（见第 1275 页）中的代码样本。

下列条件必须成立，才能访问 servlet myWebApp：

1. Bob 必须能登录系统，因为 servlet 是受保护的。
2. Bob 必须在 testing 角色中，因为 servlet 已使用部署描述符中的 auth-constraint 元素进行限制。

如果 Bob 无法登录系统，或者 Bob 不在 testing 角色中，那么将拒绝对 servlet myWebApp 进行访问。

另一个用户 Alice 可登录系统，因为 Alice 是有效用户。但 Alice 不在 testing 角色中。Alice 登录时，会显示 HTTP 403 错误（拒绝/禁止访问）。

在 xigemaAS 上设置 BasicRegistry 和角色映射

可以配置 xigemaAS 概要文件以使用基本用户注册表来认证和授权用户。

必须在 xigemaAS 的 server.xml 文件中启用 xigemaAS 功能部件 appSecurity-2.0 和 servlet-3.0。

有关 xigemaAS 中的安全性配置的更多信息，请参阅 *xigemaAS 安全性入门*（见第 1275 页）。

可以通过下列步骤在 xigemaAS 概要文件服务器的 server.xml 文件中设置基本用户注册表以及配置更多角色映射。

1. 如下所示配置基本注册表。使用对您有意义的用户名和密码。切勿在应用程序中使用此示例中的名称和密码。

```
<basicRegistry id="basic" realm="WebRealm">
   <user name="Bob" password="bobpwd" />
</basicRegistry>
```

2. 可选： 如果用户或者用户组用来执行远程系统管理活动，请为该用户或组授予 Administrator 角色。使用 *quickStartSecurity* 元素时，会自动执行此步骤；也可以通过按如下所示向 server.xml 文件中添加 administrator-role 元素来完成此步骤。

```
<administrator-role>
   <user>Bob</user>
   <group>myAdmins</group>
</administrator-role>
```

3. 对配置中的密码进行编码。可以使用 *securityUtility encode* 任务来获取编码值。
4. 可选： 添加其他用户。确保每个用户名是唯一的。

```
<basicRegistry id="basic" realm="WebRealm">
 <user name="Bob" password="bobpwd" />
 <user name="user1" password="user1pwd" />
 <user name="user2" password="user2pwd" />
</basicRegistry>
```

5. 为用户创建组。确保每个组名必须唯一。

```
<basicRegistry id="basic" realm="WebRealm">
 <user name="Bob" password="bobpwd" />
 <user name="user1" password="user1pwd" />
 <user name="user2" password="user2pwd" />

       <group name="myAdmins">
          <member name="Bob" />
          <member name="user1" />
       </group>
```

```
        <group name="users">
           <member name="user1" />
           <member name="user2" />
        </group>
</basicRegistry>
```

6. 将一些用户和组指定给应用程序的 `testing` 角色。

```
<application type="war" id="myWebApp" name="myWebApp"
             location="${server.config.dir}/apps/myWebApp.war">
   <application-bnd>
       <security-role name="testing">
           <user name="Bob" />
           <user name="user1" />
           <group name="users" />
       </security-role>
   </application-bnd>
</application>
```

在应用程序的部署描述符中配置 security-related 元素。请参阅 *xigemaAS 安全性入门*（见第 1275 页），以获取样本 web.xml 文件。

2.6.2 保护与 xigemaAS 的通信

可以配置 xigemaAS 服务器以在客户机与服务器之间提供安全通信。

使用安全套接字层 (SSL) 协议保护通信安全。SSL 协议提供了传输层安全性（包括真实性、数据签名和数据加密），以确保使用 xigemaAS 的客户机和服务器之间能够建立安全连接。SSL 的基本技术是公用密钥密码术，该技术可以保证当某个实体使用公用密钥对数据进行加密后，只有具备相应的专用密钥的实体才能对该数据进行解密。xigemaAS 服务器使用 Java™ 安全套接字扩展 (JSSE) 作为安全连接的 SSL 实现。JSSE 负责处理 SSL 提供的握手协商和保护功能，以确保大多数协议之间能够建立安全连接。JSSE 依赖基于 X.509 证书的非对称密钥对来保护安全连接和进行一些数据加密。密钥对可有效地对基于会话的密钥进行加密，而基于会话的密钥可对大型数据块进行加密。SSL 实现负责管理 X.509 证书。

要配置安全通信，可以在 server.xml 文件中指定最低 *SSL* 配置或详细 *SSL* 配置。最低配置只需要指定 SSL 功能部件和密钥库条目。

指定为缺省 SSL 配置的 SSL 配置用于通过 `SSLContext.setDefault()` 方法创建进程的缺省 SSLContext。缺省 SSL 配置可以是最低 SSL 配置，也可以是由 sslDefault 元素的 sslRef 属性所标识的配置（如果定义了多个 SSL 配置）。因为缺省 SSLContext 已在进程上设置，所以将无法识别 javax.net.ssl.keyStore 和 javax.net.ssl.trustStore 属性。

对 xigemaAS 启用 SSL 通信

要对 xigemaAS 启用 SSL 通信，必须满足一个最小的 SSL 配置选项集。它采用大部分 SSL 选项，而且只需要一些密钥库配置信息。

通过使用 SSL 证书进行连接握手期间，会进行 SSL 客户机认证。SSL 握手是通过 SSL 协议交换的一系列消息，以针对特定于连接的保护进行协商。在握手期间，安全服务器会请求客户机发送回用于认证的证书或证书链。要对 xigemaAS 启用 SSL，请将 `ssl-1.0` xigemaAS功能部件以及用于认证的密钥库信息的代码添加到配置根文档文件 server.xml。

缺省情况下，配置根文档文件的路径和文件名是 *path_to_xigemaas*/wlp/usr/servers/*server_name*/server.xml。*path_to_xigemaas* 是 xigemaAS 在操作系统上的安装位置，*server_name* 是服务器的名称。但是，可更改此路径。请参阅定制 *xigemaAS* 环境（见第 1114 页）。

1. 在 server.xml 文件中启用 ssl-1.0xigemaAS 功能部件。

```
<featureManager>
    <feature>ssl-1.0</feature>
</featureManager>
```

> 注：如果应用程序安全性是必需的，而且安全性信息会重定向到安全端口，那么您必须将 appSecurity-2.0 xigemaAS 功能部件添加到 server.xml 文件。

2. 将密钥库服务对象条目添加到 server.xml 文件。keyStore 元素称为 defaultKeyStore，并且包含密钥库密码。可以采用明文或编码格式输入该密码。*securityUtility encode* 选项可用于对密码进行编码。

```
<keyStore id="defaultKeyStore" password="yourPassword" />
```

此配置是创建 SSL 配置时需要的最低配置。在此配置中，如果在进行 SSL 初始化期间密钥库和证书不存在，那么服务器将创建密钥库和证书。所提供的密码长度必须至少为 6 个字符。 假定密钥库是服务器 home/resources/security 目录中称为 key.jks 的 JKS 密钥库。如果该文件不存在，那么服务器将为您创建该文件。如果服务器创建密钥库文件，那么它还会在该文件中创建证书。证书是自签名证书，有效期为 365 天，证书的 subjectDN 的 CN 值为正在运行服务器的机器的主机名，而证书的签名算法为 SHA256withRSA。

> 注：由 xigemaAS 服务器创建的证书不打算用于生产。创建这些证书是为了方便开发者。生产中使用的证书应当是由可信认证中心颁发或签署的正确链接的证书。如果您想要使用具有更长持续时间或定制的 subjectDN 的自签名证书，那么可以使用 securityUtility createSSLCertificate 任务来创建这类证书。

示例为最低配置中的 SAF 密钥环：

```
<keyStore id="defaultKeyStore" location="safkeyring:///xigemaASKeyring"
          type="JCERACFKS" password="password" fileBased="false"
          readOnly="true" />
```

在配置 RACF® 密钥环以供 xigemaAS 服务器使用之前，需要设置该密钥环。服务器将不会创建证书并将它们添加到 RACF®。

最低 SSL 配置的单个密钥库条目也可以扩展来包含位置和类型。

```
<keyStore id="defaultKeyStore" location="myKeyStore.p12" password="yourPassword"
 type="PKCS12"/>
```

location 参数可以是密钥库文件的绝对路径。如果是绝对路径，那么会假定已创建密钥库文件。如果已创建密钥库文件，那么也可以在最低 SSL 配置中指定其他类型的密钥库。如果使用最低 SSL 配置，那么会使用 SSL 配置缺省值来创建 SSL 握手的 SSL 上下文。缺省情况下，配置协议是 SSL_TLS。可以使用 HIGH 密码 128 位及更高位数的密码套件。

SSL 配置属性

SSL 配置包含用来控制 xigemaAS 上服务器 SSL 传输层的行为的属性。此主题复述适用于 SSL 配置的所有设置。

SSL 功能部件

要在服务器上启用 SSL，必须在 server.xml 文件中包括 SSL 功能部件：

```
<featureManager>
  <feature>ssl-1.0</feature>
```

```
</featureManager>
```

SSL 缺省值

可以配置多项 SSL 配置。如果配置了多项 SSL 配置，那么必须在 server.xml 文件中使用 `sslDefault` 服务配置来指定缺省 SSL 配置。

表 24: sslDefault 元素的属性

此表描述 `sslDefault` 元素的属性。

属性	描述	缺省值
sslRef	`sslRef` 属性指定要用作缺省值的 SSL 配置的名称。	缺省 SSL 配置名称是 `defaultSSLConfig`。

在 server.xml 文件中，条目如下所示：

```
<sslDefault sslRef="mySSLSettings" />
```

SSL 配置

使用 SSL 配置属性来定制 SSL 环境以适合需求。可以在 server.xml 文件中的 `ssl` 服务配置元素上设置这些属性。

表 25: SSL 元素的属性

此表描述 `ssl` 元素的属性。

属性	描述	缺省值
id	id 属性对 SSL 配置对象指定唯一名称。	没有缺省值；必须指定唯一名称。
keyStoreRef	keyStoreRef 属性指定用来定义 SSL 配置密钥库的密钥库服务对象。密钥库保存建立 SSL 连接所需的密钥。	没有缺省值；必须指定密钥库引用。
trustStoreRef	trustStoreRef 属性指定用来定义 SSL 配置信任库的密钥库服务对象。信任库保存对验证进行签名所需的证书。	`trustStoreRef` 是可选属性。如果缺少引用，那么会使用 `keyStoreRef` 所指定的密钥库。
clientAuthentication	clientAuthentication 属性确定是否需要 SSL 客户机认证。	缺省值为 false。
clientAuthenticationSupported	clientAuthenticationSupported 属性确定是否支持 SSL 客户机认证。客户机不必提供客户机证书。如果 clientAuthentication 属性设置为 true，那么会覆盖 clientAuthenticationSupported 属性的值。	缺省值为 false。

属性	描述	缺省值
sslProtocol	sslProtocol 属性定义 SSL 握手协议。协议可能依赖于 SDK，因此如果您修改协议，请确保据以运行的 SDK 支持该值。	缺省值为 SSL_TLS。
securityLevel	securityLevel 属性确定 SSL 握手要使用的密码套件组。该属性具有下列其中一个值： • HIGH（128 位密码及更高） • MEDIUM（40 位密码） • WEAK（适用于所有密码，不加密） • CUSTOM（如果定制了密码套件组）。 如果使用特定的密码列表来设置 enabledCiphers 属性，那么系统会忽略此属性。	缺省值为 HIGH。
enabledCiphers	enabledCiphers 属性用于指定唯一的密码套件列表。使用空格来分隔列表中的每个密码套件。如果设置了 enabledCiphers 属性，那么会忽略 securityLevel 属性。	没有缺省值。
serverKeyAlias	serverKeyAlias 属性指定密钥库中要用作 SSL 配置密钥的密钥。只有在密钥库中具有多个密钥条目时，才需要此属性。如果密钥库具有多个密钥条目，且此属性未指定密钥，那么 JSSE 会选取密钥。	没有缺省值。
clientKeyAlias	clientKeyAlias 属性指定密钥库中要用作 SSL 配置密钥的密钥（在启用 clientAuthentication 的情况下）。只有在密钥库中包含多个密钥条目时，才需要此属性。	没有缺省值。

注：

- SSL 握手使用密钥管理器来确定要使用的证书别名。密钥管理器不是在 server.xml 文件中进行配置，而是从 SDK 的安全性属性 `ssl.KeyManagerFactory.algorithm` 中检索。
- SSL 握手使用信任管理器来作出信任决策。信任管理器不是在 server.xml 文件中进行配置，而是从 SDK 的安全性属性 `ssl.TrustManagerFactory.algorithm` 中检索。

下面举例说明了如何在 server.xml 文件中配置 `ssl` 元素：

```
<!--  Simple ssl configuration service object. This assumes there is a keystore object named -->
```

```
<!--  defaultKeyStore and a truststore object named defaultTrustStore in the server.xml file. --
>
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       trustStoreRef="defaultTrustStore" />


<!--  A ssl configuration service object that enabled clientAuthentication -->
<!--  and specifies the TLS protocol be used. -->
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       trustStoreRef="defaultTrustStore"
       clientAuthentication="true"
       sslProtocol="TLS" />

<!-- An SSL configuration service object that names the serverKeyAlias -->
<!-- to be used by the handshake. This assumes there is a certificate -->
<!-- called "default" in the keystore defined by keyStoreRef. -->
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       serverKeyAlias="default" />
```

密钥库配置

keystore 配置由装入密钥库时所需的属性组成。可以在 server.xml 文件中的密钥库服务配置上设置这些属性。

表 26: keystore 元素的属性

此表说明了 keystore 元素的属性。

属性	描述	缺省值
id	id 属性定义密钥库对象的唯一标识。	没有缺省值，必须指定唯一名称。
location	location 属性指定密钥库文件名。值可以包含文件的绝对路径。如果未提供绝对路径，那么代码会在 ${server.config.dir}/resources/security 目录中查找该文件。	在 *SSL 最低配置*中，会假定该文件的位置为 ${server.config.dir}/resources/security/key.jks。
type	type 属性指定密钥库的类型。检查据以运行的 SDK 支持您所指定的密钥库类型。	缺省值为 jks。
password	password 属性指定用于装入密钥库文件的密码。可以采用明文或编码格式存储该密码。有关如何对密码进行编码的信息，请参阅 *security Utility encode* 选项。	必须提供。
provider	provider 属性指定要用于装入密钥库的提供程序。某些密钥库类型需要提供程序，而不是 SDK 缺省值。	缺省情况下，未指定任何提供程序。

属性	描述	缺省值
fileBased	fileBased 属性指定密钥库是否基于文件。	缺省值为 true。
pollingRate	服务器检查密钥库文件更新的频率。	500ms。
updateTrigger	该方法用于触发服务器以重新装入密钥库文件。指定 polled 以允许服务器检查密钥库文件的更改，指定 mbean 以允许服务器等待 mbean 重新装入密钥库文件，或指定 disabled 以禁用文件监视。	disabled。

如果 updateTrigger 属性设置为 polled 或 mbean，那么服务器可重新装入密钥库文件。如果启用了 polled，那么服务器会根据 pollingRate 属性中设置的速率监视密钥库文件的更改。如果 updateTrigger 属性设置为 mbean，那么服务器从 xigemaAS:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean MBean 接收到通知时将重新装入密钥库文件。缺省情况下，文件监视被禁用。

下面举例说明了如何在 server.xml 文件中配置 keystore 元素：

```
<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be monitored every 5 seconds -->
<!-- for updates -->
   <keyStore id="defaultKeyStore"
           location="MyKeyStoreFile.jks"
           type="JKS" password="myPassword"
           pollingRate="5s"
           updateTrigger="polled" />

<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be reloaded when the server -->
<!-- recieves an mbean notification to do so -->
   <keyStore id="defaultKeyStore"
           location="MyKeyStoreFile.jks"
           type="JKS" password="myPassword"
           updateTrigger="mbean" />
```

完整 SSL 配置示例

下面举例说明了 server.xml 文件中的完整 SSL 配置。此示例具有下列 SSL 配置：

- defaultSSLSettings
- mySSLSettings

缺省情况下，SSL 配置会设置为 defaultSSLConfig。

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>


<!-- default SSL configuration is defaultSSLSettings ->
  <sslDefault sslRef="defaultSSLSettings" />
  <ssl id="defaultSSLSettings"
```

```
      keyStoreRef="defaultKeyStore"
      trustStoreRef="defaultTrustStore"
      clientAuthenticationSupported="true" />
  <keyStore id="defaultKeyStore"
            location="key.jks"
            type="JKS" password="defaultPWD" />
  <keyStore id="defaultTrustStore"
            location="trust.jks"
            type="JKS" password="defaultPWD" />

  <ssl id="mySSLSettings"
      keyStoreRef="myKeyStore"
      trustStoreRef="myTrustStore"
      clientAuthentication="true" />
  <keyStore id="LDAPKeyStore"
            location="${server.config.dir}/myKey.p12"
            type="PKCS12"
            password="{xor}CDo9Hgw=" />    <keyStore id="LDAPTrustStore"
            location="${server.config.dir}/myTrust.p12"
            type="PKCS12"
            password="{xor}CDo9Hgw=" />
```

密钥库

xigemaAS 仅可创建 Java 密钥库 (JKS) 这一密钥库类型。xigemaAS 中对其他密钥库类型的支持可取决于底层 Java 运行时环境 (JRE) 支持的对象。以下是 xigemaAS 中的不同密钥库类型。

有关 `keystore` 元素的配置属性的更多信息，请参阅*SSL* 配置属性（见第 1279 页）。

JKS 和 JCEKS

Java™ 密钥库 (JKS) 和 Java™ 密码扩展密钥库 (JCEKS) 在 IBM® JRE 与 Oracle JRE 之间是通用的，可使用任一 JRE 配置为相同项。JKS 是 xigemaAS 中的缺省密钥库类型，并且是 xigemaAS 可创建的唯一密钥库类型。如果未在配置中指定任何密钥库类型，那么会使用 JKS。

JDS 密钥库配置的示例如下所示：

```
    <keyStore id="sampleJKSKeyStore"
     location="MyKeyStoreFile.jks"
     type="JKS" password="myPassword" />
```

JCEKS 密钥库配置的示例如下所示：

```
   <keyStore id="sampleJCEKSKeyStore"
    location="MyKeyStoreFile.jceks"
    type="JCEKS" password="myPassword" />
```

PKCS11 密钥库

可配置硬件密码密钥库，以便 xigemaAS 服务器可用来提供密码令牌支持。

用户必须提供特定于硬件设备的配置文件。此配置文件是文本文件，包含格式为 `attribute = value` 的条目。此文件必须至少包含 `name` 和 `library` 属性。例如：

```
name = HWDevice
library = /opt/foo/lib/libpkcs11.so
```

`name` 属性是要给予设备的此实例的名称。`library` 属性包含硬件设备提供的库的路径（用于该访问设备）。此配置文件还可包含特定于该硬件设备的配置数据。

要在 xigemaAS 中配置 PKCS11 密钥库，`keystore` 元素必须包含以下字段：

- `id` - 在配置中唯一标识 `keystore` 元素。

- `location` - 特定于硬件设备的配置文件的路径。
- `type` - PKCS11 必须指定为密钥库类型。
- `fileBased` - 必须为 false 以将此密钥库标识为设备。
- `password` - 访问设备中的密钥时所需的密码。
- `provider` - 所需提供程序。对于 IBM® JRE，该值必须为 `IBMPKCS11Impl`，对于 Oracle JRE，该值必须为 `SunPKCS11`。

以下是示例配置：

```
<keyStore id="hwKeyStore"
          location="${server.config.dir}/HWCrypto.cfg"
          type="PKCS11"
          fileBased="false"
          password="{xor}Lz4sLCgwLTs="
          provider="IBMPKCS11Impl"/>
```

PKCS12 密钥库

使用 IBM® JRE 时，可使用公用密钥密码标准 #12 (PKCS12) 密钥库，而不使用 xigemaAS 创建的密钥库。PKCS12 密钥库配置的示例如下所示：

```
    <keyStore id="samplePKCS12KeyStore"
     location="MyKeyStoreFile.p12"
     type="PKCS12" password="myPassword" />
```

CMS 密钥库

使用 IBM® JRE 时，可配置 CMS 密钥库，而不是 xigemaAS 创建的密钥库。但是，需要某些特殊配置。CMS 提供程序缺省情况下在 IBM® JRE 上不可用，因此，必须将它添加至 IBM® JRE 的 java.security 文件中的提供程序列表。在以下示例中，`com.ibm.security.cmskeystore.CMSProvider` 类被添加至列表结尾。请确保提供程序列表中的提供程序编号正确。xigemaAS 不使用 CMS 密钥库隐藏文件来获取对密钥库的访问。

```
    security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
    security.provider.2=com.ibm.crypto.provider.IBMJCE
    security.provider.3=com.ibm.security.jgss.IBMJGSSProvider
    security.provider.4=com.ibm.security.cert.IBMCertPath
    security.provider.5=com.ibm.security.sasl.IBMSASL
    security.provider.6=com.ibm.xml.crypto.IBMXMLCryptoProvider
    security.provider.7=com.ibm.xml.enc.IBMXMLEncProvider
    security.provider.8=org.apache.harmony.security.provider.PolicyProvider
    security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
    security.provider.10=com.ibm.security.cmskeystore.CMSProvider
```

为使用 CMS 密钥库，server.xml 文件中的配置如下所示：

```
    <keyStore id="sampleCMSKeyStore"
     password="myPassword"
     location="MyKeyStoreFile.kdb"
     provider="IBMCMSProvider"
     type="CMSKS"/>
```

xigemaAS 中的 SSL 缺省值

在 xigemaAS 中指定缺省 SSL 证书、密钥库和配置。

缺省证书和密钥库

作为帮助开发者启动和运行的便利工具，用户可在 `securityUtility` 命令中使用 createSSLCertificate 参数来创建自签名证书。用户可直接从命令行调用该工具，或让服务器调用它以在服务器启动时创建缺省证书和密钥库。

如果用户具有在 server.xml 文件中名为 `defaultKeyStore` 的密钥库元素，那么服务器将创建缺省密钥库和证书。例如：

```
<keyStore id="defaultKeyStore" password="yourPassword" />
```

如果服务器启动时 `defaultKeyStore` 的密钥库配置已就位但密钥库不存在，那么服务器将调用 createSSLCertificate 参数以使用该配置中指定的密钥来创建密钥库。

缺省密钥库详细信息：

- 位置：此密钥库文件名为 key.jks 并且在服务器或客户机 resources/security 目录中。
- 密钥库类型：此密钥库类型为 JKS。
- 密码：配置中提供的密码。

xigemaAS 详细信息创建的缺省证书：

- 类型：此证书为自签名证书。
- 大小：缺省证书大小为 2048。
- 签名算法：证书的签名算法为 SHA256WITHRSA。
- 有效性：证书有效期为 365 天。
- SubjectDN：证书是使用 `CN=<hostname>,OU=<client or server name>,O=VSETTAN,C=CN as the SubjectDN` 创建的。

如果用户要定制证书，那么可在命令行上调用 createSSLCertificate 参数。

👉 注：由 xigemaAS 服务器创建的证书不用于生产。这些证书是为方便开发者而创建的。生产中使用的证书必须为可信认证中心颁发或签名的正确链式证书。

缺省 SSL 配置

SSL 所需的最低配置是一个名为 `defaultKeyStore` 的密钥库元素。如果配置中存在 `defaultKeyStore`，那么运行时将围绕它构建名为 `defaultSSLConfig` 的 SSL 配置。

`defaultSSLConfig` 详细信息：

- 协议：如果使用 xigemaAS JRE，那么缺省情况下协议将设置为 SSL_TLS。如果使用 Oracle JRE，那么会将 SSL 用作协议。
- 密码：密码列表是通过从底层 JRE 获取受支持密码列表构建的。缺省情况下，此列表减少为不低于 128 位或为 `3DES` 的所有密钥。`RC4` 已被移除，因为它们被视为不安全并因而不启用。`ECDHE` 密码已被移除，因为在您访问不支持它们的服务器时，它们可能导致错误。可定制密码列表以包含这些密码。
- 客户机认证：缺省情况下，`clientAuthentication` 和 `clientAuthenticationSupported` 被禁用。
- 密钥库：在缺省配置中，`defaultKeyStore` 同时被用作密钥和信任库。

可在 server.xml 文件中输入名为 defaultSSLConfig 的 ssl 元素以定制 SSL 配置属性。名为 defaultSSLConfig 的定制 ssl 元素仍被视为缺省 SSL 配置，只要另一 SSL 配置未被标识为缺省值。有关可用来定制 SSL 配置的属性的更多详细信息，请参阅 *SSL* 配置属性。

要在配置中指定另一 ssl 元素作为缺省 SSL 配置，用户可使用 sslDefault 元素进行标识。

```
<sslDefault sslRef="customSSLConfiguration" />
```

xigemaAS 缺省 SSL 配置中的属性用于创建 SSLContext。通过使用 Java™ API SSLContext.setDefault()，该 SSLContext 在进程上设置为缺省 SSLContext。如果应用程序使用 https ULR 调用 httpURLConnection() 之类的 API 并且不提供任何 SSL 信息，那么应用程序选择进程的缺省 SSLContext，在此情况下为使用 xigemaAS 缺省 SSL 配置创建的 SSLContext。

如果 xigemaAS 中没有缺省 SSL 配置，那么将使用 JSSE 的缺省 SSLContext。JSSE 的缺省 SSLContext 对密钥库和信任库使用 cacerts 文件。如果未定义 SSL 功能或现有 SSL 配置未标识为缺省值，那么 xigemaAS 中没有缺省 SSL 配置。缺省配置将称为 defaultSSLConfig（如果已定义 defaultKeyStore，那么它可以是隐式的），也可使用 sslDefault 元素指定备用 SSL 配置。

javax 系统属性 (javax.net.ssl.keystore) 用于为缺省 SSL 上下文设置密钥库和信任库信息，不得用于其他用途。如果这些属性是在进程上设置的，那么调用 SSLContext.setDefault() 将擦除这些属性。

从命令行创建 SSL 证书

可以使用 securityUtility 命令来创建缺省 SSL 证书以供 xigemaAS 配置使用。

1. 打开命令行，然后将目录切换至 wlp/bin 目录。
2. 创建 SSL 证书。

   运行以下命令。如果未指定服务器名称或密码，那么命令不会运行。请参阅 *securityUtility* 命令（见第 1287 页）。

   ```
   securityUtility createSSLCertificate --server=server_name --password=your_password
   ```

您已为指定的服务器创建缺省密钥库 key.jks。密钥库文件位于所指定服务器的 /resources/security 目录下。如果缺省密钥库已经存在，那么该命令不会成功执行。

可以配置服务器以使用密钥库，并通过在服务器配置文件添加下列行来在服务器配置中启用 SSL：

```
    <featureManager>
        <feature>ssl-1.0</feature>
    </featureManager>

    <keyStore id="defaultKeyStore" password="keystore_password" />
```

请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）。

securityUtility 命令

securityUtility 命令支持对 xigemaAS 使用明文加密及创建 SSL 证书。

语法

命令语法如下所示：

```
securityUtility task [options]
```

其中 options 随 task 的值不同而不同。

参数

下列任务可用于 `securityUtility` 命令：

encode

使用 Base64 对所提供的 *text* 进行编码。如果未指定任何选项，那么命令会进入交互方式。否则，会对提供的 *text* 进行编码。如果 *text* 包含空格，那么必须使用引号将其引起来。

选项是：

--encoding=encoding_type

指定如何对密码进行编码。支持的编码是 `xor`、`aes`、`sm4` 和 `hash`。如果未提供此选项，那么缺省值为 `xor`。

--key=encryption_key

指定在使用 AES 或 SM4 进行编码时要使用的密钥。 当使用AES时，将对此字符串使用散列算法，以生成将用于对密码加密和解密的密钥。可以通过定义变量 `wlp.password.encryption.key`（该变量的值为密钥）来将密钥提供给服务器。 当使用SM4时，该字符串将作为加密和解密的密钥，其长度必须为16个字节。可以通过定义变量 `wlp.password.encryption.sm4.key`（该变量的值为密钥）来将密钥提供给服务器。 如果未提供此选项，那么将使用缺省密钥。

--notrim

指定是否从指定文本开头和结尾移除了空格字符。如果指定了此选项，那么所提供的文本将按原样编码。如果未指定此选项，那么将移除所指定文本的开头和末尾的空格字符。

text

将对其进行编码的文本。

另请参阅通过密码加密进行保护时存在的限制（见第 1063 页）。

createSSLCertificate

创建包含用于服务器或客户机配置的 SSL 证书的缺省密钥库。

密钥库详细信息：

位置：在服务器或客户机目录的 resource/security/key.jks 下。

类型：JKS

密码：使用 --password 选项提供的密码。密码是打开密钥库文件及从密钥库文件检索密钥时所需的。

证书详细信息：

类型：自签名证书

大小：缺省情况下为 2048，可使用 --keySize 选项指定另一大小。

签名算法 SHA256withRSA，可使用 --sigAlg 选项定制。

有效性：缺省情况下为 365 天，可使用 --validity 选项定制。

SubjectDN：缺省情况下为 `CN=<hostname>,OU=<client or server name>,O=vsettan,C=cn`，可使用 --subject 选项定制。

选项是：

--server=name

指定为其创建密钥库和证书的 xigemaAS 服务器的名称。如果指定了 --client 选项，那么不能使用此选项。

--client=name

指定为其创建密钥库和证书的 xigemaAS 客户机的名称。如果指定了 --server 选项，那么不能使用此选项。

--keySize=size

指定证书密钥位大小。缺省值为 2048。

--password=password

指定要用在密钥库中的密码，长度必须至少为 6 个字符。此选项是必需的。

--passwordEncoding=password_encoding_type

指定如何对密钥库密码进行编码。受支持的编码为 `xor` 、 `aes` 或 `sm4`。如果未提供此选项，那么将使用缺省值 `xor`。

--passwordkey=password_encryption_key

指定在使用 AES 或 SM4 对密钥库密码进行编码时要使用的密钥。 当使用AES时，将对此字符串使用散列算法，以生成将用于对密码加密和解密的加密密钥。可以通过定义变量`wlp.password.encryption.key`（该变量的值为密钥）来将密钥提供给服务器。 当使用SM4时，该字符串将作为加密和解密的密钥，其长度必须为16个字节。可以通过定义变量`wlp.password.encryption.sm4.key`（该变量的值为密钥）来将密钥提供给服务器。 如果未提供此选项，那么将使用缺省密钥。

--validity=days

指定证书的有效天数，必须等于或大于 365。如果未提供此选项，那么将使用缺省值 365。

--subject=DN

指定证书主体集和发卡者的专有名称 (DN)。如果未提供此选项，那么将使用缺省值 `CN=<hostname>,OU=<server or client name>,O=vsettan,C=cn`。系统使用 Java 方法检索 `CN` 值以获取机器的本地主机名。如果无法解析该主机名，那么将返回 IP 地址。

--sigAlg

指定用于签署自签名证书的签名算法。受支持签名算法取决于底层 JRE 支持的算法。更强签名算法可能要求 JRE 在适当位置具有不受限制的策略文件。

此命令接受 SHA256withRSA（缺省）、SHA1withRSA、SHA384withRSA、SHA512withRSA、SHA1withECDSA、SHA256withECDSA、SHA384withECDSA 和 SHA512withECDSA。以 RSA 结尾的签名算法使用 RSA 密钥创建证书，以 ECDSA 结尾的签名算法使用椭圆曲线加密算法 (EC) 密钥创建证书。

注：如果使用通过 EC 密钥创建的证书，那么服务器的 SSL 配置中需要定制密码列表以包含 EC 密码。

help

显示所指定任务的帮助信息。

用法

以下示例说明了正确的语法：

```
securityUtility encode --encoding=aes GiveMexigemaAS
securityUtility createSSLCertificate --server=myserver --password=mypassword --validity=365
                                      --subject=CN=mycompany,O=myOrg,C=myCountry
securityUtility help createSSLCertificate
```

⚠ 警告：不同操作系统可能以不同方式处理某些字符。对于 Windows™ 环境，如果输入字符串中有 !，那么需要使用 ^ 字符对其进行转义。 例如，

```
D:\xigemaAS\images\9101\xigemaAS9101\wlp\bin>securityUtility encode
 "a^!"
```

为客户机证书认证配置 Web 应用程序和服务器

可以在 xigemaAS 上使用 SSL 客户机认证来配置 Web 应用程序。

此主题假定您已创建 SSL 证书，如从命令行创建 *SSL* 证书（见第 1287 页）中所述。

如果服务器端请求客户端发送证书，那么会进行客户机证书认证。可以在 SSL 配置上为客户机证书认证配置 xigemaAS 服务器。要这样做，请将 `ssl-1.0` xigemaAS 功能部件，以及要用于认证的密钥库信息告知给服务器的代码，添加到 server.xml 文件。

有关支持的 SSL 方面的详细信息，请参阅*xigemaAS* 功能部件（见第 906 页）。

1. 确保 Web 应用程序的部署描述符将客户机证书认证指定为要使用的认证方法。

   检查部署描述符是否包含下列元素：

   ```
   <auth-method>CLIENT-CERT</auth-method>
   ```

   👉 注：可以使用 Rational® Application Developer 等工具来创建部署描述符。

2. 可选： 使用命令行生成 SSL 证书。请参阅*securityUtility* 命令（见第 1287 页）。
3. 通过将下列行添加到 server.xml 文件，配置服务器以启用 SSL 客户机认证：

   ```
   <featureManager>
        <feature>ssl-1.0</feature>
   <featureManager>

   <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
        trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
   <keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="defaultPWD" />
   <keyStore id="defaultTrustStore" location="trust.jks" type="JKS" password="defaultPWD" />
   ```

   - 如果指定 `clientAuthentication="true"`，那么服务器会请求客户机发送证书。但是，如果客户机没有证书，或者服务器不信任证书，那么握手不成功。
   - 如果指定 `clientAuthenticationSupported="true"`，那么服务器会请求客户机发送证书。但是，如果客户机没有证书，或者服务器不信任证书，那么握手仍可能成功。
   - 如果未指定 `clientAuthentication` 或 `clientAuthenticationSupported`，或者指定 `clientAuthentication="false"` 或 `clientAuthenticationSupported="false"`，那么握手期间服务器不会请求客户机发送证书。
4. 将客户机证书添加到浏览器。请参阅浏览器的文档，以了解如何添加客户机证书。
5. 确保服务器信任所使用的任何客户机证书。

6. 确保用于客户机认证的任何客户机证书已映射到注册表中的用户身份。
   - 对于基本注册表，用户身份是证书的专有名称 (DN) 中的公共名 (CN)。
   - 对于轻量级目录访问协议 (LDAP) 注册表，客户机证书中的 DN 必须位于 LDAP 注册表中。
7. 如果客户机证书认证不成功，请将下列行添加到 server.xml 文件，以仅使用基本认证、用户标识和密码。

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

> 注：如果指定 `allowFailOverToBasicAuth="false"` 或者未指定 `allowFailOvertoBasicAuth`，并且客户机证书认证不成功，那么请求会生成 `403` **认证错误**消息，而且不会提示客户机进行基本认证。

设置 xigemaAS 概要文件以在 SP800-131a 中运行

可设置 xigemaAS 概要文件以满足美国国家标准技术学会 (NIST) 制定的 SP800-131a 要求。

SP800-131a 要求更长的密钥长度和更强大的加密。该规范还提供了配置以允许用户改为使用严格实施的 SP800-131a。该配置还使用户可以在混合使用 FIPS140-2 和 SP800-131a 中设置的情况下运行。SP800-131a 可以采用以下两种方式运行：过渡方式和严格方式。提供了过渡方式，以对用户提供一个用于将其环境改变为 SP800-131a 严格方式的设置。在过渡方式下，可以选择使用 SP800-131a 必需的证书，并将协议设置为 SP800-131a

对 xigemaAS 概要文件严格实施 SP800-131a 要求包括下列事项：

- 为安全套接字层 (SSL) 上下文使用 TLSv1.2 协议。
- 证书的长度必须至少为 2048。椭圆曲线 (EC) 证书要求大小至少为 244 位曲线。
- 必须使用 SHA256、SHA384 或 SHA512 签名算法为证书签名。有效的 signatureAlgorithms 包括：
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA

  > 注：如果使用 SHA384withECDSA 或 SHA512withECDSA，那么 IBM® JDK 要求非受限策略文件就位。
- SP800-131a 核准的密码套件。

> 注：要将 xigemaAS 概要文件服务器配置为以 SP800-131a 方式运行，用户必须使用支持 SP800-131a 的 IBM® JDK 级别运行。IBM® JDK 的最低级别包括 Java™ 6 sr 10、Java™ 6.0.1 sr 2 或者 Java™ 7。

有关 SP800-131a 标准的更多信息，请参阅 *National Institute of Standards and Technology*。

可以按如下所示将 xigemaAS 概要文件配置为以 SP800-131a 严格方式或过渡方式运行：

- 将 xigemaAS 概要文件配置为以 SP800-131a 严格方式运行。
  1. 请确保您正在一个支持 SP800-131a 的 IBM® JDK 级别运行。
  2. 请确保服务器的证书满足 SP800-131a 的条件。
     - 证书的长度必须至少为 2048，椭圆曲线 (EC) 证书的最小大小为 244 位曲线。
     - 至少使用 SHA256 对证书进行了签名，或者使用先前所列的其中一种签名算法对证书进行了签名。

3. 配置 SSL 配置以使用 TLSv1.2 协议。有关更多详细信息，请参阅 对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）
4. 可选： 如果需要 Elliptical Curve (EC) 密码，请在 `enabledCiphers` 属性中列示这些密码。

   使用 SSL 配置的 `securityLevel` 属性生成密码列表时，未包括 EC 密码。
5. 通过将系统属性 `com.ibm.jsse2.sp800-131` 设置为 *strict*（例如，`-Dcom.ibm.jsse2.sp800-131=strict`），使 Java™ 安全套接字扩展 (JSSE) 能够以 SP800-131a 严格方式运行。请参阅定制 *xigemaAS* 环境（见第 1114 页），以了解如何在 jvm.options 文件中设置系统属性。

- 将 xigemaAS 概要文件配置为以 SP800-131a 过渡方式运行。

1. 请确保您正在一个支持 SP800-131a 的 IBM® JDK 级别运行。
2. 可选： 如果需要 Elliptical Curve (EC) 密码，请在 `enabledCiphers` 属性中列示这些密码。

   使用 SSL 配置的 `securityLevel` 属性生成密码列表时，未包括 EC 密码。
3. 通过将系统属性 `-Dcom.ibm.jsse2.sp800-131=transition` 设置为 *transition*（例如，`com.ibm.jsse2.sp800-131=transition`），使 JSSE 能够以 SP800-131a 过渡方式运行。请参阅定制 *xigemaAS* 环境（见第 1114 页），以了解如何在 jvm.options 文件中设置系统属性。

注：如果您将协议更改为使用 TLSv1.2，请确保浏览器支持 TLSv1.2。

配置 httpEndpoint 以使用非缺省 SSL 配置

缺省情况下，httpEndpoint 元素使用服务器缺省 SSL 配置 defaultSSLConfig。可将 httpEndpoint 配置为使用缺省 SSL 配置以外的 SSL 配置。

可通过多种方式将 httpEndpoint 配置为使用 SSL 配置。以下示例显示通过不同方式将 httpEndpoint 配置为使用缺省 SSL 配置以外的 SSL 配置。

- 直接在 httpEndpoint 上设置 SSL 选项。

  以下示例显示如何在 httpEndpoint 上设置 SSL 选项，并假定您已定义名为 xigemaasListenerSSLConfig 的 SSL 配置，此配置未包括在此示例中：

```
<httpEndpoint id="defaultHttpEndpoint"
  host="${listener.host}"
  httpPort="${http.port}"
  httpsPort="${https.port}">
  <sslOptions sslRef="xigemaasListenerSSLConfig" />
</httpEndpoint>
```

- 在 httpEndpoint 中引用 sslOption 元素。

  以下示例显示如何引用 sslOption 元素，并假定您已定义名为 xigemaasListenerSSLConfig 的 SSL 配置，此配置未包括在此示例中：

```
<sslOptions id="mySSLOptions" sslRef="xigemaasListenerSSLConfig" />

<httpEndpoint id="defaultHttpEndpoint"
  host="${listener.host}"
  httpPort="${http.port}"
  httpsPort="${https.port}"
  sslOptionsRef="mySSLOptions"
/>
```

- 将缺省 sslOptions 元素更改为指向缺省 SSL 配置以外的 SSL 配置。

此选项未变更 httpEndpoint。以下示例显示如何更改缺省 sslOptions 元素，并假定您已定义名为 xigemaasListenerSSLConfig 的 SSL 配置，此配置未包括在此示例中：

```
<sslOptions id="defaultSSLOptions" sslRef="xigemaasListenerSSLConfig" />
```

2.6.3 在 xigemaAS 中认证用户

xigemaAS 服务器使用用户注册表来认证用户，并检索有关用户和组的信息来执行与安全性相关的操作，包括认证和授权。

要了解 xigemaAS 中认证的工作原理，请参阅 认证（见第 1035 页）。

可以配置的认证任务可能随需求而不同。除非已使用只能配置一个用户的 *quickStartSecurity 元素*，否则必须至少配置用户注册表。不必为 JAAS、认证高速缓存和 SSO 任务配置值，除非要更改缺省值。仅当具有 TAI 接口的实现来处理认证时，才设置 TAI 配置。

为 xigemaAS 概要文件配置用户注册表

可将用于认证的用户和组信息存储在各种类型的注册表中。例如，可使用基本用户注册表、LDAP 注册表或定制用户注册表。（可选）您可以配置两个或两个以上的 LDAP 注册表，以便对所配置的所有注册表执行操作。例如，当您执行“搜索用户”操作时，将对所配置的所有 LDAP 注册表执行搜索。

为 xigemaAS 概要文件配置基本用户注册表

可以在 xigemaAS 中配置用于认证的基本用户注册表。

可以在 xigemaAS 服务器上通过定义用于认证的用户和组信息来使用基本用户注册表。要这样做，可以将 `appSecurity-2.0` xigemaAS 功能部件添加到 server.xml 文件，以及在 `basicRegistry` 元素中添加用户信息。

1. 将 `appSecurity-2.0`xigemaAS功能部件添加到 server.xml 文件。
2. 可选： 要使用 SSL，请将 `ssl-1.0`xigemaAS 功能部件添加到 server.xml 文件。请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）。
3. 配置服务器的基本注册表，如下所示：

```
<basicRegistry id="basic" realm="customRealm">
     <user name="mlee" password="p@ssw0rd" />
     <user name="rkumar" password="pa$$w0rd" />
     <user name="gjones" password="{xor}Lz4sLCgwLTs=" />
     <group name="students">
         <member name="mlee" />
         <member name="rkumar" />
     </group>
</basicRegistry>
```

注：

- 必须对用户和组使用唯一名称。
- 应该从用户名和组名中移除所有拖尾空格和前导空格。
- 如果用户标识或密码包含非 US-ASCII 的字符，请确保使用 UTF-8 字符编码来保存文件。
- 如果直接地编辑 server.xml 文件，那么可以使用 `securityUtility encode` 命令为每个用户对密码进行编码。`securityUtility` 命令行工具位于 `$INSTALL_ROOT/bin` 目录中。运行 `securityUtility encode` 命令时，将要编码的密码作为命令行输入来提供，或者如果

未指定任何参数，那么工具会提示您输入密码。工具随后会输出已编码的值。复制工具输出的值，然后将该值用作密码。例如，要对密码 GiveMexigemaAS 进行编码，请运行下列命令：

```
securityUtility encode GiveMexigemaAS
```

使用 xigemaAS 概要文件来配置 LDAP 用户注册表

可以使用 xigemaAS 概要文件来配置一个或多个轻量级目录访问协议 (LDAP) 服务器以进行认证。

确保 LDAP 服务器已启动且运行，而且 LDAP 服务器的主机名和端口号已经在已知列表中。

可以在xigemaAS 概要文件上使用现有 LDAP 服务器进行应用程序认证。要执行此操作，可将 appSecurity-2.0 功能部件添加到 server.xml 文件，并在 server.xml 文件中指定 ldapRegistry-3.0 功能部件以及用于连接到 LDAP 服务器的配置信息。

1. 将 xigemaAS功能部件 appSecurity-2.0 和 ldapRegistry-3.0 添加至 server.xml 文件。
2. 可选： 要与启用了 SSL 的 LDAP 服务器进行通信，请将 ssl-1.0 xigemaAS 功能部件添加到 server.xml 文件。
3. 可选： 将信任库复制到服务器配置目录。例如，可以使用 ${server.config.dir} 变量。

   要成功地与 LDAP 服务器进行 SSL 通信，必须将 LDAP 服务器的签署者证书添加到 <ldapRegistry> 元素的 sslAlias 属性所引用的信任库。在以下示例中，必须将签署者证书添加到 LdapSSLTrustStore.jks。
4. 配置服务器的 LDAP 条目。

   如果不想对 LDAP 服务器使用 SSL，请从以下示例中移除所有基于 SSL 和密钥库的行。

   可以在 server.xml 文件中来配置 LDAP 服务器。

   - 对于 IBM® Directory Server：

```
<ldapRegistry id="ldap" realm="SampleLdapIDSRealm"
    host="ldapserver.mycity.mycompany.com" port="389" ignoreCase="true"
    baseDN="o=mycompany,c=cn"
    ldapType="IBM Tivoli Directory Server"
    sslEnabled="true"
    sslRef="LDAPSSLSettings">
    <idsFilters
         userFilter="(&amp;(uid=%v)(objectclass=ePerson))"
         groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)
                  (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
    userIdMap="*:uid"
    groupIdMap="*:cn"
    groupMemberIdMap="mycompany-allGroups:member;mycompany-allGroups:uniqueMember;
                      groupOfNames:member;groupOfUniqueNames:uniqueMember">
    </idsFilters>
</ldapRegistry>

<ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore" trustStoreRef="LDAPTrustStore" />

<keyStore id="LDAPKeyStore" location="${server.config.dir}/LdapSSLKeyStore.jks"
          type="JKS" password="{xor}CDo9Hgw=" />   <keyStore id="LDAPTrustStore"
 location="${server.config.dir}/LdapSSLTrustStore.jks"
          type="JKS" password="{xor}CDo9Hgw=" />
```

   - 对于 Microsoft™ Active Directory Server：

```
<ldapRegistry id="ldap" realm="SampleLdapADRealm"
    host="ldapserver.mycity.mycompany.com" port="389" ignoreCase="true"
    baseDN="cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
    bindDN="cn=testuser,cn=users,dc=adtest,dc=mycity,dc=mycompany,dc=com"
    bindPassword="testuserpwd"
    ldapType="Microsoft Active Directory"
```

```
    sslEnabled="true"
    sslRef="LDAPSSLSettings">
    <actived Filters
    userFilter="(&amp;(sAMAccountName=%v)(objectcategory=user))"
groupFilter="(&amp;(cn=%v)(objectcategory=group))"
   userIdMap="user:sAMAccountName"
    groupIdMap="*:cn"
    groupMemberIdMap="memberOf:member" >
</actived Filters>
    </ldapRegistry>

<ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore" trustStoreRef="LDAPTrustStore" />

<keyStore id="LDAPKeyStore" location="${server.config.dir}/LdapSSLKeyStore.jks"
          type="JKS" password="{xor}CDo9Hgw=" />
          type="JKS" password="{xor}CDo9Hgw=" />
```

👉 注：在此示例中，groupMemberIdMap="memberof:member" 区分大小写。

您可使用 *securityUtility encode* 命令对 bindPassword 密码进行编码。securityUtility 命令行工具位于 $INSTALL_ROOT/bin 目录中。运行 securityUtility encode 命令时，将要编码的密码作为命令行输入来提供，或者如果未指定任何参数，那么工具会提示您输入密码。工具随后会输出已编码的值。复制工具输出的值，然后将该值用作 bindPassword 密码。

5. 可选： 为 LDAP 服务器配置证书过滤方式。

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
      host="myldap.vsettan.com" port="389" ignoreCase="true"
      baseDN="o=vsettan,c=cn"
      ldapType="IBM Tivoli Directory Server" searchTimeout="8m"
      certificateMapMode="CERTIFICATE_FILTER"
      certificateFilter="uid=${SubjectCN}">
      <idsFilters
           userFilter="(&amp;(uid=%v)(objectclass=ePerson))"
           groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)
          (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
      userIdMap="*:uid"
      groupIdMap="*:cn"
      groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;
          groupOfNames:member;groupOfUniqueNames:uniqueMember">
      </idsFilters>
</ldapRegistry>
```

有关 xigemaAS 概要文件中证书映射方式的更多信息，请参阅 *LDAP* 证书映射方式（见第 1297 页）。

6. 可选： 您可定义 LDAP 属性以及用户注册表 <externalId> 属性之间的映射。

您可定义 LDAP 属性以及用户注册表 <externalId> 属性之间的映射。配置映射后，将用户注册表 <externalId> 属性用于任何操作时，值将等于映射的 LDAP 属性的值。以下示例代码显示针对实体类型 <PersonAccount> 为用户注册表 <externalId> 属性以及 LDAP <distinguishedName> 属性定义的映射。<autoGenerate> 属性是可选的，缺省情况下，值为 false。

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
      host="myldap.vsettan.com" port="389" ignoreCase="true"
      baseDN="o=vsettan,c=cn"
      ldapType="IBM Tivoli Directory Server" searchTimeout="8m">
      <attributeConfiguration>
            <externalIdAttribute name="distinguishedName" entityType="PersonAccount"
 autoGenerate="false"></externalIdAttribute>
      </attributeConfiguration>
</ldapRegistry>
```

7. 可选： 为多个 LDAP 服务器配置故障转移。

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
      host="ldapserver1.mycity.mycompany.com" port="389" ignoreCase="true"
       baseDN="o=vsettan,c=cn" ldapType="IBM Tivoli Directory Server"
 idsFilters="ibm_dir_server">
  <failoverServers name="failoverLdapServersGroup1">
    <server host="ldapserver2.mycity.mycompany.com" port="389" />
    <server host="ldapserver3.mycity.mycompany.com" port="389" />
  </failoverServers>
  <failoverServers name="failoverLdapServersGroup2">
    <server host="ldapserver4.mycity.mycompany.com" port="389" />
  </failoverServers>
</ldapRegistry>

<idsLdapFilterProperties id="ibm_dir_server"
          userFilter="(&amp;(uid=%v)(objectclass=ePerson))"
          groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)
                 (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
          userIdMap="*:uid" groupIdMap="*:cn"
     groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;
                       groupOfNames:member;groupOfUniqueNames:uniqueMember">
</idsLdapFilterProperties>
```

8. 可选： 配置多个 LDAP 注册表。如果在 server.xml 文件中配置了多个 LDAP 注册表，那么会自动地联合这些注册表。请确保这些用户在所有联合存储库中唯一，否则用户注册表操作不会成功。

   👉 注：如果使用多个联合 LDAP 存储库，那么每个存储库必须定义唯一 baseDN。

```
<ldapRegistry host="ldapserver1.mycity1.mycompany.com" baseDN="o=mycompany,c=us"
    port="123" ldapType="IBM Tivoli Directory Server">
</ldapRegistry>

<ldapRegistry host="ldapserver2.mycity2.mycompany.com"
    baseDN="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
    port="456"
    ldapType="Microsoft Active Directory"
    bindDN="cn=testuser,cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
    bindPassword="{xor}KzosKyosOi0vKDs=">
</ldapRegistry>
```

   👉 注：

   - 不必指定 `federatedRepository` 元素也可以联合多个 LDAP 注册表，因为会自动地联合这些注册表。如果指定了 federatedRepository 元素来配置 `participatingBaseEntry` 和 `primaryRealm` 元素，那么仅对 `primaryRealm` 元素中定义的存储库执行用户注册表操作。可在 `primaryRealm` 元素下对不同用户注册表 API 定义输入和输出属性映射。
   - `participatingBaseEntry` 元素的 name 属性必须与 `ldapRegistry` 元素中所指定 baseDN 属性的值相同。在下列示例中，对主机 *ldapserver1.mycity1.mycompany.com* 上的 LDAP 注册表配置了 baseDN 和 name 属性。baseDN 属性的值必须与 LDAP 服务器中子树的值相同，而 name 属性的值必须为联合用户注册表中该子树的名称。可选择性地指定 name 属性。缺省情况下，name 属性和 baseDN 属性使用相同的值。如果在 `ldapRegistry` 元素中指定了 name 属性，那么 `participatingBaseEntry` 元素中的 name 属性和 `ldapRegistry` 元素中的 name 属性必须使用相同的值。

```
<ldapRegistry host="ldapserver1.mycity1.mycompany.com"
 baseDN="o=mycompany,ou=myou,c=us"
    port="123" ldapType="IBM Tivoli Directory Server" name="o=mybaseentry">
</ldapRegistry>

<ldapRegistry host="ldapserver2.mycity2.mycompany.com"
    baseDN="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
    port="456"
    ldapType="Microsoft Active Directory"
```

```
    bindDN="cn=testuser,cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"
    bindPassword="{xor}KzosKyosOi0vKDs=">
</ldapRegistry>

<federatedRepository>
    <primaryRealm name="RealmName" delimiter="@" allowOpIfRepoDown="true">
      <participatingBaseEntry name="o=mybaseentry"/>
      <participatingBaseEntry
 name="cn=users,dc=secfvt2,dc=mycity2,dc=mycompany,dc=com"/>
        <uniqueUserIdMapping inputProperty="uniqueName" outputProperty="uniqueName"/>
        <userSecurityNameMapping inputProperty="principalName"
 outputProperty="principalName"/>
              <userDisplayNameMapping inputProperty="principalName"
 outputProperty="principalName"/>
      <uniqueGroupIdMapping inputProperty="uniqueName" outputProperty="uniqueName"/>
              <groupSecurityNameMapping inputProperty="cn" outputProperty="cn"/>
              <groupDisplayNameMapping inputProperty="cn" outputProperty="cn"/>
          </primaryRealm>
</federatedRepository>
```

9. 可选： 可对 LDAP 注册表配置其他可选属性，例如 contextPool 或 ldapCache，如以下示例所示：

```
<ldapRegistry id="IBMDirectoryServerLDAP" realm="SampleLdapIDSRealm"
        host="host.domain.com" port="389" ignoreCase="true"
          baseDN="o=domain,c=us"
          bindDN="cn=testuser,o=domain,c=us"
          bindPassword="mypassword"
        ldapType="IBM Tivoli Directory Server"
        searchTimeout="8m">
    <contextPool enabled="true" initialSize="1" maxSize="0" timeout="0s" waitTime="3000ms"
 preferredSize="3"/>
     <ldapCache>
          <attributesCache size="4000" timeout="1200s" enabled="true" sizeLimit="2000"/>
          <searchResultsCache size="2000" timeout="600s" enabled="true"
 resultsSizeLimit="1000"/>
     </ldapCache>
</ldapRegistry>
```

注：

- 联合用户注册表使用上下文池机制来提高对 LDAP 服务器进行并行存取的性能。上下文池在高于连接池的级别工作。上下文池中的每个上下文条目对应一个与 LDAP 服务器的套接字连接。配置 LDAP 注册表时，指定此池所使用的绑定凭证。
- 联合存储库使用高速缓存机制以增强性能。它根据所执行的用户操作对有关 LDAP 用户和组的信息进行高速缓存。例如，如果对 LDAP 用户和组执行搜索操作，那么会对该操作的结果进行高速缓存。可按先前示例中所示在 server.xml 文件中启用 `ldapCache` 元素。

注：要对所有 LDAP 认证问题进行故障诊断，请在 bootstrap.properties 文件中使用以下跟踪规范：

```
com.ibm.ws.security.wim.*=all:com.ibm.websphere.security.wim.*=all
```

LDAP 证书映射方式

证书映射方式用于指定在 xigemaAS 是依照 EXACT_DN 还是 CERTIFICATE_FILTER 将 X.509 证书映射到 LDAP 目录。

EXACT_DN 意味着证书中的专有名称 (DN) 必须与 LDAP 服务器中的用户条目完全匹配（包括大小写和空格都完全匹配）。要对映射使用指定的证书过滤器，您可以使用 CERTIFICATE_FILTER。

证书过滤器

为 LDAP 过滤器指定过滤器证书映射属性。使用过滤器将客户机证书中的属性映射至 LDAP 注册表中的条目。

如果在运行时有多个 LDAP 条目与过滤器规范匹配，认证就会失败，这是因为这会导致模糊匹配。此过滤器的语法是：

```
LDAP attribute=${Client certificate attribute}
```

简单证书过滤器的示例为：`uid=${SubjectCN}`。

您还可以指定多个属性和值作为证书过滤器的一部分。过滤器规范的 LDAP 属性取决于将 LDAP 服务器配置为要使用的模式。客户机证书属性是客户机证书中的一个公共属性。客户机证书属性必须以美元符号 $ 和左花括号 { 开头，以右花括号 } 结尾。这些属性区分大小写。

支持下列 LDAP 属性：

- `uid`
- `initials`
- `sAMAccountName`
- `displayName`
- `distinguishedName`
- `displayName`
- `description`

支持下列客户机证书属性：

- `${SubjectCN}`
- `${SubjectDN}`
- `${IssuerCN}`
- `${IssuerDN}`
- `${SerialNumber}`

示例为启用了证书过滤方式的 LDAP 配置：

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
      host="myldap.vsettan.com" port="389" ignoreCase="true"
      baseDN="o=vsettan,c=cn"
      certificateMapMode="CERTIFICATE_FILTER"
      certificateFilter="uid=${SubjectCN}"
      userFilter="(&amp;(uid=%v)(objectclass=ePerson))"
      groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)
          (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
      userIdMap="*:uid"
      groupIdMap="*:cn"
      groupMemberIdMap="vsettan-allGroups:member;vsettan-
allGroups:uniqueMember;
          groupOfNames:member;groupOfUniqueNames:uniqueMember"
      ldapType="IBM Tivoli Directory Server" searchTimeout="8m" />
```

配置 SCIM 以进行用户和组成员管理

可在 server.xml 文件中配置 `scim-1.0` 功能部件以启用用户和组成员管理。跨域身份管理系统 (SCIM) 定义 REST API 以创建、检索、更新及删除 (CRUD) 用户和组。调用是通过系统管理 REST WAB 进行的。本地调用将是基于本地主机的 HTTP（仅通过 Web API）；本地调用不使用 Java API。

- 在 server.xml 文件中添加 `scim-1.0` 功能部件将启用 SCIM 功能。但是，为完成配置，还必须执行以下配置步骤：

- SSL 配置：REST 服务受保护并且只能在 HTTPS 端口上访问。有关如何完成 SSL 配置的更多信息，请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）。
- 联合注册表的配置：仅联合注册表支持 SCIM 功能。要使用 LDAP 快速设置联合注册表，请参阅使用 *xigemaAS* 概要文件来配置 *LDAP* 用户注册表（见第 1294 页）。
- 管理员角色的配置：只有管理员可访问 REST 服务，所以需要配置具有管理员角色的用户。有关将管理员角色映射至 xigemaAS 的更多信息，请参阅映射 *xigemaAS* 的管理员角色（见第 1180 页）。

  👉 注：要对 SCIM 配置管理员角色，不能使用快速启动注册表。

- HTTPS 端口的配置（可选）：必须配置 HTTP 端点。有关 `httpEndpoint` 功能部件元素配置的更多信息，请参阅*Admin Center 1.0*（见第 930 页） 中的 `httpEndpoint` 部分。

完成配置步骤后，可立即使用 `scim-1.0` 功能部件。以下示例中显示 server.xml 文件中的样本配置：

```
<server description="server1">
    <!-- Enable features -->
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>servlet-3.0</feature>
 <feature>ldapRegistry-3.0</feature>
 <feature>scim-1.0</feature>
 <feature>ssl-1.0</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" httpPort="9080" httpsPort="9090">
        <tcpOptions soReuseAddr="true" />
    </httpEndpoint>

    <ldapRegistry id="LDAP1" realm="SampleLdapIDSRealm" host="9.127.1.90" port="1389"
 ignoreCase="true"
    baseDN="o=vsettan,c=cn" ldapType="IBM Tivoli Directory Server" searchTimeout="8m"
 recursiveSearch="true"
        bindDN="cn=xxxx" bindPassword="xxxxxx">
     <ldapEntityType name="PersonAccount">
         <rdnProperty name="uid" objectClass="inetOrgPerson"/>
                <objectClass>inetOrgPerson</objectClass>
            </ldapEntityType>
            <ldapEntityType name="Group">
         <objectClass>groupofnames</objectClass>
  <objectClass>ibm-nestedGroup</objectClass>
    <rdnProperty name="cn" objectClass="groupofnames"/>
     </ldapEntityType>
    <attributeConfiguration>
        <attribute name="title" propertyName="honorificPrefix" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="initials" propertyName="middleName" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="st" propertyName="honorificSuffix" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="l" propertyName="homeStateOrProvinceName" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="street" propertyName="homeStreet" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="postalAddress" propertyName="homeCity" syntax="String"
 entityType="PersonAccount">
        </attribute>
 <attribute name="postalCode" propertyName="homePostalCode" syntax="String"
 entityType="PersonAccount">
        </attribute>
       <attribute name="postOfficeBox" propertyName="homeCountryName" syntax="String"
 entityType="PersonAccount">
        </attribute>
        <attribute name="departmentNumber" propertyName="photoURLThumbnail" syntax="String"
 entityType="PersonAccount">
```

```
        </attribute>
  <attribute name="description" propertyName="photoURL" syntax="String"
  entityType="PersonAccount">
        </attribute>
     </attributeConfiguration>
     <groupProperties>
         <memberAttribute name="member" dummyMember="uid=dummy" objectClass="groupOfNames"
  scope="direct"/>
  <memberAttribute name="ibm-memberGroup" objectClass="ibm-nestedGroup" scope="direct"/>
     </groupProperties>
   </ldapRegistry>

   <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" />
   <keyStore id="defaultKeyStore" password="xigemaAS"/>

   <administrator-role>
     <user>wasadmin</user>
   </administrator-role>
   <federatedRepository>
       <primaryRealm name="WIMRegistry">
       <participatingBaseEntry name="o=vsettan,c=cn"/>
        </primaryRealm>
    </federatedRepository>
</server>
```

xigemaAS 中的 SCIM 操作

跨域身份管理系统 (SCIM) 1.1 规范在 xigemaAS 中受支持。

xigemaAS 支持 SCIM 1.1 规范。有关此规范的更多信息，请参阅*http://www.simplecloud.info/*。

检索资源

要检索已知资源，必须将 GET 请求发送至所配置 HTTP 端点。例如，`/Users/{id}` **或** `/Groups/{id}`。

以下示例显示针对 LDAP 注册表的操作。

```
    <ldapRegistry id="LDAP1" realm="SampleLdapIDSRealm" host="9.127.1.90" port="1389"
ignoreCase="true"
                baseDN="o=vsettan,c=cn" ldapType="IBM Tivoli Directory Server"
searchTimeout="8m" recursiveSearch="true"
        bindDN="xxxxxx" bindPassword="xxxxxxx">
     <ldapEntityType name="PersonAccount">
         <rdnProperty name="uid" objectClass="inetOrgPerson"/>
                <objectClass>inetOrgPerson</objectClass>
            </ldapEntityType>
            <ldapEntityType name="Group">
         <objectClass>groupofnames</objectClass>
 <objectClass>ibm-nestedGroup</objectClass>
    <rdnProperty name="cn" objectClass="groupofnames"/>
     </ldapEntityType>
    <attributeConfiguration>
        <attribute name="title" propertyName="honorificPrefix" syntax="String"
entityType="PersonAccount">
        </attribute>
<attribute name="initials" propertyName="middleName" syntax="String"
entityType="PersonAccount">
        </attribute>
<attribute name="st" propertyName="honorificSuffix" syntax="String" entityType="PersonAccount">
        </attribute>
<attribute name="l" propertyName="homeStateOrProvinceName" syntax="String"
entityType="PersonAccount">
        </attribute>
<attribute name="street" propertyName="homeStreet" syntax="String" entityType="PersonAccount">
        </attribute>
<attribute name="postalAddress" propertyName="homeCity" syntax="String"
entityType="PersonAccount">
        </attribute>
<attribute name="postalCode" propertyName="homePostalCode" syntax="String"
entityType="PersonAccount">
```

```
        </attribute>
       <attribute name="postOfficeBox" propertyName="homeCountryName" syntax="String"
entityType="PersonAccount">
        </attribute>
        <attribute name="departmentNumber" propertyName="photoURLThumbnail" syntax="String"
entityType="PersonAccount">
        </attribute>
<attribute name="description" propertyName="photoURL" syntax="String"
entityType="PersonAccount">
        </attribute>
    </attributeConfiguration>
    <groupProperties>
        <memberAttribute name="member" dummyMember="uid=dummy" objectClass="groupOfNames"
scope="direct"/>
<memberAttribute name="ibm-memberGroup" objectClass="ibm-nestedGroup" scope="direct"/>
    </groupProperties>
    </ldapRegistry>
```

要从 LDAP 检索资源，必须将 GET 请求作为 https://localhost:9090/vsettan/api/scim/Users/uid=jsmith,o=vsettan,c=cn 发送。

查询资源

要查询资源，必须将 GET 请求发送至所配置 HTTP 端点并指定过滤器以进行搜索。还可指定 attributes 参数以从所返回资源返回值子集，并指定分页和排序参数以组织所返回资源。以下示例显示一些过滤器选项：

- 指定参数以检索值子集：例如中，`https://localhost:9090/vsettan/api/scim/Users?filter=givenname sw "Jo"` 检索名称以*"Jo"*开头的所有用户。
- 指定 attributes 参数以检索值子集：例如，`https://localhost:9090/vsettan/api/scim/Users?filter=givenname sw "Jo"&attributes=username,emails&filter=name.familyname eq "Smith"` 检索名字以*"Jo"*开头并且姓氏为*"Smith"*的用户名的电子邮件地址。
- 指定分页和排序参数以检索所组织资源。例如，`https://localhost:9090/vsettan/api/scim/Users?filter=givenname sw "Jo"&attributes=username&sortBy=username&startIndex=3&count=5` 检索已排序结果的第三页，每页包含 5 个用户名，这些用户名以*"Jo"*开头并且姓氏为*"Smith"*。

👉 注：

- 底层联合存储库确定过滤器中指定的值是否区分大小写。对于 LDAP，这些值在缺省情况下不区分大小写，除非该属性映射至区分大小写的 LDAP 属性。
- 不能在搜索过滤器中使用用户名。
- 联合不支持 SCIM 属性的基本注册表、SAF 注册表或定制注册表时，将始终针对用户名搜索过滤器模式，并且仅返回用户名属性。

创建资源

要创建新资源，必须将 POST 请求发送至资源端点，即 `Users` 或 `Groups`。POST 内容必须包含用户 `json` 对象，并且 HTTP 头 content-type 必须设置为 `application/json`。

👉 注：`application/xml` 内容类型不受支持。

要创建用户，必须将以下请求发布至 https://localhost:9090/vsettan/api/scim/Users Content-Type

```
Content-Type: application/json
Content-Length: ...

{
  "schemas":["urn:scim:schemas:core:1.0"],
  "userName":"bjensen",
```

```
  "externalId": "uid=bjensen,o=vsettan,c=cn",
  "name":{
    "familyName":"Jensen",
    "givenName":"Barbara"
  }
}
```

请求必须包含在用户注册表中成功创建用户或组所需的全部属性。例如，在后端 Tivoli Directory Server LDAP 中，创建用户所需的基本 LDAP 属性集将为 `uid`、`sn` 和 `cn`；所以，任何创建用户的请求必须包含 `userName`、`givenName` 和 `familyName`。如果用户注册表中存在任何模式违例，那么创建操作将失败。

externalId 属性将充当标识，并且必须指定。属性格式依赖于后端用户注册表的规范。在先前示例中，后端为 LDAP 服务器，且其 baseEntry 设置为 `o=vsettan,c=cn`，用户的 RDN 属性设置为 UID，所以 `externalId` 变为 `uid=bjensen,o=vsettan,c=cn`。

根据创建请求，将返回所创建用户对象。

```
{
    "schemas":["urn:scim:schemas:core:1.0"],
    "id":"uid=bjensen,o=vsettan,c=cn",
    "userName":"bjensen",
    "externalId":"uid=bjensen,o=vsettan,c=cn",
    "name":{
          "formatted":"Barbara Jensen",
          "familyName":"Jensen",
          "givenName":"Barbara"
    },
    "meta":{
    "lastModified":"2015-09-15T14:30:11",        "location":"https:\\localhost:9090\vsettan\api
\scim\Users\uid=bjensen,o=vsettan,c=cn",
         "created":"2015-09-15T14:30:11"
    }
}
```

如果配置了多个 LDAP 并且需要调用创建操作，那么必须对实体定义缺省父代。所定义父代指定将在其下创建新实体的基本条目。以下示例显示 `PersonAccount` 的缺省父代的配置。

```
<federatedRepository>
    <primaryRealm name="WIMRegistry">
    <participatingBaseEntry name="o=vsettan,c=cn"/>
        <participatingBaseEntry name="o=ldap"/>
     </primaryRealm>
    <supportedEntityType>
        <defaultParent>o=ldap</defaultParent>
        <name>PersonAccount</name>
    </supportedEntityType>
</federatedRepository>
```

修改资源

要修改资源，您需要将 PUT 请求发送至资源端点，即 `/Users` 或 `/Groups`。PUT 请求执行资源的完整更新。输入中未指定的任何属性将被删除。

注：使用 PATCH 进行的部分更新不受支持。

要修改先前创建的资源，必须将以下请求发布至 https://localhost:9090/vsettan/api/scim/Users/uid=bjensen,o=vsettan,c=cn。

```
Content-Type: application/json
Content-Length: ...

{
  "schemas":["urn:scim:schemas:core:1.0"],
  "userName":"bjensen",
```

```
  "externalId":"uid=bjensen,o=vsettan,c=cn",
  "name":{
    "familyName":"Jensen",
    "givenName":"Barb"
  }
}
```

此请求将对象的给定名称从 *Barbara* 修改为 *Barb*。URL 中指定的标识必须与用户对象中指定的 `externalId` 匹配。

要修改组 `cn=employeeGroup,o=vsettan,c=cn` 并更改组成员资格，需要指定成员属性。

```
Content-Type: application/json
Content-Length: ...
{
"id":"cn=employeeGroup,o=vsettan,c=cn",
    "schemas":["urn:scim:schemas:core:1.0"],
    "displayName":"employeeGroup",
    "externalId":"cn=employeeGroup,o=vsettan,c=cn",
    "members":[{"value":"uid=bjensen,o=vsettan,c=cn", "type":"User"},
{"value":"cn=consultants,o=vsettan,c=cn", "type":"Group"}]
}
```

要修改组 `cn=employeeGroup,o=vsettan,c=cn` 并移除组中的所有成员，需要指定空成员属性。

```
Content-Type: application/json
Content-Length: ...
{
"id":"cn=employeeGroup,o=vsettan,c=cn",
    "schemas":["urn:scim:schemas:core:1.0"],
    "displayName":"employeeGroup",
    "externalId":"cn=employeeGroup,o=vsettan,c=cn",
    "members":[]
}
```

删除资源

要删除资源，需要将 DELETE 请求发送至资源端点，即 `/Users` 或 `/Groups`。例如，要删除先前示例中创建的用户 `uid=bjensen,o=vsettan,c=cn`，请将请求发送至 https://localhost:9090/vsettan/api/scim/Users/uid=bjensen,o=vsettan,c=cn

👉 注：SCIM 仅在 Java 7 或更高版本上运行。

为用户和组配置附加属性

可为联合存储库的用户和组配置附加属性。要启用模式或属性扩展，请确保可对底层存储库读写该属性。

1. 可在 server.xml 文件中指定以下附加属性信息以启用模式或属性扩展。
   - 扩展属性名称 - 扩展属性的名称。确保所指定名称是唯一的并且与现有属性名称不匹配。
   - 数据类型 - 扩展属性的数据类型。可能的值为 Integer、Long、String、Boolean、Date、Double、BigInteger 和 BigDecimal。
   - 实体类型 - 该属性适用于的实体。可能的值为 PersonAccount 或 Group。

     单值或多值 - 可将该属性的值设置为单值或多值。还可为该属性设置缺省值。如果已创建实体并且未对该属性指定任何值，那么将使用缺省值。 对于多值属性，可添加名为 `assetId` 的扩展属性以存储分配给用户的资产。如果可对每个用户分配多个资产，那么 `assetId` 需要设置为多值。必须确保 `assetId` 映射至的属性在后端 LDAP 中也是多值属性。

   以下样本显示 server.xml 中的配置：

```
<federatedRepository>
```

```
    <primaryRealm name="WIMRegistry">
        <participatingBaseEntry name="o=vsettan,c=cn"/>
     </primaryRealm>
    <extendedProperty dataType="String" name="extendedProperty" entityType="PersonAccount"> </
extendedProperty>
</federatedRepository>
```

2. 要在代码中使用扩展属性，必须按以下示例中所示使用通用 getter/setter 方法：

```
PersonAccount person = new PersonAccount();
...
person.set("extendedProperty", "xyz");
...
String value = (String)person.get("extendedProperty");
```

3. 为确保可对 LDAP 读写该属性，您有以下两个选项：
   - 传递：如果扩展属性的名称与 LDAP 属性的名称相同，那么将传递该扩展属性并对该 LDAP 属性读写该扩展属性。
   - 属性映射：如果扩展属性的名称与 LDAP 属性的名称不同，那么需要使用属性映射来映射该扩展属性。

     以下样本配置显示扩展属性至名为 *extendedAttribute* 的属性的映射。

```
<attributeConfiguration>
    <attribute name="extendedAttribute" propertyName="extendedProperty" syntax="String"
 entityType="PersonAccount"></attribute>
</attributeConfiguration>
```

对安全性的动态更改

本主题介绍了有关动态更改配置将如何影响 xigemaAS 概要文件的安全性的一些具体信息。

动态更改用户注册表

更改用户注册表会影响服务器配置及使用该服务器的客户机。在动态更改用户注册表之前（不重新启动服务器），请考虑下列事项：

- 如果您更改用户注册表类型或域名，那么所有 Web 客户机都必须清除其 Web 单点登录令牌。
- 如果更改用户注册表类型或域名，那么必须更新在授权绑定中给 accessId 指定的任何值。accessId 接受格式 `user:realmName/uniqueId` 或 `group:realmName/uniqueId`。accessId 中的 realmName 必须与配置用户注册表的 realmName 匹配。

为 xigemaAS 开发定制用户存储库

可通过实现 xigemaAS 服务器中提供的 `com.ibm.ws.security.wim.RepositoryFactory` 和 `com.ibm.ws.security.wim.Repository` 接口来开发定制用户存储库类。

这些存储库接口启用可视化任何类型的帐户存储库的支持。

1. 实现存储库工厂 (`com.ibm.ws.security.wim.RepositoryFactory`) 接口。`RepositoryFactory` 接口获取配置参数并创建存储库的实例。

   例如，

```
public Repository getRepository(Map<String, Object> properties) throws WIMException {
return new CustomRepository(properties);}
```

2. 实现存储库 (`com.ibm.ws.security.wim.Repository`) 接口。此类具有实际存储库操作。

```
public class CustomRepository extends RepositoryConfiguration implements Repository {
```

```
    public CustomRepository(Map<String, Object> properties) {
        System.out.println("Constructor with " + properties);
    }
```

3. 将实现类转换为 OSGi 服务。有关更多信息，请参阅向 *OSGi* 声明式服务声明服务（见第 1248 页）。
4. 将定制用户存储库封装为 OSGi 捆绑软件并导出用户存储库服务。
5. 创建功能部件清单来包含 OSGi 捆绑软件。有关更多信息，请参阅产品扩展（见第 1028 页）。
6. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。

   例如：

```
<featureManager>
 ...
 <feature>usr:customRepositorySample-1.0</feature>
</featureManager>
```

请参阅存储库工厂和存储库接口的以下样本。

存储库工厂接口

```
package com.myorg;

import java.util.Map;

import org.osgi.service.component.ComponentContext;

import com.ibm.websphere.security.wim.exception.WIMException;
import com.ibm.ws.security.wim.Repository;
import com.ibm.ws.security.wim.RepositoryFactory;

public class CustomRepositoryFactory implements RepositoryFactory {

    @Override
    public Repository getRepository(Map<String, Object> properties) throws WIMException {
        System.out.println("getRepository " + properties);
        return new CustomRepository(properties);
    }

    public void activate(ComponentContext cc, Map<String, Object> properties) {
        System.out.println("In activate");
    }

    public void deactivate(ComponentContext cc) {
        System.out.println("In deactivate");
    }
}
```

存储库接口

```
package com.myorg;

import java.util.Map;

import com.ibm.websphere.security.wim.exception.WIMException;
import com.ibm.websphere.security.wim.model.Root;
import com.ibm.ws.security.wim.Repository;
import com.ibm.ws.security.wim.RepositoryConfiguration;

public class CustomRepository extends RepositoryConfiguration implements Repository {

    public CustomRepository(Map<String, Object> properties) {
        System.out.println("Constructor with " + properties);
    }

    @Override
    public Root create(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }
```

```
    @Override
    public Root delete(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }

    @Override
    public Root get(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }

    @Override
    public String getRealm() {
        return "customRepository";
    }

    @Override
    public Root login(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }

    @Override
    public Root search(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }

    @Override
    public Root update(Root arg0) throws WIMException {
        throw new WIMException("Method not supported");
    }
}
```

在 xigemaAS 上配置认证高速缓存

可以修改在 xigemaAS 上对已认证的用户进行高速缓存的方式。

因为创建主体可能会影响性能，所以 xigemaAS 会在成功认证用户之后提供认证高速缓存来存储主体。高速缓存是使用由 initialSize 属性所确定的特定数目的条目来初始化，并且具有由 maxSize 属性确定的最大数目的条目。如果达到最大大小，那么会从高速缓存中移除所使用的最早条目。如果用户处于不活动状态的时间超过了 timeout 属性所指定的时间段，那么会从高速缓存中移除该用户的条目。缺省情况下，高速缓存大小初始化为 50 个条目，最大条目数为 25000，超时值为 600 秒。

不必为 `authCache` 元素配置这些值，除非您想要更改认证高速缓存的缺省值。

有关认证案例的更多信息，请参阅认证高速缓存（见第 1036 页）。

注：

- 对 server.xml 文件中的用户注册表配置进行任何更改都会清除认证高速缓存。但是，如果对外部用户注册表（例如，LDAP）进行了更改，那么认证高速缓存不受影响。
- 必须考虑超时值对配置的下列影响：
  - 认证高速缓存超时值越大，安全性风险越高。例如，您可以撤销用户注册表或者存储库中的用户，但是，撤销的用户可以使用高速缓存在认证高速缓存中的凭证来登录，直到刷新高速缓存为止。
  - 较小的认证高速缓存超时值可能会影响性能。当此值较小时，xigemaAS 服务器会更频繁地访问用户注册表或用户资源库。
  - 用户数增加所导致的认证高速缓存中条目数越大，认证高速缓存使用的内存就越多。因此，应用程序服务器可能会变慢并影响性能。

1. 通过向 server.xml 文件中添加以下代码来启用 appSecurity-2.0 xigemaAS 功能部件。

```
<featureManager>
   <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 可选： 要更改认证高速缓存的缺省选项，请将 <authCache> 元素添加到 server.xml 文件。在以下示例中，认证高速缓存的初始大小会更改为 100 个条目（最大值为 50000 个条目），而超时值会更改为 15 分钟。

```
<authCache initialSize="100" maxSize="50000" timeout="15m"/>
```

3. 可选： 要禁用认证高速缓存，请在 <authentication> 元素中将属性 cachEnabled 设为 false，如下所示：

```
<authentication id="Basic" cacheEnabled="false" />
```

为 xigemaAS 概要文件配置 JAAS 定制登录模块

可以在您配置 xigemaAS 概要文件服务器登录模块之前或之后配置定制 Java™ 认证和授权服务 (JAAS) 登录模块。

请确保您有一个 JAR 文件包含 JAAS 定制登录模块，此模块用来实现 javax.security.auth.spi.LoginModule 接口（如为系统登录配置开发 *JAAS* 定制登录模块（见第 1410 页）中所述）。在本主题中，JAAS 定制登录模块使用 xigemaAS 概要文件服务器提供的散列表、回调或共享状态变量来将认证数据传递到系统登录模块。

可以使用定制登录模块来作出其他认证决策，或者在主体中添加信息以在应用程序内部作出更细颗粒度的权限决策。有关更详细的概述，请参阅*JAAS* 配置（见第 1036 页）和*JAAS* 登录模块（见第 1037 页）。

要配置 JAAS 定制登录模块，请完成以下步骤：

1. 在 server.xml 文件中启用 appSecurity-2.0 xigemaAS 功能部件。

```
<featureManager>
    <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 创建用于实现 LoginModule 接口的 com.sample.CustomLoginModule 类并将其打包到 CustomLoginModule.jar 文件中。
3. 使用一个指示 CustomLoginModule.jar 文件所在位置的 <fileset> 元素来创建 <library> 元素。在此示例中，库 id 是 customLoginLib。

```
<library id="customLoginLib">
    <fileset dir="${server.config.dir}" includes="CustomLoginModule.jar"/>
</library>
```

4. 创建 <jaasLoginModule> 元素。在此示例中，id 是 custom。
   a. 通过将 controlFlag 属性设置为 REQUIRED，将定制登录模块配置为需要成功认证。
   b. 将 libraryRef 属性设置为 customLoginLib（在上一步配置的 <library> 元素的 id）。此登录模块也具有两个选项：UserRegistry 是 ldap，而 mapToUser 是 user1。

```
<jaasLoginModule id="myCustom"
                 className="com.sample.CustomLoginModule"
                 controlFlag="REQUIRED" libraryRef="customLoginLib">
    <options myOption1="value1" myOption2="value2"/>
</jaasLoginModule>
```

5. 使用系统定义的 JAAS 配置 (system.WEB_INBOUND) 的 id 和唯一 name 来创建 `<jaasLogincontextEntry>` 元素。您还可以将此 JAAS 配置设置为 system.DEFAULT、WSLogin 或您自己的 JAAS 配置。在 `loginModuleRef` 属性上，添加 `custom`（在上一步创建 `jaasLoginModule` 元素的 id）。将此 id 放在列表中的最前面表示它是要调用的第一个 JAAS 登录模块。您还必须列出其他缺省登录模块：**hashtable**、**userNameAndPassword**、**certificate** 和 **token**。

```
<jaasLoginContextEntry id="system.WEB_INBOUND" name="system.WEB_INBOUND"
                  loginModuleRef="myCustom, hashtable, userNameAndPassword, certificate,
 token"  />
```

👉 注：选项名称不能以句点 (.)、config. 或 service 开头，并且必须唯一。此外，也不接受属性名 id 或 ID。

使用 xigemaAS 的 JAAS 配置文件配置应用程序 JAAS 定制登录上下文条目和登录模块

可在 JAAS 配置文件中配置 JAAS 配置信息。

我们支持 server.xml 文件、client.xml 文件和 JAAS 配置的 JAAS 配置文件。但是，建议在 server.xml 文件或 client.xml 文件中配置 JAAS 定制登录模块。有关配置 JAAS 定制登录模块的更多详细信息，请参阅为 *xigemaAS* 概要文件配置 *JAAS* 定制登录模块（见第 1307 页）。

xigemaAS 服务器读取应用程序 JAAS 定制登录上下文条目和登录模块的 JAAS 配置文件。对 JAAS 配置文件的更改由本地应用程序使用并在应用程序服务器重新启动后生效。server.xml 文件中的 JAAS 配置优先于 JAAS 配置文件中定义的配置。JAAS 配置文件中的配置条目将被 server.xml 文件中相同别名的条目覆盖。

要配置 JAAS 定制登录模块，请完成以下步骤：

1. 在 server.xml 文件中启用 `appSecurity-2.0` xigemaAS 功能部件。

```
<featureManager>
    <feature>appSecurity-2.0</feature>
    ...
</featureManager>
```

2. 创建 JAAS 定制登录模块类。

   例如，`com.sample.CustomLoginModule`，它将实现 `LoginModule` 接口并将其打包至 CustomLoginModule.jar 文件。

3. 创建缺省 jaas 目录。

对于服务器

${server.config.dir}/resources/security/jaas

对于客户机

${client.config.dir}/resources/security/jaas

👉 注：JAAS 配置文件中指定的所有 JAAS 定制登录模块必须放置在缺省 jaas 目录中。

4. 将 CustomLoginModule.jar 文件放置在缺省 jaas 目录中。
5. 创建 JAAS 配置文件。

   例如，创建 myJaas.conf 文件并将其放置在具有以下内容的 ${server.config.dir}/resources/security/jaas 目录中：

```
myCustomLoginContext {
```

```
    com.sample.CustomLoginModule required myOption1="value1"
 myOption2="value2"
};
```

6. 使用 jvm.options 文件配置 JAAS 配置文件。

   例如，

```
-Djava.security.auth.login.config=${server.config.dir}/resources/security/
jaas/myJaas.conf
```

   注：仅支持 JAAS 配置文件中的应用程序定制 JAAS 登录模块。不要将缺省系统 JAAS 配置信息放置在 JAAS 配置文件中。

   注：如果您进行了任何更改，那么 JAAS 配置文件不会动态更新。强烈建议在 server.xml 文件或 client.xml 文件中配置 JAAS 配置信息。

配置 Java™ Authentication SPI for Containers (JASPIC) 用户功能部件

通过使用 xigemaAS 服务器中提供的 `com.ibm.wsspi.security.jaspi.ProviderService` 接口，可开发 JASPIC 提供程序以认证入站 Web 请求。

Java™ 容器认证 SPI 规范 *JSR 196* 为认证服务提供者定义接口。在 xigemaAS 概要文件服务器中，必须将 JASPIC 提供程序打包为用户功能部件。您的功能部件必须实现 `com.ibm.wsspi.security.jaspi.ProviderService` 接口。

1. 创建 OSGi 组件，该组件提供用于实现 `com.ibm.wsspi.security.jaspi.ProviderService` 接口的服务。

   `ProviderService` 接口定义 `getAuthConfigProvider` 方法，xigemaAS 概要文件运行时调用此方法以检索用于实现 `javax.security.auth.message.config.AuthConfigProvider` 接口的 JASPIC 提供程序类。

   以下示例使用 OSGi 声明式服务注解：

```
@package com.mycompany.jaspi;

import java.util.Map;
import javax.security.auth.message.config.AuthConfigFactory;
import javax.security.auth.message.config.AuthConfigProvider;
import org.osgi.service.component.ComponentContext;
import com.mycompany.jaspi.SampleAuthConfigProvider;
import com.ibm.wsspi.security.jaspi.ProviderService;

@Component(service = { ProviderService.class },
           configurationPolicy = ConfigurationPolicy.IGNORE,
           immediate = true,
           property = { "myPoviderPoperty1=value1",
                        "myPoviderPoperty2=value2"})
public class SampleJaspiProviderService implements ProviderService {

    Map<String, String> configProps = null;

    // This method called by the xigemaAS profile runtime
    // to get an instance of AuthConfigProvider
    @Override
    public AuthConfigProvider getAuthConfigProvider(Map<String, String>
                                                    AuthConfigFactory factory)
```

```
    {
        return new SampleAuthConfigProvider(configProps, factory);
    }

    protected void activate(ComponentContext cc) {
        // Read provider config properties here if needed,
        // then pass them to the AuthConfigProvider factory.
        // This example reads the properties from the OSGi
        // component definition.
        configProps = (Map<String, String>) cc.getProperties();
    }

    protected void deactivate(ComponentContext cc) {}
}
```

2. 将该组件与您的 JASPIC 认证服务提供程序一起打包成用户功能部件中的 OSGi 捆绑软件。
3. 确保您的功能部件包含 OSGi 子系统内容：`com.ibm.websphere.appserver.jaspic-1.1; type="osgi.subsystem.feature"`。
4. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。例如：

```
<featureManager>
   ...
   <feature>usr:myJaspiProvider</feature>
</featureManager>
```

在 xigemaAS 概要文件上配置 LTPA

您可以配置 xigemaAS 概要文件服务器来使用特定的轻量级第三方认证 (LTPA) 密钥文件、用户定义的密码以及到期时间。

缺省情况下，首次对 xigemaAS 概要文件服务器启用安全性时，会配置 LTPA。自动生成的 LTPA 密钥文件的缺省位置是 ${server.output.dir}/resources/security/ltpa.keys。使用随机生成的密钥对这些 LTPA 密钥进行加密，而且最初使用缺省密码 `WebAS` 来保护密钥。将 LTPA 密钥导入到另一个服务器时，需要该密码。要保护 LTPA 密钥的安全性，必须更改密码。在服务器之间交换 LTPA 密钥时，此密码必须跨服务器进行匹配才能使单点登录 (SSO) 正常工作。

缺省到期超时为 120 分钟。到期值指的是 LTPA 令牌到期之前的有效时间。

要启用从另一个服务器复制 LTPA 密钥文件时动态重新载入 LTPA 密钥，可以在复制 LTPA 密钥文件之前指定文件监视时间间隔。监视时间间隔值指的是监视 LTPA 密钥文件更新的频率。

有关 LTPA 的更多信息，请参阅 *xigemaAS 概要文件中的 LTPA 概念*。

1. 在 server.xml 文件中按如下所示配置 `<ltpa>` 元素，并将该示例中的样本值替换为您自己的值：

```
<ltpa keysFileName="yourLTPAKeysFileName.keys" keysPassword="keysPassword" expiration="120" /
>
```

2. 设置 monitorInterval 属性以检查 lpta.keys 文件中是否有要动态重新装入的关键更改。指定为后跟时间单位的正整数，时间单位可以是小时 (h)、分钟 (m) 或秒 (s)。在以下示例中，每隔 5 秒钟就会检查 LTPA 密钥文件中是否有要动态重新装入的更改：

```
<ltpa keysFileName="yourLTPAKeysFileName.keys" keysPassword="keysPassword"
      expiration="120" monitorInterval="5s" />
```

3. 对配置中的密码进行编码。可以使用 *securityUtility encode* 命令来获取编码值。

4. 将现有 LTPA 密钥文件复制到 keysFileName 属性中指定的位置。缺省值为 ${server.output.dir}/resources/security/ltpa.keys。

OpenID

OpenID 是一个开放式标准，在此标准中，用户可对多个实体认证自身而不需要管理多个帐户或多组凭证。xigemaAS 概要文件支持 OpenID 2.0 在 Web 单点登录中充当依赖方的角色。

OpenID 提供程序 (OP)

一个 OpenID 认证服务器，可断言用户是否控制唯一标识。

依赖方 **(RP)**

一个实体，需要用户控制唯一标识的证明。

OpenID 标识：

一个 http 或 https URL，属于 OpenID 提供程序或用户。

访问 Web 站点之类的各种实体通常需要与每个实体相关联的唯一帐户。OpenID 允许 OpenID 提供者控制一组凭证来授予对支持 OpenID 的任意数目的实体的访问权。

如果需要登录实体（例如，支持 OpenID 并充当依赖方的 Web 站点），那么用户应通过直接与 OP 交互而不是对 RP 本身提供凭证来执行认证。OP 验证用户身份并将认证确认发送回 RP。OP 接收到此确认时，RP 接受用户通过认证。

下面描述了典型的 OpenID 认证流程：

1. 用户尝试访问受保护资源，例如，网页。
2. 充当 RP 的 xigemaAS 服务器为受保护资源提供表单登录页面。
3. 用户输入 OpenID 标识。
4. RP 获取该标识并将用户重定向至相应 OP。
5. OP 提示用户输入凭证。
6. 用户输入与 OP 相关联的帐户的凭证。
7. OP 认证用户并（可选）提示用户批准或拒绝向 RP 提供用户信息，然后将用户重定向回 RP 并显示认证结果。
8. 如果 OP 认证成功，那么 RP 会尝试对用户授权。
9. 如果用户授权成功，那么 RP 会建立与用户的已认证会话。

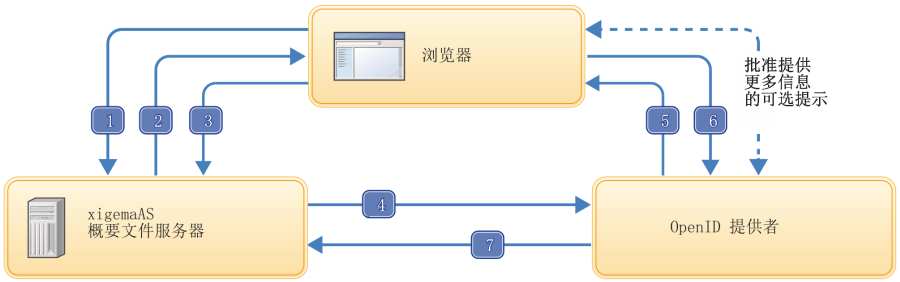


图 31: OpenID 认证流程

OpenID 有助于将错误处理用户凭证或敏感信息的机会降至最低。通过 OpenID，用户凭证仅与 OpenID 提供者交换，这意味着 OP 以外的 Web 站点不会见到用户凭证。此标准有助于减少不道德或不安全的 Web 站点危害用户身份的可能性。用户还会控制与所访问网站分享的个人信息量。例如，用户可选择允许与一个 Web 站点共享与他们的 OpenID 帐户相关联的姓名和电子邮件地址，同时选择不向另一 Web 站点提供他们的电子邮件地址或根本不向另一 Web 站点提供任何信息。

受支持的 OpenID 标准规范包括：

OpenID Authentication 2.0.

OpenID Attribute Exchange 1.0.

注：应该使用 OpenID 2.0 声明标识以针对每个规范标识用户（标识最终用户）。但是，声明标识不是用户友好的，许多依赖方选择一个 OpenID 属性来表示用户。用于表示用户的最热门属性是用户的电子邮件地址。

大家都知道 OpenID 提供者可能不会验证某些 OpenID 属性（包括电子邮件）。如果不信任提供者将为您提供已验证的电子邮件地址，那么不得将该提供者的电子邮件地址用作 xigemaAS 中的已认证用户名。

OpenID Connect

OpenID Connect 是一个简单身份协议，也是在 OAuth 2.0 协议（允许客户机应用程序依赖 OpenID Connect 提供程序执行的认证来验证用户身份）基础上构建的开放式标准。

OpenID Connect 使用 OAuth 2.0 进行认证和授权，随后构建唯一标识用户的身份。客户机应用程序还可以可互操作方式及类 REST 的方式从 OpenID Connect 提供程序处获取有关用户的基本概要文件信息。

xigemaAS 支持 OpenID Connect 1.0 并在 Web 单点登录中充当客户机、依赖方以及提供程序的角色。支持以下 OpenID Connect 规范：

OpenID Connect Core 1.0：配置为 OpenID Connect 依赖方的 xigemaAS 概要文件服务器支持使用授权代码流程进行认证。配置为 OpenID Connect 提供者的 xigemaAS 概要文件服务器支持使用授权代码流程和隐式流程进行认证。

对于将 xigemaAS 概要文件服务器用作基于 Web 的依赖方的对象，*OpenID Connect Basic Client Implementer's Guide 1.0* 是易于读取且为使用授权代码流程的基于 Web 的依赖方提供详细信息的 OpenID Connect Core 规范的一部分。

访问令牌

用于访问受保护资源的凭证。访问令牌是一个字符串，用于表示对客户机发出的授权。

授权端点

OpenID 提供程序上的资源，接受来自客户机要求执行用户认证和授权的请求。在授权代码流程中，授权端点对客户机返回授权或代码。在隐式流程中，授权端点对客户机返回标识令牌和访问令牌。

授权

一个凭证，表示用于访问资源的用户授权，被客户机用于获取访问令牌。

声明

有关实体的已断言信息。声明示例包含电话号码、名字、姓氏及其他。

标识令牌

一个 JSON Web 令牌 (JWT)，包含有关已认证用户的声明。

内省端点

OpenID 提供程序上的资源，允许持有访问令牌的客户机检索用于创建访问令牌的信息，例如，用户名、授予范围、客户机标识或其他信息。

OpenID Connect 提供程序 (OP)

OAuth 2.0 授权服务器，能够对客户机或依赖方 (RP) 提供声明。

刷新令牌

由 OP 对客户机发出，用于获取新的访问令牌（在当前访问令牌到期时）或更多访问令牌。

依赖方 **(RP)**

配置为 OpenID Connect 客户机的 xigemaAS 服务器或需要来自 OpenID 提供程序 (OP) 的声明的客户机应用程序。

范围

允许访问第三方资源的特权或许可权。

令牌端点

OP 上的资源，在交换中接受来自客户机的授权或代码以获取访问令牌、标识令牌和刷新令牌。

作为 OpenID Connect 客户机的 xigemaAS 概要文件服务器

可配置 xigemaAS 概要文件以充当 OpenID Connect 客户机。此设置允许 xigemaAS 概要文件服务器依赖另一充当用户认证和授权的 OP 的 xigemaAS 概要文件服务器。

配置为充当 OpenID Connect 客户机的 xigemaAS 概要文件服务器支持 OpenID Connect 1.0 标准的授权代码流程。

在授权代码流程中，所有令牌交换是通过使用 OpenID Connect 提供程序的令牌端点处理的。首先，客户机向 OP 的授权端点提交授权请求。向 OP 成功认证和授权时，客户机从 OP 接收授权或代码。此授权代码可通过请求发送至 OP 的令牌端点。客户机在来自令牌端点的响应时接收标识令牌、访问令牌和刷新令牌。然后，客户机验证标识令牌并检索用户的主体集标识。此概要文件流程适用于能够在他们自身与 OP 之间安全维护客户端密钥的客户端。此流程还允许客户机获取刷新令牌。

有关配置为 OpenID Connect 客户机的 xigemaAS 服务器，请参阅在 *xigemaAS* 中配置 *OpenID Connect* 客户机（见第 1351 页）

作为 OpenID Connect 提供程序的 xigemaAS 服务器

可配置 xigemaAS 以充当 OpenID Connect 提供提供程序。此设置允许 xigemaAS 服务器充当可供 OpenID Connect 客户机使用的授权服务器。

配置为充当 OpenID Connect 提供提供程序的 xigemaAS 服务器支持 OpenID Connect 1.0 标准的授权代码流程和隐式流程。每个流程确定如何将标识令牌、访问令牌和刷新令牌返回至客户机。

授权代码流程通过使用 OpenID Connect 提供程序的令牌端点处理所有令牌交换。OpenID Connect 提供程序在 OP 的授权端点接受来自客户机的授权请求。如果需要认证，那么 OpenID Connect 提供程序会执行相应认

证。OpenID Connect 提供程序还会从用户处获取任何必需的同意或授权，例如，通过在浏览器中提示用户提供授予对某些范围的访问权的许可权。如果成功，或者不需要任何认证，那么 OpenID 提供者会将授权或代码发送回客户机。然后 OpenID Connect 提供程序接受客户机提交至其令牌端点的请求，此请求包含授权代码。包含授权代码的请求由 OpenID 提供程序进行验证。成功验证时，OpenID Connect 提供者对客户机返回包含标识令牌和访问令牌的响应。

与授权代码流程不同，在隐式流程中，所有令牌都是从授权端点返回的；不会使用 OP 的令牌端点。首先，客户机准备认证请求并将其发送至 OP 的授权端点。然后，OpenID Connect 提供程序执行任何必需的认证，同时从用户处获取任何必需的同意或授权。例如，OpenID Connect 提供程序在浏览器中提示用户提供授予对某些范围的访问权的许可权。成功认证和授权时，OpenID Connect 提供程序对客户机发送回标识令牌和访问令牌。然后，客户机验证标识令牌并检索用户的主体标识。此概要文件流程适用于无法在他们自身与 OP（例如，本机应用程序）之间安全维护客户端密钥的客户机。

有关将 xigemaAS 服务器配置为 OpenID Connect提供程序的信息，请参阅在 *xigemaAS* 中配置 *OpenID Connect* 提供者（见第 1326 页）

授权代码流程

下面描述了典型的 OpenID Connect 授权代码流程：

1. 用户在 RP 上访问应用程序。
2. RP 准备认证请求并将用户重定向至 OP。
3. OP 认证用户，例如，通过提示用户输入凭证。用户授权 RP 访问应用程序所需的信息。OP 为 RP 生成一次性的授权代码。
4. OP 将用户重定向回具有授权代码的 RP。
5. RP 调用 OP 的令牌端点以交换授权代码来获取访问令牌、标识令牌和刷新令牌。
6. RP 使用标识令牌来对用户授权。

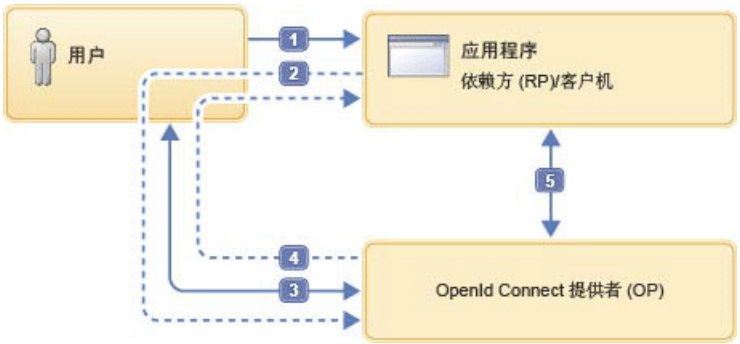


图 32: OpenID Connect 基本客户机概要文件流程

隐式流程

只有充当 OpenID Connect 提供程序的 xigemaAS 服务器才支持隐式流程。下面描述了典型的 OpenID Connect 隐式流程：

1. 用户在 RP 上访问应用程序。
2. RP 准备认证请求并将用户重定向至 OP。
3. OP 认证用户，例如，通过提示用户输入凭证。用户授权 RP 访问应用程序所需的信息。
4. OP 将用户重定向回具有标识令牌和访问令牌的 RP。
5. RP 使用标识令牌来对用户授权。

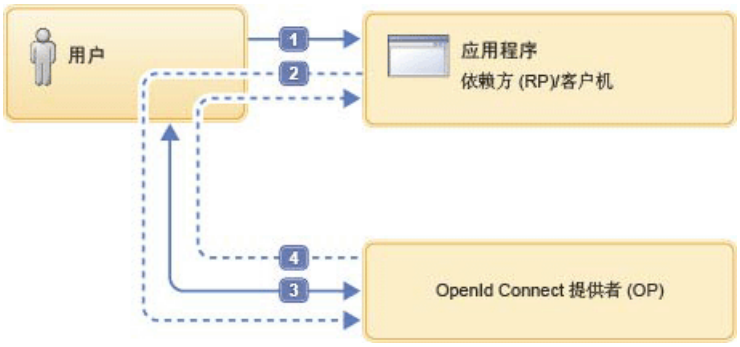


图 33: OpenID Connect 隐式客户机概要文件流程

在 xigemaAS 中配置 OpenID 依赖方

可配置 xigemaAS 服务器以充当 OpenID 依赖方，从而使用 Web 单点登录。

必须至少有一个 OpenID 提供程序 (OP) 是认证用户可信赖的。有多个第三方 OpenID 提供程序可供使用。

可通过启用 xigemaAS 概要文件的 `openid-2.0` 功能部件并输入其他可选配置信息以向 OpenID 提供者认证用户。

1. 将 `openid-2.0` xigemaAS 功能部件添加至 server.xml 文件。

   在 server.xml 文件的 `featureManager` 元素内添加以下元素声明：

   ```
   <feature>openid-2.0</feature>
   ```

2. 使用 `<openId>` 元素指定的 OpenID 依赖方配置选项更新 server.xml 文件。

   可使用 `<openId>` 元素的 providerIdentifier 属性在 server.xml 文件中预定义 OpenID 提供程序 URL，也可使用 `FormLogin` 打包应用程序，这允许用户选择提交要用于认证的 OpenID 提供程序 URL。

   如果已将 providerIdentifier 属性添加至 server.xml 文件，那么 xigemaAS 服务器会自动将用户重定向至该属性指定的 OpenID 提供程序。如果未在 server.xml 文件中定义 providerIdentifier 属性，那么在将用户重定向至 OpenID 提供程序之前，xigemaAS 服务器会先发送登录表单以要求用户选择或确认 OpenID 提供程序。

   以下是用于定义 OpenID 提供程序的样本 OpenID 配置：

   ```
   <openId id="myOpenId" providerIdentifier="https://openid.acme.com/op" userInfoRef="email">
     <userInfo id="email" alias="email" uriType="http://axschema.org/contact/email" count="1"
    required="true" />
   </openId>
   ```

   添加 `openid-2.0` 功能部件会自动实施特定最低配置。因此，不需要在 server.xml 文件中显式指定任何 `<openId>` 元素。如果未指定 `<openId>` 元素，那么以下配置是隐式配置：

   ```
   <openId id="myOpenId" userInfoRef="email">
     <userInfo id="email" alias="email" uriType="http://axschema.org/contact/
   email" count="1" required="true" />
   </openId>
   ```

   缺省情况下，从 OpenID 提供程序处返回的用户电子邮件地址用于身份断言和主体集创建。

3. 配置服务器的信任库以包含受支持的 OpenID 提供程序的签署者证书。有关密钥库的信息，请参阅对 *xigemaAS* 启用 *SSL* 通信。

a. 从 OpenID 提供程序处抽取签署者证书。大部分主流 Web 浏览器支持通过浏览器界面从 Web 站点抽取或导出证书。
b. 将 OpenID 提供程序证书导入至服务器的信任库。有关将证书导入至信任库的一种方法，请参阅 Java™ 安装目录中的 `keytool` 实用程序的 `-import` 标记功能。
c. 使用 `<openId>` 元素的 `sslRef` 属性以指向 SSL 配置。如果未指定 `sslRef` 属性，那么将使用先前提到的密钥库页面中描述的缺省 SSL 配置。SSL 配置应包含对信任库的相应引用，此信任库包含已导入的 OpenID 提供程序证书。

4. 可选：配置认证过滤器。

   如果在 server.xml 文件的 `openId` 元素内配置了 `providerIdentifier` 属性，那么您可配置 `authFilterRef` 以限制应被 `providerIdentifier` 属性定义的 OpenID 提供程序拦截的请求。

   有关配置认证过滤器的更多信息，请参阅认证过滤器（见第 1363 页）。

针对 xigemaAS 使用 LTPA cookie 来定制 SSO 配置

借助单点登录 (SSO) 配置支持，访问 xigemaAS 概要文件资源（例如 HTML、JavaServer Pages (JSP) 文件和 servlet）或访问多个 xigemaAS 概要文件服务器中共享相同轻量级第三方认证 (LTPA) 密钥的资源时，Web 用户可以认证一次。

当用户在其中一个 xigemaAS 概要文件服务器上通过认证时，会将服务器所生成的认证信息通过 Cookie 传输到 Web 浏览器。使用 cookie 来将认证信息传播到其他 xigemaAS 概要文件服务器。

LTPA 已配置好且可以立即使用。用于存储 SSO 令牌的缺省 cookie 名称称为 `ltpaToken2`。如果要对该 cookie 使用另一名称，那么可以使用 `<webAppSecurity>` 元素的 ssoCookieName 属性来定制 cookie 名称。如果定制 cookie 名称，请确保参与 SSO 的所有服务器都使用相同的 cookie 名称。

有关 SSO 的更多信息，请参阅 单点登录 *(SSO)*（见第 1039 页）。

以下示例代码将用户设置为 HTTP 会话到期后注销，并将 SSO cookie 的名称设置为 `myCookieName`：

```
<webAppSecurity  logoutOnHttpSessionExpire="true"  ssoCookieName="myCookieName" />
```

注：为使 SSO 在 xigemaAS 服务器和/或其他服务器间生效，请设置以下资源：

- 服务器必须使用相同 LTPA 密钥并共享相同用户注册表。
- 如果服务器不在相同域中，请使用 `<webAppSecurity>` 元素的 ssoDomainNames 属性来列示这些域。以下示例代码将域名设置为 `domain.com`：

  ```
  <webAppSecurity ssoDomainNames="domain.com" />
  ```

- 如果服务器在相同域中，请将 `<webAppSecurity>` 元素的 ssoUseDomainFromURL 属性设置为 `true`，或在 ssoDomainNames 属性中指定域名。以下示例代码将 ssoUseDomainFromURL 设置为 `true` 以通过请求 URL 获取域名：

  ```
  <webAppSecurity ssoUseDomainFromURL="true" />
  ```

在 xigemaAS 概要文件中配置 RunAs 认证

可以通过为 xigemaAS 概要文件配置 RunAs 规范，将认证委托给另一个身份。

通过将所指定的用户身份和可选密码映射至 RunAs 角色，可以将认证过程委派给具有 RunAs 角色的用户。

您必须启用 xigemaAS 功能部件 appSecurity-2.0 和 servlet-3.0，并且具有用户注册表供应用程序用来配置 RunAs 角色。

有关 RunAs 认证的更多信息，请参阅*RunAs() 认证*（见第 1041 页）。

要配置 RunAs 认证，请完成下列步骤：

1. 在 server.xml 文件中启用 appSecurity-2.0 和 servlet-3.0 xigemaAS 功能部件。
2. 为应用程序配置用户注册表。
3. 在应用程序的部署描述符中指定 <run-as> 元素。

   以下 web.xml 文件示例指定要将后续调用委派给映射至 Employee 角色的用户：

```
<servlet id="Servlet_1">
     <servlet-name>RunAsServlet</servlet-name>
     <display-name>RunAsServlet</display-name>
     <description>RunAsServlet</description>
     <servlet-class>web.RunAsServlet</servlet-class>
     <run-as>
          <role-name>Employee</role-name>
     </run-as>
  </servlet>
```

4. 将您在前一步骤中指定的角色映射至用户。可以在 ibm-application-bnd.xmi/xml 或 server.xml 文件中执行此操作。在 <run-as> 元素中，必须指定用户名。如果使用的是 ibm-application-bnd.xml 文件，那么密码也是必需的；如果使用的是 server.xml 文件，那么密码是可选的。如果需要密码，请在 /bin 目录中使用 securityUtility encode 命令对密码进行编码。有关 securityUtility 命令的更多信息，请参阅 *securityUtility 命令*（见第 1287 页）。

   以下示例使用 server.xml 文件的 <application-bnd> 元素中的 <run-as> 元素，其中，已将 Employee 角色映射至 RunAs 用户 user5：

```
<application-bnd>
     <security-role name="Employee">
          <user name="user1" />
          <user name="user5" />
          <run-as userid="user5" password="{xor}Lz4sLCgwLTs=" />
     </security-role>
</application-bnd>
```

   注：

   - 因为密码在 server.xml 文件中是可选的，所以您也可以对没有密码的用户使用下列代码：

```
<application-bnd>
     <security-role name="Employee">
          <user name="user1" />
          <user name="user5" />
          <run-as userid="user5" />
      </security-role>
 </application-bnd>
```

   - 如果在 server.xml 文件中指定 <application-bnd> 元素，那么您的应用程序不得位于 dropins 文件夹中。如果您让应用程序保留在 dropins 文件夹中，那么必须通过在 server.xml 文件中设置下列项目来禁用应用程序监视：

```
<applicationMonitor dropinsEnabled="false" />
```

RunAs 用户名必须是唯一的，并且在外部帐户中不存在。例如，如果向 SAML 身份提供程序或 OpenID Connect 提供程序认证用户，请确保 RunAs 用户名不在这些外部帐户中。

为 xigemaAS 概要文件配置 TAI

可以配置 xigemaAS 概要文件以使用信任关联拦截器 (TAI) 与第三方安全服务进行集成。可以在单点登录 (SSO) 之前或之后调用 TAI。

请确保您已经安装了第三方安全性服务器作为逆向代理服务器。当 xigemaAS 概要文件服务器将其自己的授权策略应用到产生的凭证（由代理服务器传递）时，第三方安全性服务器可以充当前端认证服务器。您还必须有一个 JAR 文件包含定制 TAI 类，此类可以实现 `com.ibm.wsspi.security.tai.TrustAssociationInterceptor` 接口。

👉 注：不支持监视此 JAR 文件的更改。

使用 TAI 来验证第三方安全性服务器与 xigemaAS 概要文件服务器之间的 HTTP 请求。TAI 会检查来自第三方安全性服务器的 HTTP 请求，以了解它们是否包含任何安全性属性。如果成功完成由 TAI 验证请求的过程，那么 xigemaAS 概要文件服务器会通过检查客户机用户是否具有访问资源所需的许可权来授权请求。

有关定制 TAI 以及使用 LTPA 定制 SSO 配置的更多信息，请参阅为 *xigemaAS* 开发定制 *TAI*（见第 1407 页）和针对 *xigemaAS* 使用 *LTPA cookie* 来定制 *SSO* 配置（见第 1316 页）。

您还可以使用开发者工具来配置 TAI 服务。

1. 在 server.xml 文件中启用 `appSecurity-2.0` xigemaAS 功能部件。

```
<featureManager>
    <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 将应用程序部署到 xigemaAS 概要文件服务器，并启用所有 xigemaAS 功能部件，例如 `jsp-2.2` 和 `jdbc-4.0`。
3. 将 TAI 实现库 simpleTAI.jar 放入服务器目录。
4. 使用 TAI 配置选项及 TAI 实现库的位置来更新 server.xml 文件。

   在以下 server.xml 文件中，启用了定制 TAI，但是对于不受保护的 URI 不执行任何认证，并且在 TAI 认证失败的情况下不允许回退到应用程序认证方法。如示例中所示，下列配置元素可用于 TAI 支持：

   - `trustAssociation`
   - `interceptors`
   - `properties`

```
<trustAssociation id="myTrustAssociation" invokeForUnprotectedURI="false"
                  failOverToAppAuthType="false">
    <interceptors id="simpleTAI" enabled="true"
                  className="com.sample.SimpleTAI"
                  invokeBeforeSSO="true" invokeAfterSSO="false" libraryRef="simpleTAI">
        <properties prop1="value1" prop2="value2"/>

    </interceptors>
</trustAssociation>

<library id="simpleTAI">
    <fileset dir="${server.config.dir}" includes="simpleTAI.jar"/>
</library>
...
```

   👉 注：属性名的开头不能是句点 (.)、config. 或 service。此外，也不接受属性名 id 或 ID。

   👉 注：缺省情况下，`invokeBeforeSSO` 属性设置为 true。使用此设置调用 TAI，即使提供了有效 SSO 令牌也是如此。但是，如果仅当 SSO 令牌无效或未提供时，预期行为为调用 TAI，则可以通

过将其设置为 false 并启用 invokeAfterSSO 属性来禁用此属性。仅当未提供 SSO 令牌或 SSO 令牌无效时，才可使用此设置调用 TAI。在某些情况下，此设置可能会提高系统性能。

配置定制表单登录页面

xigemaAS 允许定义定制表单登录页面以便用户提交认证凭证。

可定制您自己的定制表单登录页面，但必须按 Servlet 3.0 规范中的指定，以基于表单的必需认证格式实现此页面。在所有表单中，针对 form 元素的操作必须为 j_security_check。在支持需要用户名和密码的认证方案的表单中，该操作必须使用 j_username 输入字段以获取用户名，并使用 j_password 输入字段以获取用户密码。定制表单登录页面必须作为未受保护的 Web 资源来提供。可在全局服务器级别设置此登录页面，此页面适用于部署至服务器的所有应用程序。或者，可对各个应用程序指定此登录页面。

👉 注：确保包含在 form-login 页面中的所有文件（如外部样式表或图像）都不受保护。

1. 在需要用户名和密码的表单登录页面中，指定以下表单元素。

```
<FORM action="j_security_check" method="POST">
  User name: <INPUT type="text" name="j_username"><br>
  Password: <INPUT type="password" name="j_password"><br>
  <INPUT type="submit" name="action" value="Login">
</FORM>
```

2. 配置登录表单以供服务器上的应用程序使用。

   有两个可能的配置在部署至服务器的应用程序中使用表单登录页面。可配置定制登录页面以在单个应用程序中使用，或者您可配置该页面作为将用于部署至服务器的所有应用程序的全局登录表单。

   a. 为单个应用程序配置登录表单。

      可配置各个应用程序以将用户引导至特定表单登录页面（通过配置与应用程序一起打包的 web.xml 文件）。

      在 web.xml 文件中指定与应用程序一起打包的登录页面的路径；例如：

```
<login-config>
   <auth-method>FORM/<auth-method>
   <realm-name>MyRealm/<realm-name>
   <form-login-config id="FormLoginConfig_1">
    <form-login-page>/login.jsp/<form-login-page>
    <form-error-page>/loginError.jsp/<form-error-page>
   </form-login-config>
</login-config>
```

   b. 配置全局登录表单。

      可在与每个应用程序一起打包的 web.xml 文件中省略表单登录页面，改为在 server.xml 配置中指定登录表单在部署至服务器的应用程序中全局使用。

      在 server.xml 文件中，添加带有 loginFormURL 属性的 webAppSecurity 元素，该属性是使用登录表单页面的路径值指定的；例如：

```
<webAppSecurity loginFormURL="myGlobalFormLogin/myLogin.jsp" />
```

      确保表单登录页面打包为部署至服务器的 Web 应用程序归档 (WAR) 文件。

如果应用程序的 web.xml 文件中的 `form-login-page` 元素不存在，请使用服务器配置中指定的全局登录页面。

3. 可选：为 *OpenID* 配置定制表单登录页面。
4. 可选： 为 *OAuth* 配置定制表单登录页面。

为 OpenID 配置定制表单登录页面

如果已在 server.xml 文件中配置 `<openId>` 元素的 providerIdentifier 属性，那么可跳过此任务。

与需要用户名和密码的标准表单登录页面不同，OpenID 的表单登录只需要提交 OpenID 标识。因此，必须配置表单登录页面以接受此 OpenID 标识。

1. 配置登录表单以指定 `j_security_check` 作为操作的值。
2. 向登录表单添加输入字段并将 *name* 属性设置为 `openid_identifier` 以供用户输入 OpenID 标识。生成的表单类似以下 HTML 示例：

```
<FORM action="j_security_check" method="POST">
  OpenID: <INPUT type="text" name="openid_identifier"><br>
  <INPUT type="submit" name="action" value="Login">
</FORM>
```

如果仅使用 OpenID 执行认证，那么不需要用户名或密码输入。

为 OAuth 配置定制表单登录页面

可为特定 OAuth 服务提供程序配置定制表单登录页面。

要将定制表单登录页面用于特定 OAuth 服务提供程序，您必须在 server.xml 文件中更新服务提供程序定义。

1. 在提供程序配置中，添加 `customLoginURL` 属性并指定登录页面 URL 作为值。

   下列是提供程序定义中的示例定制登录页面条目：

```
<oauthProvider id="OAuthConfigSample"
  customLoginURL="https://acme.com:9043/oath20/login.jsp">
```

必须配置用于 OAuth 的登录表单以接受用户名和密码。

在 xigemaAS 中配置 SAML Web 浏览器 SSO

可配置 xigemaAS 服务器以充当 SAML Web 浏览器单点登录 (SSO) 服务提供程序。

通过在 xigemaAS 中启用 `samlWeb-2.0` 功能部件及其他配置信息，可将 xigemaAS 服务器配置为 SAML Web SSO 服务提供程序。

1. 通过在 `featureManager` 元素中添加以下元素声明，将 `samlWeb-2.0` xigemaAS 功能部件添加至 server.xml 文件。

```
<feature>samlWeb-2.0</feature>
```

2. xigemaAS 提供缺省 `samlWebSso20` 元素。

```
<samlWebSso20 id="defaultSP">

</samlWebSso20>
```

在此缺省配置中，将采用以下缺省值：

- AssertionConsumerService URL：

```
https://<hostname>:<sslport> /xigemaas/saml20/defaultSP/acs
```

- 服务提供程序 (SP) 元数据 URL：

```
https://<hostname>:<sslport> /xigemaas/saml20/defaultSP/samlmetadata
```

可使用浏览器以通过此 URL 下载此服务提供程序 (SP) 的元数据，并提供 SAML 身份提供程序的 URL 以在此 SP 与身份提供程序 (IdP) 之间建立联合。

- 必须将 IdP 元数据文件复制到服务器上的 resources/security 目录，并将其命名为 idpMetadata.xml。
- SAML 断言的发卡者名称将用作安全域，NameID 将用作主体以通过 SAML 断言创建已认证主体集。
- 如果未指定 KeyStoreRef 属性，那么将使用服务器的缺省密钥库中的专用密钥来签署 SAML AuthnRequest。如果未配置 keyAlias，那么 samlsp 为缺省密钥别名。如果未配置 keyAlias，并且密钥库仅包含一个专用密钥，那么将在签名中使用该专用密钥。

👉 注：如果创建新的服务提供程序实例，并且不再需要 defaultSP，那么必须通过将以下代码添加至 server.xml 文件来显式禁用 defaultSP 实例。

```
<samlWebSso20 id="defaultSP" enabled="false">
</samlWebSso20>
```

👉 注：您必须指定非空 URL 安全字符串作为 samlWebSso20 的标识。如果缺少标识，那么配置元素将被省略，并且不会作为 defaultSP 处理。

👉 注：如果已配置并启用了 SAML，那么所有未认证请求将使用 SAML 认证。要配置应使用 SAML 认证和不应使用 SAML 认证的请求类型，必须按步骤 15 中所述配置认证过滤器。

3. 可选：可将 <samlWebSso20 id="defaultSP"> 添加至 server.xml 文件，并定制 defaultSP 服务提供程序。例如：
   - idpMetadata：添加此参数以更改 IdP 元数据位置和文件名（缺省位置和文件名为 (${server.config.dir}/resources/security/idpMetadata.xml)）。
   - userIdentifier：添加此参数以选择其值将用作主体名称的 SAML 属性名。
   - groupIdentifier：添加此参数以选择其值在主体集中包含为组成员的 SAML 属性名。
   - realmName：使用此参数显式指定用于在此服务提供程序中标识 SAML 主体的域名。缺省域名为 SAML 发卡者名称。
4. 可选：可使用另一标识创建一个或多个新 samlWebSso20 元素。例如，如果使用 mySP 作为标识创建新元素，那么实际将创建具有新 AssertionConsumerService URL 的新 SAML SP 实例：

```
https://<hostname>:<sslport>/xigemaas/saml20/mySP/acs
```

👉 注：您为 samlWebSso20 选择的标识将包含在 SP 的 URL（包括 AssertionConsumerService URL 和元数据 URL）中。samlWebSso20 标识必须为非空，并且不能包含非安全 URL 字符。

5. 可选：可配置信任引擎。xigemaAS SAML SP 支持两种信任引擎类型：
   - 元数据信任引擎：针对所配置 IdP 元数据中提供的信息验证签名。
   - PKIX 信任引擎：通过 PKIX 验证来验证签名中的证书的可信任度。通过此验证的证书被认为可信。

元数据是缺省信任引擎。如果要使用 PKIX 信任引擎，您需要添加 PKIXTrustEngine 元素并定义相应 trustAnchor。

6. 可选：可配置如何通过 SAML 创建已认证主体集。缺省情况下，xigemaAS SP 直接通过 SAML 断言创建主体集而不需要本地用户注册表（相当于配置 mapToUserRegistry=No）。其他配置选项为 mapToUserRegistry=User 或 mapToUserRegistry=Group。
   - mapToUserRegistry=No：SAML 发卡者的名称为域，NameID 用于在主体集中创建主体名称和唯一安全名称，组成员未被包括。可配置 userIdentifier、realmIdentifier、groupIdentifier 和 userUniqueIdentifier 属性以创建带有定制用户名、域名、组成员资格和唯一安全标识的已认证主体集。
   - mapToUserRegistry=User：如果要针对本地用户注册表验证 SAML 用户，并根据本地注册表创建用户主体集，请选择此选项。
   - mapToUserRegistry=Group：如果要针对本地用户注册表验证 SAML 组，并创建主题以包含这些已验证组，请选择此选项。此选项类似 mapToUserRegistry=No，只是将针对本地用户注册表验证组成员资格。
7. 可选：可实现 xigemaAS SAML SPI com.ibm.wsspi.security.saml2.UserCredentialResolver 作为用户功能部件，以将 SAML 断言动态映射至 xigemaAS 主体集。
8. 可选：可定义规则以指示 IdP 如何认证用户（通过在使用 SP 启动的 Web SSO 流程时配置下列其中一个或多个属性：forceAuthn、isPassive、allowCreate、authnContextClassRef 和 authnContextComparisonType）。
9. 可选：可使用 nameIDFormat 属性定义 AuthnRequest 中的必需 NameID 格式。可指定 SAML 规范中定义的任何 NameID 格式，或使用关键字 *customize* 来指定定制 NameId 格式。
10. 可选：可通过创建多个 samlWebSso20 元素（每个 samlWebSso20 引用一个独有 authFilter 元素）来配置多个 SP 和 IdP 联合合作伙伴。所有 authFilters 都必须互斥。对于所配置的多个 samlWebSso20，每个 samlWebSso20 可使用其联合身份提供程序执行单点登录，并具有自己的认证策略和使用规则。
11. 可选：添加对 IdP 启动的非请求式 SSO 的支持。xigemaAS SAML SP 支持要求或不要求 IdP 元数据置于本地的 IdP 启动的非请求式 SSO。如果没有 IdP 元数据，或者您计划使用非请求式 SSO 以与具有一个 xigemaAS SP 的多个身份提供程序联合，那么必须添加以下配置：
    - 配置 <PKIXTrustEngine>，并将所有 IdP 签署者证书导入至 xigemaAS 服务器的缺省信任库，或导入至 PKIXTrustEngine 的 trustAnchor。
    - 配置 trustedIssuers 以列示 IdP 出现在 SAML 断言中时的发卡者名称。发卡者名称在元数据中用作 EntityID。
    - 如果计划仅支持非请求式 SSO，那么可按下一步中所述配置 SP 启动的非请求式 SSO。如果与 SAML 相关联的 SP 中的用户安全上下文变为无效，那么此方案很有用，SP 可将用户重定向回 IdP 以再次自动启动非请求式 SSO。
12. 可选：添加对 SP 启动的非请求式 SSO 的支持。xigemaAS SAML SP 使用所配置 IdP 元数据以执行请求式 SAML AuthnRequest。xigemaAS SP 可将未认证请求重定向至预先配置的登录应用程序而不使用 AuthnRequest。如果业务应用程序执行预认证处理（之后用户才能向 SAML IdP 认证）或者必须对 xigemaAS SP 隐藏 SAML IdP，那么此方案很有用。要配置此方案，应添加 loginPageURL 属性，并将其值设置为可指示用户向 SAML IdP 认证的 URL。
13. 可选：配置签名需求并考虑以下事项：
    - SAML 断言。所有 SAML 断言必须由 SAML IdP 以数字方式签名。如果您想要接受未签名的断言（极少情况），那么您可显式配置 wantAssertionsSigned=false。

- 缺省签名算法为 SHA256。如果必须更改算法，请使用 signatureMethodAlgorithm 属性进行修改。
- 如果不想签署 SAML AuthnRequest，那么可设置 authnRequestsSigned=false。

14. 可选：可配置 SP 认证会话和 cookie。验证并处理 SAML 断言后，xigemaAS SAML SP 在不使用 LTPA cookie 的情况下维护浏览器与 SP 之间的已认证会话。已认证会话超时值设置在 <saml:AuthnStatement> 中设置为 SessionNotOnOrAfter（如果存在），或者按 server.xml 文件中的配置设置为 sessionNotOnOrAfter，缺省值为 120 分钟。将自动生成会话 cookie 名称，您可使用 spCookieName 属性定制 cookie 名称以指定所需名称。

    如果希望 xigemaAS SP 通过 SAML 断言创建 LTPA cookie 并对后续认证请求使用 LTPA cookie，那么可添加配置 disableLtpaCookie=false。如果要将 LTPA cookie 与其他服务器共享，那么必须添加配置属性 allowCustomCacheKey="false"。

    注：如果配置 disableLtpaCookie="false" 和 allowCustomCacheKey="false"，请确保 SAML 用户名未直接向阻止用户拥有两个帐户的本地用户注册表认证。

15. 可选：配置认证过滤器。

    如果启用了 samlWeb-2.0 功能部件，那么任何未认证请求将通过一个 SAML SP 进行认证。如果定义一个定制 samlWebSso20 元素，那么所有认证请求由此 samlWebSso20 SP 实例处理；否则，所有认证由缺省实例 defaultSP 处理。可使用 authnFilter 定义要处理认证请求的 SP 实例。

    有关配置认证过滤器的更多信息，请参阅认证过滤器。

您现在已建立将 xigemaAS 服务器配置为 SAML 服务提供程序（支持使用 SAML 身份提供程序进行单点登录）时所需的配置。

使用 OpenID Connect

xigemaAS 服务器中的 OpenID Connect 是作为 OAuth 2.0 扩展实现的。除提供 OpenID Connect 功能外，OpenID Connect 提供程序还支持所有 OAuth 2.0 功能。

OpenID Connect 端点 URL

了解可用来与 OpenID Connect 提供者通信的 OpenID Connect 端点 URL。

配置 OpenID Connect 后，xigemaAS 上的一些端点 URL 可用，从而使 OpenID Connect 客户机能够在访问受保护资源前与 OpenID Connect 提供程序通信。缺省情况下，所有通信必须通过传输层安全性 (TLS)。

以下端点 URL 可用来与 OpenID Connect 提供程序通信：

- 授权端点 *URL*（见第 1323 页）
- 令牌端点 *URL*（见第 1324 页）
- 内省端点 *URL*（见第 1324 页）
- *UserInfo* 端点 *URL*（见第 1324 页）
- 发现端点 *URL*（见第 1325 页）
- 覆盖映射端点 *URL*（见第 1325 页）
- 注册端点 *URL*（见第 1325 页）

授权端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/authorize
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅对 *OpenID Connect* 调用授权端点。

令牌端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/token
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅对 *OpenID Connect* 调用令牌端点。

内省端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/introspect
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅对 *OpenID Connect* 调用内省端点。

UserInfo 端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/userinfo
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅对 *OpenID Connect* 调用 *UserInfo* 端点。

发现端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/.well-known/
openid-configuration
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅配置 *OpenID Connect* 提供程序以接受发现请求。

覆盖映射端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/coverage_map
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅调用覆盖映射服务。

注册端点 URL

```
https://<host_name>:<port_number>/oidc/endpoint/<provider_name>/registration
```

其中

host_name

OpenID Connect 提供程序的主机名。

port_number

xigemaAS 服务器上配置的安全端口号。

provider_name

OpenID Connect 提供程序名称。

有关更多信息，请参阅配置 *OpenID Connect* 提供程序以接受客户机注册请求。

在 xigemaAS 中配置 OpenID Connect 提供者

可配置 xigemaAS 服务器以充当 OpenID Connect 提供者或授权服务器以使用 Web 单点登录。

通过启用 xigemaAS 概要文件的 `openidConnectServer-1.0` 功能部件并配置其他配置信息，可配置 xigemaAS 概要文件服务器以充当 OpenID Connect 提供者。

1. 将 `openidConnectServer-1.0` xigemaAS 功能部件及任何其他所需功能部件添加至 server.xml 文件。对于 `openidConnectServer-1.0` 功能部件，还需要 `ssl-1.0` 功能部件。

```
<feature>openidConnectServer-1.0</feature>
<feature>ssl-1.0</feature>
```

2. 定义 OAuth 服务提供程序。OpenID Connect 基于 OAuth 2.0 协议，您必须配置有效 OAuth 服务提供程序。OAuth 服务提供程序的配置包含相应 `oauth-roles`、`oauthProvider` 和用户注册表元素。被授权使用 OpenID Connect 的任何用户还必须映射至 `authenticated oauth-role`。有关更多信息，请参阅定义 *OAuth* 服务提供程序。

   系统将针对 OpenID Connect 更新 OAuth 元数据，并且大部分添加在客户机元数据中进行。如果对客户机注册使用 `databaseStore` 方式，请参阅配置 *OpenID Connect* 提供程序以接受客户机注册请求（见第 1342 页）以了解更多信息。建议您遵循该文档中的指示来管理客户机。如果对客户机注册使用 `localStore` 方式，那么可注册 `scope`、`preAuthorizedScope`、`grantTypes`、`responseTypes`、`introspectTokens`、和 `functionalUserId` 及其他属性。

3. 添加 `openidConnectProvider` 元素，其 `oauthProviderRef` 属性引用所配置 `oauthProvider`。每个 `oauthProvider` 只能被一个 `openidConnectProvider` 引用，两个或更多 `openidConnectProvider` 元素不能引用同一 `oauthProvider`。客户机元素的 `name` 属性和 `secret` 属性必须与对应 OpenID Connect 客户机的 `client ID` 和 `client secret` 相匹配。此示例使用缺省xigemaAS 服务器 OpenID Connect 客户机。

   注：在此示例中，OP 期望客户机的 SSL 端口设置为 443。

```
<openidConnectProvider id="OidcConfigSample"
 oauthProviderRef="OAuthConfigSample" />

<oauthProvider id="OAuthConfigSample">
<localStore>
<client name="client01" secret="{xor}LDo8LTor"
displayname="client01"
scope="openid profile email"
redirect="https://server.example.com:443/oidcclient/redirect/client01"/>
</localStore>
```

```
</oauthProvider>
```

👉 注：有效客户机必须针对 authorization_code 授权类型注册其名称、重定向、范围和密钥。

4. 配置服务器的信任库以包含受支持的 OpenID Connect 依赖方或客户机的签署者证书。有关密钥库的信息，请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）
5. 修改服务器的 SSL 配置以使用所配置信任库。

```
<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings" keyStoreRef="myKeyStore"
 trustStoreRef="myTrustStore" />
<keyStore id="myKeyStore" password="{xor}Lz4sLCgwLTs=" type="jks"
 location="${server.config.dir}/resources/security/BasicKeyStore.jks" />
<keyStore id="myTrustStore" password="{xor}Lz4sLCgwLTs=" type="jks"
 location="${server.config.dir}/resources/security/BasicTrustStore.jks" />
```

OpenID Connect 配置为使用服务器指定的缺省 SSL 配置。因此，服务器的缺省 SSL 配置必须使用为 OpenID Connect 配置的信任库。

OpenID Connect 中的用户同意表单是可插拔的，这允许提供程序创建和维护他们自己的同意表单。因为此表单是通过 SSL 检索到的，所以您必须配置信任库以包含同意表单所在服务器的签署者证书。如果已使用缺省同意表单并且用于 OpenID Connect 的信任库配置为不同于 xigemaAS 服务器使用的密钥库，那么必须将 xigemaAS 服务器的签署者证书导入至 OpenID Connect 信任库。

👉 注：为使用 OpenID Connect，scope 属性必须在范围列表中添加 openid。

您现在已完成将 xigemaAS 服务器配置为 OpenID Connect 提供程序（它能够与配置为 OpenID Connect 客户机的其他 xigemaAS 服务器通信）时所需的最低配置。

将 OpenID Connect 提供程序用作 OAuth 2.0 授权服务器

OpenID Connect 提供程序可用作常规 OAuth 2.0 授权服务器以发出 OAuth 2.0 access_token 并支持所有 OAuth 2.0 授权类型。

OpenID Connect 提供程序支持 JSON Web 令牌 (JWT) 不记名令牌作为用于请求 OAuth 2.0 访问令牌的授权，请参阅用于 *OAuth* 客户机授权的 *JSON Web* 令牌 *(JWT)* 和配置 *OpenID Connect* 提供程序以接受 *JSON Web* 令牌 *(JWT)* 进行授权。

如果发出带有 authorization code grant 或 implict grant type 的授权请求，并且未包含或未批准 openid scope，那么该请求作为常规 OAuth 授权请求处理。不会发出 id_token，可发出 access_token 和 refresh_token。

OpenID Connect 提供程序可支持带有 Resource Owner Password Credentials Grant 或 Client Credentials Grant 的 OAuth 授权流程，请参阅配置 *OpenID Connect* 提供程序以启用两脚 *OAuth* 请求。

用于 OAuth 客户机授权的 JSON Web 令牌 (JWT)

用于 OAuth 客户机授权的 JWT 允许客户机在交换中将已签名 JWT 令牌发送至 OpenID Connect 提供程序以获取 OAuth 2.0 访问令牌。

用于 OAuth 客户机授权的 JWT 包含在 openidConnectServer-1.0 功能部件中。它允许客户机在交换中将已签名 JWT 令牌发送至 OpenID Connect 提供程序以获取 OAuth 2.0 访问令牌。

以下是此功能的可能示例使用方案：电子公司的客户授权每月自动从网上银行自动付款。假定电子公司和网上银行已就满足这类请求方面建立信任关系。电子公司可将带有相应声明的已签名 JWT 令牌发送至 OpenID

Connect 提供程序的令牌端点 URI，这些声明是针对网上银行配置的，用于每月请求 OAuth 2.0 访问令牌。然后电子公司可使用该访问令牌每月通过网上银行付款。

配置为 OpenID Connect 提供程序的 xigemaAS 服务器支持 用于 *OAuth 2.0* 客户机认证和授权的 *JSON Web* 令牌 *(JWT)* 概要文件 规范的某些部分。想要支持 JWT 客户机功能的用户必须使用他们自己的应用程序完成此操作。

- 授权范围
- *JSON Web* 令牌中的声明
- 提交 *JSON Web* 令牌请求

授权范围

OpenID Connect 客户机将带有 JWT 的 HTTPS 请求发送至 OpenID Connect 提供程序的令牌端点以请求访问令牌。在此过程中，用户不会见到任何有关针对使用范围授权的同意表单。JWT 处理程序将根据以下条件处理授权范围：

1. 如果未在请求中指定任何范围参数，那么 OpenID Connect 提供程序不会在访问令牌中指定任何范围。
2. 如果 OpenID Connect 客户机在 OpenID Connect 提供程序配置中被限定为 autoAuthorized 客户机，那么该客户机在请求中指定的任何范围是在访问令牌的范围列表中指定的。
3. 如果 OpenID Connect 客户机未被限定为 autoAuthorized 客户机，那么请求中包含的范围需要被客户机配置中的范围列表过滤，并且也必须在 `preAuthorizedScope` 列表中指定。如果 HTTPS 请求中的范围在客户机配置的 `scope` 和 `preAuthorizeScope` 列表中，那么可在访问令牌的范围列表中指定该范围。

如果客户机未被限定为 autoAuthorized 客户机，那么必须在客户机配置中正确配置可包含在访问令牌的范围列表中的范围。该范围必须包含在 OpenID Connect 提供程序的客户机配置的 `scope` 和 `preAuthorizedScope` 属性值中。在所显示的示例中，范围 `profile` 和 `email` 是在访问令牌的范围列表中指定的，因为这两个范围都包含在 `scope` 和 `preAuthorizedScope` 值列表中。如果某个范围未列示在客户机配置的 `scope` 属性中，那么系统会在访问令牌的范围列表中省略该范围。如果某个范围列示在 `scope` 属性中但未包含在客户机配置的 `preAuthorizedScope` 列表中，那么授权请求会在 OpenID Connect 提供程序返回的响应中触发 `invalid_grant` 错误。

```
<openidConnectProvider id="OidcConfigSample"
 oauthProviderRef="OAuthConfigSample" />
<oauthProvider id="OAuthConfigSample" ... >
        ...
        <localStore>
            <client name="client01" secret="{xor}..."
                    displayname="client01"
                    scope="profile email phone"
                    preAuthorizedScope="profile email"
                    enabled="true" />
            ...
        </localStore>
    </oauthProvider>
```

JSON Web 令牌中的声明

必须签署有效 JSON Web 令牌。 配置为 OpenID Connect 提供程序的 xigemaAS 服务器仅支持 HMAC-SHA256 作为令牌签名算法。每个 OpenID Connect 客户机的签名密钥是 OpenID Connect 提供程序的客户机配置中的 secret 属性。在所显示的示例中，所使用的签名密钥将为 "{xor}LDo8LTor"。

```
<client name="client01" displayname="client01" secret="{xor}LDo8LTor" ... />
```

OpenID Connect 提供程序也会验证 JWT 中的以下声明：

'iss' (issuer)

此声明在 JWT 中是必需的。iss 声明必须与 OpenID Connect 提供者中的客户机配置的 name 属性或 redirect 属性相匹配。在以下示例中，iss 声明必须与 client01 或 http://op201406.vsettan.com:8010/oauthclient/redirect.jsp 匹配。

```
<client name="client01" redirect="http://op201406.vsettan.com:8010/oauthcl
ient/redirect.jsp" scope="openid profile email" ... />
```

'sub' (subject)

此声明在 JWT 中是必需的。主体集的值必须是 OpenID Connect 提供程序服务器的用户注册表中的有效用户名。

'aud' (audience)

此声明在 JWT 中是必需的。如果在 openidConnectProvider 配置中指定了 issuerIdentifier 属性，那么受众声明的值为 issuerIdentifier 的名称。如果未在 openidConnectProvider 配置中指定 issuerIdentifier 属性，那么受众必须是 OpenID Connect 提供程序的令牌端点 URI。在以下示例中，受众声明的值将为 "OpenIDConnectProviderID1"。

```
<openidConnectProvider id="OidcConfigSample" oauthProviderRef="OAuthConfig
Sample" issuerIdentifier="OpenIDConnectProviderID1" />
```

'exp' (expiration)

此声明在 JWT 中是必需的，并且限制可使用 JWT 的时间范围。OpenID Connect 提供程序针对其系统时钟验证 exp，附加允许的一些时钟偏差。

'nbf' (not before)

这是可选声明。如果此声明存在，那么此令牌仅在此声明指定的时间后有效。OpenID Connect 提供程序针对其系统时钟验证此时间，附加允许的一些时钟偏差。

'iat' (issued at)

缺省情况下，这是可选声明。但是，如果 jwtGrantType 元素的 iatRequired 属性设置为 true，那么所有 JWT 必须包含 iat 声明。如果 iat 声明存在，那么该声明指示发出 JWT 的时间。发出 JWT 的时长不能超过 maxTokenLifetime。

'jti' (JWT ID)

这是可选声明，它是 JWT 令牌的唯一标识。如果 JWT 标识存在，那么发出者不能复用同一 JWT 标识。例如，如果 client01 发出 jti 为 id6098364921 的 JWT，那么 client01 发出的所有其他 JWT 的 jti 值都不能为 id6098364921。jti 声明与另一 JWT 完全相同的 JWT 被视为重放攻击。配置为 OpenID Connect 提供程序的 xigemaAS 服务器在服务器上设置 jti 高速缓存。高速缓存的大小由 jwtGrant

Type 配置中的 maxJtiCacheSize 指定。系统会针对任何新进入的 jti 标识检查保留在高速缓存中的 jti 标识。存储在高速缓存中的 jti 标识不会被废弃，除非高速缓存已满。

提交 JSON Web 令牌请求

最好使用 HTTPS 协议而不是 HTTP 来提交 JWT 请求。OpenID Connect 提供程序的令牌端点用于处理 HTTPS JWT 请求。要确定 OpenID Connect 提供程序的令牌端点，请参阅调用 *OpenID Connect* 的令牌端点或 *OAuth* 端点 *URL*。

该请求必须包含以下参数：

- grant_type - 此参数的值必须为 "urn:ietf:params:oauth:grant-type:jwt-bearer"
- assertion - 此参数的值必须包含单个已签名 JWT 令牌。
- scope - 此参数是可选的。如果省略了 scope，那么返回的访问令牌不会包含任何范围。系统会针对 OpenID Connect 提供程序配置检查 scope 参数中列示的范围值。有关更多信息，请参阅先前的“授权范围”部分。
- client_id - 此参数的值必须与 OpenID Connect 提供程序的客户机配置中的 name 属性匹配。
- client_secret - 此参数的值必须与 OpenID Connect 提供程序的客户机配置中的 secret 属性匹配。

示例 HTTPS 请求：

```
POST /token.oauth2 HTTP/1.1
    Host: oidc.vsettan.com.cn
    Content-Type: application/x-www-form-utlencoded

    client_id=client01
    &client_secret=secret
    &grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
    &assertion=eyJhbGc[---omitted---]kIn0.eyJpc[---
ommitted---]A4fQ.MB6ZFlCsHg5MJ-weIHZYz6xgF1jdSZn7ErchHs8-8Rk
    &scope=profile email
```

用于创建已签名 JWT 令牌的 Java 示例：

```
package com.ibm.sample;

import java.security.SignatureException;
import com.google.gson.JsonObject;
import net.oauth.jsontoken.crypto.HmacSHA256Signer;

import net.oauth.jsontoken.SystemClock;
import net.oauth.jsontoken.JsonToken;
import org.joda.time.Duration;
import org.joda.time.Instant;

public class SampleJWTToken {
        private static final Duration SKEW = Duration.standardMinutes(5);

        JsonToken jwtToken = null;
        String[] allPayloadKeys = { "iss", "sub", "aud", "exp", // required
                                    "nbf", "iat", "jti" }; // optional

        public SampleJWTToken(String clientId,
                              String keyId,
                              String signKey,
```

```
                                String audience,
                                String subject, // user
                                String jtiId) throws Exception { //
 InvalidKeyException

                byte[] hs256Key = signKey.getBytes();
                HmacSHA256Signer hmacSha256Signer = new HmacSHA256Signer(
                                clientId, keyId, hs256Key);
                // _rsaSha256Signer = new RsaSHA256Signer(clientId, _keyId,
                //                                        _privateKey);
                SystemClock clock = new SystemClock(SKEW);
                jwtToken = new JsonToken(hmacSha256Signer, clock);
                JsonObject headerObj = jwtToken.getHeader();
                JsonObject payloadObj = jwtToken.getPayloadAsJsonObject();

                headerObj.addProperty("alg", "HS256");

                Instant instantExp = clock.now().plus(600000); // 10 minutes
                jwtToken.setExpiration(instantExp);
                jwtToken.setAudience(audience);
                payloadObj.addProperty("iss", clientId);
                payloadObj.addProperty("sub", subject);

                // optional
                payloadObj.addProperty("jti", jtiId);
                jwtToken.setIssuedAt(clock.now()); // issued at time
        }

        public String getJWTTokenString() throws Exception {
                String signedAndSerializedString = null;
                try {
                        signedAndSerializedString = jwtToken.serializeAndSign();
                } catch (SignatureException e) {
                        throw e;
                }
                return signedAndSerializedString;
        }
}
```

配置 OpenID Connect 提供程序以接受发现请求

发现配置端点提供有关 OpenID Connect 提供程序 (OP) 服务器支持的功能的信息。

此服务返回的元数据基于 *OIDC Discovery 1.0* 规范提供程序元数据并对其进行扩展。如果未指定任何设置，那么此服务返回一组缺省配置。否则，请参阅属性列表以了解其用途及可能的可配置选项。

1. 可在发现配置服务中覆盖所选属性的缺省值。此操作是通过在 server.xml 文件中指定值来执行的。请参阅以下属性表以查看可配置属性及可能的配置选项。

表 27: 发现请求参数

属性名称	数据类型	必需/可选	描述
responseTypesSupported	输入	可选	OpenID Connect 提供程序 (OP) 服务器支持的响应类型。除非指定，否则缺省值为 code、token

属性名称	数据类型	必需/可选	描述
			和 id_token token。可指定多个值。这些值为字符串。例如，可能的值为： ◦ code ◦ token ◦ id_token token
subjectTypesSupported	仅输出	不适用	OP 服务器支持的主体集类型。此值设置为 public。此值为字符串。
idTokenSigningAlgValuesSupported	仅输出	可选	OP 服务器支持的标识令牌签名算法。在 `openidConnectProvider` 服务器配置中，此值被指定为服务器属性 `signatureAlgorithm`。除非指定，否则缺省值为 HS256。只能指定一个值。它是一个字符串。例如，在 `openidConnectProvider` 配置中，属性 `signatureAlgorithm` 的可能值为： ◦ none ◦ RS256 ◦ HS256
scopesSupported	输入	可选	OP 服务器支持的范围值。除非指定，否则缺省值为 openid、general、profile、email、address 和 phone。可指定多个值。这些值为字符串。例如，可能的值为： ◦ openid ◦ general ◦ profile ◦ email ◦ address ◦ phone
claimsSupported	输入	可选	OP 服务器支持的声明值。除非指定，否则缺省值为

属性名称	数据类型	必需/可选	描述
			sub、groupIds、name、preferred_username、picture、locale、email 和 profile。可指定多个值。这些值为字符串。例如，可能的值为： ◦ sub ◦ groupIds ◦ name ◦ preferred_username ◦ picture ◦ locale ◦ email ◦ profile
responseModesSupported	输入	可选	OP 服务器支持的响应方式。除非指定，否则缺省值为 query 和 fragment。可指定多个值。这些值为字符串。 ◦ query ◦ fragment
grantTypesSupported	输入	可选	OP 服务器支持的授权类型。除非指定，否则缺省值为 authorization_code、implicit、refresh_token、client_credentials、password 和 urn:ietf:params:oauth:grant-type:jwtbearer。可指定多个值。这些值为字符串。例如，可能的值为： ◦ authorization_code ◦ implicit ◦ refresh_token ◦ client_credentials ◦ password ◦ urn:ietf:params:oauth:grant-type:jwtbearer
tokenEndpointAuthMethodsSupported	输入	可选	OP 服务器支持的令牌端点授权方法。除非指定，否则缺省值为 client_secret_post 和 client_secret

属性名称	数据类型	必需/可选	描述
			_basic。可指定多个值。这些值为字符串。例如，可能的值为： ◦ none ◦ client_secret_post ◦ client_secret_basic
displayValuesSupported	仅输出	不适用	OP 服务器支持的显示值。此值设置为 page。此值为字符串。
claimTypesSupported	仅输出	不适用	OP 服务器支持的声明类型值。此值设置为 normal。此值为字符串。
claimsParameterSupported	输入	可选	指示 OP 服务器是否支持声明参数。除非指定，否则缺省值为 false。只能指定一个值。它是一个布尔值。例如，可能的值为： ◦ true ◦ false
requestParameterSupported	输入	可选	指示 OP 服务器是否支持请求参数。除非指定，否则缺省值为 false。只能指定一个值。它是一个布尔值。例如，可能的值为： ◦ true ◦ false
requestUriParameterSupported	输入	可选	指示 OP 服务器是否支持请求 URI 参数。除非指定，否则缺省值为 false。只能指定一个值。它是一个布尔值。例如，可能的值为： ◦ true ◦ false
requireRequestUriRegistration	输入	可选	指示 OP 服务器是否支持需要请求 URI 注册。除非指定，否则缺省值为

属性名称	数据类型	必需/可选	描述
			false。只能指定一个值。它是一个布尔值。例如，可能的值为： ◦ true ◦ false

发现配置的示例

以下示例假定在端口 443 上对 xigemaAS OP 配置了 SSL。

```
https://server.example.com:443/oidc/endpoint/<provider_name>/
```

可从以下位置访问发现配置端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/.well-known/openid-
configuration
```

例如，在 server.xml 文件中，用户可按以下方式定制其 OpenID Connect 发现配置属性：

```
<openidConnectProvider id="OidcConfigSample"
 oauthProviderRef="OAuthConfigSample">
 <discovery
  responseTypesSupported="token, id_token token"
  subjectTypesSupported="public"
  scopesSupported="openid, general, profile"
  claimsSupported="sub, groupIds, name"
  responseModesSupported="query"
  grantTypesSupported="implicit"
  tokenEndpointAuthMethodsSupported="client_secret_basic"
  displayValuesSupported="page"
  claimTypesSupported="normal"
  claimsParameterSupported="true"
  requestParameterSupported="true"
  requestUriParameterSupported="true"
  requireRequestUriRegistration="true"
 />
</openidConnectProvider>
<oauthProvider id="OAuthConfigSample">
</oauthProvider>
```

定制发现配置的示例

```
Request Headers:
GET https://server.example.com:443/oidc/endpoint/<provider_name>/.well-kno
wn/openid-configuration


Response Headers:
Status: 200
Content-Type: application/json
Cache-Control:public, max-age=3600

Response Body:
{
```

```
   "introspection_endpoint":"https://server.example.com:443/oidc/endpoint/
<provider_name>/introspect",
   "coverage_map_endpoint":"https://server.example.com:443/oidc/endpoint/<
provider_name>/coverage_map",
   "issuer":"https://server.example.com:443/oidc/endpoint/<provider_name>"
,
   "authorization_endpoint":"https://server.example.com:443/oidc/endpoint/
<provider_name>/authorize",
   "token_endpoint":"https://server.example.com:443/oidc/endpoint/<provide
r_name>/token",
   "response_types_supported":[
      "token",
      "id_token token"
   ],
   "subject_types_supported":[
      "public"
   ],
   "userinfo_endpoint":"https://server.example.com:443/oidc/endpoint/<prov
ider_name>/userinfo",
   "registration_endpoint":"https://server.example.com:443/oidc/endpoint/<
provider_name>/registration",
   "scopes_supported":[
      "openid",
      "general",
      "profile"
   ],
   "claims_supported":[
      "sub",
      "groupIds",
      "name"
   ],
   "response_modes_supported":[
      "query"
   ],
   "grant_types_supported":[
      "implicit"
   ],
   "token_endpoint_auth_methods_supported":[
      "client_secret_basic"
   ],
   "display_values_supported":[
      "page"
   ],
   "claim_types_supported":[
      "normal"
   ],
   "claims_parameter_supported":true,
   "request_parameter_supported":true,
   "request_uri_parameter_supported":true,
   "require_request_uri_registration":true,
   "check_session_iframe":"https://server.example.com:443/oidc/endpoint/<p
rovider_name>/check_session_iframe",
   "end_session_endpoint":"https://server.example.com:443/oidc/endpoint/<p
rovider_name>/end_session"
}
```

缺省发现配置的示例

```
Request Headers:
```

```
GET https://server.example.com:443/oidc/endpoint/<provider_name>/.well-kno
wn/openid-configuration


Response Headers:
Status: 200
Content-Type: application/json
Cache-Control:public, max-age=3600

Response Body:
{
   "introspection_endpoint":"https://server.example.com:443/oidc/endpoint/
<provider_name>/introspect",
   "coverage_map_endpoint":"https://server.example.com:443/oidc/endpoint/<
provider_name>/coverage_map",
   "issuer":"https://server.example.com:443/oidc/endpoint/<provider_name>"
,
   "authorization_endpoint":"https://server.example.com:443/oidc/endpoint/
<provider_name>/authorize",
   "token_endpoint":"https://server.example.com:443/oidc/endpoint/<provide
r_name>/token",
   "response_types_supported":[
      "code",
      "token",
      "id_token token"
   ],
   "subject_types_supported":[
      "public"
   ],
   "userinfo_endpoint":"https://server.example.com:443/oidc/endpoint/<prov
ider_name>/userinfo",
   "registration_endpoint":"https://server.example.com:443/oidc/endpoint/<
provider_name>/registration",
   "scopes_supported":[
      "openid",
      "general",
      "profile",
      "email",
      "address",
      "phone"
   ],
   "claims_supported":[
      "sub",
      "groupIds",
      "name",
      "preferred_username",
      "picture",
      "locale",
      "email",
      "profile"
   ],
   "response_modes_supported":[
      "query",
      "fragment"
   ],
   "grant_types_supported":[
      "authorization_code",
      "implicit",
      "refresh_token",
      "client_credentials",
```

```
      "password",
      "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ],
   "token_endpoint_auth_methods_supported":[
      "client_secret_post",
      "client_secret_basic"
   ],
   "display_values_supported":[
      "page"
   ],
   "claim_types_supported":[
      "normal"
   ],
   "claims_parameter_supported":false,
   "request_parameter_supported":false,
   "request_uri_parameter_supported":false,
   "require_request_uri_registration":false,
   "check_session_iframe":"https://server.example.com:443/oidc/endpoint/<p
rovider_name>/check_session_iframe",
   "end_session_endpoint":"https://server.example.com:443/oidc/endpoint/<p
rovider_name>/end_session"
}
```


配置 UserInfo 端点返回的声明

可配置 xigemaAS OpenID Connect 提供者以定制 UserInfo 端点返回的声明。

可使用 server.xml 文件中 `openidConnectProvider` 元素的 `scopeToClaimMap` 和 `claimToUserRegistryMap` 子元素配置从 xigemaAS 概要文件服务器 OpenID Connect 提供者返回的声明。

OpenID Connect UserInfo 端点接受访问令牌作为输入，并返回有关为其创建该访问令牌的用户的一组声明。返回的声明由以下项确定：

1. 访问令牌中的范围

   一个访问令牌可有多个范围。访问令牌中的范围是创建该访问令牌的授权端点调用中提供的范围。
2. 与范围相关联的声明

   每个范围可有多个相关联的声明。
3. 与声明相关联的联合存储库属性

   一个声明只能有一个相关联的联合存储库属性。
4. 与联合存储库属性相关联的用户注册表属性

   一个联合存储库属性只能有一个相关联的用户注册表属性。

👉 注：支持检索 UserInfo 声明的唯一用户注册表类型为 LDAP。

xigemaAS 定义缺省范围、声明、联合注册表属性和缺省映射。

表 28: 范围、声明和联合注册表属性的缺省映射

范围	声明	联合注册表属性
profile	name、given_name 和 picture	displayName、givenName 和 photo URL
email	email	mail
address	address	postalAddress
phone	phone_number	telephoneNumber

下列每个步骤都是可选的。xigemaAS 服务器定义缺省范围、声明、联合注册表属性和缺省映射。只有您想要更改缺省映射或定义定制范围或声明时，才需要执行下列任何步骤。

1. 配置与范围相关联的声明。一个范围可映射至多个声明，且多个声明之间必须以逗号分隔。

   在以下示例中，范围 CUSTOM_SCOPE1 与两个声明（CUSTOM_CLAIM1 和 language）相关联，范围 CUSTOM_SCOPE2 与一个声明 (CUSTOM_CLAIM2) 相关联。

   ```
   <scopeToClaimMap CUSTOM_SCOPE1="CUSTOM_CLAIM1, language"
                    CUSTOM_SCOPE2="CUSTOM_CLAIM2" />
   ```

   注: 声明和范围名称是区分大小写的，CUSTOM_SCOPE1 与 custom_scope1 是不同范围。

   a. 要定义拼写相同但大小写不同的范围，必须使用 property 子元素。在以下示例中，定义了范围 CUSTOM_SCOPE1 和 custom_scope1。

      ```
      <scopeToClaimMap CUSTOM_SCOPE1="CUSTOM_CLAIM1, language" >
          <property name="custom_scope1" value="custom_claim1,mobile"/>
      </scopeToClaimMap>
      ```

2. 配置与声明相关联的联合存储库属性。一个声明只能映射至一个联合存储库属性。

   在以下示例中，声明 CUSTOM_CLAIM1 与联合存储库属性 departmentNumber 相关联。声明 language 与联合存储库属性 preferredLanguage 相关联，声明 CUSTOM_CLAIM2 与联合存储库属性 mail 相关联。

   ```
   <claimToUserRegistryMap CUSTOM_CLAIM1="departmentNumber"
                           language="preferredLanguage"
                           CUSTOM_CLAIM2="mail" />
   ```

   a. 要定义拼写相同但大小写不同的声明，必须使用 property 子元素。在以下示例中，定义了声明 CUSTOM_CLAIM1 和 custom_claim1。

      ```
      <claimToUserRegistryMap CUSTOM_CLAIM1="departmentNumber" >
          <property name="custom_claim1" value="employeeType" />
      </claimToUserRegistryMap>
      ```

3. 配置与联合存储库属性相关联的用户注册表属性。

   在以下示例中，联合存储库属性 photoURL 与 LDAP 注册表属性 ldapPersonPicture 相关联

   ```
   <ldapRegistry...>
     ...
   ```

```
    <attributeConfiguration>
        <attribute name="ldapPersonPicture"
                   propertyName="photoURL"
                   entityType="PersonAccount" />
    </attributeConfiguration>
   ...
</ldapRegistry>
```

👉 注：必须在 LDAP 注册表的模式中定义该 LDAP 属性。

您现在已完成在定制 UserInfo 端点返回的声明时需要的配置。

配置 OpenID Connect 提供程序以启用两脚 OAuth 请求

典型 OAuth 流程由客户机与授权服务器之间的交互的三只“脚”或三个阶段构成。在两脚 OAuth 方案中，客户机使用预授权范围以便不需要与用户交互（不需要在典型流程中执行其中一只脚）。具体地说，用户不必向授权服务器认证或同意共享所请求范围指定的信息。反而全部所请求范围参数被视为预授权的，并自动添加至请求令牌，然后请求令牌被发送至授权服务器。

此任务要求您具有正确配置为 OpenID Connect 提供程序的 xigemaAS 服务器。

在具有两只脚或两个阶段的方案中，Open ID Connect 客户机可发送 grant type 为 `client_credential` 或 `resource owner password` 的两脚 HTTP 请求。这些请求不会通过授权端点，所以不存在用户确认并批准所请求范围的范围同意表单；但是，OpenID Connect 提供程序仍需要处理其 `access_token` 内容中的授权范围。

配置为 OpenID Connect 提供程序（配备为处理两脚 OAuth 请求）的 xigemaAS 服务器通过使用以下条件来批准预授权范围：

1. 如果请求的 grant_type 参数值为 `client_credential` 或 `resource owner password` 并且该请求为 OAuth 2.0 请求，那么在该请求中定义的所有范围被批准并复制到访问令牌的内容中。这是 OAuth 2.0 功能部件的现有行为。
2. 如果该请求为 OpenID Connect 请求、JWT 令牌 OAuth 请求或 OpenID Connect 请求，那么使用以下条件：
   - 如果未在请求中指定任何范围参数，那么 OpenID Connect 提供程序不会接受该请求。
   - 所请求范围必须出现在客户机配置的 `scope` 属性定义的范围列表中，并且必须同时在客户机配置的 `preAuthorizedScope` 列表中指定。

此任务演示如何配置将充当 OpenID Connect 提供程序以启用两脚 OAuth 请求的 xigemaAS 服务器。

1. 要对客户机指定预授权范围列表，请将必需范围添加至 server.xml 文件的相应 `<oauthProvider>` 元素内的客户机配置的 `scope` 和 `preAuthorizedScope` 属性。在所显示的示例中，范围 `profile` 和 `email` 被限定为 OpenID Connect 提供程序返回的访问令牌的范围列表中指定。`phone` 范围不会被视为预授权范围，因为它不在 `preAuthorizedScope` 列表中。

```
<oauthProvider id="OAuthConfigSample" ...>
 ....
          <localStore>
            <client name="client01" secret="{xor}..."
              displayname="client01"
              scope="profile email phone"
              preAuthorizedScope="profile email"
              enabled="true" />
            ....
          </localStore>
```

```
</oauthProvider>
```

注：如果所请求范围未列示在客户机配置的 scope 属性中，那么系统会在所返回访问令牌的范围列表中省略该范围。如果所请求范围列示在客户机配置的 scope 属性中但未包含在客户机配置的 preAuthorizedScope 列表中，那么它会在 OpenID Connect 提供程序返回的响应中触发 invalid_grant 错误。

配置 OpenID Connect 提供程序以使用 RSA-SHA256 算法签署标识令牌

可配置 OpenID Connect 提供程序以使用 RS256 算法签署标识令牌。

通过将 signatureAlgorithm 设置为 RS256 并配置含有签名专用密钥的密钥库，可配置 OpenID Connect 提供程序以使用 RSA-SHA256 签名算法签署标识令牌。

1. 在 server.xml 文件中，创建引用物理密钥库的 keystore 元素，该物理密钥库包含能够执行 RSA-SHA256 签名算法的专用密钥。例如：

```
<keyStore id="opTestKeyStore" location="${server.config.dir}/
opKeyStore.jks" type="JKS" password="keystorePwd" />
```

2. 将 OpenID Connect 提供程序 signatureAlgorithm 属性设置为 RS256，将 keyStoreRef 属性设置为步骤 1 中使用的 keystore 元素的 id 值，并设置 keyAliasName 以在密钥库中查找专用密钥。如果步骤 1 中使用的 keystore 元素标识为 opKeyStore，那么设置 keyStoreRef 是可选操作。例如：

```
<openidConnectProvider id="OAuthConfigSample"
 oauthProviderRef="OAuthConfigSample" signatureAlgorithm="RS256"
 keyStoreRef="opTestKeyStore" keyAliasName="myOpKeyAlias" />
```

您现在已配置 OpenID Connect 提供程序以使用 RSA-SHA256 签署标识令牌。

配置 OpenID Connect 提供程序以接受 JSON Web 令牌 (JWT) 进行授权

可配置充当 OpenID Connect 提供程序的 xigemaAS 服务器以在交换中接受 JSON Web 令牌来获取访问令牌。

通过启用 xigemaAS 的 openidConnectServer-1.0 和 ssl-1.0 功能部件并输入其他可选配置信息，可配置充当 OpenID Connect 提供程序的 xigemaAS 服务器以接受 JSON Web 令牌。

1. 确保 server.xml 文件的功能部件清单中包含 ssl-1.0 和 openidConnectServer-1.0 功能部件。

```
<featureManager>
    <feature>ssl-1.0</feature>
    <feature>openidConnectServer-1.0</feature>
</featureManager>
```

2. 可选：在相应 oauthProvider 元素中配置 jwtGrantType 元素。jwtGrantType 元素是可选的。如果未包含 jwtGrantType 元素，那么系统会对所有属性使用缺省值；例如：

```
<oauthProvider id="OAuthConfigSample" ...>
    <jwtGrantType clockSkew="5m" iatRequired="false"
 tokenMaxLifetime="120m" maxJtiCacheSize="10000"/>
    ...
</oauthProvider>
```

配置 OpenID Connect 提供程序以接受客户机注册请求

客户机注册端点是管理员管理的服务，用于注册、更新、删除和检索有关计划使用 OpenID Connect 提供程序的 OpenID Connect 依赖方的信息。反过来，注册过程可提供信息以供依赖方使用它，包括 OAuth 2.0 客户机标识和客户端密钥（如果未指定）。

客户机注册服务按以下两种方式的其中一种运行：本地存储器或数据库存储器。这些方式由以下因素确定：xigemaAS 服务器配置其客户机存储器的方式，以及客户机是使用 server.xml 文件中的 `oauthProvider localStore` 属性定义的（本地存储器）还是使用数据库配置的（数据库存储器）。

在本地存储器配置中，客户机注册服务被限制为仅检索 OpenID Connect 依赖方信息。可修改 server.xml 文件来添加更多操作以注册、更新或删除 OpenID Connect 依赖方。

在数据库存储器配置中，没有针对客户机注册服务的限制，可通过 `REST` 接口完成所有操作。

👉 注：xigemaAS 服务器不得同时使用本地存储器和数据库存储器配置其客户机存储器。仅应选择一个配置路由。

客户机注册端点是具有 `clientManager` 角色的受保护管理端点。为使用户能够访问此端点，管理员必须向此用户授予 `clientManager` 角色。

`clientManager` 角色是对 `oauthProvider` 定义的其中一个 `oauth-roles`。以下是样本配置，显示向 `clientAdministrator` 组中的用户 Alice 或成员授予 `clientManager` 角色的过程。

```
<oauth-roles>
<authenticated>
<special-subject type="ALL_AUTHENTICATED_USERS"/ >
</authenticated>
<clientManager>
<group name="clientAdministrator" />
<user name="Alice" />
</clientManager>
</oauth-roles>
```

有关 OpenID Connect 依赖方的客户机注册信息主要用于定义客户机的使用方案约束。此外，对客户机不透明的 OP 的其他操作使用客户机注册元数据来制定权限决策。

以下示例假定在端口 443 上对xigemaAS OP 配置了 SSL。

```
https://server.example.com:443/oidc/endpoint/<provider_name>/registration
```

以上示例还假定对 server.xml 文件配置了用户名 `clientAdmin` 和密码 `clientAdminPassword`，并且使用 `oauth-role clientManager`。

客户机注册元数据由以下参数组成：

表 29: 客户机注册参数

属性名称	数据类型	必需/可选	描述
client_id	输入/输出	可选	正向 OP 注册的客户机标识。除非指定，否则，在缺省情况下，此参数值是在注册期间生成的。它是一个字符串。

属性名称	数据类型	必需/可选	描述
client_secret	输入/输出	可选	正向 OP 注册的客户端密钥。除非指定，否则，在缺省情况下，此参数值是在注册期间生成的。它是一个字符串。在更新操作期间，参数值“*”保留现有值。空白参数值生成新 client_secret。非空白参数值使用新指定的值覆盖现有值。
client_name	输入/输出	可选	正向 OP 注册的客户机的描述。除非指定，否则，在缺省情况下，此参数设置为 client_id 参数值。它是一个字符串。
application_type	输入	可选	用于描述客户机的应用程序类型。除非指定，否则缺省值为 web。它是一个字符串。例如，可能的值为： • <an empty value is valid> • web • native
response_types	输入	可选	此客户机使用的响应类型约束。除非指定，否则缺省值为 code。它是一个 JSON 数组。例如，可能的值为： • <an empty value is valid> • code • token • id_token token (order reversible) 对于特定 response_type，必须指定对应 grant_types。
grant_types	输入	可选	此客户机使用的授予类型约束。除非指定，否则缺省值为 authorization_cod

属性名称	数据类型	必需/可选	描述
			e。它是一个 JSON 数组。例如，可能的值为： • <an empty value is valid> • authorization_code • implicit • refresh_token • client_credentials • urn:ietf:params:oauth:grant-type:jwtbearer • password
redirect_uris	输入	可选	客户机被约束为使用的重定向 URI 数组。它是一个 JSON 数组。
post_logout_redirect_uris	输入	可选	客户机被约束为使用的注销后重定向 URI 数组。它是一个 JSON 数组。
trusted_uri_prefixes	输入	可选	客户机在发送访问令牌时被认定为安全的可信 URI 前缀数组。它是一个 JSON 数组。
scope	输入	可选	客户机被约束为使用的空格定界范围值。它是一个字符串。如果允许客户机请求任何范围，那么可使用值 ALL_SCOPES。
preauthorized_scope	输入	可选	客户机被预授权的空格定界范围值（不需要用户同意）。它是一个字符串。
subject_type	输入	可选	客户机描述的主体集类型约束。它是一个字符串。例如，可能的值为： • <an empty value is valid> • public
token_endpoint_auth_method	输入	可选	客户机使用的令牌端点认证方法约束。除非指定，否则缺省值为 client_secret_basic。它是一个字符串。例如，可能的值为：

属性名称	数据类型	必需/可选	描述
			• <an empty value is valid> • client_secret_basic • client_secret_post • none
functional_user_id	输入	可选	此参数指示要与代表 client_credentials 授予类型的客户机发出的请求相关联的用户标识。它是一个字符串。
functional_user_groupIds	输入	可选	要与使用客户机凭证授予类型的客户机获取的访问令牌相关联的组标识列表。该值为生效用户所属组的组标识列表，其中组标识是区分大小写的字符串。字符串由授权服务器定义。如果此值包含多个组标识，它们的顺序也无关紧要。如果列表为空，那么会省略声明。如果指定了此客户机元数据元素，那么系统会在来自内省端点的 functional_user_groupIds 响应参数中为具有客户机凭证授权的客户机发出的访问令牌返回值。如果未使用 functional_user_id 参数，那么此参数被忽略。 👉 注：授权服务器不得信任自我断言此参数的客户机。
introspect_tokens	输入	可选	一个参数值，用于指示所指定客户机是否有权对 OP 发出的访问令牌执行内省。它是一个布尔值。
registration_client_uri	仅输出	不适用	响应中返回的一个参数，其值指示已注册客户机的唯一 URL。它是一个字符串。

属性名称	数据类型	必需/可选	描述
client_secret_expires_at	仅输出	不适用	响应中返回的一个参数，其值指示客户端密钥的到期时间，以从 1970-01- 01T0:0:0Z 开始的秒数（以 UTC 为标准进行测量）计算。值 0 指示无到期时间。
client_id_issued_at	仅输出	不适用	响应中返回的一个参数，其值指示客户机标识的发放时间，以从 1970-01- 01T0:0:0Z 开始的秒数（以 UTC 为标准进行测量）计算。值 0 指示未标识客户机标识发放时间。

1. 注册客户机，如以下示例中所示：

   请求头：

```
POST https://server.example.com:443/oidc/endpoint/<provider_name>/
registration
Accept: application/json
Content-Type: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

   请求有效内容：

```
{
   "token_endpoint_auth_method":"client_secret_basic",
   "scope":"openid profile email general",
   "grant_types":[
      "authorization_code",
      "client_credentials",
      "implicit",
      "refresh_token",
      "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ],
   "response_types":[
      "code",
      "token",
      "id_token token"
   ],
   "application_type":"web",
   "subject_type":"public",
   "post_logout_redirect_uris":[
      "https://server.example.com:9000/logout/",
      "https://server.example.com:9001/exit/"
   ],
   "preauthorized_scope":"openid profile email general",
   "introspect_tokens":true,
   "trusted_uri_prefixes":[
      "https://server.example.com:9000/trusted/"
```

```
   ],
   "redirect_uris":[
      "https://server.example.com:443/resource/redirect1",
      "https://server.example.com:9000/resource/redirect2"
   ]
}
```

响应头：

```
Status: 201
Cache-Control: private
ETag: "1B2M2Y8AsgTpgAmY7PhCfg=="
Content-Type: application/json
```

响应主体：

```
{
   "client_id_issued_at":1401776782,
   "registration_client_uri":"https://server.example.com:8020/oidc/
endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a312d8",
   "client_secret_expires_at":0,
   "token_endpoint_auth_method":"client_secret_basic",
   "scope":"openid profile email general",
   "grant_types":[
      "authorization_code",
      "client_credentials",
      "implicit",
      "refresh_token",
      "urn:ietf:params:oauth:grant-type:jwt-bearer"
   ],
   "response_types":[
      "code",
      "token",
      "id_token token"
   ],
   "application_type":"web",
   "subject_type":"public",
   "post_logout_redirect_uris":[
      "https://server.example.com:9000/logout/",
      "https://server.example.com:9001/exit/"
   ],
   "preauthorized_scope":"openid profile email general",
   "introspect_tokens":true,
   "trusted_uri_prefixes":[
      "https://server.example.com:9000/trusted/"
   ],
   "client_id":"b0a376ec4b694b67b6baeb0604a312d8",

 "client_secret":"nmrOQ20CrMdwd4pjqaimutZTcbQPzIoYgItjaccb9Wk33rKarhM3WDLmWIoE",
   "client_name":"b0a376ec4b694b67b6baeb0604a312d8",
   "redirect_uris":[
      "https://server.example.com:443/resource/redirect1",
      "https://server.example.com:9000/resource/redirect2"
   ]
}
```

2. 更新客户机，如以下示例中所示：

请求头：

```
PUT https://server.example.com:443/oidc/endpoint/<provider_name>/
registration/registration/b0a376ec4b694b67b6baeb0604a312d8
Accept: application/json
Content-Type: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

请求有效内容：

```
{
   "token_endpoint_auth_method":"client_secret_basic",
   "scope":"openid profile",
   "grant_types":[
      "authorization_code"
   ],
   "response_types":[
      "code"
   ],
   "application_type":"native",
   "subject_type":"public",
   "post_logout_redirect_uris":[
      "https://server.example.com:9000/logout/"
   ],
   "preauthorized_scope":"openid",
   "introspect_tokens":false,
   "trusted_uri_prefixes":[
      "https://server.example.com:9003/trusted/"
   ],
   "client_id":"b0a376ec4b694b67b6baeb0604a312d8",
   "client_secret":"*",
   "client_name":"updated client",
   "redirect_uris":[
      "https://server.example.com:443/resource/redirect1"
   ]
}
```

响应头：

```
Status: 200
ETag: "3DD7affTGS91mfhPZ83B39Y=="
Content-Type: application/json
```

响应主体：

```
{
   "client_id_issued_at":1401776782,
   "registration_client_uri":"https://server.example.com:8020/oidc/
endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a312d8",
   "client_secret_expires_at":0,
   "token_endpoint_auth_method":"client_secret_basic",
   "scope":"openid profile",
   "grant_types":[
      "authorization_code"
   ],
   "response_types":[
      "code"
   ],
```

```
   "application_type":"native",
   "subject_type":"public",
   "post_logout_redirect_uris":[
      "https://server.example.com:9000/logout/"
   ],
   "preauthorized_scope":"openid",
   "introspect_tokens":false,
   "trusted_uri_prefixes":[
      "https://server.example.com:9003/trusted/"
   ],
   "client_id":"b0a376ec4b694b67b6baeb0604a312d8",
   "client_secret":"*",
   "client_name":"updated client",
   "redirect_uris":[
      "https://server.example.com:443/resource/redirect1"
   ]
}
```

3. 检索客户机，如以下示例中所示：

   请求头：

```
GET https://server.example.com:443/oidc/endpoint/<provider_name>/
registration/registration/b0a376ec4b694b67b6baeb0604a312d8
Accept: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

   响应头：

```
Status: 200
Cache-Control: private
ETag: "3DD7affTGS91mfhPZ83B39Y=="
Content-Type: application/json
```

   响应主体：

```
{
   "client_id_issued_at":1401776782,
   "registration_client_uri":"https://server.example.com:8020/oidc/
endpoint/OIDC/registration/b0a376ec4b694b67b6baeb0604a312d8",
   "client_secret_expires_at":0,
   "token_endpoint_auth_method":"client_secret_basic",
   "scope":"openid profile",
   "grant_types":[
      "authorization_code"
   ],
   "response_types":[
      "code"
   ],
   "application_type":"native",
   "subject_type":"public",
   "post_logout_redirect_uris":[
      "https://server.example.com:9000/logout/"
   ],
   "preauthorized_scope":"openid",
   "introspect_tokens":false,
   "trusted_uri_prefixes":[
      "https://server.example.com:9003/trusted/"
   ],
```

```
    "client_id":"b0a376ec4b694b67b6baeb0604a312d8",
    "client_secret":"*",
    "client_name":"updated client",
    "redirect_uris":[
       "https://server.example.com:443/resource/redirect1"
    ]
}
```

4. 检索客户机（头请求），如以下示例中所示：

   请求头：

```
HEAD https://server.example.com:443/oidc/endpoint/<provider_name>/
registration/registration/b0a376ec4b694b67b6baeb0604a312d8
Accept: application/json
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

   响应头：

```
Status: 200
Cache-Control: private, no-cache=set-cookie
ETag: "3DD7affTGS91mfhPZ83B39Y=="
Content-Type: application/json
```

5. 删除客户机，如以下示例中所示：

   请求头：

```
DELETE https://server.example.com:443/oidc/endpoint/<provider_name>/
registration/registration/b0a376ec4b694b67b6baeb0604a312d8
Authorization: Basic Y2xpZW50QWRtaW46Y2xpZW50QWRtaW5QYXNzd29yZA==
```

   响应头：

```
Status: 204
Content-Length: 0
Content-Language: zh-CN
```

   注：本主题中的信息也适用于 OAuth 2.0 客户机的客户机注册服务及 OpenID Connect 依赖方。

OpenID Connect 定制表单

可以替换用于用户认证的缺省表单登录页面，或者开发您自己的用户同意表单来收集客户机授权数据。

认证用户

OpenID Connect 提供程序支持使用传统 Java™ Platform Enterprise Edition (J2EE) FormLogin 进行用户认证。

可定制登录表单，请参阅 *OpenID Connect* 定制表单。

OpenID Connect 提供程序可配置为支持其他认证方法。

OpenID Connect 提供程序将用户认证委派给第三方认证服务

如果您配置信任关联拦截器 (TAI) 以拦截针对 OpenID Connect 授权端点 (`/oidc/<provider name>/authorize`) 的请求，那么系统不会显示登录表单，用户认证由所配置 TAI 执行。

OpenID Connect 提供程序使用 HTTP 基本认证来认证用户

如果希望 OpenID Connect 提供程序使用 HTTP 基本认证来认证用户，那么按基本认证方案中的定义，`OpenID Connect autorization` 请求必须包含用户标识和密码。

OpenID Connect 提供程序使用客户机证书来认证用户

如果希望 OpenID Connect 提供程序使用客户机证书来认证用户，那么您需要在 `openidConnectProvider` 配置引用的 `oauthProvider` 配置元素内显式添加属性 `certAuthentication=true`，并且用户代理必须能够为 OpenID Connect 授权请求提供客户机证书。

OpenID Connect 提供程序将用户认证委托给第三方 OpenID Connect 提供程序

可配置 OpenID Connect 提供程序以将用户认证委托给第三方 OpenID Connect 提供程序。要启用此认证委托，应将 OP 配置为 OpenID Connect 依赖方。（可选）可添加认证过滤器以将 `openIDConnectClient-1.0` 功能部件限制为仅保护 OpenID Connect 授权端点 (`/oidc/<provider name>/authorize`)。

在 xigemaAS 中配置 OpenID Connect 客户机

可配置 xigemaAS 概要文件服务器以充当 OpenID Connect 客户机或依赖方，以使用 Web 单点登录并将 OpenID Connect 提供者用作身份提供者。

通过启用 xigemaAS 的 `openidConnectClient-1.0` 功能部件并配置其他配置信息，可配置 xigemaAS 概要文件服务器以充当 OpenID Connect 客户机。

1. 将 `openidConnectClient-1.0` xigemaAS 功能部件及任何其他所需功能部件添加至 server.xml 文件。对于 `openidConnectClient-1.0` 功能部件，还需要 `ssl-1.0` 功能部件。在 server.xml 文件的 `featureManager` 元素内添加以下元素声明：

```
<feature>openidConnectClient-1.0</feature>
<feature>ssl-1.0</feature>
```

2. 配置 `openidConnectClient` 元素。以下是使用缺省 xigemaAS 概要文件服务器 OpenID Connect 提供者的基本配置的示例。

   对于该客户机，在给定 URL 模式中必须有相应的已配置应用程序，以便能够处理来自 OpenID Connect 提供者的重定向请求。此 URL 还必须与客户机向 OP 注册的重定向 URL 精确匹配。

   注：在此示例中，客户机期望 OP 的 SSL 端口设置为 443。

```
<openidConnectClient id="client01"
    clientId="client01"
    clientSecret="{xor}LDo8LTor"
    authorizationEndpointUrl="https://server.example.com:443/oidc/
endpoint/OidcConfigSample/authorize"
    tokenEndpointUrl="https://server.example.com:443/oidc/endpoint/
OidcConfigSample/token">
</openidConnectClient>
```

   在此基本配置样本中，采用以下缺省值：

   - scope=`openid profile`：范围 `openid` 是必需的，您可使用 `scope` 属性以编辑必需范围。例如，可将必需 `scope` 更改为 `openid profile email`。

- 此 RP 向 OP 将其重定向 URL 注册为 `https://<host name>:<ssl port>/oidcclient/redirect/client01`，其中主机名和 ssl 端口是自动解析的，`client01` 是 `openidConnectClient` 配置元素的标识。如果 RP 前端有代理，那么您可使用属性 `redirectToRPHostAndPort` 覆盖主机名和端口，并将 `redirectToRPHostAndPort` 设置为 `https://<host name>:<ssl port>`。

3. 配置用户注册表。缺省情况下，OP 返回的用户身份不会映射至注册表用户，所以不需要在注册表中配置用户。但是，如果 `openidConnectClient` 元素的 *mapIdentityToRegistryUser* 属性设置为 `true`，那么从 OP 返回的相应身份必须有对应的用户条目，认证和授权才能成功。有关配置用户注册表的更多信息，请参阅为 *xigemaAS* 概要文件配置用户注册表（见第 1293 页）。
4. 配置服务器的信任库以包含受支持的 OpenID Connect 提供程序的签署者证书。有关密钥库的信息，请参阅对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）。
5. 修改服务器的 SSL 配置以使用所配置信任库。

```
<sslDefault sslRef="DefaultSSLSettings" />
<ssl id="DefaultSSLSettings" keyStoreRef="myKeyStore"
 trustStoreRef="myTrustStore" />
<keyStore id="myKeyStore" password="{xor}EzY9Oi0rJg==" type="jks"
 location="${server.config.dir}/resources/security/BasicKeyStore.jks" />
<keyStore id="myTrustStore" password="{xor}EzY9Oi0rJg==" type="jks"
 location="${server.config.dir}/resources/security/BasicTrustStore.jks" />
```

   OpenID Connect 配置为使用服务器指定的缺省 SSL 配置。因此，服务器的缺省 SSL 配置必须使用为 OpenID Connect 配置的信任库。
6. 可选：配置第三方 OpenID Connect 提供程序。

   要配置 xigemaAS OpenID Connect 客户机以使用第三方 OpenID Connect 提供者（例如，Microsoft™ Azure 或 Google），必须配置以下属性。可通过调用 OP 的发现端点获取属性值，此端点在通过将字符串 `/.well-known/openid-configuration` 连接至发出者形成的路径上提供 JSON 文档。

   a. 将 `jwkEndpointUrl` 属性设置为 OP 的 JSON Web 密钥集 `JWK` 文档的 URL，此 URL 在发现文件中定义为 `jwks_uri`。
   b. 将 `issuerIdentifier` 属性设置为发现文件中定义的 `issuer`。未包含此值作为 `iss` 声明的标识令牌会被拒绝。例如，如果您将 Google 用作 OP，那么可设置 `issuerIdentifier="accounts.google.com"`。
   c. 设置 `signatureAlgorithm="RS256"`。xigemaAS OpenID Connect 客户机的缺省签名算法为 HS256。
   d. 将 `userIdentityToCreateSubject` 属性设置为表示用户唯一标识的供应商标识令牌使用的声明名称。例如，如果您使用 Google 的 OP，那么您可设置 `userIdentityToCreateSubject ="email"`，如果您使用 Microsoft™ Azure，那么您可设置 `userIdentityToCreateSubject ="upn"` 或 `userIdentityToCreateSubject ="unique_name"`。
   e. 将 `groupIdentifier` 属性设置为表示用户组成员资格或角色的声明名称。例如，如果您使用 Microsoft™ Azure，那么可设置 `groupIdentifier="groups"`。

7. 可选：认证过滤器。

   如果已启用 `openidConnectClient-1.0` 功能部件并且未对 `openidConnectClient` 元素配置 `authFilterRef` 属性，那么系统通过 OpenID Connect 提供程序认证任何未认证的请求。

   有关配置认证过滤器的更多信息，请参阅认证过滤器（见第 1363 页）。
8. 支持多个 OpenID Connect 提供程序。

通过创建多个 `openidConnectClient` 元素和多个认证过滤器，可将 xigemaAS 配置为多个 OpenID Connect 提供程序的 OpenID Connect 依赖方。每个 `openidConnectClient` 元素定义与一个 OpenID Connect 提供程序的单点登录关系，并使用 `authFilterRef` 属性引用一个认证过滤器。

9. 配置受支持的标识令牌签名算法。

   可配置 xigemaAS OpenID Connect 客户机以在标识令牌中支持 RS256 签名算法。xigemaAS OpenID Connect 客户机的缺省签名算法为 HS256。如果通过设置 `signatureAlgorithm="RS256"` 将 RS256 配置为标识令牌的签名算法，那么必须配置 `trustStoreRef` 和 `trustAliasName`，除非 OP 支持 JWK 端点。

10. 可选：配置“隐式”授权类型。

    `openidConnectClient-1.0` 功能部件使用“授权代码”授权类型以请求用户认证令牌，您可配置 xigemaAS `openidConnectClient-1.0` 功能部件以通过将 `grantType="implicit"` 添加至 server.xml 文件以使用“隐式”授权类型。如果 xigemaAS 服务器和 OpenID Connect 提供程序在不同防火墙中，那么必须使用此配置选项。

您现在已建立将 xigemaAS 服务器配置为 OpenID Connect 客户机（它能够与配置为 OpenID Connect 提供程序的其他 xigemaAS 服务器通信）时所需的最低配置。

对 OpenID Connect 调用授权端点

在 OpenID Connect 中，授权端点处理用户的认证和授权。

例如，从浏览器内的客户机应用程序或脚本语言（例如，JavaScript）中实现的客户机应用程序启动授权端点时，不需要将 xigemaAS 概要文件服务器配置为 OpenID Connect 客户机。

授权端点接受包含 `OAuth 2.0` 和 `OpenID Connect 1.0` 规范定义的参数的认证请求。

在授权代码流程中，授权端点用于认证和授权，并返回针对客户机的授权。此授权可由客户机在针对标识令牌、访问令牌和刷新令牌的请求中通过交换传递至令牌端点。在隐式流程中，授权端点仍执行认证和授权，但还会直接在其响应中对客户机返回标识令牌和访问令牌；不会执行与令牌端点的交互。

已启用 OpenID Connect 的xigemaAS 概要文件服务器可访问位于以下 URL 的 OpenID Connect 授权端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/authorize
```

👉 注：在此示例中，OP 的 SSL 端口应该为 443。

1. 准备包含以下必需参数和建议参数的 HTTP GET 或 POST 请求。

   - scope：（必需）OpenID Connect 请求必须包含 `openid` 范围值。也可存在其他范围
   - response_type：（必需）确定要使用的授权处理流程。使用授权代码流程时，此值为 `code`。使用隐式流程时，此值为 `id_token token` 或 `id_token`。此值为 `id_token` 时，不返回任何访问令牌
   - client_id：（必需）在 OpenID Connect 提供程序上有效的客户机标识。
   - redirect_uri：（必需）响应将发送至的重定向 URI。此值必须与 OP 上已注册客户机的某个重定向 URI 值完全匹配。
   - state：（建议）不透明值，用于在请求与回调之间维护状态。
   - nonce：（对于隐式流程是必需的）字符串值，用于将客户机会话与标识令牌相关联并减少重放攻击。

   可在请求中包含更多参数。有关其他受支持参数的描述，请参阅 OpenID Connect Core 1.0 规范。

   我们只是不支持 `id_token` 响应类型。使用隐式流程时，必须始终使用 `id_token token` 并将返回访问令牌。

2. 将 GET 或 POST 请求发送至授权端点 URL。

完成这些步骤后，您将获得有效 HTTP 请求，此请求将发送至授权端点。授权端点按“示例”一节中描述的方式返回响应。

接收到来自客户机的请求后，OpenID Connect 提供程序尝试对用户进行认证和授权。

在授权代码流程中，如果认证和授权成功，那么 OpenID Connect 提供程序会发出授权代码并将其作为参数包含在客户机的 `OAuth 2.0` 授权响应中。如果初始请求包含 state，那么授权响应还将包含初始请求中包含的确切 state 值。通过使用 `application/x-www-form-urlencoded` 格式，code 和 state 参数作为查询参数添加至授权请求中指定的 redirect_uri 值。

在隐式流程中，如果认证和授权成功，那么系统从授权端点返回以下参数。

- access_token：访问令牌。除非初始请求中的 [response_type] 值为 [id_token]，否则返回此项。
- token_type：OAuth 2.0 令牌类型。对于 OpenID Connect，此值为 `Bearer`。
- id_token：标识令牌。
- state：包含在授权请求中时为必需项。
- expires_in：（可选）从生成响应开始算起访问令牌的到期时间（以秒计）。

这些参数会添加至授权请求中指定的 redirect_uri 值的片段部分中，而不是作为查询参数（例如，在授权代码流程中）。

以下示例显示授权和隐式代码流程的格式。

以下显示授权代码流程的请求示例：

```
GET /authorize?
    response_type=code
    &scope=openid profile email
    &client_id=client01
    &state=af0ifjsldkj
    &redirect_uri=https://server.example.com:443/oidcclient/redirect/client01
HTTP/1.1
```

以下显示隐式流程的请求示例：

```
GET /authorize?
    response_type=id_token token
    &scope=openid profile
    &client_id=client01
    &state=af0ifjsldkj
    &redirect_uri=https://server.example.com:443/oidcclient/redirect/client01
    &nonce=n-0S6_WzA2Mj HTTP/1.1
```

以下显示授权代码流程中的授权端点返回的响应示例：

```
HTTP/1.1 302 Found
Location: https://server.example.com:443/oidcclient/redirect/client01
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

以下显示隐式流程中的授权端点返回的响应示例：

```
HTTP/1.1 302 Found
Location: https://server.example.com:443/oidcclient/redirect/client01
```

```
    access_token=SlAV32hkKG
    &token_type=Bearer
    &id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso
    &expires_in=3600
    &state=af0ifjsldkj
```

对 OpenID Connect 调用令牌端点

在 OpenID Connect 授权代码流程中，客户机使用令牌端点来获取标识令牌、访问令牌和刷新令牌。

从浏览器内的客户机应用程序或脚本语言（例如，JavaScript）中实现的客户机应用程序启动令牌端点时，不需要将 xigemaAS 服务器配置为 OpenID Connect 客户机。

令牌端点接受客户机的请求，此请求包含授权端点对客户机发出的授权代码。验证授权代码后，系统返回相应令牌作为对客户机的响应。

OpenID Connect 隐式流程中未使用令牌端点。

已启用 OpenID Connect 的xigemaAS概要文件服务器可访问位于以下 URL 的 OpenID Connect 令牌端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/token
```

👉 注：在此示例中，OP 的 SSL 端口应该为 443。带有令牌端点的所有通信必须使用 TLS。

1. 使用以下参数准备 HTTP POST 请求。
   - grant_type：此参数的值必须为 `authorization_code`。
   - code：从授权端点接收到的授权代码。

   这些参数必须通过使用 `application/x-www-form-urlencoded` 格式添加。
2. 将请求发布至令牌端点 URL。

完成这些步骤后，您将获得有效 HTTP POST 请求，此请求将发送至令牌端点。令牌端点按“示例”一节中描述的方式返回响应。

OpenID Connect 提供程序验证从客户机接收到的令牌请求时，OpenID Connect 提供程序将带有 `application/json` 格式的 JSON 对象的 HTTP 200 响应返回至客户机。此响应包含标识令牌、访问令牌和刷新令牌及以下附加参数：

- token_type：OAuth 2.0 令牌类型。对于 OpenID Connect，此值为 `Bearer`。
- expires_in：从生成响应开始算起访问令牌的到期时间（以秒计）。

对于来自包含令牌、密钥或其他敏感信息的令牌端点的所有响应，其 Cache-Control 头值设置为 no-store，Pragma 头值设置为 no-cache。

.

下面显示 HTTP POST 请求和响应的示例

以下显示请求示例：

```
POST /token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    grant_type=authorization_code
    &code=SplxlOBeZQQYbYS6WxSbIA
```

```
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

以下显示响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
    "access_token": "SlAV32hkKG",
    "token_type": "Bearer",
    "refresh_token": "8xLOxBtZp8",
    "expires_in": 3600,
    "id_token": "eyJ ... zcifQ.ewo ... NzAKfQ.ggW8h ... Mzqg"
}
```

对 OpenID Connect 调用内省端点

内省端点允许访问令牌的持有者向发出该访问令牌的 OpenID Connect 提供程序请求有关访问令牌的一组元数据。该访问令牌必须是通过 OpenID Connect 或 OAuth 认证获取的访问令牌。

资源服务或客户机应用程序调用内省端点时，它必须将自身注册为 OpenID Connect 服务器的正常 OAuth 2.0 客户机。所注册客户机元数据必须包含属性 `introspectTokens = true`。

OpenID Connect 和 OAuth 2.0 中使用的访问令牌内包含的信息对客户机是不透明的。这允许受保护资源或客户机根据 OpenID Connect 提供程序返回的有关访问令牌的元数据制定权限决策。

已启用 OpenID Connect 的 xigemaAS 服务器可访问位于以下 URL 的 OpenID Connect 内省端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/introspect
```

👉 注：在此示例中，OP 的 SSL 端口应该为 443。

1. 在 `GET` 或 `POST` 请求的 HTTP 基本授权头中使用已注册 OpenID Connect 客户机的客户机标识和密码设置客户机认证。此客户机标识和密码是使用 `application/x-www-form-urlencoded` 编码算法进行编码的。所编码客户机标识用作用户名，所编码密码用作密码。
2. 在针对内省端点的 `GET` 或 `POST` 请求中包含访问令牌的字符串值作为参数。
3. 向内省端点 URL 发送 `GET` 或 `POST` 请求。

完成这些步骤后，您将获得有效 HTTP 请求，此请求将按“示例”一节中所示发送至内省端点。

对于有效请求，内省端点返回带有 `application/json` 格式的 `JSON` 对象的 `HTTP 200` 响应，根据访问令牌是处于活动状态还是已到期，该对象包含以下信息。

如果访问令牌处于活动状态，那么此端点返回 `active:true`，并且 `JSON` 对象包含以下附加信息：

active

布尔指示符，指示访问令牌是否处于活动状态。

client_id

请求访问令牌的 OpenID Connect 客户机的客户机标识。

sub

授予访问令牌的资源所有者。

scope

与访问令牌相关联的范围的空格分隔列表。

iat

自 1970 年 1 月 1 日 (UTC) 以来的整数时间戳记（以秒计），指示访问令牌的发出时间。

exp

自 1970 年 1 月 1 日 (UTC) 以来的整数时间戳记（以秒计），指示访问令牌的到期时间。

realmName

资源所有者的域名。

uniqueSecurityName

资源所有者的唯一安全名称。

tokenType

访问令牌类型。对于 OpenID Connect，此值为 `Bearer`。

grant_type

一个字符串，指示用于生成访问令牌的授权类型。可能的值为：`authorization_code`、`password`、`refresh_token`、`client_credentials`、`resource_owner`、`implicit` 和 `urn:ietf:params:oauth:grant-type:jwt-bearer`。

如果访问令牌已到期，但所提供认证有效，或者如果所提供访问令牌为错误类型，那么此端点在 `JSON` 对象中返回 `active:false`。

注：如果客户机或资源服务要执行访问令牌内省，那么该客户机或资源服务必须将其自身注册为 OpenID Connect 提供程序的客户机，并且客户机元数据必须将 `introspect_tokens` 设置为 `true`。

下面显示处于活动状态和已到期的访问令牌及请求的示例。

以下显示请求示例：

```
POST /register HTTP/1.1
 Accept: application/x-www-form-urlencoded
 Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3
     token=SOYleDziTitHeKcodp6vqEmRwKPjz3lFZTcsQtVC
```

活动访问令牌的响应示例：

```
 HTTP/1.1 200 OK
 Content-Type: application/json
 Cache-Control: no-store
{
   "exp"                : 1415307710,
   "realmName"          : "BasicRealm",
   "sub"                : "testuser",
   "scope"              : "openid scope2 scope1",
   "grant_type"         : "authorization_code",
   "uniqueSecurityName" : "testuser",
   "active"             : true,
   "token_type"         : "Bearer",
```

```
   "client_id"           : "pclient01",
   "iat"                 : 1415307700
}
```

到期访问令牌的响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
    "active":"false"
}
```

调用覆盖映射服务

覆盖映射服务是一个不受保护的端点，它返回以斜杠终结的 URI 前缀的 JavaScript™ 对象表示法 (JSON) 数组。此 URI 前缀数组指定哪些 Web 上下文包含在单点登录 (SSO) 组中，从而允许客户机知道 URL 目标是否被认为是可向其发送访问令牌的安全对象。

覆盖映射服务返回 URI 前缀的 JSON 数组，这是从已注册客户机中指定的 trusted_uri_prefixes 参数值的汇总派生的独有集合。因此，填充覆盖映射服务的一个典型用例是注册客户机并指定 trusted_uri_prefixes 值。

以下示例假定在端口 443 上对 xigemaAS OpenID Connect 提供程序配置了 SSL。

```
https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map
```

此外，此示例假定客户机已使用所指定 trusted_uri_prefixes 注册。

1. 对 coverage_map 端点指定 token_type URI 查询参数。以下示例请求假定客户机已使用所指定 trusted_uri_prefixes 注册。

   ```
   https://server.example.com:443/oidc/endpoint/<provider_name>/coverage_map?
   token_type=bearer
   ```

   唯一受支持的 token_type 值为 token_type=bearer。
2. 获取不记名令牌类型的覆盖映射，如以下示例中所示。

   请求头：

   ```
   GET https://server.example.com:443/oidc/endpoint/<provider_name>/
   coverage_map?token_type=bearer
   ```

   响应头：

   ```
   Status: 200
   CacheControl: public, maxage=600
   ETag:"vvhkgXkRx+BzR3Q4kwCCqw=="
   ContentType: application/json
   ```

   响应主体：

   ```
   [
     "http://res1.vsettan.com.cn/",
     "https://trusted.server.vsettan.com.cn:9554/resources/"
   ]
   ```

对 OpenID Connect 调用 UserInfo 端点

UserInfo 端点返回有关使用 OpenID Connect 认证进行认证的用户的声明。

要获取用户的声明，客户机通过将访问令牌用作凭证向 UserInfo 端点发出请求。该访问令牌必须是通过 OpenID Connect 认证获取的访问令牌。访问令牌表示的用户的声明作为包含声明的一组“名称/值”对的 JSON 对象返回。UserInfo 端点是受 OAuth 2.0 保护的资源，这意味着需要访问该端点的凭证是访问令牌。

UserInfo 端点返回的声明可使用 OpenID Connect 提供程序配置进行定制，请参阅配置 *UserInfo* 端点返回的声明。

已启用 OpenID Connect 的 xigemaAS 概要文件服务器可访问位于以下 URL 的 OpenID Connect UserInfo 端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/userinfo
```

👉 注：在此示例中，OP 的 SSL 端口应该为 443。

1. 使用通过 OpenID Connect 认证获取的访问令牌设置认证。可在 HTTP 基本授权头中提供访问令牌，也可使用 `access_token` 请求参数提供访问令牌。在任一情况都不必对访问令牌进行编码。
2. 将 `GET` 或 `POST` 请求发送至 UserInfo 端点 URL。

完成这些步骤后，您将获得有效 HTTP 请求，此请求将按“示例”一节中所示发送至 UserInfo 端点。

对于有效请求，UserInfo 端点返回带有 `application/json` 格式的 JSON 对象的 `HTTP 200` 响应，该对象包含针对 OpenID Connect 提供程序配置的声明。

以下示例演示带有有效令牌和无效令牌的请求。

- 使用 HTTP Bearer 授权头传递访问令牌的请求
- 针对有效访问令牌的响应
- 无效访问令牌

使用 HTTP Bearer 授权头传递访问令牌的请求示例：

```
POST /register HTTP/1.1
Accept: application/x-www-form-urlencoded
Authorization: Bearer fAAdLO1c6QWDbPs9HrWHz5e7nRWVAnxqTTP7i88G
```

还可使用 `access_token` 请求参数传递令牌：

```
 POST /register HTTP/1.1
 Accept: application/x-www-form-urlencoded
     access_token=fAAdLO1c6QWDbPs9HrWHz5e7nRWVAnxqTTP7i88G
```

最好使用 HTTP 授权头而不是 `access_token` 请求参数，因为可能包含敏感信息的 HTTP 请求参数可能会保存在浏览器历史记录或高速缓存中。

以下是针对有效访问令牌的响应示例。始终返回 `sub` 和 `groupIds` 声明。此处显示的其他声明是 OpenID Connect 提供程序的缺省声明。

```
 HTTP/1.1 200 OK
 Content-Type: application/json
 Cache-Control: no-store
 Pragma: no-cache
```

```
{
   "sub"           : "bob",
   "groupIds"      : [ "bobsdepartment","administrators" ],
   "given_name"    : "Bob",
   "name"          : "Bob Smith",
   "email"         : "bob@mycompany.com",
   "phone_number" : "+1 (604) 555-1234;ext5678",
   "address"       : { "formatted" : "123 Main St., Anytown, TX 77777" },
   "picture"       : "http://mycompany.com/bob_photo.jpg"
}
```

对于无效访问令牌，UserInfo 端点在 WWW-AUTHENTICATE 头中返回 HTTP 401 状态代码和错误消息。

```
HTTP/1.1 401 Unauthorized
CONTENT-LENGTH : 0
WWW-AUTHENTICATE : Bearer error=invalid_token,
   error_description=CWWKS1617E: A userinfo request was made with
       an access token that was not recognized. The request URI was
          /oidc/endpoint/MyOAuthProvider/userinfo.
```

对 OpenID Connect 调用会话管理端点

会话管理端点允许 OpenID Connect 依赖方监视使用特定 OpenID Connect 提供程序 (OP) 的用户的登录状态，同时将网络流量降至最低。在会话管理端点的帮助下，依赖方 (RP) 可注销已从 OpenID Connect 提供程序注销的用户。

OP 会话管理端点 URL 是从 OP 的发现端点返回的发现信息中的 check_session_iframe 属性获取的。此 URL 必须用作需要会话管理功能的 RP 应用程序的 iframe 的目标。RP 应用程序还必须知道 iframe 的 id 属性以向其提交 Window.postMessage() 请求。

为帮助确定用户的登录状态，RP 装入其 src 目标设置为 OP 的会话管理端点的 iframe。会话管理端点可访问用于存储用户的登录状态或浏览器状态的 cookie。此浏览器状态 cookie 在用户从 OP 注销时更新。然后，RP 可使用客户机端脚本调用 OP iframe 的 Window.postMessage() 函数，以在消息文本中发送客户机标识和当前已知会话状态。如果 RP 接收到从 OP 框架返回的 postMessage 且其值为 changed，那么 OP 上的用户的登录状态已更改，并且 RP 可决定是否注销该用户。如果返回的值为 unchanged，那么该用户在 OP 上仍处于已登录状态。

已启用 OpenID Connect 的xigemaAS 概要文件服务器可访问位于以下 URL 的 OpenID Connect 会话管理端点：

```
https://server.example.com:443/oidc/endpoint/<provider_name>/
check_session_iframe
```

👉 注：在此示例中，OP 的 SSL 端口应该为 443。

1. 在能够装入以 OP 会话管理端点为目标的 iframe 的相应 RP 应用程序中创建 Web 资源。此 Web 资源还需要具有对授权响应的 session_state 参数中返回的会话状态值的访问权。例如，会话状态值可存储在 cookie 中，或通过允许 Web 资源中的客户机端脚本知道此值的任何其他方式存储。以下是这类 iframe 的样本 HTML 片段。

   ```
   <iframe id="iframeOP" src="https://server.example.com:443/oidc/endpoint/
   OidcConfigSample/check_session_iframe" frameborder="0" width="0"
    height="0"></iframe>
   ```

2. 要检查用户的登录状态，请调用 OP `iframe` 的 `Window.postMessage()` 函数（将客户机标识和会话状态作为消息参数以 `Client ID + " " + Session State` 格式传递，并将 OP 的主机名作为目标起源参数传递）。在以下 JavaScript™ 样本函数中，该脚本期望会话状态值存储在名为 `session_state` 的 cookie 中，并且 `getCookieValue()` 函数返回存储在 `session_state` cookie 中的值。

```
var targetOP = "https://server.example.com:443";
function checkStatus() {
          var client = "client01";
          var sessionState = getCookieValue("session_state");
          var text = client + " " + sessionState;
          var iframe = document.getElementById("iframeOP");
          iframe.contentWindow.postMessage(text, targetOP);
}
```

3. 配置 Web 资源以侦听来自 OP 的 `postMessages`，它们包含值 `changed` 或 `unchanged` 以反映用户的对应登录状态。然后，RP 可根据从 OP 返回的值决定是否让该用户从 RP 注销。此函数必须确保 `postMessage` 的源与预期 OP 主机名匹配。不匹配的所有消息被拒绝。以下 JavaScript™ 示例说明如何向 Web 资源添加事件侦听器以侦听这类消息。

```
var targetOP = "https://server.example.com:443";
window.addEventListener("message", receiveMessage, false);
function receiveMessage(event) {
          if (event.origin !== targetOP) {
                    // Origin did not come from the OP; this message must
be rejected.
                    return;
          }
          if (event.data === "unchanged") {
                    // User is still logged in to the OP
                  } else {
                    // User has logged out of the OP
          }
}
```

现在 RP 上具有能够在xigemaAS 概要文件服务器 OP 上使用 OpenID Connect 的会话管理功能的 Web 资源。在 OP `iframe` 中维护的浏览器状态会在用户登录 OP 或从 OP 注销时更新。在 OP 上成功登录后，系统会在授权响应中向 RP 提供新会话状态值。然后 RP 可使用客户机端脚本验证用户的会话状态，以确定用户的登录状态在 OP 上是否已更改而不必广播额外的网络流量。

以下 HTML 示例显示一个使用 OpenID Connect 会话管理的完整 HTML 页面。OP `iframe` 的 `src` 属性设置为从 OP 获取的会话管理端点 URL。系统以 60 秒一次的频率自动调用 `startChecking()` 函数以检查用户的登录状态。此页面具有 `message` 事件侦听器，接收到 `postMessage` 时，此侦听器会调用 `receiveMessage()` 函数。此函数确保 `postMessage` 来自 OP 的预期域，并检查所返回消息的值以了解用户的登录状态是 `changed` 还是 `unchanged`。

可将此 HTML 页面本身作为 RP 内另一 Web 资源中的不可见 `iframe` 装入。这允许装入此 `iframe` 的任何 Web 资源在客户端监视用户的登录状态。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>iFrame RP Page</title>
</head>
```

```
 <body onload="javascript:startChecking()">
            <iframe id="iframeOP" src="https://localhost:8999/oidc/endpoint/
OidcConfigSample/check_session_iframe" frameborder="0" width="0" height="0"></
iframe>
 </body>
 <script>
            var targetOP = "https://server.example.com:443";

            window.addEventListener("message", receiveMessage, false);

            function startChecking() {
                      checkStatus();
                      // Check status every 60 seconds
                      setInterval("checkStatus()", 1000*60);
            }

            function getCookieValue(cookieName) {
                      var name = cookieName + "=";
                      var cookies = document.cookie.split(';');
                      if (!cookies) {
                                return null;
                      }
                      for (var i = 0; i < cookies.length; i++) {
                                var cookie = cookies[i].trim();
                                if (cookie.indexOf(name) == 0) {
                                          return cookie.substring(name.length,
 cookie.length);
                                }
                      }
                      return null;
            }

            function checkStatus() {
                      var client = "client01";
                      var sessionState = getCookieValue("session_state");
                      var text = client + " " + sessionState;
                      var iframe = document.getElementById("iframeOP");
                      iframe.contentWindow.postMessage(text, targetOP);
            }

            function receiveMessage(event) {
                      if (event.origin !== targetOP) {
                                // Origin did not come from the OP; this message
 must be rejected
                                return;
                      }
                      if (event.data === "unchanged") {
                                // User is still logged in to the OP
                              } else {
                                // User has logged out of the OP
                      }
            }
 </script>
 </html>
```

认证过滤器

可使用认证过滤器确定特定 HTTP servlet 请求是否由特定提供程序处理。

xigemaAS 服务器认证过滤器使用 server.xml 文件的 `authFilter` 元素中指定的过滤条件来确定特定 HTTP servlet 请求是否由特定提供程序（例如，OpenID、OpenID Connect 或 SPNEGO）处理以获取认证。

如果满足 `authFilter` 元素中的所有条件，那么该 HTTP servlet 请求由 `authFilter` 元素引用的特定提供程序处理。如果没有满足 `authFilter` 元素内的任何条件，那么该 HTTP servlet 请求不会被该提供程序处理。

支持的元素

`authFilter` 元素支持以下元素：`userAgent`、`host`、`webApp`、`remoteAddress` 和 `requestUrl`。

- `userAgent` 元素将与从传入 HTTP servlet 请求抽取的对应头值进行比较。`userAgent` 元素将与“User-Agent”HTTP 请求头进行比较，后者标识源请求使用的客户机软件。对于 Web 客户机浏览器，此值反映用于发出请求的浏览器类型（Internet Explorer、Firefox、Safari 等等）。
- `host` 元素的用法与 `userAgent` 元素类似。`host` 元素将与“Host” HTTP 请求头进行比较，后者标识请求的目标主机名。
- `webApp` 元素用于指定此认证过滤器保护的 xigemaAS 服务器上的应用程序或应用程序列表。
- `remoteAddress` 元素与发送 HTTP 请求的客户机应用程序的 TCP/IP 地址进行比较。可配置通配符以指定子网和范围（通过使用 `matchType` 属性的 `lessThan` 或 `greaterThan` 值），如本主题中之后示例所示。
- `requestUrl` 元素与客户机应用程序使用的 URL 进行比较以发出此请求。将配置单 URL 模式或若干值的管道列表，如本主题中之后示例所示。

认证过滤器示例

请求 **URL** 包含模式

以下示例显示认证过滤器的典型配置。此处，带有包含 `"/SimpleServlet"` 的请求 URL 的任何传入请求由配置为使用此过滤器的服务处理。

```
<authFilter id="myAuthFilter">
         <requestUrl id="myRequestUrl" urlPattern="/SimpleServlet" matchTy
pe="contains"/>
</authFilter>
```

请求 **URL** 包含一组模式中的一个

在以下示例中，指定了请求 URL 模式管道列表。要使用配置为使用此过滤器的服务处理传入请求，传入请求 URL 必须包含 `"/SimpleServlet"`、`"/EmployeeRoleServlet"` 或 `"/AllRoleServlet"` 的其中任何一项。

```
<authFilter id="myAuthFilter">
         <requestUrl id="myURL" urlPattern="/SimpleServlet|/EmployeeRoleSe
rvlet|/AllRoleServlet" matchType="contains" />
</authFilter>
```

Web 应用程序名称包含模式

在以下示例中，认证过滤器中指定了 Web 应用程序名称。传入请求必须以将由配置为使用此过滤器的服务处理的 "myApp" 应用程序作为目标。

```
<authFilter id="myAuthFilter">
        <webApp id="myWebApp" name="myApp" matchType="contains"/>
</authFilter>
```

Web 应用程序名称包含一组模式中的一个

在以下示例中，指定了 Web 应用程序的管道列表。要使用配置为使用此过滤器的服务处理传入请求，传入请求 URL 必须以 "myApp1"、"myApp2" 或 "myApp3" 应用程序的其中任何一项作为目标。

```
<authFilter id="myAuthFilter">
        <webApp id="myWebApp" name="myApp1|myApp2|myApp3" matchType="cont
ains"/>
</authFilter>
```

请求源自某个 **IP** 地址

以下示例显示如何在 remoteAddress 元素中使用通配符。通过此配置，如果传入请求来自 127.0.0.* 范围中的任何位置的 IP 地址，那么配置为使用此过滤器的服务将处理该传入请求。

```
<authFilter id="myAuthFilter">
        <remoteAddress id="myRemoteAddress" ip="127.0.0.*" matchType="equ
als"/>
</authFilter>
```

排除模式

以下示例显示如何在 requestUrl 元素中使用值的管道列表。与列表中的任何模式匹配足以满足该特定元素的要求。在此示例中，请求 URL 必须包含 "/SimpleServlet"、"/EmployeeRoleServlet" 或 "/AllRoleServlet"。此外，请求 URL 不得包含 "/ManagerRoleServlet"，并且请求必须来自 Internet Explorer 用户代理。

```
<authFilter id="myAuthFilter">
        <requestUrl id="myURL1" urlPattern="/SimpleServlet|/EmployeeRoleS
ervlet|/AllRoleServlet" matchType="contains" />
        <requestUrl id="myURL2" urlPattern="/ManagerRoleServlet" matchTyp
e="notContain" />
        <userAgent id="myAgent" agent="IE" matchType="contains" />
</authFilter>
```

使用所有子元素的示例

要使用配置为使用此过滤器的服务处理传入请求，请求必须满足以下条件：

- 在请求 URL 中包含模式 "/SimpleServlet"
- 以包含 "host.example.com" 的域为目标
- 来自 IP 地址 127.0.0.1
- 来自 Firefox 浏览器

- 目标应用程序的名称为 myApp

```
<authFilter id="myAuthFilter">
         <requestUrl id="myRequestUrl" urlPattern="/SimpleServlet" matchTy
pe="contains"/>
         <host id="myHost" name="host.example.com" matchType="contains"/>
         <remoteAddress id="myAddress" ip="127.0.0.1" matchType="equals" /
>
         <userAgent id="myUserAgent" agent="Firefox" matchType="equals"/>
         <webApp id="myWebApp" name="myApp" matchType="contains"/>
</authFilter>
```

2.6.4 授予对 xigemaAS 中资源的访问权

授权的目的是确定用户或组是否有必要的权限来访问资源。

要了解 xigemaAS 中授权的工作原理，请参阅 授权（见第 1055 页）。

在 xigemaAS 上为应用程序配置授权

为应用程序配置授权是要验证用户或组是否属于指定的角色，以及此角色是否具有访问资源的权限。

xigemaAS 服务器从用户注册表中抽取用户和组映射信息，然后检查应用程序的授权配置，以确定是否已将用户或组指定给所需角色。然后，服务器会读取应用程序的部署描述符，以确定用户或组是否具有访问资源的权限。

1. 在 server.xml 文件中启用 appSecurity-2.0 xigemaAS 功能部件。

   例如：

   ```
   <featureManager>
       <feature>appSecurity-2.0</feature>
   </featureManager>
   ```

2. 在 xigemaAS 服务器上配置用于认证的用户注册表。

   请参阅在 *xigemaAS* 中认证用户（见第 1293 页）。

3. 确保应用程序的部署描述符包含安全性约束及其他安全性相关信息。

   注：您也可以使用 Rational® Application Developer 等工具来创建部署描述符。

4. 配置授权信息，例如用户和组到角色的映射。

   可以采用下列方式来配置和授权表：

   - 如果具有 EAR 文件，那么可以将授权配置定义添加到 ibm-application-bnd.xml 或 ibm-application-bnd.xmi 文件。
   - 如果具有独立 WAR 文件，那么可以将授权表定义添加到 server.xml 文件中相应的应用程序元素下。

   注：

   - 如果具有 EAR 文件，那么授权配置可能已存在。在已写入当前规范的 EAR 文件中，此信息存储在 ibm-application-bnd.xml 文件中；在较旧的 EAR 文件中，此信息存储在 ibm-application-bnd.xmi 文件中。
   - 如果 EAR 文件尚不含 ibm-application-bnd.xm* 文件，那么由于创建该文件的任务不是很直观，您可能更愿意将授权配置添加到 server.xml 文件。

- 如果 EAR 文件的授权配置是在 ibm-application-bnd.xm* 文件中进行定义，并且也在 server.xml 文件中进行定义，那么会将这两个表合并。如果存在任何冲突，那么将使用 server.xml 文件中的信息。
- 如果修改用户注册表，请确保复审授权表以进行必要的更改。例如，如果要指定 access-id 元素并更改注册表的域名，那么也必须在 access-id 元素中更改域名。
- 如果在 server.xml 文件中指定 application-bnd 元素，那么您的应用程序不得位于 dropins 文件夹中。如果您的应用程序保留在 dropins 文件夹中，那么必须通过在 server.xml 文件中设置下列项目来禁用应用程序监视：

```
<applicationMonitor dropinsEnabled="false" />
```

可以将角色映射到用户、组或特殊主体集。这两种特殊主体集是 EVERYONE 和 ALL_AUTHENTICATED_USERS。将角色映射到 EVERYONE 特殊主体集时，没有任何安全性，因为每个人都可以访问，而且不会提示您输入凭证。将角色映射到 ALL_AUTHENTICATED_USERS 特殊主体集时，应用程序服务器已认证的任何用户随后都可以访问受保护资源。

下面的代码举例说明了在 server.xml 文件中配置用户和组到角色的映射：

```
<application type="war" id="myapp" name="myapp" location="${server.config.dir}/apps/
myapp.war">
 <application-bnd>
    <security-role name="user">
      <group name="students" />
  </security-role>
    <security-role name="admin">
    <user name="gjones" />
            <group name="administrators" />
  </security-role>
  <security-role name="AllAuthenticated">
   <special-subject type="ALL_AUTHENTICATED_USERS" />
  </security-role>
 </application-bnd>
</application>
```

在此示例中，admin 角色映射到用户标识 gjones 及组 administrators 中的所有用户。AllAuthenticatedRole 映射到特殊主体集 ALL_AUTHENTICATED_USERS，这表示任何用户只要提供有效的凭证进行认证，就具有访问权。

OAuth

OAuth 是委派授权的开放式标准。使用 OAuth 授权框架，用户可以授予第三方应用程序对使用其他 HTTP 服务来存储的信息的访问权，而不必共享其访问许可权或全部数据。

在 OAuth 中，客户机或第三方应用程序会请求对资源所有者所控制及资源服务器所主管的资源的访问权，并且会获发放不同于资源所有者的凭证集。客户机不是使用资源所有者的凭证来访问受保护资源，而是获取访问令牌，该令牌是一个表示特定作用域、生存期及其他访问属性的字符串。授权服务器会在资源所有者的批准之下将访问令牌发放给第三方客户机。客户机使用访问令牌来访问资源服务器所主管的受保护资源。

OAuth 2.0 与 OAuth 1.0 不兼容。OAuth 2.0 可让客户机应用程序开发者轻松地使用，并为不同类型的客户机应用程序提供授权流程。

xigemaAS 支持 OAuth 2.0，而且可用作 OAuth 服务提供程序端点和 OAuth 保护的资源实施端点。

xigemaAS 支持下列 OAuth 标准规范：

- OAuth 2.0 授权框架
- OAuth 2.0 授权框架：不记名令牌用法

以下列表显示了 xigemaAS OAuth 2.0 服务中的功能汇总。

- xigemaAS 充当 OAuth 服务提供程序 (SP) 来处理 OAuth 2.0 协议请求。
- xigemaAS 充当受保护的资源实施端点来授权或拒绝对已部署 Web 资源的请求。
- 允许多个服务提供程序共存。
- 允许管理员撤销访问令牌。
- 允许客户机撤销用户给予客户机的授权。
- 可以选择性地为资源应用程序提供主体（Subject），以进行认证的下游调用或执行程序化 J2EE 安全性。
- 支持如协议中所定义的 4 种典型的 OAuth 2.0 流程。
- 支持持久 OAuth 服务。

OAuth 2.0 服务

xigemaAS OAuth 服务包括 OAuth 授权服务和 Web 资源权限决策服务。

OAuth 2.0 授权服务提供所有 OAuth 2.0 协议端点 URL，并且负责客户机授权和令牌发放。

Web 资源权限决策服务已构建到 xigemaAS Web 认证代码中。当客户机访问 OAuth 保护的 Web 资源时，OAuth 令牌会加以验证并映射到 Web 请求据以运行的 xigemaAS 平台安全性主体。

定义 OAuth 服务提供程序

OAuth 服务提供程序是指定用于 OAuth 的配置选项集。在对授权和令牌端点的入站请求的 URL 中指定提供程序的 `id` 或名称。处理请求时将使用该提供程序的配置选项集。此过程允许一个带有一个端点 servlet 的服务器有效地提供多项 OAuth 配置。例如，使用定义给 OAuth 提供程序 `photoShare` 的 OAuth 配置选项集来处理 `https://my.company.com:8021/oauth2/endpoint/photoShare/authorize` URL。使用定义给 OAuth 提供程序 `calendarAuthz` 的 OAuth 配置选项集来处理 `https://my.company.com:8021/oauth2/endpoint/calendarAuthz/authorize` URL。

使用 server.xml 文件中的 `oauthProvider` 元素来定义 OAuth 服务提供程序。要定义 OAuth 服务提供程序，您可以编辑 server.xml 文件。本任务描述如何定义最低 OAuth 配置。

1. 添加 `oauth-2.0` 和 `ssl-1.0` 功能部件。

   OAuth 是安全协议，因此需要 SSL。在 xigemaAS 上，您必须使用 `keyStore` 元素来为 SSL 提供密钥库密码。没有缺省密钥库密码。

   ```
   <featureManager>
     <feature>oauth-2.0</feature>
     <feature>ssl-1.0</feature>
   </featureManager>
   ```

2. 使用 `oauth-roles` 元素为 OAuth Web 应用程序设置角色映射。

   OAuth 是基于 HTTP 的协议，而且提供了 Web 应用程序来处理授权和令牌端点。该 Web 应用程序是内置的，而且会在您指定 `oauth-2.0` 功能部件时自动启动。但是，必须将**已认证的**角色映射到一个或多个用户、组或特殊主体。提供了另一个角色 `clientManager` 来管理客户机配置，但要让 OAuth 授权正常工作，映射该角色不是必要的。

   ```
   <oauth-roles>
     <authenticated>
       <user>testuser</user>
     </authenticated>
   </oauth-roles>
   ```

3. 使用 `oauthProvider` 元素来定义一个或多个提供程序。

提供程序必须至少定义了一个客户机。可以在本地使用 localStore 和 client 元素来定义客户机。此外，也可以使用 databaseStore 元素在关系数据库中定义客户机。

```
<oauthProvider id="SampleProvider" filter="request-url%=ssodemo">
  <localStore>
    <client name="client01" secret="{xor}LDo8LTor"
            displayname="Test client number 1"
            redirect="http://localhost:1234/oauthclient/redirect.jsp"
            enabled="true" />
  </localStore>
</oauthProvider>
```

4. 定义用户注册表，可为 LDAP 注册表（通过指定 ldapRegistry-3.0 功能部件和 ldapRegistry 配置元素来定义）或者基本注册表（通过指定 basicRegistry 配置元素来定义）。

```
<basicRegistry id="basic" realm="BasicRealm">
  <user name="testuser" password="testuserpwd" />
</basicRegistry>
```

5. 将 allowFailOverToBasicAuth Web 应用程序安全性属性设为 true。

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

您已定义最低 OAuth 配置。

以下示例显示了样本 server.xml 文件，该文件使用一个客户机来定义简单的 OAuth 提供程序：

```
<server>

  <featureManager>
    <feature>oauth-2.0</feature>
    <feature>ssl-1.0</feature>
  </featureManager>

  <keyStore password="keyspass" />

  <oauth-roles>
    <authenticated>
      <user>testuser</user>
    </authenticated>
  </oauth-roles>

  <oauthProvider id="SampleProvider" filter="request-url%=ssodemo">
    <localStore>
      <client name="client01" secret="{xor}LDo8LTor"
              displayname="Test client number 1"
              redirect="http://localhost:1234/oauthclient/redirect.jsp"
              enabled="true" />
    </localStore>
  </oauthProvider>

  <webAppSecurity allowFailOverToBasicAuth="true" />

  <basicRegistry id="basic" realm="BasicRealm">
    <user name="testuser" password="testuserpwd" />
  </basicRegistry>

</server>
```

配置自动授权

确保已通过执行定义 *OAuth* 服务提供程序中的步骤来启用了 OAuth 2.0 功能部件及配置了 OAuth 服务提供程序。

要在未经资源所有者批准的情况下授权 OAuth 客户机，请启用 xigemaAS OAuth 服务提供程序的自动授权功能。

1. 在 server.xml 文件中，使用 `autoAuthorizeParam` 属性，及 `<oauthProvider>` 元素的 `<autoAuthorizeClient>` 子元素来配置自动许可：

```
<oauthProvider id="OAuthConfigSample" autoAuthorizeParam="autoauthz" ...>
  ...
  <autoAuthorizeClient>client01</autoAuthorizeClient>
  <autoAuthorizeClient>client02</autoAuthorizeClient>
</oauthProvider>
```

`client01` 和 `client02` OAuth 客户机已针对自动授权进行配置。

OAuth 端点 URL

在启用 OAuth 2.0 之后，会在 xigemaAS 上配置几个端点 URL，以便 OAuth 客户机能在访问 OAuth 保护的资源之前与 OAuth 服务提供程序通信。

为 OAuth 服务提供程序配置了下列端点 URL：

- 授权端点 URL

  ```
  https://host_name:port_number/oauth2/endpoint/provider_name/authorize
  ```

  其中

  - `host_name` 是 OAuth 服务提供程序的主机名。
  - `port_number` 是 xigemaAS 上配置的安全端口号。
  - `provider_name` 是 OAuth 提供程序名称。
- 令牌端点 URL

  ```
  https://host_name:port_number/oauth2/endpoint/provider_name/token
  ```

  其中

  - `host_name` 是 OAuth 服务提供程序的主机名。
  - `port_number` 是 xigemaAS 上配置的安全端口号。
  - `provider_name` 是 OAuth 提供程序名称。
- 基于信任关联拦截器 (TAI) 的用户认证的授权端点 URL

  ```
  https://host_name:port_number/oauth2/declaritiveEndpoint/provider_name/authorize
  ```

  其中

  - `host_name` 是 OAuth 服务提供程序的主机名。
  - `port_number` 是 xigemaAS 上配置的安全端口号。
  - `provider_name` 是 OAuth 提供程序名称。

  通过使用此授权端点，适用的用户认证会包含 TAI。

OAuth 2.0 服务调用

已注册的 OAuth 客户机可以调用xigemaAS OAuth 服务授权端点来请求访问令牌。已注册的 OAuth 客户机还可以调用 xigemaAS OAuth 服务令牌端点来请求访问令牌。客户机随后可以使用访问令牌从 xigemaAS 请求受保护的 Web 资源。

xigemaAS OAuth 2.0 服务支持下列流程。

授权代码流程

调用授权端点来请求授权代码。

OAuth 客户机会将资源所有者或用户重定向至 xigemaAS OAuth 2.0 授权服务，方法是添加其客户机标识、客户机密钥、状态、重定向 URI 以及可选作用域。

```
https://host_name:port_number/oauth2/endpoint/provider_name/authorize
```

或

```
https://host_name:port_number/oauth2/declarativeEndpoint/provider_name/authorize
```

调用 OAuth 令牌端点来请求访问令牌。

OAuth 客户机通过添加 `authorization_code` 授权类型、`authorization code`、`redirect_url` 和 `client_id` 作为请求参数来实现从 xigemaAS OAuth 2.0 令牌端点请求访问令牌。

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

以下示例说明了使用授权代码时构造 URI 以及使用访问令牌来访问 Web 资源：

```
String charset = "UTF-8";
String param1 = "code";

if (isAuthorizationCode){
  String query = String.format("response_type=%s&
                               client_id=%s&
                               client_secret=%s&
                               state=%s&
                               redirect_uri=%s&
                               scope=%s",
                               URLEncoder.encode(param1, charset),
                               URLEncoder.encode(clientId, charset),
                               URLEncoder.encode(clientSecret, charset),
                               URLEncoder.encode(state, charset),
                               URLEncoder.encode(redirectURI, charset),
                               URLEncoder.encode(scope, charset));
  String s = authorizationEndPoint + "?" + query;
  System.out.println("Visit: " + s + "\nand grant permission");
  System.out.print("Now enter the OAuth code you have received in redirect uri :");
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  String code = br.readLine();
  param1 = "authorization_code";
  query = String.format("grant_type=%s&
                         code=%s&
                         client_id=%s&
                         client_secret=%s&
                         state=%s&
                         redirect_uri=%s&
                         scope=%s",
                         URLEncoder.encode(param1, charset),
                         URLEncoder.encode(code, charset),
                         URLEncoder.encode(clientId, charset),
                         URLEncoder.encode(clientSecret, charset),
                         URLEncoder.encode(state, charset),
                         URLEncoder.encode(redirectURI, charset),
                         URLEncoder.encode(scope, charset));
  URL url = new URL(tokenEndPoint);
  HttpsURLConnection con = (HttpsURLConnection)url. openConnection();
  con.setRequestProperty("Content-Type", "application/x-www-form-urlencoded;charset="  +
 charset);
  con.setDoOutput(true);
  con.setRequestMethod("POST");
  OutputStream output = null;
        try {
    output = con.getOutputStream();
    output.write(query.getBytes(charset));
    output.flush();
```

```
  } finally {
    if (output != null) try {
      output.close();
    } catch (IOException logOrIgnore) {}
  }
  con.connect();
  System.out.println("response message is = " + con.getResponseMessage());
  // read the output from the server
  BufferedReader reader = null;
  StringBuilder stringBuilder;
  reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
  stringBuilder = new StringBuilder();
  String line = null;
        try {
    while ((line = reader.readLine()) != null) {
      stringBuilder.append(line + "\n");
    }
  } finally {
    if (reader != null) try {
      reader.close();
    } catch (IOException logOrIgnore) {}
  }
  String tokenResponse = stringBuilder.toString();
  System.out.println ("response is = " + tokenResponse);
  JSONObject json = JSONObject.parse(tokenResponse);
  if (json.containsKey("access_token")) {
    accessToken = (String)json.get("access_token");
    this.accessToken = accessToken;
  }
  if (json.containsKey("refresh_token")) {
    refreshToken = (String)json.get("refresh_token");
  }
  //sendRequestForAccessToken(query);
  if (accessToken != null) {
    String query = String.format("access_token=%s",
                                 URLEncoder.encode(accessToken, charset));
    URL urlResource = new URL(resourceEndPoint);
    HttpsURLConnection conn = (HttpsURLConnection) urlResource.openConnection();
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Content-type", "application/x-www-form-urlencoded");
    conn.setDoOutput(true);
    output = null;
          try {
      output = conn.getOutputStream();
      output.write(query.getBytes(charset));
      output.flush();
    } finally {
      if (output != null) try {
        output.close();
      } catch (IOException logOrIgnore) {}
    }
    conn.connect();
    System.out.println("response to the resource request is = " + conn.getResponseMessage ());
    reader = null;
    if(conn.getResponseCode()>=200 && conn.getResponseCode() < 400) {
      reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      stringBuilder = new StringBuilder();
      String line = null;
            try {
        while ((line = reader.readLine()) != null) {
          stringBuilder.append(line + "\n");
        }
      } finally {
        if (reader != null) try {
          reader.close();
        } catch (IOException  logOrIgnore) {}
      }
      System.out.println ("response message to the request resource is = " +
 stringBuilder.toString());
    } else {
      isValidResponse = false;
    }
  }
}
```

隐式授权流程

OAuth 客户机从 xigemaAS OAuth 2.0 授权端点请求访问令牌，方法是添加令牌 `response_type`、`redirect_url`、`client_id`、`scope` 和 `state` 作为请求参数。

```
https://host_name:port_number/oauth2/endpoint/provider_name/authorize
```

或

```
https://host_name:port_number/oauth2/declarativeEndpoint/provider_name/authorize
```

以下示例显示使用隐式授权时 URI 的构造：

```
if (isImplicit) {
  param1 = "token";
  String query = String.format("response_type=%s&
                               client_id=%s&
                               state=%s&
                               redirect_uri=%s&
                               scope=%s",
                               URLEncoder.encode(param1, charset),
                               URLEncoder.encode(clientId, charset),
                               URLEncoder.encode(state, charset),
                               URLEncoder.encode(redirectURI, charset),
                               URLEncoder.encode(scope, charset));
  String s = authorizationEndPoint + "?" + query;
  System.out.println("Visit: " + s + "\nand grant permission");
  System.out.print("Now enter the access token you have received in redirect uri :");
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  accessToken = br.readLine();
  if (accessToken != null) {
    // send Resource Request using the access token
  }
}
```

客户机凭证流程

OAuth 客户机使用客户机标识和客户机密钥来访问令牌端点，并交换访问令牌以用于将来的资源请求。在此流程中，客户机访问令牌端点，方法是添加 `client_credentials` 授权类型、`client_id` 和 `client_secret` 作为请求参数。

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

以下示例显示使用客户机凭证时 URI 的构造：

```
if (isClientCredentials){
  param1 = "client_credentials";
  String query = String.format("grant_type=%s&
                               scope=%s&
                               client_id=%s&
                               client_secret=%s",
                               URLEncoder.encode(param1, charset),
                               URLEncoder.encode(scope, charset),
                               URLEncoder.encode(clientId, charset),
                               URLEncoder.encode(clientSecret, charset));
  accessToken = sendRequestForAccessToken(query);
  if (accessToken != null) {
    //send Resource Request using (accessToken);
  }
}
```

资源所有者密码凭证流程

资源所有者密码凭证流程将资源所有者的用户标识和密码直接传递到令牌端点。在此流程中，OAuth 客户机访问令牌端点，方法是添加 `password` 授权类型、`client_id`、`client_secret`、`username`、`password`、`scope` 和 `state` 作为请求参数。

```
https://host_name:port_number/oauth2/endpoint/provider_name/token
```

以下示例显示使用资源所有者密码时 URI 的构造：

```
if (isResourceOwnerCredentials) {
  param1 = "password";
  String query = String.format("grant_type=%s&
                               username=%s&
                               password=%s&
                               scope=%s&
                               client_id=%s&
                               client_secret=%s",
                               URLEncoder.encode(param1, charset),
                               URLEncoder.encode(resOwnerName, charset),
                               URLEncoder.encode(resOwnerPassword, charset),
                               URLEncoder.encode(scope, charset),
                               URLEncoder.encode(clientId, charset),
                               URLEncoder.encode(clientSecret, charset));
  accessToken = sendRequestForAccessToken(query);
  if (accessToken != null) {
    //send Resource Request using (accessToken);
  }
}
```

如果访问令牌到期，那么可以发送刷新令牌以获取有效的访问令牌。以下示例显示如何发送刷新令牌：

```
if(isAccessToken) {
  if (this.accessToken != null) {
    if (!sendResourceRequest(this.accessToken)) {
      // resource request failed...
      //get refresh token
      param1 = "refresh_token";
      String query = String.format("grant_type=%s&
                                   client_id=%s&
                                   client_secret=%s&
                                   refresh_token=%s&
                                   scope=%s",
                                   URLEncoder.encode(param1, charset),
                                   URLEncoder.encode(clientId, charset),
                                   URLEncoder.encode(clientSecret, charset),
                                   URLEncoder.encode(this.refreshToken, charset),
                                   URLEncoder.encode(scope, charset));
      accessToken = sendRequestForAccessToken(query);
      if (accessToken != null) {
        sendResourceRequest(accessToken);
      }
    }
  }
}
```

定制 OAuth 提供程序

xigemaAS OAuth 服务提供程序具有用于定制的插件点。可以替换用于用户认证的缺省表单登录页面，或者开发您自己的用户许可表单来收集客户机授权数据。xigemaAS OAuth 提供程序也通过使用 mediator 来允许对 OAuth 令牌发放中的主要事件进行定制后处理。

定制中介者（Mediator）

在进行 OAuth 2.0 消息处理以执行定制后处理期间，将 OAuth 2.0 中介者用作回调。

编写 OAuth 2.0 中介者

要编写中介者，必须实现名称为 com.ibm.oauth.core.api.oauth20.mediator.OAuth20Mediator 的接口。您可以实现下列其中一个或多个方法来执行定制后处理。

```
void init(OAuthComponentConfiguration config)
```

创建此对象的实例时，工厂会调用此方法。

```
void mediateAuthorize(AttributeList attributeList)
```

进行基本的消息验证和处理之后，核心组件会调用此方法，以允许 processAuthorization 方法中的组件使用者进行任何定制后处理。

```
void mediateAuthorizeException(AttributeList attributeList, OAuthException exception)
```

发生协议异常时，核心组件会调用此方法，以允许 processAuthorization 方法中的组件使用者进行任何定制后处理。

```
void mediateResource(AttributeList attributeList)
```

进行基本的消息验证和处理之后，核心组件会调用此方法，以允许 processResourceRequest 方法中的组件使用者进行任何定制后处理。

```
void mediateResourceException(AttributeList attributeList, OAuthException exception)
```

发生协议异常时，核心组件会调用此方法，以允许 processResourceRequest 方法中的组件使用者进行任何定制后处理。

```
void mediateToken(AttributeList attributeList)
```

进行基本的消息验证和处理之后，核心组件会调用此方法，以允许 processTokenRequest 方法中的组件使用者进行任何定制后处理。

```
void mediateTokenException(AttributeList attributeList, OAuthException exception)
```

发生协议异常时，核心组件会调用此方法，以允许 processTokenRequest 方法中的组件使用者进行任何定制后处理。

对 OAuth 提供程序启用 OAuth 2.0 中介者

要将定制中介者添加到特定 OAuth 2.0 服务提供程序，请在 server.xml 文件中更新提供者定义。添加 oauthProvider 元素的 mediatorClassname 属性，并为中介者指定类名。您也可以使用 oauthProvider 元素的 mediatorClassname 子元素来为中介者指定多个类名。如果指定多个中介者，那么会以指定中介者的顺序来启动那些中介者。您还必须定义包含中介者类的 library 元素，并使用 libraryRef 属性来引用该 library 元素。

以下示例会显示 server.xml 文件中提供者定义内的样本定制中介者条目：

```
<oauthProvider id="OAuthConfigSample" libraryRef="myLib"
  mediatorClassname="com.ibm.ws.security.oauth20.mediator.ResourceOwnerValidationMediator" ...>
  ...
```

```
</oauthProvider>

<library id="myLib">
  <fileset dir="C:\mydir" includes="myLib.jar" />
</library>
```

以下代码样本通过在资源所有者密码凭证流程中使用 xigemaAS 用户注册表来实现凭证验证。

```
package com.ibm.ws.security.oauth20.mediator;

import com.ibm.oauth.core.api.attributes.AttributeList;
import com.ibm.oauth.core.api.config.OAuthComponentConfiguration;
import com.ibm.oauth.core.api.error.OAuthException;
import com.ibm.oauth.core.api.error.oauth20.OAuth20MediatorException;
import com.ibm.oauth.core.api.oauth20.mediator.OAuth20Mediator;
import com.ibm.oauth.core.internal.oauth20.OAuth20Constants;
import com.ibm.websphere.security.CustomRegistryException;
import com.ibm.websphere.security.PasswordCheckFailedException;
import com.ibm.websphere.security.UserRegistry;

import java.rmi.RemoteException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ResourceOwnerValidationMedidator implements OAuth20Mediator {
  private static final String CLASS = ResourceOwnerValidationMedidator.class.getName();
  private static final Logger LOG = Logger.getLogger(CLASS);
  private UserRegistry reg = null;

  public void init(OAuthComponentConfiguration config) {
         try {
      InitialContext ctx = new InitialContext();
      reg = (UserRegistry) ctx.lookup("UserRegistry");
    } catch(NamingException ne) {
      LOG.log(Level.SEVERE, "Cannot lookup UserRegistry", ne);
    }
  }

  public void mediateAuthorize(AttributeList attributeList)
    throws OAuth20MediatorException {
    // nothing to do here
  }

  public void mediateAuthorizeException(AttributeList attributeList,
                                         OAuthException exception)
    throws OAuth20MediatorException {
    // nothing to do here
  }

  public void mediateResource(AttributeList attributeList)
    throws OAuth20MediatorException {
    // nothing to do here
  }

  public void mediateResourceException(AttributeList attributeList,
                                       OAuthException exception)
    throws OAuth20MediatorException {
    // nothing to do here
  }

  public void mediateToken(AttributeList attributeList)
    throws OAuth20MediatorException {
    final String methodName = "mediateToken";
    LOG.entering(CLASS, methodName, attributeList);
    if("password".equals(attributeList.getAttributeValueByName("grant_type"))) {
      String username = attributeList.getAttributeValueByName("username");
      String password = attributeList.getAttributeValueByName("password");
           try {
        reg.checkPassword(username, password);
      } catch (PasswordCheckFailedException e) {
        throw new OAuth20MediatorException("User doesn't exist or the
```

```
                                          password doesn't match.", e);
      } catch (CustomRegistryException e) {
        throw new OAuth20MediatorException("Cannot validate resource owner.", e);
      } catch (RemoteException e) {
        throw new OAuth20MediatorException("Cannot validate resource owner.", e);
      }
    }
    LOG.exiting(CLASS, methodName);
  }

  public void mediateTokenException(AttributeList attributeList,
                                    OAuthException exception) throws OAuth20MediatorException {
    final String methodName = "mediateTokenException";
    LOG.entering(CLASS, methodName, new Object[] {attributeList, exception});
    if("password".equals(attributeList.getAttributeValueByName("grant_type"))) {
      // clear sensitive data
      attributeList.setAttribute("access_token",
                                 OAuth20Constants.ATTRTYPE_RESPONSE_ATTRIBUTE,
                                 new String[0]);
      attributeList.setAttribute("refresh_token",
                                 OAuth20Constants.ATTRTYPE_RESPONSE_ATTRIBUTE,
                                 new String[0]);
    }
    LOG.exiting(CLASS, methodName);
  }

}
```

定制许可表单模板

OAuth 授权服务器提供一个模板，用以获取有关获授权来访问给定作用域中受保护资源的 OAuth 客户机的用户许可信息。来自 OAuth 客户机的授权请求包含了模板中所请求作用域的列表。

xigemaAS 允许许可表单模板是静态 HTML 页面或动态网页。在两种情况下，都必须提供该模板作为未受保护的 Web 资源。在访问此模板 URL 时，xigemaAS 集成中的表单检索器并不执行任何认证。

xigemaAS OAuth 提供程序随附样本许可表单模板，而且允许使用 `oauthFormData` 变量进行定制。

要定制许可表单，必须使用 JavaScript 来编辑 `oauthFormData` 变量。表单数据中包含下列变量：

- `authorizationUrl` - 要提交表单的授权 URL
- `clientDisplayName` - 客户机的显示名称
- `nonce` - 用以阻止跨网站请求伪造 (CSRF) 的随机生成号码
- `client_id` - 请参阅 OAuth 2.0 规范
- `response_type` - 请参阅 OAuth 2.0 规范
- `redirect_uri` - 请参阅 OAuth 2.0 规范
- `state` - 请参阅 OAuth 2.0 规范
- `scope` - 请参阅 OAuth 2.0 规范

表单模板的开发者必须通过使用 JavaScript 来包含 `oauthFormData` 变量的内容。开发者必须将作用域值解释成对用户有意义的值。当用户授权请求时，开发者可以调用 `submitForm(oauthFormData)` 方法来执行授权。缺省情况下，会提供 `submitForm` 方法。但是，如果开发者熟悉 OAuth 2.0 协议，那么开发者可以通过实现他们自己的功能来提交 OAuth 授权请求。

`cancel(oauthFormData)` 方法是缺省情况下提供的，可用于允许用户取消授权请求。

还可修改同意表单以允许缓存用户的同意选择。这意味着，如果同一 OpenID Connect 客户机发出新的授权请求（具有相同批准范围或缩减范围），那么系统不会提示用户提供同意表单。反而先前允许的范围被视为已授权并相应传递至受保护资源。

如果客户机注册处于 localStore 方式，那么用户的同意选择缓存在浏览器会话中。系统会缓存给定用户的已批准范围，直到会话关闭或一定量的时间（在服务器配置中指定）已过去。

如果客户机注册处于 databaseStore 方式，那么用户的同意选择可保存在数据库表 OAuthDBSchema.OAUTH20CONSENTCACHE 中。系统会缓存给定用户的范围，直到一定量的时间（在服务器配置中指定）已过去。OpenID Connect 提供程序尝试自动创建同意缓存表，但建议为 OAuth2.0 提供程序和 OpenID Connect 提供程序配置数据库时用户显式创建同意表，有关进一步详细信息，请参阅持久 *OAuth* 服务配置。

要使用此功能，表单模板的开发者必须将 prompt 值包含在 oauthFormData JavaScript 对象中。为缓存用户的肯定响应并阻止同一会话中再次显示同意表单，prompt 值设置为字符串 none。为允许用户提交肯定响应而不缓存批准，prompt 值设置为字符串 consent。

可以使用动态页面，根据请求中的 Accept-Language 头来返回全球化内容。检索模板时，会转发 Accept-Language 头，并且模板开发者必须决定要返回哪些有关首选语言的内容。

👉 注：clientDisplayName 变量不会在 HTML 中进行转义。模板开发者必须净化值，因为值是由用户在客户机注册期间输入。

要将定制同意表单模板页面用于特定 OAuth 2.0 服务提供程序，您必须在 server.xml 文件中更新服务提供程序定义。在提供程序配置中，必须使用 oauthProvider 元素的 authorizationFormTemplate 属性，并将模板 URL 添加为值。以下示例显示了提供程序配置中的样本模板条目：

```
<oauthProvider id="OAuthConfigSample"
  authorizationFormTemplate="https://acme.com:9043/oath20/template.html
  ...>
```

以下示例说明了样本同意表单：

```
<oauthProvider id="OAuthConfigSample"
  authorizationLoginURL="https://acme.com:9043/oath20/login.jsp"
  ...>

function escapeHTML(str) {
    var ele = document.createElement("div");
    ele.innerText = ele.textContent = str;
    return ele.innerHTML;
}
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>OAuth authorization form</title>
<script language="javascript">
function init() {
  var scope = oauthFormData.scope;
  var scopeEle = document.getElementById("oauth_scope");
  var ul = document.createElement("ul");
  if(scope) {
    for(var i=0; i< scope.length; i++) {
      var n = document.createElement("li");
      n.innerHTML = scope[i];
      ul.appendChild(n);
  }
 }
  scopeEle.appendChild(ul);
  // set client name
  var clientEle = document.getElementById("client_name");
  clientEle.innerHTML = escapeHTML(oauthFormData.clientDisplayName);
}

function escapeHTML(str) {
    var ele = document.createElement("div");
    ele.innerText = ele.textContent = str;
```

```
    return ele.innerHTML;
}
</script>
</head>
<body onload="init()">
       <div>Do you want to allow client <span id=client_name style="font-weight:bold">xxxxxxx</
span> to access your data?</div>
       <div id="oauth_scope">
       </div>
       <div>
              <form action="javascript:submitForm(oauthFormData);">
                     <input type="submit" value="Allow, remember my decision"
 onclick="javascript:oauthFormData.prompt = 'none';"/>
                     <input type="submit" value="Allow once"
 onclick="javascript:oauthFormData.prompt = 'consent';"/>
                     <input type="button" value="Cancel"
 onclick="javascript:cancel(oauthFormData);"/>
              </form>
       </div>
</body>
</html>
```

定制用户登录表单

xigemaAS OAuth 服务提供程序包含一个表单登录页面，供用户提交用户名和密码。

您可以定制您自己的表单登录页面，但它必须根据 servlet 规范中基于表单的认证中的需要进行实现。在此表单中，操作必须是 `j_security_check`，并且使用 `j_username` 输入字段来获取用户名。该操作也必须使用 `j_password` 输入字段来获取用户密码。定制表单登录页面必须作为未受保护的 Web 资源来提供。

要将定制表单登录页面用于特定 OAuth20 服务提供程序，您必须在 server.xml 文件中更新服务提供程序定义。在提供程序配置中，必须添加 `customLoginURL` 属性并指定登录页面 URL 作为值。

下列是提供程序定义中的示例定制登录页面条目：

```
<oauthProvider id="OAuthConfigSample"
  customLoginURL="https://acme.com:9043/oath20/login.jsp"
  ...>
```

👉 注：请确保表单登录页面中包含的所有文件（例如，外部样式表或图像）都不受保护。

持久 OAuth 服务配置

xigemaAS 通过将 OAuth 令牌和客户机持久存储到数据库来支持持久 OAuth 2.0 服务。借助持久 OAuth 2.0 服务，在 OAuth 服务重新启动之后，已授权的客户机可以访问 OAuth 2.0 服务。

要配置持久 OAuth 2.0 服务，请完成下列步骤：

1. 配置 OAuth 2.0 服务提供程序。

   要使用数据库存储器，必须指定 `<oauthProvider>` 元素的 `<databaseStore>` 子元素。`<databaseStore>` 元素上的唯一必需属性是 `<dataSourceRef>`，其值必须是 `<dataSource>` 元素的标识。

   以下示例是使用 Derby 数据库存储器的 OAuth 提供程序的样本 server.xml 文件：

```
<server>

  <featureManager>
    <feature>oauth-2.0</feature>
    <feature>ssl-1.0</feature>
    <feature>jdbc-4.0</feature>
    <feature>jndi-1.0</feature>
  </featureManager>
```

```
  <keyStore password="keyspass" />

  <oauth-roles>
    <authenticated>
      <user>testuser</user>
    </authenticated>
  </oauth-roles>

  <oauthProvider id="OAuthConfigDerby" filter="request-url%=ssodemo"
                 oauthOnly="false">
    <databaseStore dataSourceRef="OAuthFvtDataSource" />
  </oauthProvider>

  <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib" />

  <library id="DerbyLib" fileSetRef="DerbyFileset" />

  <fileset id="DerbyFileset" dir="${DERBY_JDBC_DRIVER_PATH}"
           includes="derby.jar" />

  <dataSource id="OAuthFvtDataSource" jndiName="jdbc/OAuth2DB"
              jdbcDriverRef="DerbyEmbedded">
    <properties.derby.embedded databaseName="D:\oauth2db"
                               createDatabase="create"/>
  </dataSource>

  <webAppSecurity allowFailOverToBasicAuth="true" />

  <basicRegistry id="basic" realm="BasicRealm">
    <user name="testuser" password="testuserpwd" />
  </basicRegistry>

</server>
```

2. 设置数据库和表以存储 OAuth 令牌和客户机。

   a. 创建用于持久 OAuth 服务的数据库。请参阅供应商文档以了解如何创建数据库。在此示例中，数据库名称是 `D:\oauth2db`。

   b. 通过以下 SQL 语句创建 3 个所定义的 OAuth 表：

```
----- CREATE TABLES -----
CREATE TABLE OAuthDBSchema.OAUTH20CACHE
(
  LOOKUPKEY VARCHAR(256) NOT NULL,
  UNIQUEID VARCHAR(128) NOT NULL,
  COMPONENTID VARCHAR(256) NOT NULL,
  TYPE VARCHAR(64) NOT NULL,
  SUBTYPE VARCHAR(64),
  CREATEDAT BIGINT,
  LIFETIME INT,
  EXPIRES BIGINT,
  TOKENSTRING VARCHAR(2048) NOT NULL,
  CLIENTID VARCHAR(64) NOT NULL,
  USERNAME VARCHAR(64) NOT NULL,
  SCOPE VARCHAR(512) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  STATEID VARCHAR(64) NOT NULL
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
);

CREATE TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
(
  COMPONENTID VARCHAR(256) NOT NULL,
  CLIENTID VARCHAR(256) NOT NULL,
  CLIENTSECRET VARCHAR(256),
  DISPLAYNAME VARCHAR(256) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  ENABLED INT
  CLIENTMETADATA CLOB NOT NULL DEFAULT '{}'
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE
(
```

```
  CLIENTID VARCHAR(256) NOT NULL,
  USERID VARCHAR(256),
  PROVIDERID VARCHAR(256) NOT NULL,
  SCOPE VARCHAR(1024) NOT NULL,
  EXPIRES BIGINT,
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
);

----- ADD CONSTRAINTS -----
ALTER TABLE OAuthDBSchema.OAUTH20CACHE
  ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
  ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

----- CREATE INDEXES -----
CREATE INDEX OAUTH20CACHE_EXPIRES ON OAUTHDBSCHEMA.OAUTH20CACHE (EXPIRES ASC);
```

3. 配置 xigemaAS。

   配置 xigemaAS 数据源。必须将数据源 Java 命名和目录接口 (JNDI) 名称设为 `jdbc/OAuth2DB`。JNDI 名称必须与 server.xml 文件中 `dataSource` 元素的 `<jndiName>` 属性匹配。输入数据库名称，例如，`D:\oauth2db`。

   有关为 OAuth 持久服务配置 DB2® 和 Derby 的更多信息，请参阅将 *IBM® DB2®* 用于持久 *OAuth* 服务和将 *Derby* 数据库用于持久 *OAuth* 服务。您可以将它们用作样本模板来配置其他数据库。

4. 将已注册的 OAuth 客户机添加至数据库。

   要将客户机持久存储在数据库中，您必须将该客户机保存到该数据库中。下列 SQL 语句会将 OAuth 客户机 `dbclient01` 和 `dbclient02` 添加至 Derby 数据库：

```
CONNECT 'jdbc:derby:D:\oauth2db';
INSERT INTO OAuthDBSchema.OAUTH20CLIENTCONFIG VALUES
(
  'OAuthConfigDerby',
  'dbclient01',
  'secret',
  'dbclient01',
  'http://localhost:9080/oauthclient/redirect.jsp',
  1
),
(
  'OAuthConfigDerby',
  'dbclient02',
  'secret',
  'dbclient02',
  'http://localhost:9080/oauthclient/redirect.jsp',
  1
);
DISCONNECT CURRENT;
```

   注：`Componentid` 必须与 server.xml 文件中的 `oauthProvider` 元素的标识相同。

将 Derby 数据库用于持久 OAuth 服务

可以将 Derby 数据库用于持久 OAuth 服务。为了方便起见和供您参考，本主题说明了为 OAuth 持久服务配置 Derby 数据库（相对于 OAuth 服务而言处于远程位置或本地位置）时需要执行的步骤。

要为持久 OAuth 服务配置 Derby 数据库，请完成下列步骤：

1. 创建数据库和表。

   编辑并运行以下 SQL 语句以创建 OAuth 数据库和表：

```
--- Change oauth2db to the name you want for the database
--- Connect to Derby, choose one connection option to uncomment
```

```
--- if connecting to Derby as network server
--- CONNECT 'jdbc:derby://localhost:1527/oauth2db;create=true';

--- if connecting to embedded derby, you can change D:\oauth2db to location of database
--- CONNECT 'jdbc:derby:D:\oauth2db;create=true';

--- if creating tables in existing Derby database, remove the create=true parameter.

----- CREATE TABLES -----
CREATE TABLE OAuthDBSchema.OAUTH20CACHE (
  LOOKUPKEY VARCHAR(256) NOT NULL,
  UNIQUEID VARCHAR(128) NOT NULL,
  COMPONENTID VARCHAR(256) NOT NULL,
  TYPE VARCHAR(64) NOT NULL,
  SUBTYPE VARCHAR(64),
  CREATEDAT BIGINT,
  LIFETIME INT,
  EXPIRES BIGINT,
  TOKENSTRING VARCHAR(2048) NOT NULL,
  CLIENTID VARCHAR(64) NOT NULL,
  USERNAME VARCHAR(64) NOT NULL,
  SCOPE VARCHAR(512) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  STATEID VARCHAR(64) NOT NULL
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
);

CREATE TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG (
  COMPONENTID VARCHAR(256) NOT NULL,
  CLIENTID VARCHAR(256) NOT NULL,
  CLIENTSECRET VARCHAR(256),
  DISPLAYNAME VARCHAR(256) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  ENABLED INT
  CLIENTMETADATA CLOB NOT NULL DEFAULT '{}'
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE (
  CLIENTID VARCHAR(256) NOT NULL,
  USERID VARCHAR(256),
  PROVIDERID VARCHAR(256) NOT NULL,
  SCOPE VARCHAR(1024) NOT NULL,
  EXPIRES BIGINT,
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
);

----- ADD CONSTRAINTS -----
ALTER TABLE OAuthDBSchema.OAUTH20CACHE
  ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
  ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

----- CREATE INDEXES -----
CREATE INDEX OAUTH20CACHE_EXPIRES ON OAUTHDBSCHEMA.OAUTH20CACHE (EXPIRES ASC);

DISCONNECT CURRENT;
```

通过使用以下命令启动 `ij` 来运行 createTables.sql 文件：

```
ij createTables.sql
```

2. 配置 xigemaAS 服务器。

   以下示例是使用 Derby 数据库存储器的 OAuth 提供程序的样本 server.xml 文件：

```
<server>

  <featureManager>
    <feature>oauth-2.0</feature>
    <feature>ssl-1.0</feature>
    <feature>jdbc-4.0</feature>
    <feature>jndi-1.0</feature>
```

```
  </featureManager>

  <keyStore password="keyspass" />

  <oauth-roles>
    <authenticated>
      <user>testuser</user>
    </authenticated>
  </oauth-roles>

  <oauthProvider id="OAuthConfigDerby" filter="request-url%=ssodemo"
                 oauthOnly="false">
    <databaseStore dataSourceRef="OAuthDerbyDataSource" />
  </oauthProvider>

  <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib" />

  <library id="DerbyLib" filesetRef="DerbyFileset" />

  <fileset id="DerbyFileset" dir="${DERBY_JDBC_DRIVER_PATH}"
           includes="derby.jar" />

  <dataSource id="OAuthDerbyDataSource" jndiName="jdbc/OAuth2DB"
              jdbcDriverRef="DerbyEmbedded">
    <properties.derby.embedded databaseName="D:\oauth2db"
                               createDatabase="create"/>
  </dataSource>

  <webAppSecurity allowFailOverToBasicAuth="true" />

  <basicRegistry id="basic" realm="BasicRealm">
    <user name="testuser" password="testuserpwd" />
  </basicRegistry>
</server>
```

注：Componentid 必须与 server.xml 文件中的 oauthProvider 元素的标识相同。

将 IBM® DB2® 用于持久 OAuth 服务

IBM® DB2® 可用于持久 OAuth 服务。为了便利和参考的目的，此主题记录了为 OAuth 持久服务配置 DB2® 所需的步骤。

要为持久 OAuth 服务配置 DB2®，请完成下列步骤：

1. 创建数据库和表。

   编辑并运行以下 SQL 语句以创建 OAuth 数据库和表：

```
-- Change oauth2db to the name you want for the database

CREATE DATABASE oauth2db USING CODESET UTF8 TERRITORY US;
CONNECT TO oauth2db;

---- CREATE TABLES ----
CREATE TABLE OAuthDBSchema.OAUTH20CACHE
(
  LOOKUPKEY VARCHAR(256) NOT NULL,
  UNIQUEID VARCHAR(128) NOT NULL,
  COMPONENTID VARCHAR(256) NOT NULL,
  TYPE VARCHAR(64) NOT NULL,
  SUBTYPE VARCHAR(64),
  CREATEDAT BIGINT,
  LIFETIME INT,
  EXPIRES BIGINT,
  TOKENSTRING VARCHAR(2048) NOT NULL,
  CLIENTID VARCHAR(64) NOT NULL,
  USERNAME VARCHAR(64) NOT NULL,
  SCOPE VARCHAR(512) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  STATEID VARCHAR(64) NOT NULL
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
```

```
);

CREATE TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
(
  COMPONENTID VARCHAR(256) NOT NULL,
  CLIENTID VARCHAR(256) NOT NULL,
  CLIENTSECRET VARCHAR(256),
  DISPLAYNAME VARCHAR(256) NOT NULL,
  REDIRECTURI VARCHAR(2048),
  ENABLED INT
  CLIENTMETADATA CLOB NOT NULL DEFAULT '{}'
);

CREATE TABLE OAuthDBSchema.OAUTH20CONSENTCACHE (
  CLIENTID VARCHAR(256) NOT NULL,
  USERID VARCHAR(256),
  PROVIDERID VARCHAR(256) NOT NULL,
  SCOPE VARCHAR(1024) NOT NULL,
  EXPIRES BIGINT,
  EXTENDEDFIELDS CLOB NOT NULL DEFAULT '{}'
);

---- ADD CONSTRAINTS ----
ALTER TABLE OAuthDBSchema.OAUTH20CACHE
  ADD CONSTRAINT PK_LOOKUPKEY PRIMARY KEY (LOOKUPKEY);

ALTER TABLE OAuthDBSchema.OAUTH20CLIENTCONFIG
  ADD CONSTRAINT PK_COMPIDCLIENTID PRIMARY KEY (COMPONENTID,CLIENTID);

---- CREATE INDEXES ----
CREATE INDEX OAUTH20CACHE_EXPIRES ON OAUTHDBSCHEMA.OAUTH20CACHE (EXPIRES ASC);

---- GRANT PRIVILEGES ----
---- UNCOMMENT THE FOLLOWING IF YOU USE AN ACCOUNT OTHER THAN ADMINISTRATOR FOR DB ACCESS
 ----

-- Change dbuser to the account you want to use to access your database
-- GRANT ALL ON OAuthDBSchema.OAUTH20CACHE TO USER dbuser;
-- GRANT ALL ON OAuthDBSchema.OAUTH20CLIENTCONFIG TO USER dbuser;

---- END OF GRANT PRIVILIGES ----

DISCONNECT CURRENT;
```

缺省 DB2® 侦听端口是 50000。如果您想要查找此端口，请运行以下命令并查找 SVCENAME 参数的值。如果它是一个数字，那么它是端口号。如果是名称，请在 /etc/services 文件中查找名称，或者查找 Windows™ 等价项（如果使用的是 Windows™）。

```
Linux/Unix: db2 get dbm cfg | grep SVCENAME
Windows:    db2 get dbm cfg | findstr SVCENAME
```

可以通过运行下列语句，在 DB2® 中创建数据库和表：

```
db2 -tvf createTables.sql
```

2. 配置 xigemaAS 概要文件服务器。

   以下示例是使用 DB2® 存储器的 OAuth 提供程序的样本 server.xml 文件：

```
<server>
  <featureManager>
    <feature>oauth-2.0</feature>
    <feature>ssl-1.0</feature>
    <feature>jdbc-4.0</feature>
    <feature>jndi-1.0</feature>
  </featureManager>

  <keyStore password="keyspass" />

  <oauth-roles>
    <authenticated>
```

```
      <user>testuser</user>
    </authenticated>
  </oauth-roles>

  <oauthProvider id="DBOAuth20Provider" oauthOnly="true"
                 filter="request-url%=AnnuityOAuthWeb/index.jsp">
    <databaseStore dataSourceRef="OAUTH2DBDS" />
  </oauthProvider>

  <jdbcDriver id="db2Universal" libraryRef="DB2JCC4LIB" />

  <library apiTypeVisibility="spec,ibm-api,third-party" filesetRef="db2jcc4"
           id="DB2JCC4LIB" />

  <fileset dir="${shared.resource.dir}/db2" id="db2jcc4"
           includes="db2jcc4.jar db2jcc_license_cu.jar" />

  <dataStore id="OAUTH2DBDS" jdbcDriverRef="db2Universal"
             jndiName="jdbc/oauthProvider">
    <properties.db2.jcc databaseName="OAUTH2DB" driverType="4"
                        user="bob" password="abcdefg="
                        portNumber="50000"
                        serverName="db2.server.mycompany.com" />
  </dataStore>

  <webAppSecurity allowFailOverToBasicAuth="true" />

  <basicRegistry id="basic" realm="BasicRealm">
    <user name="testuser" password="testuserpwd" />
  </basicRegistry>
</server
```

以下示例将客户机添加到 DB2®：

```
INSERT INTO OAuthDBSchema.OAUTH20CLIENTCONFIG
(
  COMPONENTID,
  CLIENTID,
  CLIENTSECRET,
  DISPLAYNAME,
  REDIRECTURI,
  ENABLED
)
VALUES
(
  'DBOAuth20Provider',
  'key',
  'secret',
  'My Client',
  'https://localhost:9443/oauth/redirect.jsp',
  1
)
```

👉 注：`Componentid` 必须与 server.xml 文件中 `oauthProvider` 元素的标识相同。

2.6.5 在 xigemaAS 中配置公共安全互操作性 V2 (CSIv2)

xigemaAS 支持各种级别的 CSIv2 安全性，例如，消息认证（认证层）、身份断言（属性层）和客户机证书认证（传输层）。通过使用 CSIv2 功能部件，您可以对面向下游服务器的入站和出站请求指定认证类型。如果在 server.xml 文件中配置了 `appSecurity-2.0` 和 `ejbRemote-3.2` 功能部件，那么系统会自动启用 CSIv2 功能部件。可在 xigemaAS 概要文件中配置 CSIv2，以在 Java™ Platform Enterprise Edition 供应商之间启用互操作性。

1. 以下是所使用的缺省配置，如果配置了 appSecurity-2.0 和 ejbRemote-3.2 功能部件，那么不必在 server.xml 文件中指定该配置。

```
<orb id="defaultOrb">
    <serverPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="false"/>
            <authenticationLayer mechanisms="LTPA,GSSUP"
 establishTrustInClient="Required"/>
            <transportLayer/>
        </layers>
    </serverPolicy.csiv2>
    <clientPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="false"/>
            <authenticationLayer mechanisms="LTPA,GSSUP"
 establishTrustInClient="Supported"/>
            <transportLayer/>
        </layers>
    </clientPolicy.csiv2>
</orb>
```

可更改 serverPolicy.csiv2 和 clientPolicy.csiv2 中的每个层以定制入站和出站 CSIv2 设置。

在 xigemaAS 概要文件中配置入站 CSIv2

公共安全互操作性 V2 (CSIv2) 功能部件入站配置确定入站请求的所接受认证类型。如果在 server.xml 文件中配置了 appSecurity-2.0 和 ejbRemote-3.2 功能部件，那么系统会自动启用 CSIv2 功能部件。

要了解 CSIv2 概念，请参阅公共安全互操作性 *V2 (CSIv2)* 以了解更多信息。

以下安全层可用于入站请求：

1. 配置入站 *CSIv2* 属性层。
2. 配置入站 *CSIv2* 认证层。
3. 配置入站 *CSIv2* 传输层。

配置入站 CSIv2 属性层

可配置 xigemaAS 服务器以声明对针对入站 CSIv2 请求的身份断言的支持。

缺省情况下，已对 xigemaAS 概要文件服务器的入站 CSIv2 属性层禁用身份断言。通过 identityAssertionEnabled 属性启用身份断言后，服务器支持通过充当客户机的上游服务器进行主体名称和匿名身份断言。可使用 trustedIdentities 属性指定能够对此服务器断言身份的可信上游服务器的身份。

警告：如果已设置假定信任，请确保只有可信实体与服务器通信。

1. 在 server.xml 文件中添加 appSecurity-2.0 和 ejbRemote-3.2 功能部件。

```
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>ejbRemote-3.2</feature>
    </featureManager>
```

以下是缺省配置，不必在 server.xml 文件中指定该配置。

```
    <orb id="defaultOrb">
        <serverPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
                <transportLayer/>
            </layers>
        </serverPolicy.csiv2>
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                <transportLayer/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

2. 可选：如果需要更改缺省入站属性层配置，请按如下所示在 server.xml 文件中添加 <orb> 元素，或将 attributeLayer 元素添加至现有文件。将示例中的样本值替换为您的值。

```
   <orb id="defaultOrb">
       <serverPolicy.csiv2>
           <layers>
               <attributeLayer identityAssertionEnabled="true"/>
           </layers>
       </serverPolicy.csiv2>
   </orb>
```

   注：<orb> 元素中的 ID 值 defaultOrb 是预先定义的，不能修改。

3. 通过将示例值更改为每个上游服务器的 trustedIdentity 来设置 trustedIdentities 属性。如果有多个断言客户机，那么必须使用管道字符 (|) 分隔值。

```
<attributeLayer identityAssertionEnabled="true" trustedIdentities="yourAssertingUpstreamServer|
anotherAssertingUpstreamServer"/>
```

4. 替代方法：可使用字符 (*) 设置 trustedIdentities 属性以指示该服务器支持假定信任，而不是在步骤 2中对 trustedIdentities 设置指定值。通过使用假定信任，任何上游服务器能够断言身份，并且仅当上游服务器可限定为一组可信服务器时，才必须使用上游服务器。因此，使用此值时应特别谨慎。

```
<attributeLayer identityAssertionEnabled="true" trustedIdentities="*"/>
```

5. 如果发送证书链的上游服务器可信，请将证书链的颁发者专有名称添加至 trustedIdentities 属性。例如，

```
    <attributeLayer identityAssertionEnabled="true"
 trustedIdentities="CN=localhost,O=vsettan,C=cn"/>
```

6. 可选：如果需要更改服务器支持的缺省身份断言令牌类型，应将 identityAssertionTypes 属性添加至 server.xml 文件中的 attributeLayer 元素并指定值的逗号分隔列表。有效值为 ITTAnonymous、ITTPrincipalName、ITTX509CertChain 和 ITTDistinguishedName。例如，可配置服务器以支持带有 X509 证书链或专有名称的身份断言。将示例中的样本值替换为您的值。

```
<orb id="defaultOrb">
    <serverPolicy.csiv2>
```

```
        <layers>
            <attributeLayer identityAssertionEnabled="true"
 identityAssertionTypes="ITTX509CertChain, ITTDistinguishedName"/>
        </layers>
    </serverPolicy.csiv2>
</orb>
```

注：上游服务器身份是从服务器在认证层或传输层中发送的安全信息中获取的。认证层身份优先于传输身份，如果在认证层未发送任何安全信息，那么使用传输身份。有关 authenticationLayer 和 transportLayer 元素的样本语法和更多信息，请参阅配置入站 *CSIv2* 认证层和配置入站 *CSIv2* 传输层。

如果省略层，那么系统对该层使用缺省值。

现在已配置入站 CSIv2 属性层以用于身份断言。

配置入站 CSIv2 认证层

您可以配置 xigemaAS 概要文件服务器以对入站 CSIv2 请求使用特定认证机制。

缺省情况下，已启用 xigemaAS 概要文件服务器的入站 CSIv2 认证层，并具有对 LTPA 和 GSSUP 认证机制的支持。缺省情况下，认证层的 establishTrustInClient 关联选项设置为 Required 以指示所指定认证机制是必需的。使用 LTPA 机制时，确保通信 xigemaAS 概要文件服务器和其他服务器共享相同 LTPA 密钥。

1. 在 server.xml 文件中添加 appSecurity-2.0 和 ejbRemote-3.2 功能部件。

```
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>ejbRemote-3.2</feature>
    </featureManager>
```

以下示例显示缺省配置，不必在 server.xml 文件中指定该配置。

```
    <orb id="defaultOrb">
        <serverPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
 establishTrustInClient="Required"/>
                <transportLayer/>
            </layers>
        </serverPolicy.csiv2>
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
 establishTrustInClient="Supported"/>
                <transportLayer/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

2. 可选：如果需要更改缺省入站认证层配置，请按如下所示在 server.xml 文件中添加 <orb> 元素，或将 authenticationLayer 元素添加至现有文件。将示例中的样本值替换为您的值。

```
    <orb id="defaultOrb">
        <serverPolicy.csiv2>
            <layers>
```

```
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
            </layers>
        </serverPolicy.csiv2>
        </orb>
```

注：<orb> 元素中的 ID 值 defaultOrb 是预先定义的，不能修改。

3. 可选：将 mechanisms 属性设置为 LTPA 或 GSSUP，以仅将 LTPA 或 GSSUP（用户名和密码）用作认证机制。

```
<authenticationLayer mechanisms="LTPA" establishTrustInClient="Supported"/>
或
<authenticationLayer mechanisms="GSSUP" establishTrustInClient="Supported"/>
```

4. 可选：将 establishTrustInClient 属性设置为 Required、Supported 或 Never 以指示服务器需要、支持（可选）或从不声明使用指定机制的认证。

注：

- establishTrustInClient 属性设置为 Required 时，只有需要或支持（至少一个）可兼容认证机制的客户机才能向服务器发送安全上下文。
- establishTrustInClient 属性设置为 Supported 时，客户机可选择是否在认证层中发送认证信息。
- 如果 establishTrustInClient 属性设置为 Never，那么系统禁用入站 CSIv2 认证层，并且必须启用至少一个其他 CSIv2 层以进行认证。

  如果省略层，那么系统对该层使用缺省值。

  有关 attributeLayer 和 transportLayer 元素的更多信息，请参阅配置入站 *CSIv2* 属性层和配置入站 *CSIv2* 传输层。

现在已配置入站 CSIv2 认证层。

配置入站 CSIv2 传输层

可配置 xigemaAS 概要文件服务器以声明对针对入站 CSIv2 请求的客户机证书认证的支持。

缺省情况下，xigemaAS 概要文件服务器的入站 CSIv2 传输层的客户机证书认证已禁用。可配置 transportLayer 以指定要使用的 SSL 配置。您可将 SSL 元素配置为支持或需要客户机证书认证。系统针对服务器用户注册表认证所接收的证书，仅当 CSIv2 请求中未发送任何其他形式的认证（例如，属性层中的身份断言或认证层中的认证令牌）时，才使用其身份。

使用客户机证书认证时，确保此服务器支持 SSL。

1. 在 server.xml 文件中添加 appSecurity-2.0 和 ejbRemote-3.2 功能部件。

```
<featureManager>
    <feature>appSecurity-2.0</feature>
    <feature>ejbRemote-3.2</feature>
</featureManager>
```

以下示例是缺省配置，不必在 server.xml 文件中指定该配置。

```
<orb id="defaultOrb">
    <serverPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="false"/>
```

```
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
                <transportLayer/>
            </layers>
        </serverPolicy.csiv2>
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                <transportLayer/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

2. 按对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页）页面中描述配置 SSL 支持。
3. 配置 SSL 元素以使用 `clientAuthentication` 或 `clientAuthenticationSupported`。

   例如，

```
    <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
         trustStoreRef="defaultTrustStore" clientAuthentication="true" />
```

   或

```
    <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
         trustStoreRef="defaultTrustStore"
clientAuthenticationSupported="true" />
```

   - 如果指定 `clientAuthentication="true"`，那么服务器会请求客户机发送证书。但是，如果客户机没有证书，或者服务器不信任证书，那么握手不会成功。
   - 如果指定 `clientAuthenticationSupported="true"`，那么服务器会请求客户机发送证书。但是，即使客户机没有证书，或者服务器不信任证书，握手可能还是会成功。
   - 如果未指定 `clientAuthentication` 或 `clientAuthenticationSupported`，或者指定 `clientAuthentication="false"` 或 `clientAuthenticationSupported="false"`，那么握手期间服务器不会请求客户机发送证书。
4. 可选：如果需要更改缺省入站传输层配置，请按如下所示在 server.xml 文件中添加 <orb> 元素，或将 `transportLayer` 元素添加至现有文件。将示例中的样本值替换为您的值。

```
        <orb id="defaultOrb">
            <serverPolicy.csiv2>
                <layers>
                    <transportLayer sslRef="defaultSSLConfig"/>
                </layers>
            </serverPolicy.csiv2>
        </orb>
```

   注：<orb> 元素中的 ID 值 `defaultOrb` 是预先定义的，不能修改。

5. 确保服务器信任所使用的任何客户机证书。
6. 确保用于客户机认证的任何客户机证书已映射到注册表中的用户身份。
   - 对于基本注册表，用户身份是证书的专有名称 (DN) 中的公共名 (CN)。
   - 对于轻量级目录访问协议 (LDAP) 注册表，客户机证书中的 DN 必须位于 LDAP 注册表中。

如果省略层，那么系统对该层使用缺省值。有关 attributeLayer 和 authenticationLayer 元素的更多信息，请参阅配置入站 *CSIv2* 属性层和配置入站 *CSIv2* 认证层。

现在已配置入站 CSIv2 传输层以用于客户机证书认证。

在 xigemaAS 中配置出站 CSIv2

公共安全互操作性 V2 (CSIv2) 功能部件出站配置确定针对出站请求发送的认证信息类型。如果在 server.xml 文件中配置了 appSecurity-2.0 和 ejbRemote-3.2 功能部件，那么系统会自动启用 CSIv2 功能部件。

要了解 CSIv2 概念，请参阅公共安全互操作性 *V2 (CSIv2)* 以了解更多信息。

以下安全层对出站请求可用：

1. 配置出站 *CSIv2* 属性层。
2. 配置出站 *CSIv2* 认证层。
3. 配置出站 *CSIv2* 传输层。

配置出站 CSIv2 属性层

您可以配置 xigemaAS 概要文件服务器以对出站 CSIv2 请求执行身份断言。

在 xigemaAS 概要文件服务器的出站 CSIv2 属性层中，身份断言在缺省情况下被禁用。通过 identityAssertionEnabled 属性启用身份断言后，充当客户机的服务器支持将主体名称和匿名身份断言发送至下游服务器。您还必须配置上游服务器并启用身份断言以便客户机可断言身份。 认证层机制为 GSSUP 时，可使用 trustedIdentity 和 trustedPassword 属性指定客户机身份以供下游服务器进行信任验证。如果认证层中的认证机制为 LTPA，那么可设置 trustedIdentity 属性而不设置 trustedPassword 属性。

1. 在 server.xml 文件中添加 appSecurity-2.0 和 ejbRemote-3.2 功能部件。

```
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>ejbRemote-3.2</feature>
    </featureManager>
```

以下示例是缺省配置，不必在 server.xml 文件中指定该配置。

```
    <orb id="defaultOrb">
        <serverPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
                <transportLayer/>
            </layers>
        </serverPolicy.csiv2>
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                <transportLayer/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

2. 可选：如果需要更改缺省出站属性层配置，请按如下所示在 server.xml 文件中添加 <orb> 元素，或将 attributeLayer 元素添加至现有文件。将示例中的样本值替换为您的值。

```
    <orb id="defaultOrb">
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="true"/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

   注：<orb> 元素中的 ID 值 defaultOrb 是预先定义的，不能修改。

3. 指定上游服务器身份以供下游服务器进行信任验证。所指定的 trustedIdentity 必须存在于目标服务器的用户注册表中。

   - 如果在认证层中使用 GSSUP 机制，那么必须通过将示例值更改为充当客户机的上游服务器的身份和密码来设置 trustedIdentity 和 trustedPassword 属性。

     ```
     <attributeLayer identityAssertionEnabled="true" trustedIdentity="yourTrustedId"
      trustedPassword="yourTrustedIdPwd"/>
     ```

     对配置中的密码进行编码。可使用 securityUtility 编码命令来获取编码值。
   - 在认证层中使用 LTPA 机制时，必须通过将示例值更改为充当客户机的上游服务器身份来设置 trustedIdentity 属性。

     ```
     <attributeLayer identityAssertionEnabled="true" trustedIdentity="yourTrustedId"/>
     ```

4. 可选：如果需要更改服务器支持的缺省身份断言令牌类型，应将 identityAssertionTypes 属性添加至 server.xml 文件中的 attributeLayer 元素并指定值的逗号分隔列表。有效值为 ITTAnonymous、ITTPrincipalName、ITTX509CertChain 和 ITTDistinguishedName。例如，可配置服务器以支持带有 X509 证书链或专有名称的身份断言。将示例中的样本值替换为您的值。

```
<orb id="defaultOrb">
    <clientPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="true"
 identityAssertionTypes="ITTX509CertChain, ITTDistinguishedName"/>
        </layers>
    </clientPolicy.csiv2>
</orb>
```

   注：

   - 如果同时在认证层中配置了 LTPA 和 GSSUP，并且下游服务器支持 LTPA，那么 LTPA 优先于 GSSUP。
   - 如果同时在认证层中配置了 LTPA 和 GSSUP，并且下游服务器仅支持 GSSUP，那么系统将使用 GSSUP 并且必须指定 trustedIdentity 和 trustedPassword 属性。
   - 如果使用传输证书链来对下游服务器标识服务器，那么不需要 trustedIdentity 属性。（在 authenticationLayer 中，identityAssertionEnabled 属性设置为 true，establishTrustInClient 设置为 Never。）
   - 如果省略层，那么系统对该层使用缺省值。

   有关 authenticationLayer 和 transportLayer 元素的更多信息，请参阅配置出站 *CSIv2* 认证层和配置出站 *CSIv2* 传输层。

现在已配置出站 CSIv2 属性层以用于身份断言。

配置出站 CSIv2 认证层

您可以配置 xigemaAS 概要文件服务器以对出站 CSIv2 请求使用特定认证机制。

缺省情况下，已启用 xigemaAS 概要文件服务器的出站 CSIv2 认证层，并具有对 LTPA 和 GSSUP 认证机制的支持。缺省情况下，认证层的 `establishTrustInClient` 关联选项设置为 Supported 以指示所指定认证机制受支持并且是可选的。

使用 LTPA 机制时，确保通信 xigemaAS 服务器和其他服务器共享相同 LTPA 密钥。

1. 在 server.xml 文件中添加 `appSecurity-2.0` 和 `ejbRemote-3.2` 功能部件。

```
        <featureManager>
            <feature>appSecurity-2.0</feature>
            <feature>ejbRemote-3.2</feature>
        </featureManager>
```

   以下示例是缺省配置，不必在 server.xml 文件中指定该配置。

```
    <orb id="defaultOrb">
            <serverPolicy.csiv2>
                <layers>
                    <attributeLayer identityAssertionEnabled="false"/>
                    <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
                    <transportLayer/>
                </layers>
            </serverPolicy.csiv2>
            <clientPolicy.csiv2>
                <layers>
                    <attributeLayer identityAssertionEnabled="false"/>
                    <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                    <transportLayer/>
                </layers>
            </clientPolicy.csiv2>
    </orb>
```

2. 可选：如果需要更改缺省出站认证层配置，请按如下所示在 server.xml 文件中添加 `<orb>` 元素，或将 `authenticationLayer` 元素添加至现有文件。将示例中的样本值替换为您的值。

```
        <orb id="defaultOrb">
            <clientPolicy.csiv2>
                <layers>
                    <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                </layers>
            </clientPolicy.csiv2>
        </orb>
```

   👉 注：`<orb>` 元素中的 ID 值 `defaultOrb` 是预先定义的，不能修改。

3. 可选：将 mechanisms 属性设置为 LTPA 或 GSSUP，以仅将 LTPA 或 GSSUP（用户名和密码）用作认证机制。

```
    <authenticationLayer mechanisms="LTPA"
establishTrustInClient="Supported"/>
```

或

```
    <authenticationLayer mechanisms="GSSUP"
establishTrustInClient="Supported"/>
```

4. 可选：将 `establishTrustInClient` 属性设置为 Required、Supported 或 Never 以指示充当客户机的服务器需要、支持（可选）或从不执行使用指定机制的认证。

   注：

   - `establishTrustInClient` 属性设置为 Required 时，客户机只能将其中一种指定机制的认证令牌发送至需要或支持相同认证机制的服务器。
   - `establishTrustInClient` 属性设置为 Supported 时，客户机可选择是否在认证层中发送认证信息。如果对下游服务器配置了 Supported 或 Required，那么客户机发送兼容认证令牌。
   - 如果 `establishTrustInClient` 属性设置为 Never，那么系统禁用出站 CSIv2 认证层，并且必须启用至少一个其他 CSIv2 层以向下游服务器认证。
   - 如果省略层，那么系统对该层使用缺省值。

   有关 `attributeLayer` 和 `transportLayer` 元素的更多信息，请参阅配置出站 *CSIv2* 属性层和配置出站 *CSIv2* 传输层。

现在已配置出站 CSIv2 认证层。

配置出站 CSIv2 传输层

可配置 xigemaAS 服务器以对出站 CSIv2 请求执行客户机证书认证。

缺省情况下，已对 xigemaAS 概要文件服务器禁用出站 CSIv2 传输层的客户机证书认证。可配置 `transportLayer` 以指定要使用的 SSL 配置。

您可将 SSL 元素配置为支持或需要客户机证书认证。系统针对下游服务器用户注册表认证发送至下游服务器的证书，仅当 CSIv2 请求中未发送任何其他形式的认证（例如，属性层中的身份断言或认证层中的认证令牌）时，才使用其身份。

使用客户机证书认证时，确保此服务器支持 SSL。

1. 在 server.xml 文件中添加 `appSecurity-2.0` 和 `ejbRemote-3.2` 功能部件。

```
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>ejbRemote-3.2</feature>
    </featureManager>
```

以下示例是缺省配置，不必在 server.xml 文件中指定该配置。

```
    <orb id="defaultOrb">
        <serverPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Required"/>
                <transportLayer/>
            </layers>
        </serverPolicy.csiv2>
        <clientPolicy.csiv2>
            <layers>
                <attributeLayer identityAssertionEnabled="false"/>
```

```
                <authenticationLayer mechanisms="LTPA,GSSUP"
establishTrustInClient="Supported"/>
                <transportLayer/>
            </layers>
        </clientPolicy.csiv2>
    </orb>
```

2. 按 对 *xigemaAS* 启用 *SSL* 通信（见第 1278 页） 中所述配置 SSL 支持。
3. 可选：配置 SSL 元素以使用 `clientAuthentication` 或 `clientAuthenticationSupported`。

   例如，

```
    <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
         trustStoreRef="defaultTrustStore" clientAuthentication="true" />
```

   或

```
    <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
        trustStoreRef="defaultTrustStore"
clientAuthenticationSupported="true" />
```

4. 可选：如果需要更改缺省出站传输层配置，请按如下所示在 server.xml 文件中添加 `<orb>` 元素，或将 `transportLayer` 元素添加至现有文件。将示例中的样本值替换为您的值。

```
   <orb id="defaultOrb">
       <clientPolicy.csiv2>
           <layers>
               <transportLayer sslRef="defaultSSLConfig"/>
           </layers>
       </clientPolicy.csiv2>
   </orb>
```

   注：`<orb>` 元素中的 id 值 *defaultOrb* 是预先定义的，不能修改。

5. 确保下游服务器信任从此服务器发送的任何客户机证书。
6. 确保用于客户机认证的所有客户机证书已映射至下游服务器用户注册表中的用户身份。
   - 对于基本注册表，用户身份是证书的专有名称 (DN) 中的公共名 (CN)。
   - 对于轻量级目录访问协议 (LDAP) 注册表，客户机证书中的 DN 必须位于 LDAP 注册表中。

   注：
   - 在 `<ssl>` 元素中，`clientAuthentication` 属性设置为 true 时，客户机只能将客户机证书发送至需要或支持客户机证书认证的服务器。
   - 在 `<ssl>` 元素中，`clientAuthenticationSupported` 属性设置为 true 时，客户机可选择是否根据下游服务器使用的 <ssl> 元素配置发送客户机证书。
   - 如果未在 `<ssl>` 元素中设置 `clientAuthentication` 和 `clientAuthenticationSupported` 属性，那么系统不会对充当客户机的服务器启用客户机证书认证。

   如果省略层，那么系统对该层使用缺省值。有关 `attributeLayer` 和 `authenticationLayer` 元素的更多信息，请参阅配置出站 *CSIv2* 属性层和配置出站 *CSIv2* 认证层。

现在已配置出站 CSIv2 传输层以用于客户机证书认证。

2.6.6 为 xigemaAS 应用程序客户机容器及其应用程序配置安全性

在 xigemaAS 应用程序客户机容器及其应用程序上配置安全性，以确保客户机与服务器之间的通信是安全的。还可配置安全性以确保客户机的凭证流向服务器。

对 xigemaAS 应用程序客户机容器启用 SSL 通信

xigemaAS 应用程序客户机容器可能需要一些 SSL 配置以供客户机容器与服务器通信。为应用程序客户机容器配置 SSL 需要使用相同的 SSL 功能部件 `ssl-1.0`，服务器在启用 SSL 时需要同一功能。对于应用程序客户机和服务器，配置元素和属性是相同的；但是，对于应用程序客户机容器，这些值是在 client.xml 文件中指定的。

SSL 握手是一系列消息，系统在客户机与服务器之间通过 SSL 协议交换这些消息以进行协商，从而实现特定于连接的保护。要对 xigemaAS 概要文件应用程序客户机容器启用 SSL，SSL 功能部件 `ssl-1.0` 必须包含生成客户机使用的 SSL 配置时所需的基本信息。构成 SSL 配置时所需的基本信息是密钥库和密码。

可使用 `securityUtility createSSLCertificatecommand` 以创建客户机密钥库并为您提供有关该配置的信息。可选择使用工具，因为您也可创建密钥库及关联配置以实现其他客户定义用途。

1. 将密钥库元素添加至 client.xml 文件。`id` 属性必须为 `defaultKeyStore`，`password` 属性包含密钥库密码。可以采用明文或编码格式输入该密码。使用 `securityUtility` 编码选项以对密码进行编码。

   ```
   <keyStore id="defaultKeyStore" password="yourPassword" />
   ```

   这是创建 SSL 配置时所需的最低配置。在此配置中，如果 SSL 初始化期间密钥库和证书不存在，那么客户机会创建密钥库和证书。提供的密码长度必须至少为 6 个字符。`JKS` 是缺省密钥库类型，缺省密钥库称为 `key.jks`，这些缺省值位于 <client home>/resources/security 目录。

   使用先前配置时，客户机将在首次启动时创建 `defaultKeyStore`，但让客户机创建缺省证书会降低性能。为避免降低性能，建议使用 `securityUtilitiy createSSLCertificate` 命令创建 `defaultKeyStore` 配置中使用的缺省密钥库。

   如果需要定制 SSL 配置，请参阅 *SSL* 配置属性（见第 1279 页）。

接受签署者证书

如果客户机没有与服务器建立信任关系，那么与客户机的通信会提示用户并询问他们是否接受来自服务器的证书。如果用户回答**是**，那么证书被接受并存储在客户机密钥库配置中，命令继续执行。如果用户指定**否**，那么不会建立信任关系，调用结束并产生错误。

有关提示外观的示例：

```
*** SSL SIGNER EXCHANGE PROMPT ***
The SSL signer from target host is not found in trust store C:/xigemaas/wo
rkspace/build.image/wlp/usr/clients/myTestClient/resources/security/key.jk
s.

Here is the signer information (verify the digest value matches what is di
splayed at the server):
Subject DN:    CN=localhost, O=vsettan, C=cn
Issuer DN:     CN=localhost, O=vsettan, C=cn
Serial number: 1327582458
Expires:       Sun Jan 04 06:54:18 CST 2099
SHA-1 Digest:  00:6F:25:F1:78:5D:EB:00:B1:E2:99:DB:E8:D7:DF:3B:F8:E0:20:9A
Add signer to the trust store now? (y/n)
```

如果用户在被要求将签署者添加至信任库时指定**否**，那么您可能会接收到以下错误消息：

```
[ERROR   ] CWPKI0022E: SSL HANDSHAKE FAILURE:  A signer with SubjectDN CN=lo
calhost, O=vsettan, C=cn
sent from the target host.  The signer might need to be added to local trust
 store C:/xigemaas/workspace/build.image/wlp/usr/clients/myTestClient/resou
rces/security/key.jks, located in SSL configuration alias defaultSSLConfig.
 The extended error message from the SSL handshake exception is: PKIX path b
uilding failed: java.security.cert.CertPathBuilderException: PKIXCertPathBui
lderImpl could not build a valid CertPath.; internal cause is:
        java.security.cert.CertPathValidatorException: The certificate issue
d by SubjectDN CN=localhost, O=vsettan, C=cn is not trusted; internal cause
 is:
        java.security.cert.CertPathValidatorException: Certificate chaining
 error
throw able: javax.net.ssl.SSLHandshakeException: java.security.cert.Certific
ateException: PKIX path building failed: java.security.cert.CertPathBuilderE
xception
: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause
 is:

        java.security.cert.CertPathValidatorException: The certificate issue
d by
 SubjectDN CN=localhost, O=vsettan, C=cn is not trusted; internal cause is:
        java.security.cert.CertPathValidatorException: Certificate chaining
 error
```

自动接受签署者证书

如果客户机不希望收到要求提供签署者证书的提示，并选择接受服务器签署者证书而不检查该证书，那么该用户可在客户机容器命令行中添加 `-autoAcceptSigner` 标记。

```
client run client_name --autoAcceptSigner
```

客户机认证

如果客户机要与已启用客户机认证的服务器通信，那么服务器与客户机需要相互信任。客户机的密钥库中必须具有密钥和个人证书。如果使用 `securityUtility createSSLCertificate` 命令，那么密钥库包含个人证书。要与客户机应用程序容器通信的服务器必须信任客户机，所以需要将客户机中的签署者添加至服务器的信任库。可使用 Java 工具 `keytool` 从应用程序客户机容器的密钥库中抽取签署者，并将客户机中的证书添加至服务器的信任库。

在 xigemaAS 应用程序客户机容器中配置 JAAS 程序化登录

xigemaAS 概要文件应用程序客户机容器可配置为使用 JAAS 程序化登录。

查看在应用程序客户机容器上认证用户的不同方法，并决定程序化登录选项是否为您的环境的最佳选择。有关进一步详细信息，请参阅*xigemaAS 应用程序客户机容器上的认证*（见第 1050 页）。

程序化登录是一种表单登录类型，它支持使用应用程序表示登录表单来进行认证。此方法要求应用程序开发者收集用户的凭证并认证该用户。此方法使用 JAAS 框架将用户的凭证发送至服务器以进行认证。JAAS 框架包含通过指定 JAAS 登录配置和使用回调处理程序收集用户凭证来创建登录上下文。从登录上下文获取主体集时，可使用 xigemaAS 安全性 API 在该线程上设置该主体集，它将用于您对服务器的出站调用。

JAAS 登录配置指定登录模块如何用于认证及哪些登录模块用于认证。以下是 xigemaAS 概要文件在客户机上提供的 JAAS 登录配置：

- WSLogin JAAS 登录配置：一种通用 JAAS 登录配置，xigemaAS 概要文件应用程序客户机容器可使用此配置执行基于用户标识和密码的认证。但是，此配置不支持客户机应用程序模块的部署描述符中指定的 CallbackHandler 处理程序。
- ClientContainer JAAS 登录配置：此 JAAS 登录配置识别客户机应用程序模块的部署描述符中指定的 CallbackHandler 处理程序（如果已指定）。如果未在该部署描述符中指定处理程序，那么将使用通过程序指定的处理程序。

  JAAS 登录配置指定的登录模块实现某种认证技术。登录模块可使用 javax.security.auth.callback.CallbackHandler 接口收集来自用户的凭证。xigemaAS 概要文件提供 CallbackHandler 接口的非提示实现，此实现称为 com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl。此实现允许应用程序开发者在应用程序中直接指定凭证而不必提示用户。可通过两种方法指定 CallbackHandler 实现：

  - 通过程序指定实现（作为 javax.security.auth.login.LoginContext 构造函数的自变量）；例如：

    ```
    LoginContext logincontext = new LoginContext("ClientContainer", new
     WSCallbackHandlerImpl("user", "password"));
    ```

  - 在客户机应用程序模块的部署描述符 (application-client.xml) 中指定实现名称；例如：

    ```
    <callbackhandler>com.acme.callbackhandler.WSCallbackHandlerImpl/
    <callbackhandler>
    ```

👉 注：WSLogin 登录配置不识别在部署描述符中指定 CallbackHandler 处理程序的第二个选项。

1. 将 appSecurityClient-1.0 功能部件添加至 client.xml 文件。

   ```
   <feature>appSecurityClient-1.0</feature>
   ```

2. 为客户机配置 SSL：

   a. 可选：使用 securityUtility 命令为客户机创建 SSL 证书；例如：

   ```
   securityUtility createSSLCertificate --client=myClient --
   password=xigemaas
   ```

   b. 建议：使用 securityUtility 命令生成 xor 编码密码。例如，对密码 xigemaas 进行编码：

   ```
   securityUtility encode xigemaas
   ```

   c. 将 keyStore 元素添加至 client.xml 文件。以下示例使用缺省 SSL 配置：

   ```
   <keyStore id="defaultKeyStore" password="{xor}MzY9Oi0rJg==" /> <!-- pwd:
   xigemaas -->
   ```

3. 在应用程序代码中，使用 ClientContainer JAAS 登录配置和 WSCallbackHandlerImpl 回调处理程序创建 Subject。

   a. 应用程序发出出站请求之前，添加以下代码。将 userName 和 userPassword 更改为目标服务器的用户注册表中的现有用户的有效凭证。

   ```
   CallbackHandler wscbh = new WSCallbackHandlerImpl("userName",
    "userPassword");
   ```

```
LoginContext ctx = null;
try {
      ctx = new LoginContext("ClientContainer", wscbh);
} catch (LoginException le) {
      le.printStackTrace();
}
try {
          ctx.login();
} catch (LoginException le) {
      le.printStackTrace();
}
    Subject subject = ctx.getSubject();
```

4. 在线程上设置先前步骤中获取的 Subject，并使用该 Subject 查找 EJB。使用 WSSubject.doAs 或 doAsPrivilieged API 完成此操作。com.ibm.websphere.security.auth.WSSubject.doAs 或 com.ibm.websphere.security.auth.WSSubject.doAsPrivileged 代码块中的该主体用于 Java™ Platform Enterprise Edition (J2EE) 资源授权检查。

```
WSSubject.doAs(subject, new PrivilegedAction() {
    public Object run() {
           try {
                  //Perform EJB lookup and invocation
           } catch (Exception ex) {
                  ex.printStackTrace();
           }
           return null;
    }
});
```

5. 如果已在客户机上启用 Java™ 2 安全性，并且应用程序代码将调用 JAAS 或 xigemaAS 安全性 API，请将必需的 Java™ 2 安全性许可权添加至应用程序的 permissions.xml 文件或 client.xml 文件。有关进一步详细信息，请参阅 *Java™ 2 安全性*。

与在服务器上一样，您可使用定制登录模块来制定更多认证决策，或向主体集添加信息以在客户机应用程序内制定更详细的授权决策。有关进一步详细信息，请参阅在 *xigemaAS* 应用程序客户机容器上配置 *JAAS* 定制登录模块。

为 xigemaAS 应用程序客户机容器配置 JAAS 定制登录模块

可配置 xigemaAS 概要文件应用程序客户机容器以使用定制 Java™ 认证和授权服务 (JAAS) 登录模块。

请确保您有一个 JAR 文件包含 JAAS 定制登录模块，此模块用来实现 javax.security.auth.spi.LoginModule 接口。

可以使用定制登录模块来作出其他认证决策，或者在主体集中添加信息以在应用程序内部作出更细颗粒度的权限决策。要配置 JAAS 定制登录模块，请完成以下步骤。

1. 将 appSecurityClient-1.0 功能部件添加至 client.xml 文件。

```
<feature>appSecurityClient-1.0</feature>
```

2. 创建用于实现 LoginModule 接口的 com.sample.CustomLoginModule 类并将其打包到 CustomLoginModule.jar 文件中。

3. 创建一个 <library> 元素，该元素使用一个 <fileset> 元素指示 CustomLoginModule.jar 文件所在的位置。在此示例中，该文件在客户机的配置目录中，库标识为 customLoginLib。

```
<library id="customLoginLib">
         <fileset dir="${server.config.dir}"
 includes="CustomLoginModule.jar"/>
</library>
```

4. 创建 <jaasLoginModule> 元素。在此示例中，id 为 myCustom。

   a. 通过将 controlFlag 属性设置为 REQUIRED，将定制登录模块配置为需要成功认证。

   b. 将 libraryRef 属性设置为 customLoginLib（上一步中配置的 <library> 元素的标识）。

```
<jaasLoginModule id="myCustom"  className="com.sample.CustomLoginModule"
 controlFlag="REQUIRED" libraryRef="customLoginLib"/>
```

5. 在应用程序客户机容器 ClientContainer 上创建 <jaasLogincontextEntry> 元素，此元素带有系统定义的 JAAS 配置的标识和名称。您还可以将此 JAAS 配置设置为 WSLogin 或您自己的 JAAS 配置。在 loginModuleRef 属性中，添加 proxy（代理登录模块的标识）和 myCustom（上一步中创建的 jaasLoginModule 元素的标识）。

```
<jaasLoginContextEntry id="ClientContainer" name="ClientContainer"
               loginModuleRef="proxy, myCustom"/>
```

在 xigemaAS 应用程序客户机容器中配置公共安全互操作性 V2 (CSIv2)

xigemaAS 概要文件应用程序客户机容器支持各种级别的 CSIv2 安全性，例如，消息认证（消息层）和客户机证书认证（传输层）。通过使用 CSIv2 功能部件，您可以对面向服务器的出站请求指定认证类型。CSIv2 功能部件在缺省情况下已启用。可在 xigemaAS 概要文件应用程序客户机容器中配置 CSIv2，以允许 Java™ Platform Enterprise Edition 供应商之间互操作。

在xigemaAS应用程序客户机容器中配置出站 CSIv2

CSIv2 功能部件在缺省情况下已启用。可在 xigemaAS 应用程序客户机容器中配置 CSIv2，以允许 Java™ Platform Enterprise Edition 供应商之间互操作。

要了解 CSIv2 概念，请参阅公共安全互操作性 *V2 (CSIv2)* 以了解更多信息。

在 xigemaAS 概要文件应用程序客户机容器中配置出站 CSIv2 认证层

可配置xigemaAS概要文件应用程序客户机容器以对出站 CSIv2 请求使用特定认证机制。

缺省情况下，已启用 xigemaAS 概要文件应用程序客户机容器的出站 CSIv2 认证层，并具有对 GSSUP 认证机制的支持。缺省情况下，认证层的 establishTrustInClient 关联选项设置为 Supported 以指示所指定认证机制受支持并且是可选的。

1. 按如下所示在 client.xml 文件中配置 orb 元素，或将 authenticationLayer 元素添加至现有文件，以将示例中的样本值替换为您的值：

```
<orb id="defaultOrb">
   <clientPolicy.clientContainerCsiv2>
     <layers>
      <authenticationLayer user="userId" password="{xor}PDc+MTg6Ejo="/>
     </layers>
   </clientPolicy.clientContainerCsiv2>
</orb>
```

👉 注：orb 元素中的 id 值 defaultOrb 是预先定义的，不能修改。

👉 注：不能使用散列编码来对密码加密，因为不能通过散列值对原始密码解码。

mechanisms 和 establishTrustInClient 属性是可选的。对于 mechanisms 属性，唯一受支持值和缺省值为 GSSUP。

如果未指定 <orb> 元素，那么以下配置是隐式的。

```
<orb id="defaultOrb">
    <clientPolicy.clientContainerCsiv2>
        <layers>
            <authenticationLayer mechanisms="GSSUP"
 establishTrustInClient="Supported"/>
            <transportLayer/>
        </layers>
    </clientPolicy.clientContainerCsiv2>
</orb>
```

2. 可选：使用有效用户标识和密码来设置 user 和 password 属性以访问服务器。缺省情况下，服务器需要 GSSUP 机制以进行入站连接，这意味着服务器必须接收用户和密码，因为此需求，client.xml 文件中需要用户和密码值，除非应用程序实现程序化登录。
3. 可选：将 establishTrustInClient 属性设置为 Required、Supported（缺省值）或 Never 以使用指定机制来执行认证。例如，

```
<orb id="defaultOrb">
   <clientPolicy.clientContainerCsiv2>
     <layers>
      <authenticationLayer user="userId" password="{xor}PDc+MTg6Ejo="
 establishTrustInClient="Required" />
     </layers>
   </clientPolicy.clientContainerCsiv2>
</orb>
```

👉 注：

- establishTrustInClient 属性设置为 Required 时，客户机只能将其中一种指定机制的认证令牌发送至需要或支持相同认证机制的服务器。
- establishTrustInClient 属性设置为 Supported（缺省值）时，客户机可选择是否在认证层中发送认证信息。对于相同认证机制，如果对服务器配置了 Supported 或 Required，那么客户机将发送兼容认证令牌。
- 如果 establishTrustInClient 属性设置为 Never，那么系统禁用出站 CSIv2 认证层，并且必须启用 CSIv2 传输层以向服务器认证。

现在已配置出站 CSIv2 认证层。

在 xigemaAS 应用程序客户机容器中配置出站 CSIv2 传输层

可配置 xigemaAS 概要文件应用程序客户机容器以对出站 CSIv2 请求执行客户机证书认证。

缺省情况下，不会对 xigemaAS 概要文件应用程序客户机容器使用出站 CSIv2 传输层的客户机证书认证。可配置 transportLayer 以指定要使用的 SSL 配置。

您可将 SSL 元素配置为支持或需要客户机证书认证。系统针对服务器用户注册表认证发送至服务器的证书，仅当 CSIv2 请求中未发送任何其他形式的认证（例如，属性层中的身份断言或认证层中的认证令牌）时，才使用其身份。

1. 按对 *xigemaAS* 应用程序客户机容器启用 *SSL* 通信（见第 1395 页）中所述配置 SSL 支持。
2. 可选：配置 SSL 元素以使用 `clientAuthentication` 或 `clientAuthenticationSupported`。

   例如，

   ```
       <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
            trustStoreRef="defaultTrustStore" clientAuthentication="true" />
   ```

   或

   ```
       <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
           trustStoreRef="defaultTrustStore"
    clientAuthenticationSupported="true" />
   ```

3. 按如下所示在 client.xml 文件中配置 `<orb>` 元素，或将 `transportLayer` 元素添加至现有文件并将样本中的样本值替换为您的值：

   ```
   <orb id="defaultOrb">
       <clientPolicy.clientContainerCsiv2>
           <layers>
               <transportLayer sslRef="defaultSSLConfig"/>
           </layers>
       </clientPolicy.clientContainerCsiv2>
   </orb>
   ```

   如果未指定 `<orb>` 元素，那么以下配置是隐式的。

   ```
   <orb id="defaultOrb">
       <clientPolicy.clientContainerCsiv2>
           <layers>
               <authenticationLayer mechanisms="GSSUP"
    establishTrustInClient="Supported"/>
               <transportLayer/>
           </layers>
       </clientPolicy.clientContainerCsiv2>
   </orb>
   ```

4. 确保服务器信任从此服务器发送的任何客户机证书。
   - 在 ssl 元素中，`clientAuthentication` 属性设置为 true 时，客户机只能将客户机证书发送至需要或支持客户机证书认证的服务器。
   - 在 ssl 元素中，`clientAuthenticationSupported` 属性设置为 true 时，客户机可选择是否根据服务器使用的 ssl 元素配置发送客户机证书。
   - 如果未在 ssl 元素中设置 `clientAuthentication` 和 `clientAuthenticationSupported` 属性，那么系统不会对充当客户机的服务器启用客户机证书认证。

现在已配置出站 CSIv2 传输层以用于客户机证书认证。

2.6.7 配置 Java™ Servlet 3.1 支持以实现安全性

按 Java™ Servlet 3.1 规范中的定义，xigemaAS 概要文件支持所有安全性更新。

在 xigemaAS 概要文件上采用 Java™ Servlet 3.1 功能部件。

1. 在 server.xml 文件中添加 servlet-3.1 功能部件：

```
<feature>servlet-3.1</feature>
```

2. 确定要使用的以下 Java™ Servlet 3.1 功能：

   - 在登录表单中指定 autocomplete=off。

     对表单登录页面使用 HTML 时，将密码表单字段设置为 autocomplete="off" 以禁止在 Web 浏览器中自动填写密码。例如：

```
<form method="POST" action="j_security_check">
<input type="text" name="j_username">
<input type="password" name="j_password" autocomplete="off">
</form>
```

   - 指定所有认证安全性约束 (**)。

     特殊角色名称 ** 指定任何已认证用户。** 显示在授权约束中时，如果用户通过认证，那么该用户具有对该约束中指定的方法的访问权。用户不必在应用程序绑定中映射至此角色。例如：

```
<security-constraint id="SecurityConstraint_1">
  <web-resource-collection id="WebResourceCollection_1">
   <web-resource-name>Protected with ** role</web-resource-name>
   <url-pattern>/AnyAuthSecurityConstraint</url-pattern>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint id="AuthConstraint_1">
   <role-name>**</role-name>
  </auth-constraint>
</security-constraint>
```

     使用角色名 ** 调用 isUserInRole() 方法时，如果用户通过认证，那么 isUserInRole() 返回 true。如果 ** 是安全角色的配置中的已定义角色，那么它不会被视为任何特殊已认证用户角色。该用户必须在应用程序绑定中映射至该角色，isUserInRole 才能返回 true。

   - 在 web.xml 文件中指定 deny-uncovered-http-methods 标记。

     如果在 web.xml 文件中指定了 deny-uncovered-http-methods 元素，那么该容器拒绝符合以下条件的任何已发现 HTTP 方法：最符合请求 URL 的 URL 模式的组合安全性约束内未枚举这些方法。将返回 403 (SC_FORBIDDEN) 状态码。例如：

```
<servlet-mapping id="ServletMapping_1">
  <servlet-name>MyServlet</servlet-name>
  <url-pattern>/MyURLPattern</url-pattern>
 </servlet-mapping>

<deny-uncovered-http-methods/>

<!-- SECURITY CONSTRAINTS -->
<security-constraint id="SecurityConstraint_1">
  <web-resource-collection id="WebResourceCollection_1">
   <web-resource-name>Protected with Employee or Manager roles</web-
resource-name>
   <url-pattern>/MyURLPattern</url-pattern>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
  </web-resource-collection>
```

```
  <auth-constraint id="AuthConstraint_1">
   <role-name>Employee</role-name>
       <role-name>Manager</role-name>
  </auth-constraint>
</security-constraint>
```

如果在 web.xml 文件中指定了 deny-uncovered-http-methods 元素，那么 messages.log 文件中将对每个 servlet 中的每个 URL 模式记录一条消息，此消息指示已发现方法，并且带有一个注释，说明这些已发现方法不受保护并且不可访问。例如：

```
对于 servlet MyServlet
中的 URL MyURLPattern，发现以下 HTTP 方法并且它们不可访问：DELETE OPTIONS HEAD
 PUT TRACE
```

如果未在 web.xml 文件中指定 deny-uncovered-http-methods 元素，那么 messages.log 文件中将对每个 servlet 中的每个 URL 模式记录一条消息，此消息指示已发现方法，并且带有一个注释，说明这些已发现方法不受保护并且可访问。例如：

```
对于 servlet MyServlet
中的 URL MyURLPattern，发现以下 HTTP 方法并且它们可访问：DELETE OPTIONS HEAD
 PUT TRACE
```

您现在已保护您的应用程序。

2.6.8 配置与 xigemaAS 的安全 JMX 连接

您可以在 xigemaAS 上使用 SSL 来访问受保护的 Java™ 管理扩展 (JMX) 连接器。由 xigemaAS 功能部件 restConnector-1.0 来启用安全 JMX 连接。

通过 xigemaAS 功能部件 restConnector-1.0 来启用 REST 连接器。单个管理员角色会保护通过 REST 连接器进行的远程访问。此外，需要 SSL 才能保持通信机密。restConnector-1.0 功能部件已经包含 ssl-1.0 功能部件。

👉 注：xigemaAS 上部署的应用程序可以不受限制地访问其 MBeanServer 目录。

下列部分描述如何在 xigemaAS 上配置和访问 REST 连接器。

1. 在 server.xml 文件中使用下列代码来启用 REST 连接器。

```
<featureManager>
     <feature>restConnector-2.0</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 *SSL* 证书。

   确保证书 subjectDN 的 CN 值是运行该服务器的机器的主机名，并且信任库的 jConsole 连接中包含该服务器的证书。

3. 在 server.xml 文件中将用户或组配置到管理员角色。

   - 映射 *xigemaAS* 的管理员角色（见第 1180 页）

4. 访问 REST 连接器。

   可从 Java 客户机或直接通过 HTTPS 调用来访问 xigemaAS REST 连接器。Java 客户机使用该连接器的客户端（在 wlp/clients/restConnector.jar 中）并实现 javax.management.MBeanServerConnection

接口。HTTPS 调用使用该连接器的服务器端。至于服务器端的 HTTPS 调用，可进行 HTTPS 调用的任何编程语言（例如，C++、JavaScript、curl、Ruby 和 Perl）都可使用 REST API。REST API 包含管理端点（JMX）及文件传输。

- 从 *JMX* 客户机应用程序或者通过使用 Java SDK 中提供的 jConsole 工具来访问 REST 连接器。使用 -J 标志将系统属性作为 Java 选项来传递，以及将类路径设为包含连接器类文件。连接器类文件打包在 clients/restConnector.jar 文件中。
  - 使用 SSL 证书的下列属性：

```
-J-Djavax.net.ssl.trustStore=<location of your client trust store>
-J-Djavax.net.ssl.trustStorePassword=<password for the trust store>
-J-Djavax.net.ssl.trustStoreType=<type of trust store>
```

    以下示例显示了使用 SSL 配置的 jConsole 工具：

```
jconsole -J-Djava.class.path=%JAVA_HOME%/lib/jconsole.jar;
                             %JAVA_HOME%/lib/tools.jar;
                             %WLP_HOME%/clients/restConnector.jar
         -J-Djavax.net.ssl.trustStore=key.jks
         -J-Djavax.net.ssl.trustStorePassword=xigemaas
         -J-Djavax.net.ssl.trustStoreType=jks
```

    在 jConsole 启动后，选择**远程进程**，然后输入 JMX 服务 URL：`service:jmx:rest://<host>:<port>/xigemaJMXConnectorREST`。端口号是 HTTPS 端口。您还必须提供用户名和密码。
- 使用 HTTPS 调用直接访问 REST 连接器。

  要使用 HTTPS 调用访问 REST 连接器，您需要 xigemaAS 9.0.0.1 或更高版本。

  1. 使用浏览器打开 https://*<host>*:*<port>*/xigemaJMXConnectorREST/api，然后输入您在步骤 3 中指定的管理凭证。
  2. 检查可用 REST API。每项具有其行为、输入、输出、查询参数和头的描述。

2.6.9 为 xigemaAS 概要文件配置与 Web 安全性相关的属性

可以为 xigemaAS 概要文件配置与 Web 安全性相关的属性，例如 SSO 和客户机证书认证。

可以使用 `webAppSecurity` 元素来配置 xigemaAS 概要文件的 Web 容器应用程序安全性。确保将 `appSecurity-2.0`、`servlet-3.0` 及其他必需 xigemaAS 功能部件添加到 xigemaAS 概要文件的 server.xml 文件。

针对 xigemaAS 使用 LTPA cookie 来定制 SSO 配置

借助单点登录 (SSO) 配置支持，访问 xigemaAS 概要文件资源（例如 HTML、JavaServer Pages (JSP) 文件和 servlet）或访问多个 xigemaAS 概要文件服务器中共享相同轻量级第三方认证 (LTPA) 密钥的资源时，Web 用户可以认证一次。

当用户在其中一个 xigemaAS 概要文件服务器上通过认证时，会将服务器所生成的认证信息通过 Cookie 传输到 Web 浏览器。使用 cookie 来将认证信息传播到其他 xigemaAS 概要文件服务器。

LTPA 已配置好且可以立即使用。用于存储 SSO 令牌的缺省 cookie 名称称为 `ltpaToken2`。如果要对该 cookie 使用另一名称，那么可以使用 `<webAppSecurity>` 元素的 ssoCookieName 属性来定制 cookie 名称。如果定制 cookie 名称，请确保参与 SSO 的所有服务器都使用相同的 cookie 名称。

有关 SSO 的更多信息，请参阅 单点登录 *(SSO)*（见第 1039 页）。

以下示例代码将用户设置为 HTTP 会话到期后注销，并将 SSO cookie 的名称设置为 `myCookieName`：

```
<webAppSecurity  logoutOnHttpSessionExpire="true"  ssoCookieName="myCookieName" />
```

注：为使 SSO 在 xigemaAS 服务器和/或其他服务器间生效，请设置以下资源：

- 服务器必须使用相同 LTPA 密钥并共享相同用户注册表。
- 如果服务器不在相同域中，请使用 `<webAppSecurity>` 元素的 ssoDomainNames 属性来列示这些域。以下示例代码将域名设置为 `domain.com`：

  ```
  <webAppSecurity ssoDomainNames="domain.com" />
  ```

- 如果服务器在相同域中，请将 `<webAppSecurity>` 元素的 ssoUseDomainFromURL 属性设置为 `true`，或在 ssoDomainNames 属性中指定域名。以下示例代码将 ssoUseDomainFromURL 设置为 `true` 以通过请求 URL 获取域名：

  ```
  <webAppSecurity ssoUseDomainFromURL="true" />
  ```

为客户机证书认证配置 Web 应用程序和服务器

可以在 xigemaAS 上使用 SSL 客户机认证来配置 Web 应用程序。

此主题假定您已创建 SSL 证书，如从命令行创建 *SSL* 证书（见第 1287 页）中所述。

如果服务器端请求客户端发送证书，那么会进行客户机证书认证。可以在 SSL 配置上为客户机证书认证配置 xigemaAS 服务器。要这样做，请将 `ssl-1.0` xigemaAS 功能部件，以及要用于认证的密钥库信息告知给服务器的代码，添加到 server.xml 文件。

有关支持的 SSL 方面的详细信息，请参阅*xigemaAS* 功能部件（见第 906 页）。

1. 确保 Web 应用程序的部署描述符将客户机证书认证指定为要使用的认证方法。

   检查部署描述符是否包含下列元素：

   ```
   <auth-method>CLIENT-CERT</auth-method>
   ```

   注：可以使用 Rational® Application Developer 等工具来创建部署描述符。

2. 可选： 使用命令行生成 SSL 证书。请参阅*securityUtility* 命令（见第 1287 页）。
3. 通过将下列行添加到 server.xml 文件，配置服务器以启用 SSL 客户机认证：

   ```
   <featureManager>
        <feature>ssl-1.0</feature>
   <featureManager>

   <ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore"
        trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />
   <keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="defaultPWD" />
   <keyStore id="defaultTrustStore" location="trust.jks" type="JKS" password="defaultPWD" />
   ```

   - 如果指定 `clientAuthentication="true"`，那么服务器会请求客户机发送证书。但是，如果客户机没有证书，或者服务器不信任证书，那么握手不成功。
   - 如果指定 `clientAuthenticationSupported="true"`，那么服务器会请求客户机发送证书。但是，如果客户机没有证书，或者服务器不信任证书，那么握手仍可能成功。

- 如果未指定 `clientAuthentication` 或 `clientAuthenticationSupported`，或者指定 `clientAuthentication="false"` 或 `clientAuthenticationSupported="false"`，那么握手期间服务器不会请求客户机发送证书。

4. 将客户机证书添加到浏览器。请参阅浏览器的文档，以了解如何添加客户机证书。
5. 确保服务器信任所使用的任何客户机证书。
6. 确保用于客户机认证的任何客户机证书已映射到注册表中的用户身份。
   - 对于基本注册表，用户身份是证书的专有名称 (DN) 中的公共名 (CN)。
   - 对于轻量级目录访问协议 (LDAP) 注册表，客户机证书中的 DN 必须位于 LDAP 注册表中。
7. 如果客户机证书认证不成功，请将下列行添加到 server.xml 文件，以仅使用基本认证、用户标识和密码。

```
<webAppSecurity allowFailOverToBasicAuth="true" />
```

> 注：如果指定 `allowFailOverToBasicAuth="false"` 或者未指定 `allowFailOvertoBasicAuth`，并且客户机证书认证不成功，那么请求会生成 `403` **认证错误**消息，而且不会提示客户机进行基本认证。

配置 xigemaAS 概要文件服务器以跟踪已注销 LTPA 令牌

可配置 xigemaAS 概要文件服务器以跟踪已注销轻量级第三方认证 (LTPA) 令牌。

用户使用表单注销或程序注销时，系统会从 cookie 中移除用于单点登录的 LTPA 令牌。系统还会从本地认证高速缓存中移除用于 SSO 的 LTPA 令牌，并使会话失效。如果令牌被保存下来并再次呈现，那么系统会根据到期时间和 LTPA 加密密钥验证该令牌。

启用此元素后，系统会跟踪服务器上已注销的 LTPA SSO 令牌，如果这些令牌在同一服务器上再次出现，那么系统不会使用这些令牌。已执行注销，用户需要再次认证。

此配置仅在同一服务器上生效。这意味着，只能在用户已注销的服务器上跟踪 LTPA 令牌，如果同一 LTPA 令牌出现在另一服务器上，那么将使用该 LTPA 令牌，如果这些 LTPA 密钥是共享的并且该令牌未到期，那么将使用该令牌直到它也在该服务器上注销。

1. 要在特定 xigemaAS 概要文件服务器上跟踪已注销的令牌，可在 server.xml 中启用以下元素：

```
<webAppSecurity trackLoggedOutSSOCookies="true"/>
```

启用此元素后，它可能影响您的单点登录 (SSO) 方案。例如，如果用户“bob”从多个浏览器登录同一服务器，从一个浏览器注销，然后尝试使用另一浏览器访问资源，那么该用户必须登录，因为所呈现的令牌被废弃。

2.6.10 为 xigemaAS 配置认证别名

可以配置认证数据别名，以与 xigemaAS 概要文件上的认证资源引用一起使用。

可以通过在 xigemaAS 概要文件中定义用于认证的用户和密码来使用认证数据别名。要这样做，请将 `jdbc-4.0` xigemaAS 功能部件添加到 server.xml 文件，然后至少添加一个 `authData` 元素。

> 注：未提供任何认证别名主体映射模块支持。

1. 在 server.xml 文件中添加 `jdbc-4.0` 功能部件。

```
    <featureManager>
```

```
        <feature>jdbc-4.0</feature>
    </featureManager>
```

2. 在 server.xml 文件中配置 authData 元素，如下所示。如果 authData 元素用作顶级配置元素，那么必须将 id 属性值设置为唯一认证别名。

```
    <authData id="auth1" user="dbuser1" password="dbuser1pwd"/>
```

3. 通过在资源引用中使用 authentication-alias 元素来配置应用程序的 xigemaAS 部署描述符，例如，ibm-web-bnd.xml 文件。name 属性值必须与 server.xml 文件中定义的 id 属性匹配。

```
<resource-ref name="jdbc/mydbresource" binding-name="jdbc/mydbresource">
  <authentication-alias name="auth1"/>
</resource-ref>
```

2.6.11 开发 xigemaAS 概要文件安全性基础结构的扩展

xigemaAS 概要文件服务器提供各种插入点，以便您能扩展安全性基础结构。

此部分阐述了下列主题：

- 遵循为 *xigemaAS* 开发定制 *TAI*（见第 1407 页） 中的指示信息，以开发定制信任关联拦截器 (TAI) 来扩展 xigemaAS 概要文件服务器的安全性基础结构。
- 遵循为系统登录配置开发 *JAAS* 定制登录模块（见第 1410 页）中的指示信息，以开发 JAAS 定制登录模块来扩展 xigemaAS 概要文件服务器的安全性基础结构。

为 xigemaAS 开发定制 TAI

要开发定制信任关联拦截器 (TAI) 类，可以实现 xigemaAS 概要文件服务器随附的 com.ibm.wsspi.security.tai.TrustAssociationInterceptor 接口。

信任关联接口是一个服务提供程序 API，支持将第三方安全服务与 xigemaAS 服务器集成在一起。处理 Web 请求时，xigemaAS 服务器会向外调用，并将 HttpServletRequest 和 HttpServletResponse 传递到信任关联拦截器。HttpServletRequest 会调用拦截器的 isTargetInterceptor 方法来查看拦截器是否可以处理该请求。在选择相应的信任关联拦截器之后，HttpServletRequest 会由拦截器的 negotiateValidateandEstablishTrust 方法加以处理，而且结果是在 TAIResult 对象中返回。可以将您自己的逻辑代码添加到定制 TAI 类的每个方法。

另请参阅 TAI 接口的 Java™ API 文档。

👉 注:

下面是称为 SimpleTAI 的样本 TAI 类，它也列出 TrustAssociationInterceptor 接口中的所有可用方法。

```
package com.ibm.websphere.security.sample;

import java.util.Properties;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.ibm.websphere.security.WebTrustAssociationException;
import com.ibm.websphere.security.WebTrustAssociationFailedException;
import com.ibm.wsspi.security.tai.TAIResult;
import com.ibm.wsspi.security.tai.TrustAssociationInterceptor;

public class SimpleTAI implements TrustAssociationInterceptor {
   public SimpleTAI() {
```

```
        super();
    }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#isTargetInterceptor
 * (javax.servlet.http.HttpServletRequest)
 */
   public boolean isTargetInterceptor(HttpServletRequest req)
                  throws WebTrustAssociationException {
      //Add logic to determine whether to intercept this request
        return true;
    }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#negotiateValidateandEstablishTrust
 * (javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
 */
   public TAIResult negotiateValidateandEstablishTrust(HttpServletRequest req,
                    HttpServletResponse resp) throws WebTrustAssociationFailedException {
        // Add logic to authenticate a request and return a TAI result.
        String tai_user = "taiUser";
        return TAIResult.create(HttpServletResponse.SC_OK, tai_user);
     }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#initialize(java.util.Properties)
 */
    public int initialize(Properties arg0)
                    throws WebTrustAssociationFailedException {
        return 0;
    }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getVersion()
 */
    public String getVersion() {
        return "1.0";
    }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#getType()
 */
    public String getType() {
        return this.getClass().getName();
    }

/*
 * @see com.ibm.wsspi.security.tai.TrustAssociationInterceptor#cleanup()
 */
    public void cleanup()

    {}
}
```

将 TAI 类添加至 xigemaAS 服务器。

使用下列方法之一来将 TAI 类添加到 xigemaAS 服务器：

- 将定制 TAI 类放入 JAR 文件（例如 `simpleTAI.jar`），然后使 JAR 文件可作为共享库提供。请参阅为 *xigemaAS* 概要文件配置 *TAI*（见第 1318 页）。
- 将定制 TAI 类打包为功能部件。请参阅将定制 *TAI* 作为 *xigemaAS* 功能部件来开发（见第 1269 页）。

开发定制 SIP TAI

在开发会话启动协议 (SIP) 应用程序时，可以创建定制信任关联拦截器 (TAI)。

开发 SIP TAI 与开发其他在信任关联中使用的任意定制拦截器相似。实际上，SIP 应用程序的定制 TAI 是信任关联拦截器模型的扩展。

TAI 可由 SIP Servlet 请求或 SIP Servlet 响应进行调用。要实现定制 SIP TAI，您需要编写自己的 Java™ 类。

1. 编写一个 Java™ 类，此类扩展 `com.ibm.wsspi.security.tai.extension.BaseTrustAssociationInterceptor` 类并实现 `com.ibm.websphere.security.tai.extension.SIPTrustAssociationInterceptor` 接口。这些类是在 ${wlp.install.dir}/dev/api/ibm/com.ibm.websphere.appserver.api.sipServletSecurity.1.0_1.0.10.jar 文件中定义的。
2. 声明以下 Java™ 方法：

```
public int initialize(Properties properties) throws WebTrustAssociationFailed
Exception;
```

这在处理第一个消息之前调用的，因此实现就可以分配任何它需要的资源。例如，它可以建立与数据库的连接。`WebTrustAssociationFailedException` 是在 ${wlp.install.dir}/lib/com.ibm.websphere.security_1.0.10.jar 文件中定义的。`properties` 参数的值来自 `<trustAssociation>` 配置。

```
public void cleanup();
```

它是在 TAI 可释放任何它占用的资源时调用的。例如，它可以关闭与数据库的连接。

```
public boolean isTargetProtocolInterceptor(SipServletMessage sipMsg) throws W
ebTrustAssociationFailedException;
```

您的定制 TAI 可使用此方法来处理 `sipMsg` 消息。如果此方法返回 `false`，那么 xigemaAS 将对 `sipMsg` 忽略您的 TAI。

```
public TAIResult negotiateValidateandEstablishProtocolTrust (SipServletReques
t req, SipServletResponse resp) throws WebTrustAssociationFailedException;
```

此方法返回 `TAIResult`，它指示要处理的消息的状态，以及尝试认证的用户的用户标识或唯一标识。如果认证成功，那么 `TAIResult` 包含 `HttpServletResponse.SC_OK` 状态和主体。如果认证失败，那么 `TAIResult` 将包含返回码 `HttpServletResponse.SC_UNAUTHORIZED (401)`、`SC_FORBIDDEN (403)` 或 `SC_PROXY_AUTHENTICATION_REQUIRED (407)`。它仅指示容器是否应该接受消息以便进一步处理。为质询入局请求，TAI 实现必须生成并发送自己的 `SipServletResponse`（包含质询）。系统可针对内部 TAI 错误抛出该异常。表 *1* 描述 `negotiateValidateandEstablishProtocolTrust` 方法的参数值及产生的操作。

表 30: negotiateValidateandEstablishProtocolTrust 参数和操作的描述

此表提供 `negotiateValidateandEstablishProtocolTrust` 参数和操作的描述

参数或操作	对于 **SIP** 请求	对于 **SIP** 响应
`req` 参数的值	入局请求	Null
`resp` 参数的值	Null	入局响应
对于有效响应凭证的操作	返回包含 `SC_OK` 及用户标识或唯一标识的 `TAIResult.status`	返回包含 `SC_OK` 及用户标识或唯一标识的 `TAIResult.status`
对于不正确响应凭证的操作	返回 `TAIResult` 及 4*xx* 状态	返回 `TAIResult` 及 4*xx* 状态

事件的顺序如下：

1. SIP 容器通过使用每个应用程序部署描述符中的规则将初始请求映射至应用程序；后续消息根据 *JSR289* 机制进行映射。

2. 如果任何应用程序需要安全性，SIP 容器则为该消息调用任何定义的 TAI 实现。
3. 如果该消息传递安全性，此容器会调用相应的应用程序。

您的 TAI 实现可以修改 SIP 消息，但由于此修改的消息在容器调用 TAI 之前已完成，因此在请求映射进程内将不可用。

${wlp.install.dir}/lib/com.ibm.ws.security.authentication.tai_1.0.10.jar 文件中定义的 `com.ibm.wsspi.security.tai.TAIResult` 类有三个用于创建 `TAIResult` 的静态方法。`TAIResult create` 方法采用 `int` 类型作为第一个参数。xigemaAS 要求结果是有效 HTTP 请求返回码并进行如下解释：

如果值为 `HttpServletResponse.SC_OK`，那么此响应告知 该 TAI 已完成协商。此响应还会告知 使用 `TAIResult` 中的信息来创建用户身份。

所创建 `TAIResults` 的含义显示在表 *2* 中。

表 31: TAIResults 的含义

下表列示 `TAIResults` 的含义

TAIResult	说明
public static TAIResult create(int status);	对 xigemaAS 指示状态。此状态不应该为 `SC_OK`，因为已提供标识信息。
public static TAIResult create(int status, String principal);	对 xigemaAS指示状态并提供此用户的用户标识或唯一标识。 将通过查询用户注册表来创建凭证。
public static TAIResult create(int status, String principal, Subject subject);	对 xigemaAS 指示状态、该用户的用户标识或唯一标识及定制主体集。如果主体集包含散列表，那么会忽略主体。主体集的内容成为最终用户主体集的一部分。

`public String getVersion();`

此方法返回当前 TAI 实现的版本号。

`public String getType();`

此方法的返回值是依赖于实现的。

3. 实现它之后编译该实现，以创建您自己的 SIP TAI `jar` 文件。
4. 遵循为 *xigemaAS* 配置 *TAI* 主题中描述的步骤 3 到 4 以将 xigemaAS 服务器配置为使用 SIP TAI。

为系统登录配置开发 JAAS 定制登录模块

对于 xigemaAS 服务器，存在多个用于配置系统登录的 Java™ 认证和授权服务 (JAAS) 插件点。xigemaAS 概要文件使用系统登录配置来认证入局请求。可以开发定制 JAAS 登录模块以将信息添加到系统登录配置的 Subject。

servlet 应用程序调用应用程序登录配置来获取基于特定认证信息的主体。如果编写插入 xigemaAS 概要文件应用程序登录或系统登录配置中的登录模块，那么必须开发登录配置逻辑以了解特定信息的存在时间及其用法。有关更多详细信息，请参阅 *JAAS* 配置（见第 1036 页）和 *JAAS* 登录模块（见第 1037 页）。

要为系统登录配置开发 JAAS 定制登录模块，请遵循下列过程中的步骤：

- 了解可用回调及其工作原理。

注：xigemaAS 仅支持下列回调：

```
callbacks[0] = new javax.security.auth.callback.NameCallback("Username: ");
callbacks[1] = new javax.security.auth.callback.PasswordCallback("Password: ", false);
callbacks[2] = new
 com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl("Credential Token:
 ");
callbacks[3] = new
 com.ibm.websphere.security.auth.callback.WSServletRequestCallback("HttpServletRequest:
 ")
callbacks[4] = new
 com.ibm.websphere.security.auth.callback.WSServletResponseCallback("HttpServletResponse:
 ");
callbacks[5] = new
 com.ibm.websphere.security.auth.callback.WSAppContextCallback("ApplicationContextCallback:
 ");
callbacks[6] = new WSRealmNameCallbackImpl("Realm Name: ", default_realm);
callbacks[7] = new WSX509CertificateChainCallback("X509Certificate[]: ");
callbacks[8] = wsAuthMechOidCallback = new WSAuthMechOidCallbackImpl("AuthMechOid: ");
```

- 了解共享状态变量及其工作原理。

  如果要访问 xigemaAS 在登录期间创建的对象，请参阅下列共享状态变量。。

com.ibm.wsspi.security.auth.callback.Constants.WSPRINCIPAL_KEY

指定 `java.security.Principal` 接口的已实现对象。此共享状态变量适用于只读用途。不要在定制登录模块的共享状态中设置此变量。缺省登录模块会设置此变量。

com.ibm.wsspi.security.auth.callback.Constants.WSCREDENTIAL_KEY

指定 `com.ibm.websphere.security.cred.WSCredential` 对象。此共享状态变量适用于只读用途。不要在定制登录模块的共享状态中设置此变量。缺省登录模块将设置此变量。

com.ibm.wsspi.security.auth.callback.Constants.WSSSOTOKEN_KEY

指定 `com.ibm.wsspi.security.token.SingleSignonToken` 对象。不要在定制登录模块的共享状态中设置此变量。缺省登录模块会设置此变量。

- 可选： 了解 xigemaAS 中定制 JAAS 登录模块的散列表。请参阅散列表登录模块（见第 1042 页），以了解更多详细信息。
- 开发使用回调和共享状态的样本定制登录模块。

  可以使用以下样本来了解如何使用某些回调和共享状态变量。

```
public class CustomCallbackLoginModule implements LoginModule {

  protected Map<String, ?> _sharedState;
  protected Subject _subject = null;
  protected CallbackHandler _callbackHandler;
  private final String customPrivateCredential = "CustomLoginModuleCredential";

  /**
   * Initialization of login module
   */
  public void initialize(Subject subject, CallbackHandler callbackHandler,
                         Map<String, ?> sharedState, Map<String, ?> options) {
     _sharedState = sharedState;
    _subject = subject;
    _callbackHandler = callbackHandler;
  }

  public boolean login() throws LoginException {
          try {
      AccessController.doPrivileged(new PrivilegedExceptionAction<Object>() {
        public Object run() throws Exception {
          _subject.getPrivateCredentials().add(customPrivateCredential);
          return null;
```

```
        }

      });
    } catch (PrivilegedActionException e) {
      throw new LoginException(e.getLocalizedMessage());
    }

    String username = null;
    char passwordChar[] = null;
    byte[] credToken = null;
    HttpServletRequest request = null;
    HttpServletResponse response = null;
    Map appContext = null;
    String realm = null;
    String authMechOid = null;
    java.security.cert.X509Certificate[] certChain = null;

    NameCallback nameCallback = null;
    PasswordCallback passwordCallback = null;
    WSCredTokenCallbackImpl wsCredTokenCallback = null;
    WSServletRequestCallback wsServletRequestCallback = null;
    WSServletResponseCallback wsServletResponseCallback = null;
    WSAppContextCallback wsAppContextCallback = null;
    WSRealmNameCallbackImpl wsRealmNameCallback = null;
    WSX509CertificateChainCallback wsX509CertificateCallback = null;
    WSAuthMechOidCallbackImpl wsAuthMechOidCallback = null;

    Callback[] callbacks = new Callback[9];
    callbacks[0] = nameCallback = new NameCallback("Username: ");
    callbacks[1] = passwordCallback = new PasswordCallback("Password: ", false);
    callbacks[2] = wsCredTokenCallback = new WSCredTokenCallbackImpl("Credential Token: ");
    callbacks[3] = wsServletRequestCallback = new
WSServletRequestCallback("HttpServletRequest: ");
    callbacks[4] = wsServletResponseCallback = new
WSServletResponseCallback("HttpServletResponse: ");
    callbacks[5] = wsAppContextCallback = new
WSAppContextCallback("ApplicationContextCallback: ");
    callbacks[6] = wsRealmNameCallback = new WSRealmNameCallbackImpl("Realm name:");
    callbacks[7] = wsX509CertificateCallback = new
WSX509CertificateChainCallback("X509Certificate[]: ");
    callbacks[8] = wsAuthMechOidCallback = new WSAuthMechOidCallbackImpl("AuthMechOid: ");

         try {
      _callbackHandler.handle(callbacks);
    } catch (Exception e) {
      // handle exception
    }

    if (nameCallback != null)
      username = nameCallback.getName();

    if (passwordCallback != null)
      passwordChar = passwordCallback.getPassword();

    if (wsCredTokenCallback != null)
      credToken = wsCredTokenCallback.getCredToken();

    if (wsServletRequestCallback != null)
      request = wsServletRequestCallback.getHttpServletRequest();

    if (wsServletResponseCallback != null)
      response = wsServletResponseCallback.getHttpServletResponse();

    if (wsAppContextCallback != null)
      appContext = wsAppContextCallback.getContext();

    if (wsRealmNameCallback != null)
      realm = wsRealmNameCallback.getRealmName();

    if (wsX509CertificateCallback != null)
      certChain = wsX509CertificateCallback.getX509CertificateChain();

    if (wsAuthMechOidCallback != null)
      authMechOid = wsAuthMechOidCallback.getAuthMechOid();

    _subject.getPrivateCredentials().add("username = " + username);
```

```
    _subject.getPrivateCredentials().add("password = " + String.valueOf(passwordChar));
    _subject.getPrivateCredentials().add("realm = " + realm);
    _subject.getPrivateCredentials().add("authMechOid = " + authMechOid.toString());

       return true;
  }

  public boolean commit() throws LoginException {
       return true;
  }

  public boolean abort() {
       return true;
  }

  public boolean logout() {
       return true;
  }

}
```

- 可选： 开发使用散列表登录的样本定制登录模块。

  可以使用以下样本来了解如何使用散列表登录。

```
package com.ibm.websphere.security.sample;

import java.util.Map;

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginException;
import javax.security.auth.spi.LoginModule;

import com.ibm.wsspi.security.token.AttributeNameConstants;

/**
 * Custom login module that adds another PublicCredential to the subject
 */
@SuppressWarnings("unchecked")
public class CustomHashtableLoginModule implements LoginModule {

  protected Map<String, ?> _sharedState;
  protected Map<String, ?> _options;

/**
 * Initialization of login module
 */
  public void initialize(
    Subject subject, CallbackHandler callbackHandler, Map<String, ?> sharedState,
 Map<String, ?> options) {
       _sharedState = sharedState;
      _options = options;
  }

  public boolean login() throws LoginException {
          try {
      java.util.Hashtable<String, Object> customProperties = (java.util.Hashtable<String,
 Object>)
        _sharedState.get(AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY);
      if (customProperties == null) {
        customProperties = new java.util.Hashtable<String, Object>();
      }

      customProperties.put(AttributeNameConstants.WSCREDENTIAL_USERID, "userId");
      // Sample of creating custom cache key
      customProperties.put(AttributeNameConstants.WSCREDENTIAL_CACHE_KEY, "customCacheKey");

      /*
       * Sample for creating user ID and security name
       * customProperties.put(AttributeNameConstants.WSCREDENTIAL_UNIQUEID, "userId");
       * customProperties.put(AttributeNameConstants.WSCREDENTIAL_SECURITYNAME,
 "securityName");
       * customProperties.put(AttributeNameConstants.WSCREDENTIAL_REALM, "realm");
```

```
         * customProperties.put(AttributeNameConstants.WSCREDENTIAL_GROUPS, "groupList");
         */
        /*
         * Sample for creating user ID and password
         * customProperties.put(AttributeNameConstants.WSCREDENTIAL_USERID, "userId");
         * customProperties.put(AttributeNameConstants.WSCREDENTIAL_PASSWORD, "password");
         */
       Map<String, java.util.Hashtable> mySharedState = (Map<String, java.util.Hashtable>)
 _sharedState;
       mySharedState.put(AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY,
 customProperties);
    } catch (Exception e) {
        throw new LoginException("LoginException: " + e.getMessage());
    }

       return true;
  }

  public boolean commit() throws LoginException {
       return true;
  }

  public boolean abort() {
       return true;
  }

  public boolean logout() {
       return true;
  }
}
```

将定制登录模块添加到 server.xml 文件的 WEB_INBOUND 和缺省 Java™ 认证和授权服务 (JAAS) 系统登录配置。将定制登录模块类放入 JAR 文件（例如 `customLoginModule.jar`），然后使 JAR 文件可供 xigemaAS 服务器使用。请参阅为 *xigemaAS* 概要文件配置 *JAAS* 定制登录模块（见第 1307 页）。

为 xigemaAS 开发定制 JASPIC 认证服务提供程序

通过创建用于实现 JSR 196: Java™ 容器认证服务提供程序接口规范中说明的必需接口的类，可开发定制 Java™ Authentication SPI for Containers (JASPIC) 认证服务提供程序。

查看 JSR 196: Java™ 容器认证服务提供程序接口规范中针对 JASPIC 认证服务提供程序和模块的特定接口实现需求。

xigemaAS 支持使用第三方认证服务提供程序，该提供程序符合 Java™ Authentication SPI for Containers (JASPIC) V1.1 中指定的 servlet 容器概要文件。

servlet 容器概要文件定义了一些接口，安全性运行时环境将这些接口与 xigemaAS 中的 Web 容器配合使用以在应用程序处理 Web 请求前后调用认证模块。仅当在安全配置中启用了 JASPIC 时，才会执行使用 JASPIC 模块的认证。

要开发定制认证服务提供程序，请创建用于实现 JSR 196: Java™ 容器认证服务提供程序接口规范中说明的必需接口的类。提供程序可以使用一个或多个认证模块进行认证。模块可以使用回调来执行认证，它们也可以将必需的用户身份信息手动添加至客户机主体集。

1. 创建用于实现 `javax.security.auth.message.config.AuthConfigProvider` 接口的类。

   `AuthConfigProvider` 实现类必须定义一个具有两个自变量的公用构造函数以及 `getServerAuthConfig` 公用方法：

```
import java.util.Map;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.config.AuthConfigFactory;
```

```
import javax.security.auth.message.config.AuthConfigProvider;
import javax.security.auth.message.config.ServerAuthConfig;

public class SampleAuthConfigProvider implements AuthConfigProvider {

        public SampleAuthConfigProvider(Map<String, String> properties,
 AuthConfigFactory factory) {
                ...
        }
        public ServerAuthConfig getServerAuthConfig(String layer, String
 appContext, CallbackHandler handler)
                throws AuthException {
                ...
        }
}
```

要由应用程序的 Web 模块处理的请求到达时，xigemaAS 将使用 `AuthConfigProvider` 实现类的实例。`getServerAuthConfig` 方法用于获取 `ServerAuthConfig` 实例。`authentication` 模块使用方法调用中的 `CallbackHandler` 自变量。

2. 创建用于实现 `javax.security.auth.message.config.ServerAuthConfig` 接口的类。

   `ServerAuthConfig` 实现类必须定义 `getAuthContextID` 和 `getAuthContext` 公用方法：

```
import java.util.Map;
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.config.ServerAuthConfig;
import javax.security.auth.message.config.ServerAuthContext;

public class SampleServerAuthConfig implements ServerAuthConfig {

        public String getAuthContextID(MessageInfo messageInfo) throws
 IllegalArgumentException {
                ...
        }
        public ServerAuthContext getAuthContext(String authContextID,
 Subject serviceSubject, Map properties)
                throws AuthException {
                ...
        }
}
```

   `ServerAuthConfig` 实现类中的 `getAuthContextID` 和 `getAuthContext` 方法用于获取 `ServerAuthContext` 实例。

3. 创建用于实现 `javax.security.auth.message.config.ServerAuthContext` 接口的类。

   `ServerAuthContext` 实现类必须定义 `validateRequest` 和 `secureResponse` 公用方法：

```
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.config.ServerAuthContext;

public class SampleServerAuthContext implements ServerAuthContext {
```

```
        public AuthStatus validateRequest(MessageInfo messageInfo, Subject
 clientSubject, Subject serviceSubject)
                throws AuthException {
                ...
        }
        public AuthStatus secureResponse(MessageInfo messageInfo, Subject
 serviceSubject)
                throws AuthException {
                ...
        }
}
```

ServerAuthContext 实现类中的 validateRequest 方法用于调用认证所接收 Web 请求消息的模块。如果认证结果成功，那么 Web 容器将分派目标 Web 模块在应用程序中处理的所接收的 Web 请求消息。如果认证结果不成功，那么将使用相应的响应状态来拒绝该请求。

4. 创建用于实现 javax.security.auth.message.module.ServerAuthModule 接口的类。

ServerAuthModule 实现类必须定义 initialize、validateRequest 和 secureResponse 公用方法：

```
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import javax.security.auth.message.MessagePolicy;
import javax.security.auth.message.module.ServerAuthModule;

public class SampleAuthModule implements ServerAuthModule {

        public void initialize(MessagePolicy requestPolicy, MessagePolicy
 responsePolicy, CallbackHandler handler, Map options)
                throws AuthException {
                ...
        }

        public AuthStatus validateRequest(MessageInfo messageInfo, Subject
 clientSubject, Subject serviceSubject)
                throws AuthException {
                ...
        }

        public AuthStatus secureResponse(MessageInfo messageInfo, Subject
 serviceSubject)
                throws AuthException {
                ...
        }

        public void cleanSubject(MessageInfo messageInfo, Subject
 subject)
                throws AuthException {
                ...
        }
}
```

ServerAuthContext 实现类调用 ServerAuthModule 实现类中的 initialize 方法以初始化 authentication 模块并使其与 ServerAuthContext 实例相关联。

此类中的 validateRequest 和 secureResponse 方法用于认证所接收的 javax.security.auth.message.MessageInfo 中包含的 javax.servlet.http.HttpServletRequest 和 javax.servlet.http.HttpServletResponse。这些方法可使用 initialize 方法中接收到的 CallbackHandler 实例以与 xigemaAS 安全运行时交互来验证用户密码，并与活动用户注册表交互以检索用户的唯一标识和组成员资格。检索到的数据放在客户机主体集中的专用凭证集的**散列表**中。CallbackHandler 的 xigemaAS 实现支持以下三个回调：

- CallerPrincipalCallback
- GroupPrincipalCallback
- PasswordValidationCallback

xigemaAS PasswordValidationCallback.getUsername() 和 CallerPrincipalCallback.getName() 获取的名称值相同。如果它们不相同，那么会产生不可预测的结果。CallbackHandler 的 handle() 方法依次处理该方法的自变量数组中给出的每个回调。因此，客户机主体集的专用凭证中的名称值集就是从所处理的最后一个回调获得的名称值集。

如果认证模块未使用 CallbackHandler，并且 validateRequest 返回成功状态，那么 xigemaAS 要求 Hashtable 实例与用户身份信息一起包含在 clientSubject 中，以使可执行定制登录以获取用户的凭证。可如以下示例中所示将此**散列表**添加至客户机主体集：

```
import java.util.Hashtable;
import java.util.String;
import javax.security.auth.Subject;
import javax.security.auth.message.AuthException;
import javax.security.auth.message.AuthStatus;
import javax.security.auth.message.MessageInfo;
import com.ibm.wsspi.security.registry.RegistryHelper;
import
 com.ibm.wsspi.security.token.AttributeNameConstants.AttributeNameConstants;

public AuthStatus validateRequest(MessageInfo messageInfo, Subject
 clientSubject, Subject serviceSubject)
       throws AuthException {
       ...
       UserRegistry reg = RegistryHelper.getUserRegistry(null);
       String uniqueid = reg.getUniqueUserID(username);

       Hashtable hashtable = new Hashtable();
       hashtable.put(AttributeNameConstants.WSCREDENTIAL_UNIQUEID,
 uniqueid);
       hashtable.put(AttributeNameConstants.WSCREDENTIAL_SECURITYNAME,
 username);
       hashtable.put(AttributeNameConstants.WSCREDENTIAL_PASSWORD,
 password);
       hashtable.put(AttributeNameConstants.WSCREDENTIAL_GROUPS,
 groupList); //optional
       clientSubject.getPrivateCredentials().add(hashtable);
       ...
}
```

有关**散列表**需求及定制登录的更多信息，请参阅开发用于系统登录配置的 *JAAS* 定制登录模块。

开发 Java™ Authorization Contract for Containers (JACC) 授权提供程序

可开发 JACC 提供程序以对 Java™ Platform Enterprise Edition (J2EE) 应用程序制定定制授权决策，方法是实现 xigemaAS 服务器中提供的 `com.ibm.wsspi.security.authorization.jacc.ProviderService` 接口。

缺省情况下，应用程序模块装入会延迟，直到处理针对该应用程序的请求，但是，准备处理应用程序之前，需要先处理应用程序中整个模块的安全性约束。延迟模块装入需要被禁用。下面显示如何对其进行禁用：

1. 对于 WebContainer：

   在 server.xml 文件中，需要设置以下元素：

   ```
   <webContainer deferServletLoad="false"/>
   ```

2. 对于 EJBContainer：

   在 server.xml 文件中，需要设置以下元素：

   ```
   <ejbContainer startEJBsAtAppStart="true"/>
   ```

👉 注：如果未设置以上元素，那么启动服务器时，完整安全性约束信息可能不会填充至第三方 JACC 提供程序。因此，第三方 JACC 提供程序可能无法强制实施正确的授权决策。

Java™ 容器授权合同规范 *JSR 115* 为授权提供程序定义接口。在 xigemaAS 概要文件服务器中，必须将 JACC 提供程序打包为用户功能部件。您的功能部件必须实现 `com.ibm.wsspi.security.authorization.jacc.ProviderService` 接口。

1. 创建 OSGi 组件，该组件提供用于实现 `com.ibm.wsspi.security.authorization.jacc.ProviderService` 接口的服务。

   `ProviderService` 接口定义以下两个方法：`getPolicy`（xigemaAS 概要文件运行时调用此方法以检索用于实现 `java.security.Policy` 抽象类的 Policy 的实例）和 `getPolicyConfigFactory`（xigemaAS 概要文件运行时调用此方法检索用于实现 `javax.security.jacc.PolicyConfigurationFactory` 抽象类的 `PolicyConfigurationFactory` 类的实例）。

   以下示例使用 OSGi 声明式服务注解：

   ```
   @package com.mycompany.jacc;

   import com.mycompany.jacc.MyAuthConfigProvider;
   import com.ibm.wsspi.security.authorization.jacc.ProviderService;

   // The property value of javax.security.jacc.policy.provider which defines
    the implementation class of Policy and
   // javax.security.jacc.PolicyConfigurationFactory.provider which defines
    the implementation class of PolicyConfigurationFactory, are required for
    propagating the properties to the xigemaAS runtime.

   @Component(service = ProviderService.class,
              immediate = true,
              property = {
                          "javax.security.jacc.policy.provider
                            =com.mycompany.jacc.MyPolicy",

    "javax.security.jacc.PolicyConfigurationFactory.provider
   ```

```
 =com.mycompany.jacc.MyPolicyConfigurationFactoryImpl"
                        }
          )
public class MyJaccProviderService implements ProviderService {
// This method called by the xigemaAS profile runtime
    // to get an instance of Policy class
    @Override
    public Policy getPolicy() {
        return new myPolicy();
    }

    // This method called by the xigemaAS profile runtime
    // to get an instance of PolicyConfigurationFactory class
    @Override
    public Policy getPolicyConfigurationFactory() {
        ClassLoader cl = null;
        PolicyConfigurationFactory pcf = null;
        System.setProperty(JACC_FACTORY, JACC_FACTORY_IMPL);
        try {
            cl = Thread.currentThread().getContextClassLoader();
            Thread.currentThread().setContextClassLoader(
                            this.getClass().getClassLoader());
            pcf =
 PolicyConfigurationFactory.getPolicyConfigurationFactory();
        } catch (Exception e) {
            return null;
        } finally {
            Thread.currentThread().setContextClassLoader(cl);
        }
        return pcf;
    }

    protected void activate(ComponentContext cc) {
        // Read provider config properties here if needed,
        // then pass them to the Provider ctor.
        // This example reads the properties from the OSGi
        // component definition.
        configProps = (MapString, String) cc.getProperties();
    }

    protected void deactivate(ComponentContext cc) {}
}
```

2. 将该组件与您的 JACC 提供程序一起打包成用户功能部件中的 OSGi 捆绑软件。
3. 确保您的功能部件包含 OSGi 子系统内容：`com.ibm.ws.javaee.jacc.1.5; location:="dev/api/spec/"`。
4. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。例如：

```
<featureManager>
   ...
   <feature>usr:myJaccProvider</feature>
</featureManager>
```

开发 customPasswordEncryption 提供者

可以开发 customPasswordEncryption 提供者以对 Java™ 平台企业修订版 (J2EE) 应用程序制定定制授权决策，方法是实现 xigemaAS 服务器中提供的 com.ibm.wsspi.security.crypto.CustomPasswordEncryption 接口。

1. 创建 OSGi 组件，该组件提供用于实现 com.ibm.wsspi.security.crypto.CustomPasswordEncryption 接口的服务。

   CustomPasswordEncryption 接口定义以下三种方法：decrypt（xigemaAS 服务器运行时调用该方法以对字符串解密）、encrypt（xigemaAS 服务器运行时调用该方法以对字符串加密）和 initialize（保留供将来使用）。

   以下示例使用 OSGi 声明式服务注释：

```
package com.mycompany.custom;

import org.osgi.service.component.ComponentContext;
import org.osgi.service.component.annotations.Activate;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.ConfigurationPolicy;
import org.osgi.service.component.annotations.Deactivate;
import org.osgi.service.component.annotations.Modified;

import com.ibm.wsspi.security.crypto.CustomPasswordEncryption;
import com.ibm.wsspi.security.crypto.EncryptedInfo;
import com.ibm.wsspi.security.crypto.PasswordDecryptException;
import com.ibm.wsspi.security.crypto.PasswordEncryptException;

/**
 */
@Component(service = CustomPasswordEncryption.class,
                immediate = true,
                name = "com.mycompany.CustomPasswordEncryptionImpl",
                configurationPolicy = ConfigurationPolicy.OPTIONAL,
                property = { "someKey=someValue" })
public class CustomPasswordEncryptionImpl implements
 CustomPasswordEncryption {

    @Activate
    protected synchronized void activate(ComponentContext cc, Map<String,
 Object> props) {
    }

    @Modified
    protected synchronized void modify(Map<String, Object> props) {
    }

    @Deactivate
    protected void deactivate(ComponentContext cc) {
    }

    /**
     * The encrypt operation takes a UTF-8 encoded String in the form of a
 byte[].
     * The byte[] is generated from String.getBytes("UTF-8"). An encrypted
 byte[]
     * is returned from the implementation in the EncryptedInfo object.
```

```
     * Additionally, a logically key alias is returned in EncryptedInfo so
which
     * is passed back into the decrypt method to determine which key was
used to
     * encrypt this password. The xigemaAS runtime has no
     * knowledge of the algorithm or key used to encrypt the data.
     *
     * @param decrypted_bytes
     * @return com.ibm.wsspi.security.crypto.EncryptedInfo
     * @throws com.ibm.wsspi.security.crypto.PasswordEncryptException
     **/
    @Override
    public EncryptedInfo encrypt(byte[] input) throws
PasswordEncryptException {
        byte[] output = null;
        String key = null;
        try {
            :
            <do some encryption>
            :
            return new EncryptedInfo(output, key);
        } catch (Exception e) {
            throw new PasswordEncryptException("Exception is caught", e);
        }
    }

    /**
     * The decrypt operation takes the EncryptedInfo object containing a
byte[]
     * and the logical key alias and converts it to the decrypted byte[].
The
     * xigemaAS runtime will convert the byte[] to a String
     * using new String (byte[], "UTF-8");
     *
     * @param info
     * @return byte[]
     * @throws PasswordEncryptException
     * @throws com.ibm.wsspi.security.crypto.PasswordDecryptException
     **/
    @Override
    public byte[] decrypt(EncryptedInfo info) throws
PasswordDecryptException {
        byte[] input = info.getEncryptedBytes();
        String key = info.getKeyAlias();
        byte[] output = null;
        try {
            :
            <do some decryption>
            :
            return output;
        } catch (Exception e) {
            throw new PasswordEncryptException("Exception is caught", e);
        }
    }

    /**
     * This is reserved for future use and is currently not called by the
     * WebSphere Application Server runtime.
     *
     * @param initialization_data
```

```
     **/
    @SuppressWarnings("rawtypes")
    @Override
    public void initialize(Map initialization_data) {}

}
```

2. 将该组件封装到作为用户功能部件的组成部分的 OSGi 捆绑软件中。确保该捆绑软件包含 OSGi 服务清单。

   以下示例显示 OSGi 服务清单的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
 name="com.mycompany.custom.CustomPasswordEncryptionImpl"
 configuration-policy="optional" immediate="true" activate="activate"
 deactivate="deactivate" modified="modify">
  <implementation
 class="com.mycompany.custom.CusomPasswordEncryptionImpl"/>
  <service>
    <provide
 interface="com.ibm.wsspi.security.crypto.CustomPasswordEncryption"/>
    </service>
  <property name="<someKey>" type="String" value="<someValue>"/>
</scr:component>
```

3. 确保您的功能部件清单包含带有 start-phase:="SERVICE_EARLY" 的 OSGi 子系统内容。

   例如：

```
Manifest-Version: 1.0
IBM-Feature-Version: 2
IBM-ShortName: customPasswordEncryption-1.0
Subsystem-Type: osgi.subsystem.feature
Subsystem-Version: 1.0.0
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: customPasswordEncryption-1.0;visibility:=public
Subsystem-Content:
com.mycompany.custom; version="[1,1.0.100)"; start-phase:="SERVICE_EARLY"
```

4. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。

```
<featureManager>
   ...
   <feature>usr:customPasswordEncryption-1.0</feature>
</featureManager>
```

定制应用程序登录以使用 JAAS 来执行身份断言

可以使用 Java™ 认证和授权服务 (JAAS) 登录框架来创建 JAAS 登录配置，可以使用此 JAAS 登录配置来登录到 xigemaAS 上的身份断言。

通过使用信任验证来配置身份断言，应用程序可以使用 JAAS 登录配置来执行程序化身份断言。有关更多详细信息，请参阅 *IdentityAssertionLoginModule*。

1. 将信任验证委托给用户实现的插入点。

由定制登录模块完成信任验证。此定制登录模块执行任何所需的信任验证，然后设置共享状态下要传递到身份断言登录模块的信任和身份信息。下列共享状态键中需要映射：

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state
```

如果缺少状态，那么 IdentityAssertionLoginModule 类会报告 WSLoginFailedException 问题。

共享状态键中的映射必须包含具有下列键名称的信任密钥：

```
com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trust
```

如果此键设为 true，那么建立信任。如果此键设置为 false，那么不会建立任何信任，并且 IdentityAssertionLoginModule 类会创建 WSLoginFailedException 问题。

共享状态键中的映射也必须设置下列其中一个资源：

- 身份键。可以在下列键中设置 java.security.Principal：

  ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal
  ```

- java.security.cert.X509Certificate[]。可以在下列键中设置此证书：

  ```
  com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certficates
  ```

如果同时提供主体和证书，那么会使用主体，并报告警告。

2. 为应用程序登录创建 JAAS 配置。

   JAAS 配置将包含用户实现的信任验证定制登录模块和 IdentityAssertionLoginModule 类。然后，要设置应用程序登录配置，请在 server.xml 文件中配置下列代码：

   ```
   <jaasLoginContextEntry id="CustomIdentityAssertion" name="CustomIdentityAssertion"
                          loginModuleRef="customIdentityAssertion,identityAssertion" />
   <jaasLoginModule id="customIdentityAssertion"
                    className="com.ibm.ws.security.authentication.IdentityAssertionLoginModule"
                    controlFlag="REQUIRED" libraryRef="customLoginLib"/>
    <library id="customLoginLib">
       <fileset dir="${server.config.dir}" includes="IdentityAssertionLoginModule.jar"/>
    </library>
   ```

   应用程序将使用此 JAAS 配置来执行身份断言。

3. 执行可编程身份断言。

   现在，程序可以使用 JAAS 登录配置来执行程序化身份断言。应用程序可为步骤 2 中所创建的 JAAS 配置创建登录上下文，然后使用将断言的身份来登录到该登录上下文。如果登录成功，就会在当前运行进程中设置该身份。以下示例说明了此过程：

   ```
   NameCallback handler = new NameCallback(new MyPrincipal("Joe"));
   LoginContext lc = new LoginContext("customIdentityAssertion", handler);
   lc.login();  //assume successful
   Subject s = lc.getSubject();
   WSSubject.setRunAsSubject(s);
   // From here on , the runas identity is "Joe"
   ```

   👉 注：MyPrincipal 类是示例中 java.security.Principal 接口的实现。

通过使用 JAAS 登录框架和两个用户实现的登录模块，可以创建可用来登录到身份断言的 JAAS 登录配置。

为 xigemaAS 开发定制用户注册表

要开发定制用户注册表类，可以实现 xigemaAS 概要文件服务器中所提供的 `com.ibm.websphere.security.UserRegistry` 接口。

UserRegistry 接口是服务编程接口 (SPI)，可以实现对几乎任何类型的帐户存储库的支持。

1. 实现定制用户注册表。
2. 将实现类转换为 OSGi 服务。可以采用下列方式来执行转换：
   - 将 UserRegistry 类转换为声明式服务 (DS) 组件。有关更多信息，请参阅向 *OSGi* 声明式服务声明服务（见第 1248 页）。
   - 编写作为 DS 组件的新 UserRegistry 类并将其委派给 UserRegistry 类。
   - 使用 OSGi 核心 API 直接在服务注册表 (SR) 中注册 UserRegistry 类。有关更多信息，请参阅使用 *OSGi* 服务注册表（见第 1244 页）。
3. 将定制用户注册表打包为 OSGi 捆绑软件并导出 UserRegistry 服务。
4. 创建功能部件清单来包含 OSGi 捆绑软件。

   有关更多信息，请参阅产品扩展（见第 1028 页）。
5. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。

   例如：

```
<featureManager>
   ...
   <feature>usr:customRegistrySample-1.0</feature>
</featureManager>
```

开发 JAAS 定制登录模块以进行数据库认证

可开发 Java™ 认证和授权服务 (JAAS) 定制登录模块，以添加用于向数据库认证的用户名和密码。

可开发 JAAS 定制登录模块，创建需要认证的数据库连接时可调用此模块。JAAS 定制登录模块用于创建密码凭证，此凭证包含用户名、密码和受管连接工厂。此登录模块必须将密码凭证添加至主体集的专用凭证集，此凭证集用于向数据库认证。

1. 创建用于实现 `javax.security.auth.spi.LoginModule` 接口的类。
2. 在初始化方法中保存必需字段。例如：

```
/** {@inheritDoc} */
@SuppressWarnings("unchecked")
@Override
public void initialize(Subject subject, CallbackHandler callbackHandler,
 Map<String, ?> sharedState, Map<String, ?> options) {
  this.callbackHandler = callbackHandler;
  this.subject = subject;
  this.sharedState = (Map<String, Object>) sharedState;
  this.options = options;
}
```

3. 在登录方法中处理 `WSManagedConnectionFactoryCallback` 和 `WSMappingPropertiesCallback` 回调。例如：

```
/** {@inheritDoc} */
@Override
```

```
public boolean login() throws LoginException {
  ...
  Callback callbacks[] = new Callback[2];
  callbacks[0] = new WSManagedConnectionFactoryCallback("Target
 ManagedConnectionFactory: ");
  callbacks[1] = new WSMappingPropertiesCallback("Mapping Properties
 (HashMap): ");
  callbackHandler.handle(callbacks);
```

4. 在登录方法中获取受管连接工厂和属性。例如：

```
// The method getManagedConnectionFacotry must be used as shown for
 compatiblity with WAS Classic
ManagedConnectionFactory managedConnectionFactory =
 ((WSManagedConnectionFactoryCallback)
 callbacks[0]).getManagedConnectionFacotry();
Map properties = ((WSMappingPropertiesCallback)
 callbacks[1]).getProperties();
```

5. 根据认证数据别名或某些其他条件获取用户名和密码。例如：

```
String alias = (String)
 properties.get(com.ibm.wsspi.security.auth.callback.Constants.MAPPING_ALIAS);
String user = getUser(alias); // Implementation specific
char[] password = getPassword(alias); // Implementation specific
```

6. 使用用户名和密码创建 `javax.resources.spi.PasswordCredential` 对象并设置受管连接工厂。例如：

```
javax.resource.spi.security.PasswordCredential passwordCredential = new
 PasswordCredential(user, password);
passwordCredential.setManagedConnectionFactory(managedConnectionFactory);
```

7. 将密码凭证添加至落实方法中的主体集。例如：

```
/** {@inheritDoc} */
@Override
public boolean commit() throws LoginException {
  // Verify that the login was successful before adding the
 PasswordCredential to the subject.
  subject.getPrivateCredentials().add(passwordCredential);
    return true;
}
```

开发程序化登录以获取认证数据

可以使用 Java 认证和授权服务 (JAAS) 登录框架以通过应用程序获取认证数据。

通过使用 `DefaultPrincipalMapping` JAAS 上下文条目名称在专用凭证集（包含为 `authData` 元素配置的用户名和密码）中获取具有 `javax.resource.spi.security.PasswordCredential` 实例的主体集对象，应用程序可执行 JAAS 程序化登录。

1. 在 server.xml 文件中添加 `appSecurity-2.0`、`passwordUtilities-1.0` 和 `jca-1.7` 功能部件。还可以添加 `appSecurity-2.0`、`passwordUtilities-1.0` 和 `jca-1.6`。例如：

```
<featureManager>
   <feature>appSecurity-2.0</feature>
   <feature>passwordUtilities-1.0</feature>
```

```
    <feature>jca-1.7</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 authData 元素。例如：

```
<authData id="myAuthData" user="myUser" password="myPassword"/> <!--
 password can also be encoded -->
```

对配置中的密码进行编码。可使用 securityUtility encode 命令获取编码值。

3. 使用 DefaultPrincipalMapping JAAS 登录上下文条目名称通过应用程序 servlet 或企业 bean 执行程序化登录（将映射别名替换为您需要的别名）。例如：

```
HashMap map = new HashMap();
map.put(com.ibm.wsspi.security.auth.callback.Constants.MAPPING_ALIAS,
 "myAuthData"); // Replace value with your alias.
CallbackHandler callbackHandler = new
 com.ibm.wsspi.security.auth.callback.WSMappingCallbackHandler(map, null);
LoginContext loginContext = new LoginContext("DefaultPrincipalMapping",
 callbackHandler);
loginContext.login();
Subject subject = loginContext.getSubject();
Set<javax.resource.spi.security.PasswordCredential> creds =
 subject.getPrivateCredentials(javax.resource.spi.security.PasswordCredential.class);
PasswordCredential passwordCredential = creds.iterator().next();
```

注：为简单起见，不显示错误处理。如果所请求认证别名不存在或格式错误，那么将返回 javax.security.auth.login.LoginException。

4. 从 PasswordCredential 获取用户名和密码。例如：

```
String userName = passwordCredential.getUserName();
char[] password = passwordCredential.getPassword();
// Do something with the userName and password.
```

5. 如果启用了 Java 2 安全性，那么必须对应用程序授予 javax.security.auth.PrivateCredentialPermission。例如，在应用程序的 META-INF/permissions.xml 文件中授予该许可权以访问 PasswordCredential 对象：

```
<?xml version="1.0" encoding="UTF-8"?>
<permissions xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
permissions_7.xsd" version="7">

  <permission>
    <class-name>javax.security.auth.PrivateCredentialPermission</class-
name>
    <name>javax.resource.spi.security.PasswordCredential * "*"</name>
    <actions>read</actions>
  </permission>

  <!-- Other permissions -->

</permissions>
```

有关 Java 2 安全性的更多信息，请参阅 *xigemaAS: Java 2 安全性*。

开发定制线程身份服务

要开发定制线程身份服务类，需实现xigemaAS 服务器中所提供的 `com.ibm.wsspi.kernel.security.thread.ThreadIdentityService` 接口。`ThreadIdentityService` 接口是服务编程接口 (SPI)，支持接收用户身份交换的通知。

1. 通过实现 `ThreadIdentityService` 接口来创建定制线程身份服务。
2. 将实现类转换为 OSGi 服务。

   可以采用以下任一方法来执行转换：

   a. 将 `ThreadIdentityService` 类转换为声明服务 (DS) 组件。

      有关更多信息，请参阅向 *OSGi* 声明式服务声明服务（见第 1248 页）。

   b. 编写作为 DS 组件的新类 `ThreadIdentityService`，并将其委派给 `ThreadIdentityService` 类。使用 OSGi 核心 API 直接在服务注册表 (SR) 中注册 `ThreadIdentityService` 类。

      有关更多信息，请参阅使用 *OSGi* 服务注册表（见第 1244 页）。
3. 将定制线程身份服务封装为 OSGi 捆绑软件并导出 `ThreadIdentityService` 服务。
4. 创建功能部件清单来包含 OSGi 捆绑软件。

   有关更多信息，请参阅产品扩展（见第 1028 页）。
5. 将功能部件安装到用户产品扩展位置之后，使用功能部件名称来配置 server.xml 文件。

```
<featureManager>
 ...
  <feature>usr:sampleThreadIdentityService-1.0</feature>
</featureManager>
```

2.6.12 Web Service 安全性

安全性是支持 QoS 的 Web Service 的一项主要的服务质量 (QoS) 需求。可以通过多种方式为 Web Service 提供安全性，其中两种方式包括传输层安全性和消息级别安全性。

在传输层保护 Web Service

传输级别安全性基于安全套接字层 (SSL) 或传输层安全性 (TLS)，用于保护点到点的 HTTP 消息内容。

在消息级别保护 Web Service

消息级别的安全性保护 Web Service 的 HTTP 消息中所含的 SOAP 内容。

在传输级别保护 Web Service

传输级别安全性是已知并且经常使用的机制，用于保护 HTTP 因特网和内部网通信。可以使用传输级别安全性保护 Web Service 消息。传输级别安全性功能独立于消息级别安全性 (WS-Security) 或 HTTP 基本认证所提供的功能。可以使用传输级别安全性绑定来保护 Web Service 客户机和 Web Service 提供程序之间的通信。

ibm-ws-bnd.xml 文件必须位于基于 Web 的 Web Service 应用程序（WAR 文件）的 /WEB-INF 目录中，或基于 EJB 的 Web Service 应用程序（JAR 文件）的 /META-INF 目录中。

注：如果客户机正在xigemaAS 概要文件应用程序客户机容器中运行，那么 ibm-ws-bnd.xml 文件必须在客户机 EAR 的 JAR 文件的 /META-INF 目录中。

传输级别安全性基于在 HTTP 下运行的安全套接字层 (SSL) 或传输层安全性 (TLS)。

SSL 和 TLS 提供安全性功能，它包含认证、数据保护和对安全 HTTP 连接的密码令牌支持。要用 HTTPS 运行，服务端口地址必须以 `https://` 格式表示。当您使用 SSL 和 TLS 时，会确认传输数据的完整性和机密性，其包括 SOAP 消息和 HTTP 基本认证。

xigemaAS 概要文件使用 Java™ 安全套接字扩展 (JSSE) 包来支持 SSL 和 TLS。下列安全性配置支持在 xigemaAS 概要文件中保护 Web Service：

- 安全 HTTP
- 基本认证
- 客户机证书

对于您可以在 ibm-ws-bnd.xml 文件中配置的所有可用元素，请参阅 *ibm-ws-bnd.xml 文件*（见第 1538 页）。

启用 SSL 通信以访问 Web Service

可以配置 SSL 通信以便客户机应用程序访问 Web Service。

在启用 SSL 通信以访问 Web Service 之前，您必须满足下列先决条件：

- 为客户机和提供程序生成一对自签名的公用密钥和专用密钥：

```
keytool -genkey -alias default -keystore myKey.jks -dname "CN=myServer, O=VSETTAN, C=CN"
    -storepass passw0rd -keypass passw0rd -storetype jks  -validity 1000 -keyalg RSA
```

- 将 myKey.jks 文件复制到 xigemaAS 服务器的 ${server.config.dir}/resources/security 目录。

👉 注：可以在 Java™ 安装目录中找到 `keytool` 实用程序。

如果您需要将 Web Service 客户机应用程序与安全 HTTP 协议一起使用来访问受保护的 Web Service 资源，那么将按照 SSL 规范对所有消息进行加密。

1. 在 server.xml 文件中启用 `jaxws-2.2`、`servlet-3.0`（或 `servlet-3.1`）和 `appSecurity-2.0` 功能部件。

```
<featureManager>
  <feature>jaxws-2.2</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 SSL 元素。

```
<sslDefault sslRef="customizeSSLConfig" />
<ssl id="customizeSSLConfig" keyStoreRef="serverKeyStore" trustStoreRef="serverTrustStore" />
<keyStore id="serverKeyStore" location="myKey.jks" type="JKS" password="passw0rd" />
<keyStore id="serverTrustStore" location="myKey.jks" type="JKS" password="passw0rd" />
```

3. 通过指定 Web Service 端点来配置服务提供程序。

   a. 创建 Web Service。

```
@WebService(serviceName = "SayHelloPojoService",
            portName = "SayHelloPojoPort")
public class SayHelloPojoService implements SayHelloService {
 ...
}

@WebService(serviceName = "SayHelloStatelessService",
            portName = "SayHelloStatelessPort",
            endpointInterface =
 "com.ibm.ws.jaxws.transport.server.security.SayHelloService")
```

```
@Stateless(name = "SayHelloSessionBean")
public class SayHelloStatelessService implements SayHelloLocal {
  ...
}
```

b. 为服务提供程序配置 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee
    http://websphere.ibm.com/xml/ns/javaee/ibm-ws-bnd_1_0.xsd "
  version="1.0">
  <http-publishing>
  <webservice-security>
   <security-constraint>
    <web-resource-collection>
     <web-resource-name>All</web-resource-name>
     <url-pattern>/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
    </web-resource-collection>
    <user-data-constraint>
     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
   </security-constraint>
  </webservice-security>
 </http-publishing>
</webservices-bnd>
```

注：ibm-ws-bnd.xml 文件必须位于 Web 应用程序的 /WEB-INF 目录中，或者位于基于 EJB 的 Web Service 应用程序（JAR 归档）的 /META-INF 目录中。

4. 通过指定 Web Service 端点来配置服务客户机。例如，客户机应用程序是一个名为 TransportSecurityClient.war 的 Web 应用程序。

a. 在 server.xml 文件中配置客户机应用程序。

```
<application id="TransportSecurityClient" name="TransportSecurityClient"
    location="TransportSecurityClient.war"
    context="TransportSecurityClient" type="war" />
```

b. 配置客户机应用程序的 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd id="idvalue0" version="1.0" xmlns="http://websphere.ibm.com/xml/ns/
javaee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee

    http://websphere.ibm.com/xml/ns/javaee/ibm-ws-bnd_1_0.xsd " >
  <!-- POJO service reference binding-->
  <service-ref name="service/SayHelloPojoService">
    <port name="SayHelloPojoPort"
    namespace="http://ibm.com/ws/jaxws/transport/security/"
    ssl-ref="customizeSSLConfig"/>
    <properties http.conduit.tlsClientParameters.disableCNCheck="true" />
 </service-ref>
  <!-- Stateless service reference binding-->
  <service-ref name="service/SayHelloStatelessService">
    <port name="SayHelloStatelessPort"
    namespace="http://ibm.com/ws/jaxws/transport/security/"
    ssl-ref="customizeSSLConfig"/>
    <properties http.conduit.tlsClientParameters.disableCNCheck="true" />
</service-ref>
</webservices-bnd>
```

👉 注：

- ibm-ws-bnd.xml 文件必须位于客户机 Web 应用程序的 /WEB-INF 目录中。
- ssl-ref 属性的值必须与 server.xml 文件中的 `ssl` 元素的标识值相匹配。
- 如果在 ibm-ws-bnd.xml 文件中未指定 ssl-ref 属性，那么 Web Service 引擎将使用 xigemaAS 概要文件中的缺省 SSL 配置（如果此配置存在）。
- http.conduit.tlsClientParameters.disableCNcheck 属性用来控制是否验证远程服务器。在生产环境中对此属性使用 *false*，因为如果此属性是 *true*，那么将忽略 hostName 验证。

c. 通过 WSDL 位置来生成客户机存根。

```
@WebServiceClient(name = "SayHelloPojoService",
                  targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
                  wsdlLocation = "https://localhost:8020/TransportSecurityProvider/
unauthorized/employPojoService?wsdl")
public class SayHelloPojoService extends Service
{...}

@WebServiceClient(name = "SayHelloStatelessService",
    targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
    wsdlLocation = "https://localhost:8020/TransportSecurityProvider/unauthorized/
EmployStatelessService?wsdl")
public class SayHelloStatelessService extends Service
{...}
```

d. 使用 `@WebServiceRef` 注解将 Web Service 插入到 Servlet。例如，`TestJaxWsTransportSecurityServlet`。

```
@WebServiceRef(name = "service/SayHelloPojoService")
SayHelloPojoService pojoService;

@WebServiceRef(name = "service/SayHelloStatelessService")
SayHelloStatelessService statelessService;
```

启用基本认证以访问 Web Service

可以配置基本认证以便客户机应用程序访问 Web Service。

如果您需要将 Web Service 客户机应用程序与基本认证一起使用，以访问受保护的 Web Service 资源，那么客户机在与服务提供程序通信时，必须在请求中提供用户名和密码。

1. 在 server.xml 文件中启用 `jaxws-2.2`、`servlet-3.0`（或 `servlet-3.1`）和 `appSecurity-2.0` 功能部件。

```
<featureManager>
  <feature>jaxws-2.2</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 在 server.xml 文件中配置登录域，并将此域绑定至服务提供程序。

```
<application id="TransportSecurityProvider" name="TransportSecurityProvider"
             location="TransportSecurityProvider.war" type="ear">
        <application-bnd>
            <security-role name="Employee">
                <user name="employee0" />
                <group name="employeeGroup" />
            </security-role>
            <security-role name="Manager">
                <user name="manager0" />
            </security-role>
            <security-role name="AllAuthenticated">
                <special-subject type="ALL_AUTHENTICATED_USERS" />
```

```
            </security-role>
        </application-bnd>
     </application>
    <basicRegistry id="basic" realm="BasicRealm">
        <user name="employee0" password="emp0pwd" />
        <user name="employee1" password="emp1pwd" />
        <user name="manager0" password="mgr0pwd" />
        <group name="employeeGroup">
            <member name="employee0" />
            <member name="employee1" />
        </group>
    </basicRegistry>
```

3. 通过指定 Web Service 端点来配置服务提供程序。

   a. 创建 Web Service。

```
@WebService(serviceName = "SayHelloPojoService",
            portName = "SayHelloPojoPort")
public class SayHelloPojoService implements SayHelloService {
 ...
}

@WebService(serviceName = "SayHelloStatelessService",
            portName = "SayHelloStatelessPort",
            endpointInterface =
 "com.ibm.ws.jaxws.transport.server.security.SayHelloService")
@Stateless(name = "SayHelloSessionBean")
public class SayHelloStatelessService implements SayHelloLocal {
 ...
}
```

   b. 为服务提供程序配置 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
  <http-publishing>
  <webservice-security>
   <security-constraint>
    <web-resource-collection>
          <web-resource-name>Only Managers</web-resource-name>
          <url-pattern>/manager/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
    </web-resource-collection>
        <auth-constraint id="AuthConstraint_manager">
     <role-name>Manager</role-name>
    </auth-constraint>
   </security-constraint>
   <security-constraint>
    <web-resource-collection>
          <web-resource-name>Employees</web-resource-name>
          <url-pattern>/employee/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
    </web-resource-collection>
        <auth-constraint id="AuthConstraint_employee">
     <role-name>Employee</role-name>
    </auth-constraint>
   </security-constraint>
   <!-- SECURITY ROLES -->
      <security-role id="Staff">
    <role-name>Employee</role-name>
    <role-name>Manager</role-name>
   </security-role>
   <!-- AUTHENTICATION METHOD: Basic authentication -->
      <login-config id="LoginConfig">
    <auth-method>BASIC</auth-method>
        <realm-name>Authentication</realm-name>
   </login-config>   </webservice-security>
```

```
 </http-publishing>
</webservices-bnd>
```

👉 注:

- ibm-ws-bnd.xml 文件必须位于 Web 应用程序的 /WEB-INF 目录中，或者位于基于 EJB 的 Web Service 应用程序（JAR 归档）的 /META-INF 目录中。
- ibm-ws-bnd.xml 文件中的 `login-config` 元素仅在基于 EJB 的 Web Service 应用程序（JAR 归档）中生效。对于 Web 应用程序，将忽略 `login-config` 元素，将使用 web.xml 文件中的同一元素的值。

4. 通过指定 Web Service 端点来配置服务客户机。例如，客户机应用程序是一个名为 TransportSecurityClient.war 的 Web 应用程序。

   a. 在 server.xml 文件中配置客户机应用程序。

```
<application id="TransportSecurityClient" name="TransportSecurityClient"
    location="TransportSecurityClient.war"
    context="TransportSecurityClient" type="war" />
```

   b. 配置客户机应用程序的 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd" version="1.0">
  <!-- POJO service reference binding-->
  <service-ref name="service/SayHelloPojoService">
    <port name="SayHelloPojoPort"
        namespace="http://ibm.com/ws/jaxws/transport/security/"
        username="employee1"
        password="{xor}OjIvbi8oOw=="
    />
 </service-ref>
  <!-- Stateless service reference binding-->
  <service-ref name="service/SayHelloStatelessService">
    <port name="SayHelloStatelessPort"
        namespace="http://ibm.com/ws/jaxws/transport/security/"
        username="employee1"
        password="{xor}OjIvbi8oOw=="
    />
 </service-ref>
</webservices-bnd>
```

👉 注:

- ibm-ws-bnd.xml 文件必须位于客户机 Web 应用程序的 /WEB-INF 目录中。
- username 和 password 属性的值必须与 server.xml 文件中 `basicRegistry` 元素的用户名和密码相匹配。可以使用 *`securityUtility`* 命令对密码编码。

   c. 通过使用 WSDL 位置来生成客户机存根。

```
@WebServiceClient(name = "SayHelloPojoService",
                  targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
                  wsdlLocation = "https://localhost:8020/TransportSecurityProvider/
unauthorized/employPojoService?wsdl")
public class SayHelloPojoService
               extends Service
{...}

@WebServiceClient(name = "SayHelloStatelessService",
    targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
    wsdlLocation = "https://localhost:8020/TransportSecurityProvider/unauthorized/
EmployStatelessService?wsdl")
public class SayHelloStatelessService
               extends Service
```

```
{...}
```

d. 使用 `@WebServiceRef` 注解将 Web Service 插入到 Servlet。例如，`TestJaxWsTransportSecurityServlet`。

```
@WebServiceRef(name = "service/SayHelloPojoService")
SayHelloPojoService pojoService;

@WebServiceRef(name = "service/SayHelloStatelessService")
SayHelloStatelessService statelessService;
```

启用客户机证书认证以访问 Web Service

可以配置客户机证书认证以便客户机应用程序访问 Web Service。

在启用客户机证书认证以访问 Web Service 之前，您必须满足下列先决条件：

- 为提供程序生成一对自签名的公用密钥和专用密钥。

```
keytool -genkey -alias default -keystore serverKey.jks -dname "CN=myServer, O=VSETTAN, C=CN"
    -storepass passw0rd -keypass passw0rd -storetype jks  -validity 1000 -keyalg RSA
```

- 导出缺省证书，并将该证书导入到信任库。

```
keytool -export -alias default -file myserver.cer -keystore serverKey.jks
    -storepass passw0rd -storetype jks
keytool -import -file myserver.cer -alias default -keystore clientTrust.jks
    -storepass passw0rd -keypass passw0rd -storetype jks
```

- 为客户机生成两对自签名的公用密钥和专用密钥。

```
keytool -genkey -alias user0 -keystore clientKey.jks -dname "CN=employee0, O=VSETTAN, C=CN"
    -storepass passw0rd -keypass passw0rd -storetype jks  -validity 1000 -keyalg RSA
    keytool -genkey -alias admin0 -keystore clientKey.jks -dname "CN=manager0, O=VSETTAN,
 C=CN"
    -storepass passw0rd -keypass passw0rd -storetype jks  -validity 1000 -keyalg RSA
```

- 导出两个别名的证书，然后将证书导入到信任库。

```
keytool -export -alias user0 -file user0.cer -keystore clientKey.jks -storepass passw0rd -storetype jks
keytool -export -alias admin0 -file admin0.cer -keystore clientKey.jks -storepass passw0rd -storetype jks
keytool -import -file user0.cer -alias user0 -keystore serverTrust.jks -storepass passw0rd -keypass
 passw0rd -storetype jks
keytool -import -file admin0.cer -alias admin0 -keystore serverTrust.jks -storepass passw0rd -keypass
 passw0rd -storetype jks
```

- 将 serverKey.jks、serverTrust.jks、clientKey.jks 和 clientTrust.jks 文件复制到 ${server.config.dir}/resources/security 目录中。

👉 注：可以在 Java™ 安装目录中找到 `keytool` 实用程序。

如果您需要将 Web Service 客户机应用程序与客户机证书认证一起使用，以访问受保护的 Web Service 资源，那么客户机必须在请求中提供有效证书，并且必须使用 HTTPS 与服务提供程序进行通信。

1. 在 server.xml 文件中启用 `jaxws-2.2`、`servlet-3.0`（或 `servlet-3.1`）和 `appSecurity-2.0` 功能部件。

```
<featureManager>
  <feature>jaxws-2.2</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 SSL 元素以及由客户机定制的 SSL 元素。

```
<!-- Server SSL configuration -->
<ssl id="defaultSSLConfig" keyStoreRef="serverKeyStore" trustStoreRef="serverTrustStore"
  clientAuthenticationSupported="true"/>
<keyStore id="serverKeyStore" location="serverKey.jks" type="JKS" password="passw0rd" />
<keyStore id="serverTrustStore" location="serverTrust.jks" type="JKS" password="passw0rd" />

<!-- customize SSL configuration -->
<ssl id="customizeSSLConfig" keyStoreRef="clientKeyStore" trustStoreRef="clientTrustStore" /
>
<keyStore id="clientKeyStore" location="clientKey.jks" type="JKS" password="passw0rd" />
<keyStore id="clientTrustStore" location="clientTrust.jks" type="JKS" password="passw0rd" />
```

3. 在 server.xml 文件中配置登录域，并将此域绑定至服务提供程序。

```
<application id="TransportSecurityProvider" name="TransportSecurityProvider"
             location="TransportSecurityProvider.war" type="ear">
        <application-bnd>
            <security-role name="Employee">
                <user name="employee0" />
                <group name="employeeGroup" />
            </security-role>
            <security-role name="Manager">
                <user name="manager0" />
            </security-role>
            <security-role name="AllAuthenticated">
                <special-subject type="ALL_AUTHENTICATED_USERS" />
            </security-role>
        </application-bnd>
     </application>
    <basicRegistry id="basic" realm="BasicRealm">
        <user name="employee0" password="emp0pwd" />
        <user name="employee1" password="emp1pwd" />
        <user name="manager0" password="mgr0pwd" />
        <group name="employeeGroup">
            <member name="employee0" />
            <member name="employee1" />
        </group>
    </basicRegistry>
```

4. 配置服务提供程序。

   a. 创建 Web Service。

```
@WebService(serviceName = "SayHelloPojoService",
            portName = "SayHelloPojoPort")
public class SayHelloPojoService implements SayHelloService {
 ...
}

@WebService(serviceName = "SayHelloStatelessService",
            portName = "SayHelloStatelessPort",
            endpointInterface =
 "com.ibm.ws.jaxws.transport.server.security.SayHelloService")
@Stateless(name = "SayHelloSessionBean")
public class SayHelloStatelessService implements SayHelloLocal {
 ...
}
```

   b. 为服务提供程序配置 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
  version="1.0">

  <http-publishing>
  <webservice-security>
   <security-constraint>
```

```
    <web-resource-collection>
          <web-resource-name>Only Managers</web-resource-name>
          <url-pattern>/manager/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
    </web-resource-collection>
        <auth-constraint id="AuthConstraint_manager">
     <role-name>Manager</role-name>
    </auth-constraint>
    <user-data-constraint>
     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
   </security-constraint>
   <security-constraint>
    <web-resource-collection>
          <web-resource-name>Employees</web-resource-name>
          <url-pattern>/employee/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
    </web-resource-collection>
        <auth-constraint id="AuthConstraint_employee">
     <role-name>Employee</role-name>
    </auth-constraint>
    <user-data-constraint>
     <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
   </security-constraint>
   <!-- SECURITY ROLES -->
      <security-role id="Staff">
    <role-name>Employee</role-name>
    <role-name>Manager</role-name>
   </security-role>
      <!-- AUTHENTICATION METHOD: client-cert authentication -->
      <!-- login configuration -->
      <login-config id="LoginConfig">
    <auth-method>CLIENT-CERT</auth-method>
        <realm-name>Authentication</realm-name>
   </login-config>   </webservice-security>
 </http-publishing>
</webservices-bnd>
```

👉 注:

- ibm-ws-bnd.xml 文件必须位于 Web 应用程序的 /WEB-INF 目录中，或者位于基于 EJB 的 Web Service 应用程序（JAR 归档）的 /META-INF 目录中。
- ibm-ws-bnd.xml 文件中的 `login-config` 元素仅在基于 EJB 的 Web Service 应用程序（JAR 归档）中生效。对于 Web 应用程序，将忽略 `login-config` 元素，将使用 web.xml 文件中的同一元素的值。

.

5. 通过指定 Web Service 端点来配置服务客户机。例如，客户机应用程序是一个名为 TransportSecurityClient.war 的 Web 应用程序。

   a. 在 server.xml 文件中配置客户机应用程序。

```
<application id="TransportSecurityClient" name="TransportSecurityClient"
    location="TransportSecurityClient.war"
    context="TransportSecurityClient" type="war" />
```

   b. 配置客户机应用程序的 ibm-ws-bnd.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
  version="1.0">
  <!-- POJO service reference binding-->
```

```
  <service-ref name="service/SayHelloPojoService">
    <port name="SayHelloPojoPort"
        namespace="http://ibm.com/ws/jaxws/transport/security/"
        ssl-ref="customizeSSLConfig"
        key-alias="user0"/>
    <properties http.conduit.tlsClientParameters.disableCNCheck="true" />
 </service-ref>
  <!-- Stateless service reference binding-->
  <service-ref name="service/SayHelloStatelessService">
    <port name="SayHelloStatelessPort"
        namespace="http://ibm.com/ws/jaxws/transport/security/"
        ssl-ref="customizeSSLConfig"
        key-alias="user0"/>
    <properties http.conduit.tlsClientParameters.disableCNCheck="true" />
 </service-ref>
</webservices-bnd>
```

👉 注：

- ibm-ws-bnd.xml 文件必须位于客户机 Web 应用程序的 /WEB-INF 目录中。
- key-alias 属性指定在 server.xml 文件中定义的客户机密钥库中的有效客户机证书。在此示例中，客户机密钥库定义为如下所示：

  ```
  <keyStore id="clientKeyStore" location="clientKey.jks" type="JKS"
   password="passw0rd" />
  ```

- ibm-ws-bnd.xml 文件中的 ssl-ref 属性必须与 server.xml 文件中 `ssl` 元素的正确 id 属性相匹配。在此示例中，`ssl` 元素定义为如下所示：

  ```
  <ssl id="customizeSSLConfig" keyStoreRef="clientKeyStore"
   trustStoreRef="clientTrustStore" />
  ```

- key-alias 属性的值还可以覆盖 server.xml 文件中 `ssl` 元素的 clientKeyAlias 属性的值。

c. 通过 WSDL 位置来生成客户机存根。

```
@WebServiceClient(name = "SayHelloPojoService",
    targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
    wsdlLocation = "https://localhost:8020/TransportSecurityProvider/unauthorized/
employPojoService?wsdl")
public class SayHelloPojoService
                extends Service
{...}

@WebServiceClient(name = "SayHelloStatelessService",
    targetNamespace = "http://ibm.com/ws/jaxws/transport/security/",
    wsdlLocation = "https://localhost:8020/TransportSecurityProvider/unauthorized/
EmployStatelessService?wsdl")
public class SayHelloStatelessService
                extends Service
{...}
```

d. 使用 `@WebServiceRef` 注释将 Web Service 插入到 Servlet。例如，`TestJaxWsTransportSecurityServlet`。

```
@WebServiceRef(name = "service/SayHelloPojoService")
SayHelloPojoService pojoService;

@WebServiceRef(name = "service/SayHelloStatelessService")
SayHelloStatelessService statelessService;
```

消息级别的 Web Services 安全性

Web Service 消息级别安全性（Web Service 安全性或 WS-Security）是 Web Service 应用程序的安全性服务质量 (QoS)。WS-Security 标准和概要文件描述了如何为 Web Service 环境中交换的 SOAP 消息提供安全和保护。

WS-Security 作为一个 xigemaAS 功能部件提供。xigemaAS 概要文件中所提供的 WS-Security 运行时基于 Apache CXF 开放式源代码服务框架。xigemaAS 中的 WS-Security 功能部件受到 Apache CXF 框架的功能部件和功能的限制。必须通过启用 `wsSecurity-1.1` 功能部件来显式地启用 WS-Security。请确保您还将 `appSecurity-2.0`、`servlet-3.0`（或 `servlet-3.1`）和 `jaxws-2.2` 以及其他必需的 xigemaAS 功能部件添加至 xigemaAS 的 server.xml 文件。

使用 Web Service 应用程序的 WSDL 文件中的 WS-SecurityPolicy 来配置 WS-Security。要使用 WS-Security 来保护 Web Service 应用程序，JAX-WS 应用程序必须包含一个具有嵌入式 WS-Security 策略的 WSDL 文件。在 `wsdl:binding` 和/或 `wsdl:operation` 部分，必须存在对于该嵌入式 WS-Security 策略的 PolicyReference。

Web Service 安全性规范和标准

xigemaAS 支持一些结构化信息标准促进组织 (OASIS) 标准。

- Web Services Security SOAP Message Security 1.1: *http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf*
- Web Services Security Username Token Profile 1.1：*http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf*
- Web Services Security X.509 Certificate Token Profile 1.1：*http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf*
- WS-SecurityPolicy 1.3：*http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.pdf*

Web Service 安全性缺省配置

Web Service 安全性 (WS-Security) 配置是对运行时 WS-Security 策略的补充。WS-Security 配置遵循 CXF 名称/值对样式，并且保留 CXF 属性名。某些属性具有缺省值，而某些属性则没有。

在 server.xml 文件中，xigemaAS 提供了适用于所有服务的服务器级配置。此配置称为缺省 WS-Security 配置。

server.xml 文件具有两个缺省 WS-Security 配置：一个配置适用于客户机应用程序，另一个配置适用于提供程序应用程序。server.xml 文件中不能存在其他 WS-Security 配置。如果您需要应用程序具有不同于缺省配置的定制 WS-Security 配置，那么必须通过程序来完成配置。

以下示例显示了缺省客户机配置：

```
<wsSecurityClient id="default"
                  ws-security.username="user2"
                  ws-security.password="security">
  <signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"

 org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Client"
                       org.apache.ws.security.crypto.merlin.keystore.alias="x509ClientCert"
                       org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ClientDefault.jks"/>
</wsSecurityClient>
```

以下示例显示了缺省提供程序配置：

```
<wsSecurityProvider id="default"
                    ws-security.username="user2">
  <encryptionProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"

 org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Server"
                        org.apache.ws.security.crypto.merlin.keystore.alias="x509ServerCert"
                        org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ServerDefault.jks"/>
```

```
</wsSecurityProvider>
```

下表显示了 xigemaAS 中的缺省 WS-Security 用户属性。可以在 CXF 中找到这些相同的属性。

表 32: xigemaAS 和 CXF 中的缺省 WS-Security 用户属性

xigemaAS 和 CXF 中的缺省 WS-Security 用户属性

xigemaAS/CXF 属性	缺省值
ws-security.username	无
ws-security.password	无
ws-security.signature.username	无
ws-security.encryption.username	无

下表显示了 xigemaAS 中的 WS-Security 回调处理程序类和 crypto 属性以及等价的 CXF 属性（如果这些属性不同）。

表 33: xigemaAS 概要文件中的 WS-Security 回调处理程序类和 crypto 属性以及等价的 CXF 属性

xigemaAS 中的 WS-Security 回调处理程序类和 crypto 属性以及等价的 CXF 属性

xigemaAS 概要文件属性	**CXF** 属性	缺省值
ws-security.callback-handler		无
<signatureProperties>	ws-security.signature.properties	无
<encryptionProperties>	ws-security.encryption.properties	无

在 xigemaAS 中，wss4j 属性指定为 signatureProperties 或 encryptionProperties 元素的属性。以下示例显示了 wss4j 属性：

```
<signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"
                     org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Client"
                     org.apache.ws.security.crypto.merlin.keystore.alias="x509ClientDefault"
                     org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ClientDefault.jks">
</signatureProperties>
```

下表显示了 xigemaAS 中的 wss4j crypto 属性。可以在 CXF 中找到这些相同的属性。

表 34: xigemaAS 和 CXF 中的 wss4j crypto 属性

xigemaAS 和 CXF 中的 wss4j crypto 属性

xigemaAS /CXF 属性	缺省值
org.apache.ws.security.crypto.provider	org.apache.ws.security.components. crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.provider	缺省为所安装的提供程序
org.apache.ws.security.crypto.merlin.cert.provider	缺省为密钥库提供程序
org.apache.ws.security.crypto.merlin.x509crl.file	无

下表显示了 xigemaAS 概要文件中的 wss4j 密钥库属性。可以在 CXF 中找到这些相同的属性。

表 35: xigemaAS 概要文件和 CXF 中的 wss4j 密钥库属性

xigemaAS 和 CXF 中的 wss4j 密钥库属性

xigemaAS/CXF 属性	缺省值
org.apache.ws.security.crypto.merlin.keystore.file	无
org.apache.ws.security.crypto.merlin.keystore.password	无
org.apache.ws.security.crypto.merlin.keystore.type	无
org.apache.ws.security.crypto.merlin.keystore.alias	无
org.apache.ws.security.crypto.merlin.keystore.private.password	无

下表显示了 xigemaAS 概要文件中的 wss4j 信任库属性。可以在 CXF 中找到这些相同的属性。

表 36: xigemaAS 概要文件和 CXF 中的 wss4j 信任库属性

xigemaAS 和 CXF 中的 wss4j 信任库属性

xigemaAS 概要文件属性	缺省值
org.apache.ws.security.crypto.merlin.truststore.file	无
org.apache.ws.security.crypto.merlin.truststore.password	无
org.apache.ws.security.crypto.merlin.truststore.type	无

下表显示了 xigemaAS 概要文件中的 WS-Security 其他属性。可以在 CXF 中找到这些相同的属性。

表 37: xigemaAS 概要文件和 CXF 中的 WS-Security 其他属性

xigemaAS 和 CXF 中的 WS-Security 其他属性

xigemaAS/CXF 属性	缺省值
ws-security.enable.nonce.cache	true
ws-security.cache.config.file	无

下表显示了仅在 xigemaAS 概要文件中受支持的属性。

表 38: 仅在 xigemaAS 概要文件中受支持的属性

仅在 xigemaAS 中受支持的属性

xigemaAS 概要文件属性	CXF 属性	缺省值
callerToken	无	无

配置其他属性

您可以设置若干额外的属性，以向 WS-Security 运行时环境提供其他配置信息。请参阅下列链接以了解有关这些属性的详细信息：

- *http://cxf.apache.org/docs/ws-securitypolicy.html*
- *http://ws.apache.org/wss4j/config.html*

可以在 server.xml 文件中的缺省 WS-Security 配置中指定任何其他属性。

例如，要指定任何其他属性，请在 wsSecurityClient 和/或 wsSecurityProvider 部分指定这些属性。

```
<wsSecurityProvider id="default"
  <signatureProperties ... />
  <encryptionProperties ... />
  ws-security.cache.config.file = "${server.config.dir}/resources/ws-security/new_cxf-
ehcache.xml"
</wsSecurityProvider>

<wsSecurityClient id="default"
  <signatureProperties ... />
  <encryptionProperties ... />
  ws-security.username-token.always.encrypted="false"
</wsSecurityClient>
```

配置高速缓存

WS-Security 提供了 UsernameToken 中的 nonce、created 时间戳记和安全性令牌的缺省高速缓存实现。缺省高速缓存实现基于 ehCache，并且具有下列缺省设置：

```
maxEntriesLocalHeap="5000"
timeToIdleSeconds="3600"
timeToLiveSeconds="3600"
overflowToDisk="true"
maxElementsOnDisk="10000000"
diskPersistent="false"
diskExpiryThreadIntervalSeconds="120"
memoryStoreEvictionPolicy="LRU"
```

要修改缺省高速缓存设置，您可以提供 ehCache 配置 XML 文件。使用 ws-security.cache.config.file 定制属性来指定具有定制的属性的文件名以不同于缺省设置。必须将此文件放入服务器概要文件中的某个位置。可以从以下网址找到另外的样本高速缓存设置配置文件：*http://svn.apache.org/viewvc/cxf/trunk/rt/ws/security/src/main/resources/cxf-ehcache.xml?view=markup*。

配置更强大的签名算法

因为 SHA1 签名算法较脆弱，所以美国国家标准技术学会 (NIST) 建议您使用更强大的签名算法。

但是，WS-Security 策略 1.3 定义了 HmacSha1 或 RsaSha1（统称为 sha1）作为唯一的签名算法。xigemaAS 中的 WS-Security 有一个使用以下更强大的签名算法的配置选项：

- RSA-SHA-256
- RSA-SHA-384
- RSA-SHA-512

- HMAC-SHA-256
- HMAC-SHA-384
- HMAC-SHA-512

要配置 xigemaAS 中的 WS-Security 以支持更强大的签名算法，请使用 `signatureAlgorithm` 属性在 server.xml 文件的 <signatureProperties> 元素中定义必需的算法。`signatureAlgorithm` 属性的有效值为 sha256、sha384 和 sha512。例如，如果您将 `signatureAlgorithm` 属性的值指定为 sha512，那么在具有非对称密钥的签名中使用的签名算法为 RSA-SHA-512，而在具有对称密钥的签名中使用的签名算法为 HMAC-SHA-512。

以下示例显示了一个需要 sha256 签名算法的样本客户端配置：

```
<wsSecurityClient id="default"
  ws-security.password="security"
  ws-security.username="user1"
  ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.CommonPasswordCallback"
  ws-security.signature.username="soaprequester">
  <signatureProperties
    signatureAlgorithm="sha256"
    org.apache.ws.security.crypto.merlin.keystore.type="jks"
    org.apache.ws.security.crypto.merlin.keystore.password="client"
    org.apache.ws.security.crypto.merlin.keystore.alias="soaprequester"
    org.apache.ws.security.crypto.merlin.file="${server.config.dir}/dsig-sender.ks" />
</wsSecurityClient>
```

以下示例显示了一个需要 sha256 签名算法的样本提供程序端配置：

```
<wsSecurityProvider id="default"
  ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.CommonPasswordCallback"
  ws-security.signature.username="soapprovider">
  <signatureProperties
    signatureAlgorithm="sha256"
    org.apache.ws.security.crypto.merlin.keystore.type="jks"
    org.apache.ws.security.crypto.merlin.keystore.password="server"
    org.apache.ws.security.crypto.merlin.keystore.alias="soapprovider"
    org.apache.ws.security.crypto.merlin.file="${server.config.dir}/dsig-receiver.ks" />
</wsSecurityProvider>
```

使用 UsernameToken 来认证 Web Service 客户机

xigemaAS 支持 OASIS Web Services Security UsernameToken Profile 1.1 规范。此规范描述 Web Service 客户机如何提供 UsernameToken，作为确定请求者的手段，方式是使用用户名以及选择对 Web Service 提供程序使用密码或密码等价项。xigemaAS 中用于为 Web Service 提供程序处理此策略的 Web Service 安全性 (WS-Security) 运行时可以使用此标识信息来认证用户。

UsernameToken 的需求表示为 WS-Security 策略中的其中一个支持令牌。可以添加 UsernameToken 需求，作为其中一个支持令牌断言中的必需令牌，包括 `SupportingTokens`、`SignedSupportingTokens`、`SignedEndorsingSupportingTokens`、`SignedEncrypted` 和 `EncryptedSupportingTokens`。

以下示例说明了一个样本策略片段，它要求通过 SOAP 消息的安全头将具有密码文本的 UsernameToken 发送至 Web Service 提供程序：

```
<sp:SupportingTokens>
  <wsp:Policy>
    <sp:UsernameToken
      sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssUsernameToken11 />
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
```

```
</sp:SupportingTokens>
```

除了必须提供用户名或密码之外，您还可以配置策略以在 UsernameToken 中包含 nonce 和 created 时间戳记。

以下示例说明了一个样本策略片段，它要求通过 SOAP 消息的安全性头将具有密码文本、nonce 和 created 时间戳记的 UsernameToken 发送至 Web Service 提供程序：

```
<sp:SupportingTokens>
  <wsp:Policy>
    <sp:UsernameToken
      sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssUsernameToken11 />
        <sp13:Created />
        <sp13:Nonce />
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
</sp:SupportingTokens>
```

以下示例说明了一个样本策略片段，它要求具有密码摘要（而不是密码文本）的 UsernameToken：

```
<sp:SupportingTokens>
  <wsp:Policy>
    <sp:UsernameToken
      sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssUsernameToken11 />
        <sp:HashPassword />
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
</sp:SupportingTokens>
```

有关 nonce、created 以及不同密码类型的更多信息，请参阅 *OASIS WS-Security* 策略 *1.3* 规范。

在 Web Service 客户机中提供用户名和密码

在生成 UsernameToken 时，xigemaAS 中的 WS-Security 功能部件提供了多种方法来指示客户机应用程序的用户名和密码。可以通过程序来设置用户名和密码，也可以在 server.xml 文件中设置。

客户机可以使用 server.xml 文件中提供的用户名和密码来生成 UsernameToken。server.xml 文件中的用户名和密码被认为是缺省配置，会被 RequestContext 上为客户机的 Web Service 调用所提供的用户名和密码覆盖。

以下示例说明了一个样本缺省配置：

```
<wsSecurityClient id="default"
    ws-security.username="alice"
    ws-security.callback-handler="com.acme.PasswordCallback"
</wsSecurityClient>
```

要使用通过程序确定的用户名和密码来生成 UsernameToken，您可以在 RequestContext 上为客户机的 Web Service 调用设置下列 CXF 属性：

- ws-security.username - 用户名
- ws-security.password - 用户密码（如果未定义 ws-security.callback-handler）
- ws-security.callback-handler - 用来获取密码的 CallbackHandler 实现类

以下代码样本说明如何在请求上下文上提供用户名和密码：

```
Map<String, Object> requestCtx = ((BindingProvider)port).getRequestContext();
requestCtx.put("ws-security.username", "bob_username");
requestCtx.put("ws-security.password", "bob_password");
```

在 Web Service 提供程序中使用 UsernameToken

收到 UsernameToken 后，WS-Security 会自动使用 xigemaAS 安全用户注册表来验证用户名和密码（如果需要密码）。如果 UsernameToken 中的密码类型是 `PasswordDigest` 或者使用的是派生密钥，那么您必须通过在 server.xml 文件中配置密码回调处理程序的 `ws-security.callback-handler` 实现来提供该实现。此回调处理程序必须返回所期望的所有用户名的有效密码，以便 WS-Security 运行时可以计算摘要值，以与 SOAP 消息中的值进行比较。成功完成对摘要值进行比较之后，将针对用户注册表来验证用户名和密码。

以下示例说明了 server.xml 文件中用于密码摘要的样本配置：

```
<wsSecurityProvider id="default"
    ws-security.callback-handler="com.acme.PasswordCallback"
</wsSecurityProvider>
```

密码 CallbackHandler

WS-Security 使用密码 CallbackHandler 来检索用户密码。在 xigemaAS 概要文件中，此密码 CallbackHandler 类必须打包成 xigemaAS 功能部件。有关密码 CallbackHandler 的更多信息，请参阅为 *WS-Security* 开发密码回调处理程序。

有关回调处理程序实现的要求和局限性的更多信息，请参阅使用 *X.509* 令牌保护 *Web Service* 的“专用密钥密码 `CallbackHandler`”一节。

在 SOAP 消息中保护 UsernameToken

在策略中指定了 UsernameToken 时，密码类型在缺省情况下为密码文本 (`PasswordText`)。当使用密码文本来发送密码时，它在消息中将按原样发送。以下示例说明了密码类型为 `PasswordText` 的 UsernameToken：

```
<UsernameToken>
  <Username>myusername</Username>
  <Password
    Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">
    mypassword
  </Password>
</UsernameToken>
```

当您使用 `PasswordText` 来发送 UsernameToken 时，应考虑对消息进行更多保护，例如，使用 HTTPS，或者使用 `EncryptedSupportingToken` 策略断言对令牌进行加密。有关要求在策略中使用 HTTPS 传输的更多信息，请参阅 *Web Service* 安全性 *HTTPS* 传输策略断言（见第 1476 页）。

UsernameToken 密钥派生

如 Web Services Security UsernameToken Profile 1.1 规范中所述：

“可以使用与用户名相关联的密码来派生共享密钥，以实现完整性或机密性，从而保护消息内容。”

“必须注意，密码会经受多种类型的攻击，这又会导致泄露所派生的任何密钥。此密钥派生过程旨在尽可能降低对密钥进行攻击的风险，但最终还是会受到为了便于人们记住密码和在标准键盘上输入密码而对密码造成的不安全性的限制。”

“为了能够从密码派生密钥，还需要另外两个元素。这两个元素是 <wsse11:Salt> 和 <wsse11:Iteration>。这些值不是秘密，当使用了密钥派生时，必须在 Username 令牌中传递这些值。当使用了密钥派生时，密码不得包括在 Username 令牌中。接收方将利用它对密码的了解来派生与发送方相同的密钥。”

如果 UsernameToken 正在对提供程序应用程序使用密钥派生，那么您必须通过在 server.xml 文件中配置密码回调处理程序的 `ws-security.callback-handler` 实现来提供该实现。

以下示例显示了将 UsernameToken 作为通过密钥派生获得的保护令牌的策略片段：

```
<sp:SymmetricBinding>
  <wsp:Policy>
    <sp:ProtectionToken>
      <wsp:Policy>
        <sp:UsernameToken
          sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
          <wsp:Policy>
            <sp:WssUsernameToken10 />
            <sp:RequireDerivedKeys />
          </wsp:Policy>
        </sp:UsernameToken>
      <wsp:Policy>
    </sp:ProtectionToken>
  </wsp:Policy>
</sp:SymmetricBinding>
```

以下示例显示了在使用密钥派生的情况下存在于安全性头中的样本 UsernameToken：

```
<wsse:UsernameToken wsse:Id="...">
  <wsse:Username>...</wsse:Username>
  <wsse11:Salt>...</wsse11:Salt>
  <wsse11:Iteration>...</wsse11:Iteration>
</wsse:UsernameToken>
```

使用 X.509 令牌来保护 Web Service

xigemaAS 支持 Oasis Web Services Security X.509 Certificate Token Profile 1.1。可以使用 X.509 令牌对消息进行签名和加密，从而提供消息完整性和机密性。

WS-Security 策略

要使用 X.509 令牌来保护 XML 消息，必须先创建一个在 Web 服务描述语言 (WSDL) 中指定的合同。Web Service 必须在 WSDL 文件中包含 WS-Security 策略。WS-Security 策略可以包含 `AsymmetricBinding` 或 `SymmetricBinding` 断言。

X.509 令牌的需求表示为 WS-Security 策略中的 `X509Token` 断言类型。以下示例说明了一个样本 `X509Token` 断言：

```
<sp:X509Token sp:IncludeToken=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
  <wsp:Policy>
    <sp:WssX509V3Token10 />
  </wsp:Policy>
</sp:X509Token>
```

在 `AsymmetricBinding` 断言中：

- 发起方 X509Token 用于对从请求者传递到提供者的消息进行签名，以及对从提供者传递到请求者的消息进行加密。
- 接收方 X509Token 用于对从提供者传递到请求者的消息进行签名，以及对从请求者传递到提供者的消息进行加密。

下表说明了发起方令牌和接收方令牌如何用于请求/响应链的每一部分：

表 39: 发起方令牌和接收方令牌用于请求/响应链的每一部分

发起方令牌和接收方令牌用于请求/响应链的每一部分

*	发起方令牌	接收方令牌
请求传出（请求者）	签名	加密
请求传入（提供者）	验证签名	解密
响应传出（提供者）	加密	签名
响应传入（请求者）	解密	验证签名

专用密钥用来对消息进行签名和解密。公用证书用来对消息进行加密和验证签名。专用密钥用于密钥的用户（签署者或者解密者）。

在 SymmetricBinding 断言中，为 X.509 令牌保护的密钥或临时密钥供发起方和接收方共享，而密钥用于加密和解密。

以下示例说明了在 AsymmetricBinding 断言中使用 X.509 令牌的样本 WS-Security 策略。此策略使得能够使用时间戳记对请求和响应的主体进行签名和加密。

```
<wsp:Policy wsu:Id="SampleAsymmetricX509TokensPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:AsymmetricBinding>
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken=
                "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken=
                "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:RecipientToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:ProtectTokens />
          <sp:OnlySignEntireHeadersAndBody />
        </wsp:Policy>
      </sp:AsymmetricBinding>
```

```
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
      <sp:Wss10>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
        </wsp:Policy>
      </sp:Wss10>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

以下示例说明了在 SymmetricBinding 断言中使用 X.509 令牌的样本 WS-Security 策略。此策略使得能够使用时间戳记对请求和响应的主体进行签名。

```
<wsp:Policy wsu:Id="SampleSymmetricX509TokensPolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken=
                "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                <wsp:Policy>
                  <sp:WssX509V3Token11 />
                  <sp:RequireThumbprintReference />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:SignBeforeEncrypting />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:SymmetricBinding>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

令牌引用和令牌包括

令牌断言可以携带 sp:IncludeToken 属性，该属性要求将令牌包括在消息中。除了直接引用以外，若干令牌断言还支持有关引用令牌的机制。如果令牌断言中包含多个引用断言，那么对于该令牌的引用需要包含所指定的所有引用类型。例如，如果一个令牌断言携带了 sp:IncludeToken 属性，该属性的值为 ../Always，并且该令牌断言还包含嵌套的 sp:RequireIssuerSerialReference 断言，那么消息中必须包括该令牌两次。虽然这样的组合并不算错误，但是，由于效率方面的原因，最好是避免使用这样的组合。

在 xigemaAS 中强制实施了 IncludeToken 断言

WS-Security 策略支持下列令牌引用机制：

- KeyIdentifier: <sp:RequireKeyIdentifierReference... />
- IssuerSerial: <sp:RequireIssuerSerialReference ... />
- Thumbprint: <sp:RequireThumbprintReference ... />

在客户端/生成者端生成 X.509 令牌时使用这些引用（但是要求在服务器端/使用者端未强制实施这些引用）。

运行时配置

对于使用 X.509 令牌来保护消息，xigemaAS 中的 WS-Security 功能部件运行时比 WS-Security 策略提供了更多配置选项。运行时配置是在 server.xml 文件中定义。运行时配置包括密码属性（例如，`signatureProperties` 和 `encryptionProperties`）、密钥和密钥库。

以下示例说明了 `X509Token` 断言的客户机配置：

```
<wsSecurityClient id="default"
                  ws-security.callback-
handler="com.ibm.ws.wssecurity.example.cbh.KeyPasswordCallbackHandler">
  <signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"
                       org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Client"
                       org.apache.ws.security.crypto.merlin.keystore.alias="x509ClientDefault"
                       org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ClientDefault.jks" />
  <encryptionProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"
                        org.apache.ws.security.crypto.merlin.keystore.password="xigemaASXClient"
                        org.apache.ws.security.crypto.merlin.keystore.alias="x509DefaultCert"
                        org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ClientDefault.jks" />
</wsSecurityClient>
```

以下示例说明了 `X509Token` 断言的服务器配置：

```
<wsSecurityProvider id="default"
                    ws-security.callback-
handler="com.ibm.ws.wssecurity.example.cbh.KeyPasswordCallbackHandler">
  <signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"
                       org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Server"
                       org.apache.ws.security.crypto.merlin.keystore.alias="x509ServerDefault"
                       org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ServerDefault.jks" />
  <encryptionProperties org.apache.ws.security.crypto.merlin.keystore.type="jks"
                        org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Server"
                        org.apache.ws.security.crypto.merlin.keystore.alias="x509ServerDefault"
                        org.apache.ws.security.crypto.merlin.file="${server.config.dir}/
x509ServerDefault.jks" />
</wsSecurityProvider>
```

在运行时配置中，`signatureProperties` 和 `encryptionProperties` 属性等价于 WSS4J 配置中的 crypto 属性。有关更多详细信息，请参阅 *WSS4J configuration*。许多 crypto 属性具有有效的缺省值，因此，不需要指定这些属性。但是，必须指定其他属性（例如，与密钥库和密钥相关的属性）。

有关这些属性的更多信息，请参阅 *Web Service* 安全性缺省配置。

密钥库文件、密钥和证书

对消息进行签名和加密需要使用公用证书和专用密钥。这些公用证书和专用密钥存储在密钥库文件中。当您使用 `X509Token` 断言时需要密钥库文件。客户端密钥库文件包含客户机的专用密钥，以及对应于服务器公用密钥的证书链。服务器端密钥库文件包含服务器的专用密钥，以及对应于客户机公用密钥的证书链。

对 SOAP 消息进行签名和加密之后，由 WS-Security 运行时环境按如下所示来解释 `signatureProperties` 和 `encryptionProperties` 属性：

- AsymmetricBinding 断言

  由 `signatureProperties` 属性所指定的密钥用作对出站消息进行签名的专用密钥，还用于对入站消息进行解密。

由 encryptionProperties 属性所指定的密钥用作对出站消息进行加密的公用密钥，还用于验证入站消息中的签名。

- SymmetricBinding 断言

  使用了由 encryptionProperties 属性所指定的密钥，而忽略了 signatureProperties 属性。

专用密钥密码 CallbackHandler

要访问密钥库中的专用密钥，您必须知道两个不同的密码。一个密码用于存储了专用密钥的密钥库。另一个密码用于专用密钥本身。虽然在其中一个 crypto 属性（signatureProperties 或 encryptionProperties）中指定了密钥库的密码，但是 WS-Security 通常使用 CallbackHandler 类来访问密钥库中的专用密钥。在 xigemaAS 中，必须将此 CallbackHandler 类作为 xigemaAS 功能部件进行打包，并且其类名指定为 security.xml 中的 ws-security.callback-handler 定制属性的值。

WS-Security 需要此 CallbackHandler 类才能访问密钥库中的专用密钥。

有关密码 CallbackHandler 类的更多信息，请参阅*为 WS-Security 开发密码回调处理程序*。

如果未提供密码 CallbackHandler 类，那么在 crypto 属性中指定的密码将用作专用密钥的缺省密码。此属性配置为 org.apache.ws.security.crypto.merlin.keystore.private.password。

因为在 security.xml 文件的每个 wsSecurityClient 和 wsSecurityProvider 部分中仅支持一个 ws-security.callback-handler 定制属性，所以单个密码回调处理程序必须支持每种使用缺省密码回调处理程序的应用程序类型（客户机或服务器）所需要的所有密码。所有提供者应用程序必须使用缺省密码回调处理程序。客户机应用程序可以通过在客户机的 Web Service 调用的请求上下文中指定 ws-security.callback-handler 定制属性来覆盖缺省回调处理程序。

如果您需要为单个 Web Service 调用提供 X.509 专用密钥和 UsernameToken 的密码，那么您无法每次都提供不同的回调处理程序以供使用。在这种情况下，您必须实现单个回调处理程序来处理这两种密码。如前一个示例中所述，您必须使用 WSPasswordCallback.getUsage() 方法来确定您应返回哪种密码。请参阅 WSPasswordCallback API 文档以了解受支持的用法代码：*http://ws.apache.org/wss4j/apidocs/org/apache/wss4j/common/ext/WSPasswordCallback.html*。

证书撤销列表

证书撤销列表 (CRL) 是已撤销证书的序列号的列表。CRL 可以是一个独立文件，也可以将其打包在 PKCS#7 包装器中。可以将 CRL 与信任库配合使用来控制对服务的访问。

需要执行两个步骤来配置 xigemaAS 中的 WS-Security 功能部件以对 CRL 文件执行撤销检查。要对 CRL 文件启用证书撤销检查，您必须修改 server.xml 文件中的 WS-Security 运行时配置。

1. 在 wsSecurityProvider 或 wsSecurityClient 元素中，添加以下属性：

   ```
   ws-security.enableRevocation="true"
   ```

2. 在 signatureProperties 元素中，添加以下属性，并将值设置为 CRL 文件：

   ```
   org.apache.ws.security.crypto.merlin.x509crl.file
   ```

以下示例说明了一个对 CRL 文件启用证书撤销检查的样本 WS-Security 提供者配置：

```
<wsSecurityProvider id="default"
    ws-security.callback-handler="com.acme.example.cbh.CommonPasswordCallback"
    ws-security.signature.username="x509ServerDefault"
    ws-security.enableRevocation="true">
```

```
  <signatureProperties
      org.apache.ws.security.crypto.merlin.keystore.type="jks"
      org.apache.ws.security.crypto.merlin.keystore.password="xigemaASX509Server"
      org.apache.ws.security.crypto.merlin.keystore.alias="x509ServerDefault"
      org.apache.ws.security.crypto.merlin.file="${server.config.dir}/x509ServerDefault.ks"
      org.apache.ws.security.crypto.merlin.truststore.password="xigemaASX509Server"
      org.apache.ws.security.crypto.merlin.truststore.file="${server.config.dir}/
x509DefaultServer.ks"
      org.apache.ws.security.crypto.merlin.x509crl.file="${server.config.dir}/revokedCerts.crl"/
>
</wsSecurityProvider>
```

使用签署令牌来保护 Web Service

签署令牌用来对 SOAP 消息签名进行签名。签署令牌通常用来对整个 <Signature> 元素进行签名，还可以选择用来对其他消息部件进行签名。如果使用了传输安全性，那么必须使用签署令牌来对 SOAP 安全性头中的时间戳记进行签名。

签署支持令牌断言

签署令牌是 WS-Security 中的 SupportingTokens，并且可以对其签名和加密。有四种类型的签署令牌：

- EndorsingSupportingTokens - 不需要对令牌进行签名或加密。
- SignedEndorsingSupportingTokens - 必须对令牌签名。
- EndorsingEncryptedSupportingTokens - 必须对令牌加密。
- SignedEndorsingEncryptedSupportingTokens - 必须对令牌签名。

X509Token 作为签署令牌

可以将 X509Token 配置为签署令牌。将 X509Token 配置为签署令牌类似于将 X509Token 配置为 AssymmetricBinding 中的 InitiatorToken。要将 X509Token 配置为签署令牌，您需要一个策略，该策略具有先前描述的其中一个签署令牌断言中包含的 X509Token，以及 server.xml 文件中的 <SignatureProperties> 元素。

以下示例显示了 X509 签署支持令牌的策略片段：

```
<sp:EndorsingSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
  <wsp:Policy>
    <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssX509V3Token10/>
      </wsp:Policy>
    </sp:X509Token>
  </wsp:Policy>
</sp:EndorsingSupportingTokens>
```

将 X509Token 配置为签署令牌之后，您必须在 server.xml 文件中定义 <signatureProperties> 元素以标识密钥库和签名密钥。<signatureProperties> 元素的配置与消息签名的 X509Token 的配置相同。有关更多信息，请参阅使用 *X.509* 令牌来保护 *Web Service*。

将 UsernameToken 作为签署令牌

作为 endorsingToken 的 UsernameToken 的配置类似于具有 PasswordDigest 的 UsernameToken 的配置（只不过策略中没有 HashPassword）。要将 UsernameToken 配置为签署令牌， 您需要一个策略，该策略具有“签署支持令牌断言”一节中描述的其中一个签署令牌断言中包含的 UsernameToken。

以下示例显示了 UsernameToken 已签名和已加密的签署支持令牌的策略片段。使用了密钥派生：

```
<sp:SignedEndorsingEncryptedSupportingTokens>
  <wsp:Policy>
    <sp:UsernameToken
      sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssUsernameToken10 />
        <sp:RequireDerivedKeys />
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
</sp:SignedEndorsingEncryptedSupportingTokens>
```

在此样本策略片段中，使用此消息的其余内容对 UsernameToken 进行签名和加密。然后，使用从 UsernameToken 中的密码派生的密钥对消息签名进行签名。

由于 UsernameToken 签署令牌使用派生密钥，因此您必须为提供程序应用程序实现并配置密码 CallbackHandler。有关更多信息，请参阅使用 *UsernameToken* 来认证 *Web Service* 客户机。

Web Service 安全性调用者配置

可以在已认证的方式或未认证的方式下运行 Web Service。如果您想要根据用户身份来限制对资源的访问权，那么 Web Service 必须以已认证的方式运行。当 Web Service 以已认证的方式运行时，会将用户身份放在运行 Web Service 的同一个线程上。

Web Service 可以通过两种方法来实现以已认证的方式运行：

HTTP 基本认证

由 Web 容器将 HTTP 头中的身份放在线程上。

WS-Security 调用者配置

由 WS-Security 运行时将 SOAP 安全性头中其中一个令牌的身份放在线程上。

WS-Security 规范允许将多个令牌传入 SOAP 消息的安全性头。当 Web Service 需要使用 WS-Security 以已认证的方式运行时，需要某种机制以让 WS-Security 运行时环境知道要将哪个令牌用于身份。此机制称为调用者配置。

在 server.xml 文件中使用 <callerToken> 元素来指定 WS-Security 调用者配置。

下面举例说明了一个包含 UsernameToken 的调用者配置的样本 WS-Security 提供程序配置：

```
<wsSecurityProvider ...>
  ...
  <callerToken name="UsernameToken" />
  ...
</wsSecurityProvider>
```

可为 `<callerToken>` 元素指定下列值：

- `UsernameToken`
- `X509Token`
- `SamlToken`

如果将 `X509Token` 配置为调用者令牌，请确保从安全性头只能解析一个客户机的 `X509Certificate`。例如，确保只有一个客户机证书是从 `AsymmetricBinding` 中的发起方令牌解析，或者只有一个客户机证书是从签署令牌解析。

如果将 `UsernameToken` 配置为调用者令牌，那么安全性头必须只包含一个 `UsernameToken`。

如果将 SamlToken 配置为调用者令牌，那么安全性头必须只包含一个 SAML 令牌。有一些附加配置选项（例如，userIdentifier、groupIdentifier 或 realmIdentifier）用于指定 SAML 属性，此属性可在创建已认证主体集时用作主体、组或域。有关可选 SAML callerToken 配置的更多信息，请参阅创建 *WS-Security SAML* 调用者配置（见第 1451 页）。

创建 WS-Security SAML 调用者配置

可使用 server.xml 文件中的 <callerToken> 元素来配置包含 SAML 令牌的调用者的 WS-Security 提供程序配置。

1. 在 server.xml 文件中使用 <callerToken> 元素配置 WS-Security 安全性断言标记语言 (SAML) 调用者配置。以下示例显示包含 SAML 令牌的调用者的样本 WS-Security 提供程序配置：

```
<wsSecurityProvider ...>
  ...
  <callerToken name="SamlToken"  userIdentifier="userIdentifierString"
 groupIdentifier="groupIdentifierString"
 userUniqueIdentifier="uniqueIdentifierString"
 realmIdentifier="realmIdentifierString"
                includeTokenInSubject="false" mapToUserRegistry="User"
 realmName="customRealmName" allowCustomCacheKey="false"/>
 ...
</wsSecurityProvider>
```

   此配置中的唯一必需属性为“name”。缺省情况下，认证主体集是使用 SAML 断言中的信息创建的，它不需要本地用户注册表来执行认证。
2. 可选：可配置以下可选属性以帮助通过 SAML 断言创建已认证主体集。其中一些可选属性的缺省值为：

```
includeTokenInSubject=true
mapToUserRegistry="No"
allowCustomCacheKey="true"
```

   - 如果 mapToUserRegistry 为“No”，那么 SAML 发卡者的名称将用作域，NameID 将用作主体名称和主体集中的唯一安全性名称，组成员未被包括。
   - 如果 mapToUserRegistry 为“User”，那么系统将针对您的本地用户注册表验证 SAML 用户，然后运行时会根据本地注册表创建用户主体集。
   - 如果 mapToUserRegistry 为“Group”，那么系统将针对您的本地用户注册表验证 SAML 组，然后运行时会创建带有已验证组的主体集。此选项类似 mapToUserRegistry=No，只是针对本地用户注册表验证组成员资格的操作不同。

   可配置附加属性（例如，userIdentifier、realmIdentifier、groupIdentifier 和 userUniqueIdentifier）以创建带有定制用户名、域名、组成员资格和唯一安全标识的已认证主体集。

   - userIdentifier：使用此属性选择其值用作主体名称的 SAML 属性名称。
   - groupIdentifier：使用此属性选择其值包含为主体集中的组成员的 SAML 属性名称。
   - realmName：使用此属性明确指定用于对已认证主体集标识 SAML 主体的域名。缺省域名为 SAML 发卡者名称。
3. 可选：可实现 xigemaAS SAML SPI com.ibm.wsspi.security.saml2.UserCredentialResolver 作为用户功能部件，以将 SAML 断言动态映射至 xigemaAS 主体集。

WS-SecurityPolicy 和模板

使用 Web Service 应用程序的 Web 服务描述语言 (WSDL) 文件中的 WS-SecurityPolicy 来配置 xigemaAS 概要文件中的 Web Service 安全性 (WS-Security)。要使用 WS-Security 来保护 Web Service 应用程序，JAX-WS 应用程序必须包含一个具有嵌入式 WS-Security 策略的 WSDL 文件。WS-SecurityPolicy 模板中包含若干常用的 WS-Security 策略样本。可以修改这些模板以满足许多不同的使用方案。

提供了以下常用 WS-SecurityPolicy 模板列表，可以修改这些模板以满足许多不同的使用方案：

- 方案 1：基于 *SSL* 且具有密码摘要 *(HashPassword)* 和时间戳记的 *UsernameToken*
- 方案 2：基于 *SSL* 且具有密码文本、*nonce* 和 *created* 时间戳记的 *UsernameToken*
- 方案 3：*UsernameToken* 作为 *EndorsingToken*, 并且 *X509Token* 对于消息保护是对称的
- 方案 4：具有 *X509Token* 非对称消息保护 *(* 相互认证 *)* 的 *UsernameToken*
- 方案 5：将客户机 *X509Token* 作为 *EndorsingToken*, 并且服务器 *X509Token* 对于消息保护是对称的
- 方案 6：进行 *UsernameToken* 认证, 并且 *X509Token* 对于消息保护是对称的
- 方案 7：基于 *HTTP* 且具有密码文本、*nonce* 和 *created* 时间戳记的 *UsernameToken*
- 方案 8：作为基于 *SSL* 的 *SupportingToken* 的 *SamlToken*
- 方案 9：*SAMLToken*, 其 *X509Token* 对于消息签名是非对称的
- 方案 10：具有 *X509Token* 非对称消息保护 *(* 相互认证 *)* 的 *SamlToken*
- 方案 11：*SAMLToken*, 其 *X509Token* 对于消息保护是对称的

基于 SSL 且具有密码摘要 (HashPassword) 和时间戳记的 UsernameToken

此策略要求您使用 HTTPS 来保护消息，并将 UsernameToken 用于进行认证。使用 created 时间戳记和 nonce 对 UsernameToken 密码进行了散列。还有消息时间戳记。

以下策略显示了基于 SSL 且具有密码摘要 (HashPassword) 和时间戳记的 UsernameToken：

```
<wsp:Policy wsu:Id="UserNameTokenPasswordHashOverSSL">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken>
                <wsp:Policy />
              </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
              <sp:HashPassword/>
            </wsp:Policy>
          </sp:UsernameToken>
```

```
        </wsp:Policy>
      </sp:SupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

要验证该 UsernameToken 存在于 xigemaAS 服务器中，您必须通过设置 ws-security.callback-handler 属性来实现在提供者端提供密码回调处理程序类。回调处理程序中的密码必须与 PasswordDigest 中使用的密码相匹配。该密码还必须与 xigemaAS 概要文件中的用户注册表中的密码相匹配。

基于 SSL 且具有密码文本、nonce 和 created 时间戳记的 UsernameToken

此策略要求您使用 HTTPS 来保护消息，并将 UsernameToken 用于进行认证。UsernameToken 中的用户密码以明文形式发送，并且会包括时间戳记和 nonce。还有消息时间戳记。您可以修改此策略以便不需要 nonce 和时间戳记。为了进行测试，您还可以移除 TransportBinding 以通过 HTTP 来发送 UsernameToken，从而达到简化目的。

以下策略显示了基于 SSL 且具有密码文本、nonce 和 created 时间戳记的 UsernameToken：

```
<wsp:Policy wsu:Id="UserNameTokenPasswordTextOverSSL">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken>
                <wsp:Policy>
                </wsp:Policy>
              </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
              <sp13:Created />
              <sp13:Nonce />
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

UsernameToken 作为 EndorsingToken，并且 X509Token 对于消息保护是对称的

对于 X509Token 对称保护，会创建临时密钥以对消息进行签名和加密。会使用接收方的公用证书对临时密钥进行加密。会将具有派生密钥的 UsernameToken 用于认证。会使用 UsernameToken 的派生密钥对消息签名进行签名。还有消息时间戳记。

如果不支持 HTTP 传输，服务具有 X509 令牌并支持 UsernameToken，并且服务要求客户机签署消息，那么可以使用此策略模板。

以下策略显示了将 UsernameToken 作为 EndorsingToken，并且 X509Token 对于消息保护是对称的：

```
<wsp:Policy wsu:Id="UsernameTokenAsEndorsingAndX509Symmetric">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:SymmetricBinding>
      <sp:SignedEndorsingEncryptedSupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
              <sp:RequireDerivedKeys />
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SignedEndorsingEncryptedSupportingTokens>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

具有 X509Token 非对称消息保护（相互认证）的 UsernameToken

消息中包括具有发送方的公用证书的签发者/序列化表示的 X509Token。会使用 X509Token 非对称消息保护对请求中的 UsernameToken 以及请求和响应中的 SOAP 主体进行签名和加密。还有消息时间戳记和签名确认。会将具有明文密码的 UsernameToken 用于认证。

如果客户机必须同时使用 X509 客户机证书和 UsernameToken 向服务认证它自身，那么最好是使用此策略模板。

以下策略显示了具有 X509Token 非对称消息保护的 UsernameToken：

```
<wsp:Policy wsu:Id="AsymmetricX509MutualAuthenticationWithUnt">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedEncryptedSupportingTokens
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:UsernameToken
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SignedEncryptedSupportingTokens>
      <sp:AsymmetricBinding>
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                  <sp:RequireIssuerSerialReference />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken>
            <wsp:Policy>
              <sp:X509Token
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/Never">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                  <sp:RequireIssuerSerialReference />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:RecipientToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:EncryptSignature />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
```

```
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:AsymmetricBinding>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
          <sp:MustSupportRefIssuerSerial />
          <sp:MustSupportRefThumbprint />
          <sp:MustSupportRefEncryptedKey />
          <sp:RequireSignatureConfirmation />
        </wsp:Policy>
      </sp:Wss11>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

将客户机 X509Token 作为 EndorsingToken，并且服务器 X509Token 对于消息保护是对称的

客户机和服务器 X509 证书用于相互认证和消息保护。对于 X509Token 对称保护，会创建临时密钥以对消息进行签名和加密。会使用接收方的公用证书对临时密钥进行加密。消息中包括具有客户机的公用证书的指纹表示的 X509Token。会使用客户机的专用密钥对消息签名进行签名。

如果客户机必须同时使用 X509 客户机证书和 UsernameToken 向服务认证它自身，那么最好是使用此策略模板。

以下策略显示了将客户机 X509Token 作为 EndorsingToken，并且服务器 X509Token 对于消息保护是对称的：

```
<wsp:Policy wsu:Id="X509SymmetricAndEndorsing">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/Never">
                <wsp:Policy>
                  <sp:RequireDerivedKeys />
                  <sp:RequireThumbprintReference />
                  <sp:WssX509V3Token11 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
```

```
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
        </wsp:Policy>
      </sp:SymmetricBinding>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EndorsingSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
        <wsp:Policy>
          <sp:X509Token
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:RequireThumbprintReference />
              <sp:WssX509V3Token11 />
            </wsp:Policy>
          </sp:X509Token>
        </wsp:Policy>
      </sp:EndorsingSupportingTokens>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
          <sp:MustSupportRefIssuerSerial />
          <sp:MustSupportRefThumbprint />
          <sp:MustSupportRefEncryptedKey />
          <sp:RequireSignatureConfirmation />
        </wsp:Policy>
      </sp:Wss11>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- `xmlns:wsp="http://www.w3.org/ns/ws-policy"`
- `xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"`
- `xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"`

进行 UsernameToken 认证，并且 X509Token 对于消息保护是对称的

会将具有明文密码的 UsernameToken 用于认证。会使用 X509Token 对称消息保护对请求中的 UsernameToken 以及请求和响应中的 SOAP 主体进行签名和加密。还有消息时间戳记。

对于 X509Token 对称保护，会创建临时密钥以对消息进行签名和加密。会使用接收方的公用证书对临时密钥进行加密。

在此示例中，令牌引用正在使用 RequireThumbprintReference。可以更改策略以使用 RequireIssuerSerialReference 或 RequireKeyIdentifierReference。您还可以修改此策略以使用从临时密钥派生的密钥，通过添加 <sp:RequireDerivedKeys /> 断言来保护消息交换的安全。

如果客户机只能使用 UsernameToken 来认证它自身，并且必须对消息交换进行签名和加密，那么最好是使用此策略模板。

以下策略显示了 UsernameToken 认证，并且 X509Token 对于消息保护是对称的：

```
<wsp:Policy wsu:Id="X509SymmetricForMessageAndUntForClient">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedEncryptedSupportingTokens
        xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:UsernameToken
```

```
            sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10 />
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SignedEncryptedSupportingTokens>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/Never">
                <wsp:Policy>
                  <sp:RequireThumbprintReference />
                  <sp:WssX509V3Token10 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:EncryptSignature />
        </wsp:Policy>
      </sp:SymmetricBinding>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
          <sp:MustSupportRefIssuerSerial />
          <sp:MustSupportRefThumbprint />
          <sp:MustSupportRefEncryptedKey />
          <sp:RequireSignatureConfirmation />
        </wsp:Policy>
      </sp:Wss11>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

基于 HTTP 且具有密码文本、nonce 和 created 时间戳记的 UsernameToken

此策略要求客户机必须发送一个具有明文密码、nonce 和 created 时间戳记的 UsernameToken。不需要具备传输安全性。仅当不在意密码机密性或者通过其他方法提供密码机密性时，此简单策略才适用。可以在预生产环境中使用此策略来进行测试。

以下策略显示了基于 HTTP 且具有密码文本、nonce 和 created 时间戳记的 UsernameToken：

```
<wsp:Policy wsu:Id="UsernameTokenOverHTTP">
  <sp:SupportingTokens>
    <wsp:Policy>
      <sp:UsernameToken
        sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
        <wsp:Policy>
          <sp13:Created />
          <sp13:Nonce />
          <sp:WssUsernameToken11 />
        </wsp:Policy>
      </sp:UsernameToken>
    </wsp:Policy>
  </sp:SupportingTokens>
</wsp:Policy>
```

此示例中使用的命名空间为：

- xmlns:wsp="http://www.w3.org/ns/ws-policy"
- xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
- xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

作为基于 SSL 的 SupportingToken 的 SamlToken

此策略要求您使用 HTTPS 来保护消息，并将 SAML 2.0 令牌用于进行认证。

以下策略显示基于 SSL 的 SAML 2.0 令牌。

```
<wsp:Policy wsu:Id="Saml20TokenOverSSL">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding>
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken>
                <wsp:Policy/>
              </sp:HttpsToken>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SupportingTokens>
        <wsp:Policy>
          <sp:SamlToken
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssSamlV20Token11/>
```

```
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
      </sp:SupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

SAMLToken，其 X509Token 对于消息签名是非对称的

消息中包括具有发送方的公用证书的 X509Token。请求中的 SAML 令牌及请求和响应中的 SOAP 主体是使用 X509Token 非对称消息保护进行签名的。时间戳记也包含在消息中。SAML 2.0 令牌用于认证。

以下策略显示 SAMLToken（其 X509Token 对于消息签名是非对称的）：

```
<wsp:Policy wsu:Id="SamlTokenWithMessageSignature">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:SamlToken
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssSamlV20Token11/>
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <sp:AsymmetricBinding>
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10/>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10/>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:RecipientToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
        </wsp:Policy>
      </sp:AsymmetricBinding>
      <sp:Wss10>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
        </wsp:Policy>
      </sp:Wss10>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
```

```
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

具有 X509Token 非对称消息保护（相互认证）的 SamlToken

消息中包括具有发送方的公用证书的签发者和序列号表示的 X509Token。请求中的 SAML 令牌及请求和响应中的 SOAP 主体是使用 X509Token 非对称消息保护进行签名和加密的。还有消息时间戳记和签名确认。SAML 2.0 令牌用于认证。如果客户机必须同时使用 X509 客户机证书和 SAML 令牌向服务认证它自身，那么最好是使用此策略模板。

以下策略显示了具有 X509Token 非对称消息保护的 SAML 令牌：

```
<wsp:Policy wsu:Id="AsymmetricX509MutualAuthenticationWithSaml">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedEncryptedSupportingTokens
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:SamlToken
                sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssSamlV20Token11/>
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
      </sp:SignedEncryptedSupportingTokens>
      <sp:AsymmetricBinding>
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token
sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/
AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                  <sp:RequireIssuerSerialReference />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken>
            <wsp:Policy>
              <sp:X509Token
sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                <wsp:Policy>
                  <sp:WssX509V3Token10 />
                  <sp:RequireIssuerSerialReference />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:RecipientToken>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:EncryptSignature />
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </wsp:Policy>
      </sp:AsymmetricBinding>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
          <sp:MustSupportRefIssuerSerial />
```

```
          <sp:MustSupportRefThumbprint />
          <sp:MustSupportRefEncryptedKey />
          <sp:RequireSignatureConfirmation />
        </wsp:Policy>
      </sp:Wss11>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

SAMLToken，其 X509Token 对于消息保护是对称的

SAML 令牌用于认证。请求中的 SAML 令牌及请求和响应中的 SOAP 主体是使用 X509Token 对称消息保护进行签名和加密的。还有消息时间戳记。

通过 X509Token 对称保护，系统创建临时密钥以对消息进行签名和加密。临时密钥是使用接收方的公用证书加密的。

在以下样本中，令牌引用将使用 RequireThumbprintReference。可更改策略以使用 RequireIssuerSerialReference 或 RequireKeyIdentifierReference。您还可以修改此策略以使用从临时密钥派生的密钥，通过添加 <sp:RequireDerivedKeys /> 断言来保护消息交换的安全。如果客户机只能使用 SAML 令牌来认证它自身，并且必须对消息交换进行签名和加密，那么最好是使用此策略模板。

以下策略显示使用 SAML 令牌进行认证，并且 X509Token 对于消息保护是对称的：

```
<wsp:Policy wsu:Id="X509SymmetricForMessageAndSamlForClient">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SignedEncryptedSupportingTokens
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:SamlToken
              sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssSamlV20Token11/>
            </wsp:Policy>
          </sp:SamlToken>
        </wsp:Policy>
      </sp:SignedEncryptedSupportingTokens>
      <sp:SymmetricBinding>
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token
sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                <wsp:Policy>
                  <sp:RequireThumbprintReference />
                  <sp:WssX509V3Token10 />
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic128 />
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict />
            </wsp:Policy>
          </sp:Layout>
```

```
          <sp:IncludeTimestamp />
          <sp:OnlySignEntireHeadersAndBody />
          <sp:EncryptSignature />
        </wsp:Policy>
      </sp:SymmetricBinding>
      <sp:Wss11>
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier />
          <sp:MustSupportRefIssuerSerial />
          <sp:MustSupportRefThumbprint />
          <sp:MustSupportRefEncryptedKey />
          <sp:RequireSignatureConfirmation />
        </wsp:Policy>
      </sp:Wss11>
      <sp:SignedParts>
        <sp:Body />
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body />
      </sp:EncryptedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

为 WS-Security 开发密码回调处理程序

在 xigemaAS 概要文件中，可以在许多 WS-Security 方案中使用密码回调处理程序来检索密码。例如，可以检索密码以生成 UsernameToken、打开密钥库或者访问专用密钥。当您使用 PasswordDigest 时，提供者需要密码回调处理程序。必须将密码回调处理程序打包为 xigemaAS 概要文件中的用户功能部件。

此任务描述如何开发密码回调处理程序以检索用户名和密钥库密钥密码。

1. 开发密码回调处理程序。

   以下示例显示了一个回调处理程序：

```
package com.ibm.ws.wssecurity.example.cbh;

import java.util.HashMap;
import java.util.Map;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import org.apache.ws.security.WSPasswordCallback;

public class SamplePasswordCallback implements CallbackHandler {

  private Map<String, String> userPasswords =  new HashMap<String, String>();
  private Map<String, String> keyPasswords = new HashMap<String, String>();
  public SamplePasswordCallback() {
    // some example user passwords
    userPasswords.put("user1", "user1pswd");
    userPasswords.put("admin", "adminpswd");
    // some example key passwords
    keyPasswords.put("alice",  "keypwsd");
    keyPasswords.put("bob",   "keypswd");
  }
  public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
    for (int i = 0; i < callbacks.length; i++) {
      WSPasswordCallback pwcb = (WSPasswordCallback)callbacks[i];
      String id = pwcb.getIdentifier();
      String pass = null;
      switch (pwcb.getUsage()) {
        case WSPasswordCallback.USERNAME_TOKEN_UNKNOWN:
        case WSPasswordCallback.USERNAME_TOKEN:
          pass = userPasswords.get(id);
          pwcb.setPassword(pass);
              break;        case WSPasswordCallback.SIGNATURE:
        case WSPasswordCallback.DECRYPT:
          pass = keyPasswords.get(id);
          pwcb.setPassword(pass);
              break;      }
    }
```

```
  }
}
```

2. 为回调处理程序创建 MANIFEST.MF 文件。

   以下示例显示了 MANIFEST.MF 文件：

```
Manifest-Version: 1.0
Bnd-LastModified: 1359415594428
Build-Identifier: SNAPSHOT-Mon Jan 28 17:26:34 CST 2013
Bundle-Copyright: The Program materials contained in this file are Vsettan
copyright materials.
Copyright Vsettan Corp. 2015 All Rights
Reserved *
Bundle-Description: An PasswordCallbackHandler; version=1.0.0
Bundle-ManifestVersion: 2
Bundle-Name: wssecuritycbh
Bundle-SymbolicName: com.ibm.ws.wssecurity.example.cbh
Bundle-Vendor: Vsettan
Bundle-Version: 1.0.0
Created-By: 1.6.0 (Vsettan Corporation)
Export-Package: com.ibm.ws.wssecurity.example.cbh;uses:="com.ibm.websphe
re.ras.annotation,javax.security.auth.callback";version="1.0.0"
Import-Package: com.ibm.websphere.ras,com.ibm.websphere.ras.annotation,c
om.ibm.ws.ffdc,javax.security.auth.callback,org.apache.ws.security;version="[1.6,2)"
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version>=1.6))"
Tool: Bnd-2.1.0.20120920-170235
WS-TraceGroup: WSSecurity
```

3. 将回调处理程序打包在 JAR 文件中。

   创建一个具有回调处理程序类以及在先前步骤中所创建的 MANIFEST.MF 文件的 JAR 文件。

   以下示例显示了称为 SampleCbh.jar 的样本 JAR 文件的内容：

```
META-INF/MANIFEST.MF
com/ibm/ws/wssecurity/example/cbh/SamplePasswordCallback.class
```

4. 创建功能部件清单文件。

   以下示例显示了一个称为 wsseccbh-1.0.mf 的样本功能部件清单文件：

```
Subsystem-ManifestVersion: 1
Subsystem-SymbolicName: wsseccbh-1.0; visibility:=public
Subsystem-Version: 1.0.0
Subsystem-Content: com.ibm.ws.wssecurity.example.cbh; version="[1,1.0.100)";
    location:="lib/"; type="osgi.bundle"; start-phase:=APPLICATION_EARLY

Subsystem-Type: osgi.subsystem.feature
IBM-Feature-Version: 2

IBM-API-Package: com.ibm.ws.wssecurity.example.cbh; version="1.0"; type="internal"
```

5. 将回调处理程序作为 xigemaAS 中的用户功能部件来安装。

   将回调处理程序 JAR 文件和功能部件清单文件复制到 xigemaAS 的用户目录下。

   以下示例说明了在何处复制这些文件：

```
build.image/wlp/usr/extension/lib/SampleCbh.jar
build.image/wlp/usr/extension/lib/features/wsseccbh-1.0.mf
```

6. 将回调处理程序配置为 server.xml 文件中的用户功能部件。

   将回调处理程序定义为 server.xml 文件中的用户功能部件。

   以下示例说明如何将回调处理程序定义为用户功能部件：

```
<featureManager>
  <feature>usr:wsseccbh-1.0</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
```

```
  <feature>jaxws-2.2</feature>
  <feature>wsSecurity-1.1</feature>
</featureManager>
```

您已成功开发密码回调处理程序，并且已将该回调处理程序安装到 xigemaAS。

现在，您可以使用此回调处理程序来检索 WS-Security 配置中使用的 UsernameToken 和专用密钥的密码。

使用 WS-Security 策略保护 Web Service

可以使用 Vsettan xigemaAS概要文件中的 WS-Security 策略来开发和保护 Java™ API for XML Web Services (JAX-WS) Web Service。通过教程提供了一些示例来说明在开发和保护 xigemaAS Web Service 时涉及到的常规步骤。请勿在生产环境中使用这些示例。请复审您自己的安全需求，以制定 Web 服务描述语言 (WSDL) 合同和 WS-Security 策略以保护 Web Service 应用程序。

此任务描述如何开发简单的 JAX-WS Web Service 应用程序以及使用 WS-Security 策略来保护这些应用程序。

此任务使用两个密钥库：enc-sender.jceks 和 enc-receiver.jceks。

1. 创建 WSDL 合同和 WS-Security 策略。

   在此样本中，创建了一个 WSDL 合同，它包含用于 JAX-WS Web Service 提供程序应用程序的 WS-Security 策略。将使用称为具有 *X509Token* 非对称消息保护 *(* 相互认证 *)* 的 *UsernameToken* 的 WS-Security 策略模板。客户机会对 SOAP 主体进行签名和加密，并对请求消息中的 UsernameToken 进行签名和加密。在响应消息中，提供程序会对 SOAP 主体进行签名和加密。对于此示例中的安全性约束的完整描述，请参阅具有 *X509Token* 非对称消息保护 *(* 相互认证 *)* 的 *UsernameToken*。

   a. 为您的服务创建 WSDL 合同。

      以下示例显示了样本 WSDL 合同：

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://com/ibm/was/wssample/sei/echo/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="WSSampleSei"
                  targetNamespace="http://com/ibm/was/wssample/sei/echo/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://com/ibm/was/wssample/sei/echo/"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="echoStringResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="echoResponse" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="echoStringInput">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="echoInput" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="echoOperationRequest">
    <wsdl:part element="tns:echoStringInput" name="parameter" />
  </wsdl:message>
  <wsdl:message name="echoOperationResponse">
    <wsdl:part element="tns:echoStringResponse" name="parameter" />
  </wsdl:message>
  <wsdl:portType name="EchoServicePortType">
    <wsdl:operation name="echoOperation">
      <wsdl:input message="tns:echoOperationRequest" />
      <wsdl:output message="tns:echoOperationResponse" />
    </wsdl:operation>
  </wsdl:portType>
```

```
  <wsdl:binding name="Echo1SOAP" type="tns:EchoServicePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="echoOperation">
      <soap:operation soapAction="echoOperation" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Echo1Service">
    <wsdl:port binding="tns:Echo1SOAP" name="Echo1ServicePort">
      <soap:address location="http://localhost:8010/WSSampleSei/Echo1Service" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

b. 将支持安全策略所需的命名空间添加到 WSDL 中的 wsdl:definitions 元素。

以下示例显示了所添加的命名空间：

```
xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
xmlns:sp13="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802"
xmlns:wsaws="http://www.w3.org/2005/08/addressing"
```

c. 在 WSDL 中，正好位于 wsdl:binding 元素之前的位置，添加 WS-Security 策略片段。

此示例中将使用具有 *X509Token* 非对称消息保护 *(* 相互认证 *)* 的 *UsernameToken* 中的策略模板。

d. 将安全策略的 wsp:PolicyReference 添加到 wsdl:binding 元素。

以下示例显示了 wsp:PolicyReference：

```
<wsp:PolicyReference URI="#AsymmetricX509MutualAuthenticationWithUnt" />
```

e. 验证最终的 WSDL 看起来是否类似于示例。

以下示例显示了最终的 WSDL：

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://com/ibm/was/wssample/sei/echo/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="WSSampleSei"
                  xmlns:wsp="http://www.w3.org/ns/ws-policy"
                   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
                  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
                  xmlns:sp13="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802"
                  xmlns:wsaws="http://www.w3.org/2005/08/addressing"
                  targetNamespace="http://com/ibm/was/wssample/sei/echo/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://com/ibm/was/wssample/sei/echo/"
                 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="echoStringResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="echoResponse" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="echoStringInput">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="echoInput" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
```

```
  </wsdl:types>
  <wsdl:message name="echoOperationRequest">
    <wsdl:part element="tns:echoStringInput" name="parameter" />
  </wsdl:message>
  <wsdl:message name="echoOperationResponse">
    <wsdl:part element="tns:echoStringResponse" name="parameter" />
  </wsdl:message>
  <wsdl:portType name="EchoServicePortType">
    <wsdl:operation name="echoOperation">
      <wsdl:input message="tns:echoOperationRequest" />
      <wsdl:output message="tns:echoOperationResponse" />
    </wsdl:operation>
  </wsdl:portType>
  <wsp:Policy wsu:Id="AsymmetricX509MutualAuthenticationWithUnt">
    <wsp:ExactlyOne>
      <wsp:All>
        <sp:SignedEncryptedSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
          <wsp:Policy>
            <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
              <wsp:Policy>
                <sp:WssUsernameToken10 />
              </wsp:Policy>
            </sp:UsernameToken>
          </wsp:Policy>
        </sp:SignedEncryptedSupportingTokens>
        <sp:AsymmetricBinding>
          <wsp:Policy>
            <sp:InitiatorToken>
              <wsp:Policy>
                <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                  <wsp:Policy>
                    <sp:WssX509V3Token10 />
                    <sp:RequireIssuerSerialReference />
                  </wsp:Policy>
                </sp:X509Token>
              </wsp:Policy>
            </sp:InitiatorToken>
            <sp:RecipientToken>
              <wsp:Policy>
                <sp:X509Token sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/Never">
                  <wsp:Policy>
                    <sp:WssX509V3Token10 />
                    <sp:RequireIssuerSerialReference />
                  </wsp:Policy>
                </sp:X509Token>
              </wsp:Policy>
            </sp:RecipientToken>
            <sp:Layout>
              <wsp:Policy>
                <sp:Strict />
              </wsp:Policy>
            </sp:Layout>
            <sp:IncludeTimestamp />
            <sp:OnlySignEntireHeadersAndBody />
            <sp:EncryptSignature />
            <sp:AlgorithmSuite>
              <wsp:Policy>
                <sp:Basic128 />
              </wsp:Policy>
            </sp:AlgorithmSuite>
          </wsp:Policy>
        </sp:AsymmetricBinding>
        <sp:Wss11>
          <wsp:Policy>
            <sp:MustSupportRefKeyIdentifier />
            <sp:MustSupportRefIssuerSerial />
            <sp:MustSupportRefThumbprint />
            <sp:MustSupportRefEncryptedKey />
            <sp:RequireSignatureConfirmation />
          </wsp:Policy>
        </sp:Wss11>
        <sp:SignedParts>
```

```
          <sp:Body />
        </sp:SignedParts>
        <sp:EncryptedParts>
          <sp:Body />
        </sp:EncryptedParts>
      </wsp:All>
    </wsp:ExactlyOne>
  </wsp:Policy>
  <wsdl:binding name="Echo1SOAP" type="tns:EchoServicePortType">
    <wsp:PolicyReference URI="#AsymmetricX509MutualAuthenticationWithUnt" />
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="echoOperation">
      <soap:operation soapAction="echoOperation" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Echo1Service">
    <wsdl:port binding="tns:Echo1SOAP" name="Echo1ServicePort">
      <soap:address location="http://localhost:8010/WSSampleSei/Echo1Service" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

2. 使用 WSDL 来创建 Web Service 应用程序。

   可以在将 WS-Security 策略添加到 WSDL 之前或之后完成此步骤。可以使用受支持的工具根据前一节中开发的 WSDL 来创建 JAX-WS Web Service 应用程序。

   以下示例显示了使用 Rational Application Developer (RAD) 工具从 WSDL 开发的 Web Service 应用程序：

```
@javax.jws.WebService (endpointInterface="com.ibm.was.wssample.sei.echo.EchoServicePortType",
                       targetNamespace="http://com/ibm/was/wssample/sei/echo/",
                       serviceName="Echo1Service",
                       wsdlLocation = "WEB-INF/wsdl/Echo.wsdl",
                       portName="Echo1ServicePort")
public class Echo1SOAPImpl {

  public EchoStringResponse echoOperation(EchoStringInput parameter) {
    String strInput = (parameter == null ? "input_is_null" : parameter.getEchoInput() );
    try {
      com.ibm.was.wssample.sei.echo.EchoStringResponse strOutput = new EchoStringResponse();
      strOutput.setEchoResponse( "Echo1SOAPImpl>>" + strInput );
      return strOutput;
    } catch (java.lang.Exception ex) {
      ex.printStackTrace();
    }
  }

@WebService (name = "EchoServicePortType",
             targetNamespace = "http://com/ibm/was/wssample/sei/echo/")
@SOAPBinding (parameterStyle = SOAPBinding.ParameterStyle.BARE)
@XmlSeeAlso ({
    ObjectFactory.class
})

public interface EchoServicePortType {

  @WebMethod (action = "echoOperation")
  @WebResult (name = "echoStringResponse", targetNamespace = "http://com/ibm/was/wssample/
sei/echo/", partName = "parameter")
  public  EchoStringResponse echoOperation(
    @WebParam (name = "echoStringInput", targetNamespace = "http://com/ibm/was/wssample/sei/
echo/", partName = "parameter")
    EchoStringInput parameter);

}
```

以下代码显示了一个将调用 Web Service 提供者应用程序的受管 Web Service 客户机：

```
@WebServlet("ClientServlet")
public class ClientServlet extends HttpServlet {

  @WebServiceRef (value=Echo1Service.class, wsdlLocation="Echo.wsdl")
  Echo1Service echo1Service;

  public  ClientServlet() {
    super ();
  }

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response) throws ServletException, IOException {
    processRequest(request, response);
  }

  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException, IOException {
    processRequest(request, response);
  }

  private void processRequest(HttpServletRequest req,
                              HttpServletResponse resp) throws ServletException, IOException
 {

    String endpointURL = "http://localhost:8010/WSSampleSei/Echo1Service";

    Echo1ServicePortProxy proxy = new Echo1ServicePortProxy(echo1Service);
    proxy._getDescriptor().setEndpoint(endpointURL);

    echoParm = new ObjectFactory().createEchoStringInput();
    echoParm.setEchoInput("Hello");

    String retval = proxy.echoOperation(echoParm).getEchoResponse();

  }
}
```

以下示例显示了 Web Service 提供者应用程序 WAR 文件的文件结构：

```
WEB-INF/web.xml
WEB-INF/wsdl/Echo.wsdl
WEB-INF/classes/com/ibm/was/wssample/sei/echo/Echo1SOAPImpl.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/EchoServicePortType.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/EchoStringInput.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/EchoStringResponse.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/ObjectFactory.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/package-info.class
```

以下示例显示了 Web Service 客户机应用程序 WAR 文件的文件结构：

```
WEB-INF/web.xml
WEB-INF/wsdl/Echo.wsdl
WEB-INF/classes/com/ibm/was/wssample/client/ClientServlet.class
WEB-INF/classes/com/ibm/was/wssample/client/SampleClient.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/Echo1Service.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/Echo1ServicePortProxy.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/EchoStringInput.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/EchoStringResponse.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/ObjectFactory.class
WEB-INF/classes/com/ibm/was/wssample/sei/echo/package-info.class
```

3. 开发回调处理程序。

   您必须开发回调处理程序以检索用户名和密钥库密钥密码。当您生成 UsernameToken 时将使用用户名密码。密钥库密码用于访问密钥库中的专用密钥。回调处理程序必须返回明文密码，并且作为 xigemaAS 的用户功能部件来打包和安装。

以下样本代码说明了回调处理程序：

```
package com.ibm.ws.wssecurity.example.cbh;

import java.util.HashMap;
import java.util.Map;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import org.apache.ws.security.WSPasswordCallback;

public class SamplePasswordCallback implements CallbackHandler {
  private Map<String, String> userPasswords = new HashMap<String, String>();
  private Map<String, String> keyPasswords = new HashMap<String, String>();
  public SamplePasswordCallback() {
    // some example user passwords
    userPasswords.put("user1", "user1pswd");
    userPasswords.put("admin", "adminpswd");
    // some example key passwords
    keyPasswords.put("alice", "keypwsd");
    keyPasswords.put("bob", "keypswd");
  }
  public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
    for (int i = 0; i < callbacks.length; i++) {
      WSPasswordCallback pwcb = (WSPasswordCallback)callbacks[i];
      String id = pwcb.getIdentifier();
      String pass = null;
      switch (pwcb.getUsage()) {
        case WSPasswordCallback.USERNAME_TOKEN_UNKNOWN:
        case WSPasswordCallback.USERNAME_TOKEN:
          pass = userPasswords.get(id);
          pwcb.setPassword(pass);
          break;
        case WSPasswordCallback.SIGNATURE:
        case WSPasswordCallback.DECRYPT:
          pass = keyPasswords.get(id);
          pwcb.setPassword(pass);
          break;
      }
    }
  }
}
```

以下示例显示了与回调处理程序打包在一起的 MANIFEST.MF 文件。

```
Manifest-Version: 1.0
Bnd-LastModified: 1359415594428
Build-Identifier: SNAPSHOT-Mon Jan 28 17:26:34 CST 2013
Bundle-Copyright: The Program materials contained in this file are Vsettan
copyright materials.
Copyright Vsettan Corp. 2015 All Rights
Reserved *
Bundle-Description: An PasswordCallbackHandler; version=1.0.0
Bundle-ManifestVersion: 2
Bundle-Name: wssecuritycbh
Bundle-SymbolicName: com.ibm.ws.wssecurity.example.cbh
Bundle-Vendor: Vsettan
Bundle-Version: 1.0.0
Created-By: 1.6.0 (Vsettan Corporation)
Export-Package: com.ibm.ws.wssecurity.example.cbh;uses:="com.ibm.websphe
re.ras.annotation,javax.security.auth.callback";version="1.0.0"
Import-Package: com.ibm.websphere.ras,com.ibm.websphere.ras.annotation,c
om.ibm.ws.ffdc,javax.security.auth.callback,org.apache.ws.security;version="[1.6,2)"
Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version>=1.6))"
Tool: Bnd-2.1.0.20120920-170235
WS-TraceGroup: WSSecurity
```

a. 创建一个具有回调处理程序以及功能部件清单文件 wsseccbh-1.0.mf 的 JAR 文件。

创建一个具有下列内容的称为 SampleCbh.jar 的 JAR 文件：

```
META-INF/MANIFEST.MF
```

```
com/ibm/ws/wssecurity/example/cbh/SamplePasswordCallback.class
```

以下示例显示了 wsseccbh-1.0.mf 文件：

```
Subsystem-ManifestVersion: 1
Subsystem-SymbolicName: wsseccbh-1.0; visibility:=public
Subsystem-Version: 1.0.0
Subsystem-Content: com.ibm.ws.wssecurity.example.cbh; version="[1,1.0.100)";
    location:="lib/"; type="osgi.bundle"; start-phase:=APPLICATION_EARLY

Subsystem-Type: osgi.subsystem.feature
IBM-Feature-Version: 2

IBM-API-Package: com.ibm.ws.wssecurity.example.cbh; version="1.0"; type="internal"
```

b. 复制位于 xigemaAS 的用户目录下的回调处理程序 JAR 文件和功能部件清单文件。

以下示例显示了回调处理程序 JAR 文件和功能部件清单文件的复制位置：

```
build.image/wlp/usr/extension/lib/SampleCbh.jar
build.image/wlp/usr/extension/lib/features/wsseccbh-1.0.mf
```

4. 配置 xigemaAS 服务器中的 WS-Security。

启用 xigemaAS 概要文件服务器配置文件 server.xml 中的 WS-Security 功能部件，并为在先前章节中开发的样本 Web Service 客户机和提供程序应用程序配置 WS-Security。

以下示例说明如何配置 WS-Security：

```
<server>
  <featureManager>
    <feature>usr:wsseccbh-1.0</feature>
    <feature>servlet-3.0</feature>
    <feature>appSecurity-2.0</feature>
    <feature>jsp-2.2</feature>
    <feature>jaxws-2.2</feature>
    <feature>wsSecurity-1.1</feature>
  </featureManager>
  <basicRegistry id="basic" realm="customRealm">
    <user name="user1" password="user1pswd" />
    <user name="user2" password="user2pswd" />
  </basicRegistry>
  <wsSecurityProvider id="default"
    ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.SamplePasswordCallback"
    ws-security.signature.username="bob">
    <signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jceks"
      org.apache.ws.security.crypto.merlin.keystore.password="storepswd"
      org.apache.ws.security.crypto.merlin.keystore.alias="bob"
      org.apache.ws.security.crypto.merlin.file="${server.config.dir}/enc-receiver.jceks" />
    <encryptionProperties org.apache.ws.security.crypto.merlin.keystore.type="jceks"
      org.apache.ws.security.crypto.merlin.keystore.password="storepswd"
      org.apache.ws.security.crypto.merlin.keystore.alias="alice"
      org.apache.ws.security.crypto.merlin.file="${server.config.dir}/enc-receiver.jceks" />
  </wsSecurityProvider>
  <wsSecurityClient id="default"
    ws-security.password="security"
    ws-security.username="user1"
    ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.SamplePasswordCallback"
    ws-security.encryption.username="alice">
    <signatureProperties org.apache.ws.security.crypto.merlin.keystore.type="jceks"
      org.apache.ws.security.crypto.merlin.keystore.password="storepswd"
      org.apache.ws.security.crypto.merlin.keystore.alias="alice"
      org.apache.ws.security.crypto.merlin.file="${server.config.dir}/enc-sender.jceks"/>
    <encryptionProperties org.apache.ws.security.crypto.merlin.keystore.type="jceks"
      org.apache.ws.security.crypto.merlin.keystore.password="storepswd"
      org.apache.ws.security.crypto.merlin.keystore.alias="bob"
      org.apache.ws.security.crypto.merlin.file="${server.config.dir}/enc-sender.jceks" />
  </wsSecurityClient>
</server>
```

您已使用 WS-Security 策略来保护 Web Service。

在第一步中创建的样本 WS-Security 策略将生成与下列消息相似的 SOAP 请求和响应消息。

以下示例显示了 SOAP 请求消息：

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                  soap:mustUnderstand="1">
      <wsse:BinarySecurityToken
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
        wsu:Id="X509-B1165B2A578AFFC7D613649595665924">...
      </wsse:BinarySecurityToken>
      <wsu:Timestamp wsu:Id="TS-1">
        <wsu:Created>2013-04-03T03:26:06.549Z</wsu:Created>
        <wsu:Expires>2013-04-03T03:31:06.549Z</wsu:Expires>
      </wsu:Timestamp>
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-B1165B2A578AFFC7D613649595666705">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"></xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <ds:X509Data>
              <ds:X509IssuerSerial>
                <ds:X509IssuerName>CN=Bob,O=VSETTAN,C=CN</ds:X509IssuerName>
                <ds:X509SerialNumber>240546756673 89</ds:X509SerialNumber>
              </ds:X509IssuerSerial>
            </ds:X509Data>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#ED-4"></xenc:DataReference>
          <xenc:DataReference URI="#ED-5"></xenc:DataReference>
          <xenc:DataReference URI="#ED-6"></xenc:DataReference>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <xenc:EncryptedData
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="ED-6" Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                  <wsse:SecurityTokenReference
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
            xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
            <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595666705"></wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
      <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                          Id="ED-5"
                          Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                  <wsse:SecurityTokenReference
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
            xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
            <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595666705"></wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
    </wsse:Security>
  </SOAP-ENV:Header>
  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
             wsu:Id="Id-1788936596">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                        Id="ED-4"
                        Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></xenc:EncryptionMethod>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <wsse:SecurityTokenReference
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
          xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
          wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
```

```
          <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595666705"></wsse:Reference>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

以下示例显示了 SOAP 响应消息：

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
                   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
                   soap:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="TS-7">
        <wsu:Created>2013-04-03T03:26:07.286Z</wsu:Created>
        <wsu:Expires>2013-04-03T03:31:07.286Z</wsu:Expires>
      </wsu:Timestamp>
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-
B1165B2A578AFFC7D613649595673129">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"></
xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <ds:X509Data>
              <ds:X509IssuerSerial>
                <ds:X509IssuerName>CN=Alice,O=VSETTAN,C=CN</ds:X509IssuerName>
                <ds:X509SerialNumber>24054530212598</ds:X509SerialNumber>
              </ds:X509IssuerSerial>
            </ds:X509Data>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#ED-10"></xenc:DataReference>
          <xenc:DataReference URI="#ED-11"></xenc:DataReference>
          <xenc:DataReference URI="#ED-12"></xenc:DataReference>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                          Id="ED-12"
                          Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></
xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
            xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKey">
            <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595673129"></wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
      <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                          Id="ED-11"
                          Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></
xenc:EncryptionMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
            xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
```

```
            wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKey">
            <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595673129"></wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
    </wsse:Security>
  </SOAP-ENV:Header>
  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
             wsu:Id="Id-2035943749">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
                        Id="ED-10"
                        Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"></
xenc:EncryptionMethod>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <wsse:SecurityTokenReference
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
          xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd"
          wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKey">
          <wsse:Reference URI="#EK-B1165B2A578AFFC7D613649595673129"></wsse:Reference>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

现在，可以开发您自己的 WSDL 文件，并使用匹配安全性要求的 WS-Security 策略来保护 Web Service 应用程序。

将 CXF WS-Security 迁移到 xigemaAS

将 CXF WS-Security 迁移到 xigemaAS 概要文件是一项简单而又直接的任务。此迁移工作包括将 Spring 或者类似于 Spring 的配置迁移到 server.xml 文件。如果需要 CallbackHandler，那么您还必须将密码 CallbackHandler 作为 xigemaAS 用户功能部件来打包和安装。

请确保您对 xigemaAS 概要文件外部的启用了 CXF WS-Security 的 Web Service 有所了解。

要使用 WS-Security 来保护 Web Service 应用程序，JAX-WS 应用程序必须包含一个具有嵌入式 WS-Security 策略的 WSDL 文件。在 `wsdl:binding` 和/或 `wsdl:operation` 部分，必须存在对于该嵌入式 WS-Security 策略的 PolicyReference。将 Web Service 迁移到 xigemaAS 概要文件之后，您可以启用由 WS-Security 策略驱动的 WS-Security 配置。此任务描述了您可以如何将 Apache CXF WS-Security 配置迁移到 xigemaAS。

1. 将 `wsSecurity-1.1` 功能部件添加至 server.xml 文件，以在 xigemaAS 概要文件中启用 WS-Security。
2. 将 WS-Security 配置添加到 server.xml 文件。

   xigemaAS 概要文件中的 CXF WS-Security 不支持 Spring 配置文件或者其他供应商提供的等价配置文件。必须将在 Spring 或者其等价配置文件中的策略以外定义的额外配置迁移到 xigemaAS 中的 server.xml 文件。

   创建 <wsSecurityClient> 元素来保存客户端配置，创建 <wsSecurityProvider> 元素来保存服务器端配置。必须保留 CXF 和 WSS4J 中的所有配置属性“名称/值”对。可以使用 Spring 或者等价的配置文件中的相同“名称/值”对。对于 crypto 属性，您必须创建 <signatureProperties> 和 <encryptionProperties> 子元素来保存所有必需的属性。有关更多信息，请参阅 *Web Service 安全性缺省配置*。

3. 如果 Spring 配置文件中具有密码回调处理程序，请将密码回调处理程序作为 xigemaAS 概要文件用户功能部件来打包。

   有关密码 CallbackHandler 的更多信息，请参阅为 *WS-Security* 开发密码回调处理程序。

此时，您已将“WSDL 优先”Web Service 迁移到 xigemaAS。

向 SAML（安全性断言标记语言）令牌认证 Web Service 客户机

Web Services Security SAML Token Profile 1.1 定义如何将安全性断言标记语言 (SAML) V1.1 和 V2.0 断言 与 Web Service 安全性 (WSS)：SOAP 消息安全性 V1.1 规范配合使用。xigemaAS 通过不记名确认方法支持对 SAML V2.0 断言使用 Web Services Security SAML Token Profile 1.1。

xigemaAS 通过不记名确认方法 (`<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">`) 支持 SAML V2.0。Web Service 客户机将 SAML 令牌传递至 Web Service 提供程序以标识请求者，Web Service 提供程序可使用此 SAML 令牌信息来认证请求者。

SAML 令牌的需求表示为 WS-Security 策略中的其中一个支持令牌。可将 `SamlToken` 需求添加为支持令牌断言（包括 `SupportingTokens`、`SignedSupportingTokens`、`SignedEncryptedSupportingTokens` 和 `EncryptedSupportingTokens`）之一中的必需令牌。

1. 配置 Web Service 客户机以传播 SAML 令牌。

   `Saml20Token` (`com.ibm.websphere.security.saml2.Saml20Token`) 必须存在于 `RunAs` 主体集中。如果配置了 `SamlToken` 策略断言，那么 xigemaAS WS-Security 运行时将自动从 `RunAs` 主体集中抽取 `Saml20Token`，并将 SAML 断言附加至 `<wsse:Security>` SOAP 头。需要执行以下步骤：

   a. 将 `wsSecuritySaml-1.1` 添加至 server.xml 文件中的 `featureManager`。

   ```
   <featureManager>
           <feature>wsSecuritySaml-1.1</feature>
   </featuremanager>
   ```

   b. 应用程序必须指定 SAML 令牌作为 .wsdl 文件的策略中的支持令牌（包括 `SupportingTokens`、`SignedSupportingTokens`、`SignedEncryptedSupportingTokens` 和 `EncryptedSupportingTokens` 令牌断言）。

   ```
   <wsp:Policy wsu:Id="CallerHttpPolicy">
    <sp:SupportingTokens>
     <wsp:Policy>
      <sp:SamlToken
       sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
   securitypolicy/200702/IncludeToken/AlwaysToRecipient">
       <wsp:Policy>
        <sp:WssSamlV20Token11 />
       </wsp:Policy>
      </sp:SamlToken>
     </wsp:Policy>
    </sp:SupportingTokens>

   </wsp:Policy>
   ```

2. 向 SAML 令牌认证 Web Service。

如果 Web Service 提供程序接收带有 SAML 令牌的 SOAP 消息，那么它会先验证令牌，然后使用 SAML 令牌属性来完成认证过程。以下是验证 SAML 令牌涉及的步骤：

a. 将 `wsSecuritySaml-1.1` 添加至 server.xml 文件中的 `featureManager`。

```
<featureManager>
        <feature>wsSecuritySaml-1.1</feature>
</featuremanager>
```

b. 应用程序必须指定 SAML 令牌作为 .wsdl 文件的策略中的支持令牌。

c. 验证令牌时，将使用服务器配置中的以下属性。

```
<wsSecurityProvider id="default"
          ..
          ..
          <samltoken
               wantAssertionsSigned="true"
               clockSkew="5m"
               timeToLive="30m"
               requiredSubjectConfirmationMethod="bearer"
               audienceRestrictions="https://../aud1"
               audienceRestrictions="aud2" ..          />


</wsSecurityProvider>
```

- `wantAssertionsSigned` - 指定 WS-Security 提供程序接收到的 SOAP 消息中的 `<saml:Assertion>` 元素是否已签名。
- `clockSkew` - 生成 SAML 令牌的系统与接收 SAML 令牌之间的系统之间的允许时差。缺省值为 5 分钟。
- `timeToLive` - SAML 断言的生存时间。SAML 断言中缺少 `NoOnOrAfter` 条件时，此设置是必需的。缺省值为 30 分钟。
- `requiredSubjectConfirmationMethod` - SAML 断言中的主体集确认方法。缺省值为 `bearer`。
- `audienceRestrictions` - SAML 断言的允许受众。缺省值为允许所有受众。

有关认证涉及的步骤的更多详细信息，请参阅创建 *WS-Security SAML* 调用者配置。

Web Service 安全性 HTTPS 传输策略断言

可以使用 WSDL 文件中定义的 Web Service 安全性 (WS-Security) 策略中的断言来确保使用 HTTPS 安全传输保护 SOAP 消息。将 HTTPS 与安全性令牌（例如，具有明文密码的 UsernameToken）配合使用时，有助于确保机密性。

WS-Security 策略中 HTTPS 传输的断言不设置请求者与提供程序之间的 HTTPS 传输。它们仅确保在调用具有所定义的策略的 Web Service 应用程序时使用 HTTPS 传输。要对 Web Service 启用传输安全性，请参阅在传输级别保护 *Web Service*（见第 1427 页）。

确保在 Web Service 客户机和 Web Service 提供程序之间设置了 HTTPS。要使用 HTTPS 安全传输来保护 SOAP 消息，请完成下列额外步骤。

1. 必须通过添加 `wsSecurity-1.1` 功能部件来显式地启用 WS-Security。请确保您还将 `appSecurity-2.0`、`servlet-3.0`（或 `servlet-3.1`）和 `jaxws-2.2` 以及其他必需的 xigemaAS 功能部件添加至 xigemaAS 概要文件的 server.xml 文件。

2. 附加到 Web Service 应用程序的 WS-Security 策略必须包括 `TransportBinding` 断言，并且必须与 HTTPS 配置相匹配。以下示例说明了样本 `TransportBinding` 断言：

```
<wsp:Policy xmlns:wsp="..." xmlns:sp="...">
    <sp:TransportBinding>
        <wsp:Policy>
            <sp:TransportToken>
                <wsp:Policy>
                    <sp:HttpsToken />
                </wsp:Policy>
            </sp:TransportToken>
            <sp:AlgorithmSuite>
                <wsp:Policy>
                    <sp:Basic256 />
                </wsp:Policy>
            </sp:AlgorithmSuite>
            <sp:Layout>
                <wsp:Policy>
                    <sp:Strict />
                </wsp:Policy>
            </sp:Layout>
            <sp:IncludeTimestamp />
        </wsp:Policy>
    </sp:TransportBinding>
</wsp:Policy>
```

通过在 HTTPS 配置后面完成这些额外步骤，需要通过 HTTPS 才能将 SOAP 消息从 Web Service 客户机发送到 Web Service 提供程序。

2.6.13 安全注意事项

当您配置 xigemaAS 的安全性时，请考虑下列事项。

LTPA

- 请保护对于 LTPA 密钥文件的文件访问权，因为该文件包含用来对用户数据进行加密和解密的密码信息。请确保只有服务器和管理员对此文件具有访问权。
- 请确保所有服务器都使用相同的 LTPA 密钥。此外，请确保所有服务器的日期和时间同步。
- 当您指定密码时，请确保它是所有使用同一组 LTPA 密钥的服务器的同一密码。该密码不用来生成密钥，而是用来对 LTPA 密钥文件进行加密以防止读取密钥。如果您将 LTPA 密钥文件复制到另一个 xigemaAS 概要文件服务器以实现单点登录 (SSO)，那么需要输入密码才能获得对于 LTPA 密钥文件中的密钥的访问权。有关 LTPA 的更多信息，请参阅在 *xigemaAS 概要文件上配置 LTPA* 主题。

密码

- 使用 securityUtility encode 命令对密码进行加密。
- 如果您使用 `wlp.password.encryption.key` 属性来覆盖缺省加密密钥，请在存储于服务器的正常配置目录外部的独立配置文件中设置此属性。

授权

- 如果您在应用程序中指定 auth-constraint，但是没有任何角色，那么将不允许任何用户访问资源。
- 应谨慎指定 EVERYONE 特殊主体，因为这样指定就不会保护资源。

认证

- 在 <authCache> 元素中指定的认证高速缓存的超时值必须小于在 <ltpa> 元素中指定的 LTPA 令牌的到期值。

2.7 在 xigemaAS 环境中开发应用程序

xigemaAS 是一个轻量级的可组合应用程序服务器，它为 Web 应用程序和 OSGi 应用程序提供了方便的应用程序开发环境。

xigemaAS 概要文件通过提供下列主要好处和其他好处来简化应用程序开发：

- 供开发之用的免费顺畅下载
- 超轻量级模块化运行时环境，安装大小在 50 MB 以下
- 启动时间极快；例如，对于简单的 Web 应用程序，不到 5 秒
- 简化的配置，缩短了投产时间
- Web 应用程序的 Java™ EE 和 OSGi 应用程序部署支持
- LDAP 注册表支持
- 将应用程序和已配置的服务器作为一个软件包来部署
- 已打包应用程序和服务器的受管集中部署
- 对于分布式平台、 Mac OS 的平台支持

重新启动速度极快，加上其较小的大小、动态行为和易于使用等特点，使 xigemaAS 概要文件成为开发者的一个很好选择，这样在不需要传统企业应用程序服务器概要文件所需的完整 Java™ EE 环境的情况下，也可以构建 Web 应用程序。熟悉的 xigemaAS 企业服务质量（例如安全性和事务完整性）已按需要启用。

2.7.1 在 xigemaAS 上开发 EJB 应用程序

Enterprise JavaBeans™ (EJB) 应用程序是一组 bean，它们打包为 Java™ 归档文件 (jar)、Web 应用程序归档文件 (war) 或 Java™ EE 企业应用程序归档文件 (EAR) 的组合。

xigemaAS 提供对 Enterprise JavaBeans 规范的全面支持。EJB 规范还定义了精简子集，它也是 xigemaAS 中提供的可配置功能部件，另外还有一些其他可选功能部件启用仅支持完整 EJB 规范的子集的服务器配置。这些子集允许使用一些服务器配置，已安装应用程序未使用 EJB 规范的所有功能时，这些配置使用的资源较少。

Enterprise JavaBeans 核心功能部件

ejbLite

此功能部件启用 EJB 规范中定义的 EJB 精简子集。此子集包含对写至 EJB 3.x API 的本地会话 bean 的支持。从 `ejbLite-3.2` 功能部件开始，此子集还包括对非持久性 EJB 计时器和异步本地接口方法的支持。

除了在 server.xml 文件中启用 `ejbLite-3.2` 功能部件之外，此功能部件不需要任何其他配置。

mdb

此功能部件启用 EJB 技术的消息驱动的 Bean 子集，它类似于 ejbLite 功能部件对会话 bean 启用的支持。此功能部件未启用会话 bean，所以，如果需要会话 bean 和消息驱动的 Bean，那么需要在 server.xml 文件中启用这两个功能部件。

以下功能部件为 EJB API 提供基本功能集合：

注：xigemaAS 概要文件不支持 EJB 2.x API 实体 bean。为实现持久性，必须改用 Java™ Persistence API (JPA)。

Enterprise JavaBeans™ 扩展功能部件

以下功能部件提供对核心 EJB 功能部件的扩展，它们组合到一起时提供对完整 EJB API 集合的支持：

ejbHome

启用 EJB 2.x API 的支持，具体地说，启用对 javax.ejb.EJBLocalHome 接口的支持。与 ejbRemote 功能部件组合到一起时，javax.ejb.EJBHome 接口也是受支持的。因为没有 ejbLite 功能部件时此功能部件没什么用处，所以启用此功能部件会自动启用对应 ejbLite 功能部件。

除了在 server.xml 文件中启用此功能部件之外，没有任何其他配置对此功能部件可用。

ejbPersistentTimer

启用对持久性 EJB 计时器的支持。启用此功能部件会自动启用对应 ejbLite 功能部件。

除了在 server.xml 文件中启用此功能部件外，还必须配置数据源以便为计时器提供持久性存储。缺省情况下会使用 `DefaultDataSource`，因此只有该配置是必需的。

ejbRemote

启用对远程 EJB 接口的支持。启用此功能部件会自动启用对应 ejbLite 功能部件。

除了在 server.xml 文件中启用此功能部件之外，此功能部件不需要任何其他配置。

Enterprise JavaBeans™ 便利功能部件

以下功能部件提供一种便利方法来启用完整 EJB API 集合：

ejb

对所配置功能部件的规范级别启用所有核心和扩展功能部件。例如，`ejb-3.1` 启用 `ejbLite-3.1` 和 `mdb-3.1`。除了因为启用所有核心和扩展 EJB 功能部件而提供的支持外，此功能部件未提供任何其他支持。

除了在 server.xml 文件中启用此功能部件之外，此功能部件不需要任何其他配置。

- 在 server.xml 文件中启用该功能部件。

  例如：

```
<featureManager>
     <feature>ejbLite-3.2</feature>
</featureManager>
```

开发企业 Bean (EJB) 持久性计时器应用程序

EJB 持久性计时器由容器自动创建或由应用程序通过程序创建。两种类型的持久性计时器都存储在 server.xml 文件中对 EJB 计时器服务配置的数据库内的表中。

创建持久性计时器后，它存在于数据库中直到发生下列其中一种情况：

- 未对此计时器安排更多到期时间；例如，在单一操作计时器到期和成功运行后。
- 应用程序取消了该计时器。

- 通过使用 EJB 计时器服务 MBean (`EJBPersistentTimerServiceMXBean`) 取消了该计时器
- 已从数据库中手动移除该计时器。例如，如果计时器服务数据库已清除或移除。

而且，应注意 EJB 持久性计时器已根据应用程序名称、模块名称和 bean 名称与 EJB 相关联。如果应用程序已更新，并且其中任何名称更改，那么计时器将继续在数据库中存在，但显示为不再存在（因为它不再到期和运行）。

Enterprise JavaBeans 规范声明，部署应用程序时由容器创建自动计时器。对于 xigemaAS，应用程序首次在服务器进程中启动时出现此情况。第一个应用程序启动期间，自动创建的持久性计时器存储在持久性计时器数据库中，并带有说明数据库已部署的指示。自动持久性计时器与说明应用程序已部署的指示从 EJB 持久性计时器数据库中移除之前，此容器不会再次创建自动持久性计时器。

1. 在 server.xml 文件中启用 EJB 持久性计时器功能部件；例如：

```
<featureManager>
     <feature>ejbPersistentTimer-3.2</feature>
</featureManager>
```

管理企业 bean (EJB) 持久性计时器

应用程序或模块可自动创建或通过程序创建 EJB 持久性计时器，它们将存储在数据库中。MXBean 是为查看数据库中存在的持久性计时器和移除与应用程序相关联的一些或所有持久性计时器而提供的。

管理 EJB 持久性计时器的最好方法是通过创建计时器的应用程序进行。创建计时器的应用程序能够提供说明创建计时器的原因的上下文以及计时器在到期时要执行的确切操作。但是，为进行应用程序开发和诊断，EJB 计时器服务提供能够执行基本计时器管理功能的 MXBean，即 `com.ibm.websphere.ejbcontainer.mbean.EJBPersistentTimerServiceMXBean`。

MXBean 提供用于查找与应用程序、模块或特定 bean 相关联的 EJB 持久性计时器的操作。可查看信息包含与计时器相关联的 bean 的名称及计时器到期时的安排。与计时器相关联的特定于应用程序的数据可能无法查看。系统还提供了用于取消与应用程序、模块或 bean 相关联的特定计时器或所有计时器的操作。

EJB API 未提供用于在创建计时器时提供唯一标识的机制，但是，计时器存储在数据库中时，一个唯一标识将与该计时器相关联并将在数据库中用作主键。无法通过程序获取计时器标识，但该标识包含在 MXBean 的“get”操作提供的可查看信息中。遇到涉及该计时器的错误时，此标识将显示在日志中。而且，此标识是用于通过 MXBean 操作“*cancelTimer*”取消特定计时器的值。

注意：对于自动性持久性计时器和程序性持久性计时器，所有 MXBean 操作执行起来是相同的。

1. 配置应用程序服务器以在 server.xml 文件中包含支持持久性计时器和 REST 连接器的 EJB 功能部件。

```
<featureManager>
     <feature>servlet-3.1</feature>
     <feature>ejbPersistentTimer-3.2</feature>
     <feature>jdbc-4.1</feature>
     <feature>restConnector-1.0</feature>
</featureManager>
```

2. 从 JVM bin 目录启动 **“JConsole”** 并连接至 xigemaAS 服务器进程。
3. 浏览至 EJB 计时器服务 MXBean。
4. 在第一个“*getTimers*”操作的 **`String`** 字段中输入应用程序名称并提交以接收有关应用程序的持久性计时器的信息。

管理自动创建的 EJB 持久性计时器

应用程序或模块在 XML 部署描述符中使用 `javax.ejb.Schedule` 注解或 schedule 元素来指示应用程序服务器自动创建 EJB 计时器。已提供 MBean 以管理自动创建的 EJB 持久性计时器的创建。

首次在服务器进程中启动应用程序时，xigemaAS 服务器为该应用程序创建 EJB 自动持久性计时器。计时器及说明已创建计时器的指示存储在与 `PersistentExecutor` 相关联的数据库中，PersistentExecutor 是针对 EJB 计时器服务配置的。对应用程序创建自动创建的持久性计时器后，在执行操作以从数据库中专门清除用于说明已创建这些计时器的指示之前，不会再次创建这些计时器。即使移除应用程序的所有自动创建的计时器，在说明已创建这些计时器的指示也被移除之前，xigemaAS 服务器不会创建其中任何计时器。

可通过手动清除计时器数据库或使用 EJB 计时器服务 MBean

1. 配置应用程序服务器以在 server.xml 文件中包含支持持久性计时器和 REST 连接器的 EJB 功能部件。

```
<featureManager>
     <feature>servlet-3.1</feature>
     <feature>ejbPersistentTimer-3.2</feature>
     <feature>jdbc-4.1</feature>
     <feature>restConnector-1.0</feature>
</featureManager>
```

2. 配置应用程序服务器以在 server.xml 文件中包含 EJB 计时器服务的数据源。

```
<dataSource id="DefaultDataSource" jdbcDriverRef="DerbyEmbedded">
     <properties.derby.embeddedcreateDatabase="create"
 databaseName="${server.config.dir}/data/EJBTimerDB"/>
</dataSource>  <jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib"/>
<library id="DerbyLib">
     <filename="${server.config.dir}/derby/derby.jar"/>
</library>
```

3. 获取 MBean 服务器连接。
4. 获取 EJB 持久性计时器服务 MBean。

```
import javax.management.MBeanServerConnection;
import avax.management.ObjectName;
import
 com.ibm.websphere.ejbcontainer.mbean.EJBPersistentTimerServiceMXBean;
...
MBeanServerConnection mbsc = <step #3>;
ObjectName on = new
ObjectName("WebSphere:feature=ejbPersistentTimer,type=EJBPersistentTimerService,name=
EJBPersistentTimerServiceMXBean timerServiceMBean =
     JMX.newMXBeanProxy(mbsc, on, EJBPersistentTimerServiceMXBean.class);
```

5. 移除为应用程序自动创建的持久性计时器。

```
timerServiceMBean.removeAutomaticTimers("<application name>");
```

MBean 上的 `removeAutomaticTimers()` 方法移除自动创建的计数器以及说明已创建这些计时器的指示。下次服务器启动该应用程序时，将创建自动创建的持久性计时器。

为持久性计时器配置 Enterprise JavaBeans™ 计时器服务

对于持久性 EJB 计时器，可配置 EJB 计时器服务以控制用于存储这些计时器的数据源、计时器的重试时间间隔以及针对超时回调方法调用失败的重试次数。

如果启用了 ejbPersistentTimer 功能部件，那么必须配置数据源以保存计时器，否则使用持久性计时器的任何尝试将失败。如果已配置 DefaultDataSource 并且系统未配置任何其他特定数据源以供 EJB 计时器服务使用，那么 EJB 计时器服务将使用 DefaultDataSource。

而且，EJB 计时器服务会以每 5 分钟一次的频率对持久性计时器重试超时回调方法的失败调用，直到超时回调方法成功完成。

使用以下选项来配置持久性计时器。

EJB 持久性计时器安排的执行程序

指定对控制持久性计时器行为的持久性执行程序的引用。系统提供了缺省实例 `defaultEJBPersistentTimerExecutor`，它将 `DefaultDataSource` 用作持久性存储。EJB 持久性计时器安排的执行程序引用可能更改为定制配置，或者缺省实例可能被覆盖以更改特定值。以下选项对持久性执行程序可用：

重试限制

指定可重试失败超时回调方法的最大次数。如果超时回调方法在重试时成功，那么服务器将停止尝试运行该方法。如果重试失败，那么服务器将继续尝试，直到超时回调方法成功或达到重试限制。在达到重试限制后，服务器不会尝试运行超时回调方法，即使先前尝试未成功也是如此。缺省值 -1 表示重试次数无限制。值 0 指示不重试，它不符合 EJB 规范。值 1 或更高的值指示允许进行特定次数的重试。

对于不需要非持久性计时器在每个安排时间完成的应用程序，更改针对非持久性计时器重试超时回调方法的次数的配置很有用。例如，如果应用程序创建的非持久性计时器安排为每 5 分钟运行一次，那么将重试次数配置为 0 会导致该计时器每 5 分钟运行一次，而不理会它的运行是否成功。

重试时间间隔

指定失败超时回调方法的重试时间间隔。第一次重试始终立即进行，无论对此值配置的时间间隔如何都是如此。所有其他重试等待针对此值指定的时间间隔。值 0 表示立即进行所有重试。值 1 或更高的值指示重试必须等待该特定秒数。缺省值为 300 秒。

对于超时回调方法在安排时间前完成很重要的应用程序，对非持久性计时器配置不同重试时间间隔很重要。此方法也适用于可接受将计时器完成延迟至稍晚时间（例如，稍晚重试时间间隔）以启用超时回调方法的应用程序，以使应用程序有更好的机会成功运行。

持久性任务存储

指定要使用的数据源。系统提供了缺省实例 defaultDatabaseStore，它将 DefaultDataSource 用作持久性存储。

1. 配置应用程序服务器以在 server.xml 文件中包含支持持久性计时器的 EJB 功能部件。

```
<featureManager>
        <feature>ejbPersistentTimer-3.2</feature>
</featureManager>
```

2. 在 server.xml 文件中配置 EJB 计时器服务，以对持久性计时器使用特定最大重试次数值和重试时间间隔值。

例如，使用以下配置指定持久性计时器最多重试 3 次，重试间隔时间为 10 秒：

```
<persistentExecutor id="defaultEJBPersistentTimerExecutor"
 retryInterval="10s"
          retryLimit="3"/>
```

对于此配置，最多可对计时器调用 4 次超时回调方法。第一次调用在安排时间进行。如果第一次调用失败，那么第一次重试在失败后立即进行。如果超时回调方法仍然失败，那么第二次和第三次重试分别在 10 秒后和 20 秒后进行。

在以下示例中，失败超时回调方法仅重试一次。回调时间间隔不重要，因为第一次重试始终立即启动。

```
<persistentExecutor id="defaultEJBPersistentTimerExecutor" retryLimit="1"/
>
```

3. 在 server.xml 文件中配置 EJB 计时器服务，以对持久性计时器使用特定数据源。

   例如，使用以下配置指定必须存储持久性计时器（通过使用 jdbc/timerDataSource）。

```
<databaseStore id="EJBTimerDatabaseStore" tablePrefix="EJBTimer_"/>
<persistentExecutor id="defaultEJBPersistentTimerExecutor"
          taskStoreRef="EJBTimerDatabaseStore"/>
```

持久性 Enterprise JavaBeans 计时器的配置选项

持久性 EJB 计时器不需要启用 ejbPersistentTimer 功能部件和设置缺省数据源（它具有 DefaultDataSource 标识以指向数据库）以外的任何配置。可选配置设置可用于控制行为，例如，选择另一数据源、控制轮询数据库以查找持久性计时器任务的时间和频率，以及是否重试失败或回滚计时器任务及重试频率。

EJB 计时器配置是由可选 timerService 配置元素指定的。持久性 EJB 计时器的配置属性进一步组合到 persistentExecutor 配置下。缺省情况下，EJB 计时器服务使用名为 defaultEJBPersistentTimerExecutor 的持久性执行程序实例。可通过将计时器服务配置为使用另一持久性执行程序实例来定制 EJB 持久性计时器配置。但是，定制 EJB 持久性计时器配置的最佳范例是覆盖 defaultEJBPersistentTimerExecutor 实例，以便您从 defaultEJBPersistentTimerExecutor 实例继承缺省值。

例如，仅覆盖重试限制：

```
 <persistentExecutor id="defaultEJBPersistentTimerExecutor" retryLimit="50"/
>
```

对持久性计时器任务定制数据库存储

持久性 EJB 计时器任务保存至数据库。与该数据库相关的配置组合到 databaseStore 配置元素下。除非另行配置，否则名为 defaultDatabaseStore 的 databaseStore 实例用于持久性 EJB 计时器，并供需要数据库的其他产品功能部件使用。覆盖 defaultDatabaseStore 实例的示例：

```
 <databaseStore id="defaultDatabaseStore" dataSourceRef="DB2DataSource" ta
blePrefix="MYTIMERS"/>
```

将 defaultEJBPersistentTimerExecutor 实例配置为使用另一 databaseStore 实例的示例：

```
  <persistentExecutor id="defaultEJBPersistentTimerExecutor" taskStoreRef=
"MyDBStore"/>
```

```
  <databaseStore id="MyDBStore" dataSourceRef="DB2DataSource" tablePrefix=
"MYTIMERS">
    <authData user="user1" password="password1"/>
  </databaseStore>
```

启用和禁用持久性计时器任务执行

持久性 EJB 计时器任务被允许在安排它们的事务落实时运行。为阻止持久性计时器任务运行，可将 enableTaskExecution 属性配置为值 `false`，在此情况下，EJB 计时器服务仍将持久性计时器任务写至数据库，但不运行这些任务。如果值切换为 `true`，那么 EJB 计时器服务将开始运行先前安排的计时器及所有已安排的新计时器。

为计时器任务定制持久性存储的轮询

系统将在启动时执行一次持久性存储的初始轮询，以查找任何先前安排的计时器任务。安排任务时，有关所安排的下一次运行时间的信息将存储并保留在内存中，从而不必进一步轮询持久性存储。此行为在许多情况下很理想，但可能并非始终令人满意。

- 如果计时器任务与其他服务存在外部依赖关系，那么初始轮询可能发生得太快，计时器任务可能因为它们需要的外部服务不可用而失败。如果发生此情况，那么可使用初始轮询延迟以将初始轮询延迟固定时间量。
- 如果安排了大量 EJB 持久性计时器，那么它们可能消耗过多内存。在此情况下，可配置轮询时间间隔来定期轮询持久性存储以仅查找您要在该时间间隔内运行的计时器任务，直到下一次轮询。轮询大小进一步限制每个轮询时间间隔中可从数据库读取的计时器任务数，这可能导致某些任务延迟运行。

示例配置：

```
<persistentExecutor id="defaultEJBPersistentTimerExecutor" initialPollDela
y="5m" pollInterval="10m" pollSize="200"/>
```

针对失败的和回滚的持久性计时器任务的重试

持久性 EJB 计时器执行失败或标记为回滚时，系统会立即重试一次这些执行。如果立即重试失败，那么系统会按固定时间间隔重试直到成功。可通过在配置中指定重试限制来限制重试次数。还可通过在配置中指定重试时间间隔来控制重试之间的时间间隔。

示例配置：

```
<persistentExecutor id="defaultEJBPersistentTimerExecutor" retryLimit="100
" retryInterval="2m"/>
```

将 Enterprise JavaBeans 与 xigemaAS 上的远程接口配合使用

如果 EJB 由另一 Java 虚拟机 (JVM) 或同一 JVM 内的另一应用程序托管，那么您可通过远程接口远程访问 Enterprise Java™ Bean (EJB) 方法。xigemaAS 使用 RMI-IIOP 技术实现远程 EJB 接口。可使用 `ejbRemote-3.2` 功能部件启用远程 EJB 支持。

要配置 xigemaAS 服务器以运行已启用远程 EJB 支持的应用程序，必须设置 `ejbRemote-3.2` 功能部件。

使用远程 EJB 接口时，查看以下注意事项：

命名

使用 **`java:` namespace**

EJB 规范要求远程接口绑定至 java: 命名空间；例如：

```
java:global/ExampleApp/ExampleModule/ExampleBean!com.ibm.example.ExampleRe
moteInterface
java:app/ExampleModule/ExampleBean!com.ibm.example.ExampleRemoteInterface

java:module/ExampleBean!com.ibm.example.ExampleRemoteInterface
```

EJB 组件未绑定至缺省 Java 命名和目录接口 (JNDI) 命名空间，所以 ibm-*-bnd.xml 文件中的 @EJB 查找和绑定不能使用此命名空间。对于同一服务器内的 EJB 组件，这些查找必须使用 java: 名称，对于另一服务器中的 EJB 组件，这些查找必须使用 corbaname: URL。

使用 **corbaname: URL**

这些接口还会绑定至 java:global 命名空间中所使用接口的类似上下文中的 ORB CosNaming 名称服务。通过使用 corbaname: URL，可借助 JNDI 访问这些接口；例如：

```
corbaname::test.vsettan.com.cn:2809#ejb/global/ExampleApp/ExampleModule/Ex
ampleHomeBean!com.ibm.example.ExampleEJBHomecorbaname:rir:#ejb/global/Exam
pleApp/ExampleModule/ExampleHomeBean!com.ibm.example.ExampleEJBHome
```

在服务器上，URL 的 rir: 形式使用本地名称服务。在客户机上，它使用缺省的或已配置的远程名称服务。

对 **corbaname: URL** 进行转义

根据对象管理组 (OMG) 命名服务规范，corbaname: URL 中的一些字符必须转义。xigemaAS 尝试确定从 java:global 命名空间派生的 corbaname: URL 是否需要转义，然后自动进行转义。不能在所有情况下进行转义。例如，如果名称包含单个点 (.) 并且没有无效字符，那么它不能自动转义。要强制以特定方式解释名称，需要按 *OMG* 命名服务规范中所述手动对 URL 转义。

对于企业 bean，考虑以下 java:global 名称：

```
java:global/TestApp/TestModule/TestBean!test.TestRemoteInterface
```

corbaname: URL 的简单形式不能自动转义，因为它表示不同但有效的位置。因此，以下 URL 不会按预期工作：

```
corbaname:rir:#ejb/global/TestApp/TestModule/TestBean!test.TestRemoteInter
face
```

反而必须按以下方式对此 URL 进行手动转义：

```
corbaname:rir:#ejb/global/TestApp/TestModule/TestBean!test%5c.TestRemoteIn
terface
```

OMG 命名服务规范中完整描述了用于对 corbaname: URL 进行转义的语法。

通过程序使用 **JNDI** 名称

可通过程序从 InitialContext 查找这些示例中的所有 URL 和 JNDI 名称。查找 java: 名称时，产生的对象可直接转型为预期类型；例如：

```
Object found = new InitialContext().lookup("java:global/ExampleApp/Example
Module/ExampleBean!com.ibm.example.ExampleRemoteInterface");
```

```
ExampleRemoteInterface bean = (ExampleRemoteInterface) found;
```

但是，使用 corbaname: URL 检索对象时，必须使用 RMI 样式的强制类型转换（称为收缩）；例如：

```
Object found = new InitialContext().lookup("corbaname:rir:#ejb/global/Exam
pleApp/ExampleModule/ExampleBean!com.ibm.example.ExampleRemoteInterface");
ExampleRemoteInterface bean = (ExampleRemoteInterface)PortableRemoteObject
.narrow(found, ExampleRemoteInterface.class);
```

互操作性

打包在带有 V2.0 部署描述符的 EJB JAR 模块中时，支持 IIOP 协议的任何产品可调用一些企业 bean，这些企业 bean 将 EJB 2.x 远程编程模型与 EJBHome 和 EJBObject 配合使用。WLP_INSTALL_DIR/clients/ejbRemotePortable.jar 文件必须包含在远程客户机的类路径上。此文件包含与 xigemaAS 服务器通信时所需的系统值类。从 xigemaAS 中远程访问 EJB 组件时，此文件不是必需的。在 xigemaAS 上使用 EJB 3 远程编程模型的 EJB 组件可供 xigemaAS 进程远程访问。xigemaAS 未提供用于从独立 Java 进程启动 EJB 组件的瘦客户机。xigemaAS 概要文件未对远程 EJB 组件提供工作负载管理或故障转移功能，包括您启动 EJB 组件时。

存根类

启动 xigemaAS 上的远程 EJB 时，客户机必须包括存根类。在某些情况下，如果客户机为 xigemaAS，那么该产品自动生成正确的存根类：

- 如果客户机应用程序启动包含在同一应用程序内的远程 EJB，那么 xigemaAS 概要文件自动生成存根类。
- 如果目标 EJB 在另一应用程序上运行，并且将 EJB 2.x 远程编程模型与 EJBHome 和 EJBObject 配合使用，那么客户机的类路径上必须包含存根类。如果 EJB 在 xigemaAS 上，那么必须使用 Java SDK 附带的 rmic 程序为目标 EJB 生成存根类，然后必须对客户机添加这些存根类。

事务传播

xigemaAS不支持出站或入站事务传播。此外，EJB 规范要求即使产品支持出站事务传播时，它仍然必须发送一个空事务上下文。此上下文必须被使用 Required（缺省）、Mandatory 或 Supports 事务属性的 EJB 组件拒绝。如果客户机或服务器在 xigemaAS 中，那么带有活动全局事务的客户机不能启动带有缺省事务属性的 EJB。如果 EJB 更改为使用 RequiresNew 或 NotSupported 事务属性，那么客户机可启动 EJB。但是，EJB 完成的事务性工作不会作为客户机事务的一部分落实。

异步方法

EJB 远程接口可以有类型为 Future 的返回值的异步方法。服务器会将 Future 对象返回至客户机，用于检索该值。建议不要使用远程异步方法，因为未声明结果的累积可能耗尽内存。为了缓解此问题，如果客户机在 24 小时内未检索结果或最大未声明结果数超过 1000，那么服务器结果将到期。可在 server.xml 文件中调整这些值；例如：

```
<ejbContainer>
     <async maxUnclaimedRemoteResults="10"unclaimedRemoteResultTimeout="10
m"/>
</ejbContainer>
```

1. 要启用此功能部件，请更新 server.xml 文件以添加 ejbRemote-3.2 功能部件；例如：

```
<featureManager>
```

```
      <feature>ejbRemote-3.2</feature>
</featureManager>
```

2. 为部署描述符 <ejb-ref> 中定义的或使用源代码注解 @EJB 定义的远程 EJB 引用配置应用程序绑定文件（ibm-*-bnd.xml 文件）。

   对于在注解上或部署描述符中提供查找名称的 EJB 引用，绑定不是必需的。在绑定文件中，可使用 EJB 的某个 java: 名称或使用某个 corbaname: 名称绑定 EJB 引用；例如：

   对于 EJB 引用：

```
@EJB(name="TestBean")
   TestRemoteInterface testBean;
```

   已定义绑定：

```
 <ejb-ref name="TestBean" binding-name="corbaname:rir:#ejb/global/TestApp/
TestModule/TestBean!test%5c.TestRemoteInterface"/>
```

3. 配置应用程序客户机以包含存根类。
4. （可选）通过在远程客户机的类路径上添加 WLP_INSTALL_DIR/clients/ejbRemotePortable.jar 文件，为支持 IIOP 协议的应用程序配置互操作性以使用 EJB 远程接口。

使用 Enterprise JavaBeans™ 应用程序，这些应用程序调用另一应用程序中的本地 EJB 组件

Enterprise JavaBeans™ (EJB) 规范要求打包在同一应用程序中的 EJB 组件支持本地客户机视图。这包括本地 home 接口、本地业务接口和非接口视图。未打包在同一应用程序内的 EJB 组件也支持本地客户机视图，但需要额外配置。

对于打包在另一应用程序中的 EJB 组件，该产品支持访问本地客户机视图，但有一些限制：

- 对于调用应用程序及目标 EJB 应用程序的类加载器，本地接口以及本地接口所使用的所有参数类型、返回类型和异常类型必须均为可见。可通过使用与服务器类加载器相关联的共享库或者使用均与两个应用程序关联的公共库引用来确保它们可见。
- 如果目标 EJB 应用程序已停止，那么必须通过重新启动调用应用程序来刷新对 EJB 的所有已缓存引用。最简单的解决方案是，每当重新启动执行调用的应用程序所依赖的目标 EJB 应用程序时，就重新启动该应用程序。
- 将该配置添加至 server.xml 文件；例如：

```
<library id="ejbInterfaceLib">
        <file name="${server.config.dir}/lib/ejbInterfaceLib.jar"/>
    </library>
    <webApplication id="ejbClient" location="ejbClient.war">
        <classloader commonLibraryRef="ejbInterfaceLib"/>
    </webApplication>
    <ejbApplication id="ejbApp" location="ejbApp.ear">
        <classloader commonLibraryRef="ejbInterfaceLib"/>
    </ejbApplication>
```

2.7.2 在 xigemaAS 上开发 SIP 应用程序

会话启动协议 (SIP) 应用程序是打包在 SIP 应用程序归档文件 (SAR) 中的一组 SIP Servlet。

SIP Servlet 是应用程序组件，由执行 SIP 信号发送的 SIP 容器进行管理。编程和部署模块类似于 Web Servlet，并因此映射至 xigemaAS 管理模块。 可在 SAR 文件中添加 Web Servlet（与所需 web.xml 部署描述符）来创建称为聚合应用程序的对象。有关 SIP 应用程序、servlet、聚合应用程序和状态代码的详细信息，请参阅*JSR 116*和*JSR 289*。

在 xigemaAS 上开发支持 PRACK 的 SIP 应用程序

对 INVITE 请求的 SIP 响应可能是最终的或临时的。最终响应始终可靠地发送，而临时响应通常并非这样。对于需要可靠地发送临时响应的情况，可以使用 PRACK（临时响应应答）方法。

要开发支持 PRACK 的应用程序，必须符合下列条件：

- 为指示客户机支持 PRACK，发送 INVITE 请求的客户机必须将 100rel 标记放置在 Supported 或 Require 头中。
- SIP servlet 必须通过调用 sendReliably() 方法而不是 send() 方法进行响应。

PRACK 按下列标准进行了描述：

- *RFC 3262*（“Reliability of Provisional Responses in the Session Initiation Protocol (SIP)”），它通过添加 PRACK 和选项标记 100rel 扩展了 *RFC 3261*（“SIP: Session Initiation Protocol”）。
- *JSR 289*（“SIP Servlet Specification V1.1”）的第 5.7.1 节（“Reliable Provisional Responses”）。
- 如果您要开发充当代理的应用程序，请让应用程序生成并发送一个可靠的临时响应给在“至”字段中任何没有标记的 INVITE 请求。
- 如果要开发充当用户代理客户机 (UAC) 的应用程序，请执行以下操作：
  - 使应用程序将 100rel 标记添加至出局 INVITE 请求。此选项标记必须出现在 Supported 头或 Require 头中。
  - 要创建 PRACK 请求并将该请求发送至用户代理服务器 (UAS)，请在应用程序的 doProvisionalResponse(...) 方法中对入局响应调用 createPrack() 方法。该容器按 JSR 289 第 5.7.1 节中的定义处理 RAck 头。
  - 充当 UAC 的应用程序接收针对 PRACK 的最终 2*xx*-6*xx* 响应，UAC 可使用 doResponse() 方法处理该响应。
- 如果要开发充当用户代理服务器 (UAS) 的应用程序，请执行以下操作：
  - 👉 注：如果入局的 INVITE 请求需要 100rel 标记，那么通过使用 send() 方法尝试不可靠地发送 101-199 响应会导致抛出异常。
  - 在应用程序中，声明 SipErrorListener 对象以在可靠的临时响应没有在 64*T1 秒内应答（其中 T1 是 SIP 计时器）时接收 noPrackReceived() 事件。在 noPrackReceived() 事件处理中，该应用程序应按照 JSR 116 第 6.7.1 节为关联 INVITE 请求生成并发送 5*xx* 错误响应。
  - 该应用程序可具有最多一个未完成的且未应答的可靠临时响应。第一个响应尚未应答就尝试再发送一个响应会导致异常。
  - 确保该应用程序强制对包含会话描述的 PRACK 请求实施 RFC 3262 提供/应答语义。具体来说，如果任何未应答的临时响应包含会话描述，那么 servlet 不得发送 2*xx* 最终响应。

开发背靠背 (B2B) 应用程序

可使用 `javax.servlet.sip.B2buaHelper` 接口以开发背靠背用户代理 (B2BUA) 应用程序。

阅读 *JSR289 规范*的第 12 节，它提供有关 B2BUA 操作的详细信息。SIP Servlet 1.1 规范定义帮助程序类 javax.servlet.sip.B2buaHelper 以创建 B2BUA 应用程序。有关更多信息，请参阅有关 *B2buaHelper 方法*的文档。

B2BUA 是一个会话启动协议 (SIP) 应用程序，它包含在 SIP 调用中，用于转发两个或更多 SIP 对话之间的请求和响应。典型 B2BUA 应用程序管理两个用户代理之间的 SIP 会话。但是，B2BUA 应用程序还可处理更复杂的场景，在这些场景中，它需要管理来自下游派生的不同分支的响应。请求拆分为不同分支及消息从用户代理客户机转发至用户代理服务器时，发生下游派生。`B2buaHelper` 类包含用于创建这类 B2BUA 应用程序的所有必需方法。`B2buaHelper` 类的行为及用于创建新请求的规则是在 JSR289 规范第 12.2 节中定义的。可使用 `B2buaHelper` 类函数来链接 SIP 背靠背用户代理呼叫的两个分支：入局呼叫分支和出局呼叫分支。

1. 通过调用 `SipServletRequest.getB2buaHelper()` 方法来检索 `B2buaHelper` 实例。
2. 检索此实例后，可通过调用 `B2buaHelper.createRequest()` 来创建新的 SIP 请求。

```
private void doInvite(SipServletRequest req) {
 B2buaHelper b2buaHelper = req.getB2buaHelper();
 SipServletRequest newRequest = b2buaHelper.createRequest(req, true,
 headerMap);
}
```

   所创建的 `newRequest` 与在此方法的第一个参数中作为原始请求提供的 `SipServletRequest req` 完全相同。Boolean 参数指示是否已链接原始请求 `req` 和 `newRequest`。`headerMap` 提供要在 `newRequest` 中覆盖的非系统头的映射。
3. 确保已链接两个分支。如果调用 `createRequest` 方法时指定了 `false` 属性，那么可通过调用以下项以显式链接 SIP 会话：

```
B2buaHelper.linkSipSessions(session1, session2)
```

4. 链接 SIP 会话和 SIP 请求后，可通过调用以下项来检索所链接会话：

```
SipSession linkedSession = B2buaHelper.getLinkedSession(req.getSession());
```

   
   注：在 B2BUA 派生场景中，该应用程序创建多个 SIP 请求，所以不会链接所有 SIP 会话。必须显式链接这些会话。
5. 从用户代理服务器 (UAS) 接收响应后，创建针对其分支的响应。进行检查以确定响应上是否存在所链接会话，如果存在，请使用所链接请求创建针对用户代理客户机 (UAC) 的响应。

```
SipServletRequest linked =
 b2bHelper.getLinkedSipServletRequest(resp.getRequest());
linked.createResponse(resp.getStatus(),resp.getReasonPhrase()).send();
```

   如果没有与响应相关联的所链接会话，那么该应用程序需要调用 `B2buaHelper.createResponseToOriginalRequest` 并创建新的派生会话，此会话是与原始请求相关联的 `SipSession` 的副本。

   
   注：JSR289 规范第 12.5 节提供一种方法以创建针对原始请求的多个成功响应。如果有多个 `SipSession` 与原始 `INVITE` 消息相关联，并且需要将多个成功响应发送至 INVITE 消息，请使用 `B2buaHelper.createResponseToOriginalRequest` 方法。

6. 如果使用 `createResponseToOriginalRequest` 方法，请确保已链接两个分支。应用程序负责将新的派生会话链接至其配对会话，以便它可检索所链接会话以发送与其相关的后续响应。

在 xigemaAS 上的会话启动协议 (SIP) 应用程序中执行 DNS 查找

在应用程序中使用域解析器 API，以通过 RFC 3263 协议执行 SIP URI 的域名系统 (DNS) 查找。可执行同步查找以避免必须处于保留状态，异步回调时需要此保留状态。或者，如果需要提高接口性能，那么您可执行异步查找。

在 server.xml 文件中，安装并配置带 `domainResolver` 元素的 `sipServlet-1.1` 功能部件。有关更多信息，请参阅在 *xigemaAS* 上管理会话启动协议 *(SIP)*（见第 1225 页）。

1. 在应用程序中访问域解析器 API。
   - 通过将 `com.ibm.websphere.sip.resolver` 用作键以从 ServletContext 方法中获取属性。

     ```
     getServletContext().getAttribute("com.ibm.websphere.sip.resolver ")
     ```

   - 使用资源注入。

     ```
     @resource
     DomainResolver resolver
     ```

2. 执行 URI 查找。
   - 要同步使用 API，请调用 locate(SIPURI) 方法，此方法返回 URI 解析请求响应的结果。

     ```
     DomainResolver
     locate(SIPURI)
     ```

   - 要异步使用 API，请调用 locate(SIPURI, Listener) 方法，此方法在完成后向侦听器发出信号。缓存结果后，将在同一调用者线程上触发该侦听器。

     ```
     DomainResolver
     locate(SIPURI, Listener)
     ```

在 xigemaAS 上接收会话启动协议 (SIP) 中的不匹配消息

使用 UnmatchedMessageListener API 接收 SIP 容器无法处理的入局会话启动协议 (SIP) 请求或响应消息。

在 server.xml 文件中，安装并配置 `sipServlet-1.1` 功能部件。有关更多信息，请参阅在 *xigemaAS* 上管理会话启动协议 *(SIP)*（见第 1225 页）。

SIP 容器提供接口以允许应用程序接收无法与任何现有对话匹配的所有入局请求或响应消息。与现有对话不匹配的所有请求或响应消息称为不匹配消息。

不匹配请求具有 `To` 和 `From` 标记的请求，但在 SIP 容器中找不到相关对话，因为从未创建此对话或此对话处于失效状态。如果 SIP 容器接收到入局不匹配请求，那么该容器的响应为 `481 Call/Transaction Does Not Exist`，然后将此入局请求转发至侦听器类。入局请求将作为 UnmatchedRequestEvent 事件的一部分进行转发，此事件是在 com.ibm.websphere.sip.unmatchedMessages.events 包中定义的。此事件包含接口以获取入局不匹配请求及与接收此事件的应用程序相关的 ServletContext。此应用程序可使用 ServletContext 以创建新的 SIP 活动。侦听器无法为所接收不匹配请求创建响应或代理此请求。如果发生此情况，那么 SIP 容器中将出现 IllegalStateException 异常。

不匹配响应是在 SIP 容器中接收的，但与任何出局请求不匹配。SIP 容器废弃不匹配响应前，不匹配响应将作为 UnmatchedResponseEvent 事件的一部分发送至 UnmatchedMessageListener 侦听器。与不匹配请求一样，此应用程序访问不匹配响应及相关应用程序 ServletContext。

如果应用程序有多个已定义 UnmatchedMessageListener 侦听器，那么每个侦听器以相互独立的方式启动。如果单个应用程序服务器上有多个应用程序并且多个应用程序具有 UnmatchedMessageListener 侦听器，那么所有侦听器以随机顺序启动。如果一个应用程序向同一服务器上的另一应用程序发送请求（作为应用程序组合的一部分），那么 SIP 容器确定此请求是其无法处理的请求，然后该服务器上的所有 UnmatchedMessageListener 侦听器将激活。

UnmatchedMessageListener API 是在 com.ibm.websphere.sip.unmatchedMessages 包中的 com.ibm.websphere.appserver.api.sipServlet 文件内定义的。

- 可在应用程序代码中使用下列其中一个方法来访问 UnmatchedMessageListener API：
- 将 UnmatchedMessageListener 实现类显式添加为 sip.xml 文件中的侦听器部署描述符元素。

```
<listener>
 <listener-class>com.example.MyTimerListener</listener-class>
</listener>
```

- 使用 @javax.servlet.sip.annotation.SipListener 注释。

```
@SipListener
public class MySipUnmatchedMessagesListener
   implements UnmatchedMessageListener {
....
}
```

xigemaAS 上的 SIP 专有头字段

您可以创建包括专有头字段的会话启动协议 (SIP) Servlet 请求。SIP 专有头字段可让您逐条消息实现特定 SIP 设置。在 SIP 容器级别所设的 SIP 设置适用于该 SIP 容器处理的所有 SIP 消息。

要在消息中包括一个或多个专有头字段，请对 SIP Servlet 请求进行设置，以便它包括一个或多个 SipServletMessage.setHeader(*string_name*, *string value*) 方法。当应用程序调用 SipServletRequest.send() 以发送请求时，传递至 SIP 堆栈以进行传输的消息对象包括专有头信息。然后，SIP 堆栈会创建客户机事务以发送请求，并根据消息对象中包括的任何专有头字段调整此特定请求的 SIP 配置设置。该堆栈会先移除专有头字段，然后再将消息送出到网络中。

用于指定计时器值的专有头字段

以下专有头字段可用于指定特定消息的计时器值。应用程序可以在一个消息实例中设置多个计时器值，但是不能为同一个专有头字段指定多个值。

IBM-TransactionTimeout

使用此头字段来指定客户机事务的超时时间长度（以毫秒为单位）。此头等价于为 INVITE 客户机事务中的计时器 B 和非 INVITE 客户机事务中的计时器 F 指定值。

IBM-RetransmissionInterval

使用此头字段来指定请求重新传输时间间隔的长度（以毫秒为单位）。此头等价于为 INVITE 客户机事务中的计时器 A 和非 INVITE 客户机事务中的计时器 E 指定值。

IBM-RetransmissionMaxInterval

使用此头字段来指定最大重新传输时间间隔（以毫秒为单位）。此头等价于为非 INVITE 客户机事务中的计时器 T2 和 INVITE 客户机事务中的计时器 B 指定值。

xigemaAS 上的 SIP SipServletRequest 和 SipServletResponse 类

SipServletRequest 和 SipServletResponse 类与您开发 Web 应用程序时使用的 HttpServletRequest 和 HttpServletResponse 类相似。

每个类都提供了访问 SIP 消息中的头并对其进行操作的功能。由于请求和响应的异步性质，SipServletRequest 类还将为请求创建新响应。扩展 doInvite 方法时，只会将 SipServletRequest 类传递给该方法。要将响应发送给客户机，必须对 Request 对象调用 createResponse 方法以创建响应，如以下示例中所示：

```
protected void doInvite(SipServletRequest req) throws
 javax.servlet.ServletException, java.io.IOException {

  //send back a provisional Trying response
  SipServletResponse resp = req.createResponse(100);
resp.send();
```

因为其异步性质，所以 SIP Servlet 可能看起来很复杂；但是，像以上代码样本一样简单的代码也能向客户机发送响应。

以下示例显示较复杂的 SIP Servlet。通过使用 SIP Servlet 中包含的以下方法，Servlet 将阻止所有来自 *example.com* 域之外的调用。

```
protected void doInvite(SipServletRequest req) throws
 javax.servlet.ServletException, java.io.IOException {

//check to make sure that the URI is a SIP URI
            if (req.getFrom().getURI().isSipURI()){
                    SipURI uri = (SipURI)req.getFrom.getURI();
                if (!uri.getHost().equals("example.com")) {
                   //send forbidden response for calls outside domain

 req.createResponse(SipServletResponse.SC_FORBIDDEN).send();
                   return;
                }
             }
             //proxy all other requests on to their original destination
             req.getProxy().proxyTo(req.getRequestURI);
        }
```

有关这些类的更多信息，请参阅 SIP Servlet 规范 1.1，即 *JSR 289*。

xigemaAS 上的 SIP SipSession 和 SipApplicationSession 类

SipSession 和 SipApplicationSession 类都可充当存储用于分布式环境或高可用性环境的应用程序中的数据的主要场所。

SipSession 类是两个实体间具体的点到点通信的最佳代表，与 HttpSession 对象最为接近。由于以前 HTTP Servlet 中并没有针对 HTTP 请求的代理或分叉，因此并不存在对超过单个点到点会话的需求。SIP 用户预计会遇到需要多层 SIP 会话管理的代理和分叉活动。SipSession 类是最低的点到点层。

SipApplicationSession 类表示更高层的 SIP 会话管理。一个 SipApplicationSession 类可以有一个或多个 SipSession 对象。但是，每个 SipSession 类只能与一个 SipSession 对象相关。SipApplicationSession 类还支持附加任意数量的其他协议会话。当前，只有 HTTP 会话受到所有实现的支持。SipApplicationSession 类有一个 getSessions 方法，该方法以所请求的协议类型作为参数。

在许多应用程序中，组合 HTTP 和 SIP 可能很有用。例如，您可以使用这种方法将 HTTP 和 SIP 会话绑定到一起，以通过富 HTTP 图形用户界面监视电话呼叫或启动电话呼叫。

有关这些类的更多信息，请参阅 SIP Servlet 规范 1.1，即 *JSR 289*。

2.7.3 在 xigemaAS 中开发 WebSocket 应用程序

可以配置 xigemaAS 来使用 WebSocket 协议以允许应用程序使用全双工连接进行通信。

要配置 xigemaAS 服务器以运行已启用 WebSocket 1.0 的应用程序，必须设置 `websocket-1.0` 功能部件（对于 WebSocket 1.0）或 `websocket-1.1` 功能部件（对于 WebSocket 1.1）。

注：

除了 *JSR 356* (Java API for WebSocket V1.1) 中定义的 WebSocket API 外，还向 xigemaAS 实现添加了一个 API，此 API 允许 servlet 或过滤器请求当前 `HTTP Request`“升级”以启动 WebSocket 会话。

Websocket 端点可使用模板以使端点与 URI 匹配。Web 应用程序也可使用不应映射至 WebSocket 端点的 URI（即使它们与 WebSocket 模板匹配）。将 WebSocket 端点映射至 URI 或允许 URI 被视为“非 WebSocket”`HTTP Request` 之间的区别是由 `HTTP Request` 中带有值“websocket”的“Upgrade”头是否存在造成的。

WebSocket

WebSocket 是一种标准协议，它允许 Web 浏览器或客户机应用程序使用一种全双工连接与 Web 服务器应用程序通信。

HTTP 不适用于两个应用程序之间的长时间实时全双工通信。在许多实例中，用户的 Web 服务器应用程序或 servlet 想要在长时间实时全双工会话中与客户机浏览器或应用程序通信。换言之，这两个应用程序想要自由地来回读写数据。此类型应用程序的一个示例是证券交易员 Web 浏览器上经常显示不断变化的货币汇率的应用程序。涉及实现此类型的通信的现有 HTTP 技术的当前解决方案繁琐且低效。用于浏览器与服务器之间的恒定双向通信的 HTTP 解决方案多以轮询和/或仅处理单向流量的两个开放式 HTTP 连接组成。

WebSocket 使用标准 HTTP 请求/响应序列以建立连接。建立连接后，WebSocket API 提供读写接口以通过所建立的连接（异步全双工方式）读写数据。WebSocket 还提供接口以从任一端异步关闭连接。

因为 WebSocket 使用标准 HTTP 请求/响应序列建立连接，所以此连接以与 HTTP 连接相同的方式连通防火墙和代理。WebSocket 需要全双工通信，包括在同一连接上同时读写。WebSocket 还可使用 SSL 以进行安全连接和数据传输。此协议按 HTTP 协议使用 SSL 的方式使用 SSL。

xigemaAS WebSocket 功能部件实现以下规范：

- *WebSocket 协议 - RFC 6455*
- *Java™ API for WebSocket - JSR 356*

xigemaAS 支持 WebSocket 1.0 和 WebSocket 1.1 规范。与 WebSocket 1.0 相比，WebSocket 1.1 支持以更健壮的方式指定消息处理程序。

2.8 在 xigemaAS 中部署应用程序

可在 xigemaAS 中部署 Web 应用程序、企业应用程序和 OSGi 应用程序。要部署应用程序，请将应用程序拖放到先前定义的 dropins 目录中，或者在服务器配置中添加一个应用程序条目。

可按本主题中所述部署应用程序。

此主题假定您尚未禁用运行时配置的动态更新，如控制动态更新（见第 1135 页）中所述。

缺省情况下，自动监视“dropins”目录。如果将应用程序拖放到此目录，那么会自动将该应用程序部署在服务器上。类似地，如果将该应用程序从此目录中删除，那么会自动将该应用程序从服务器中移除。“dropins”目录可用于不需要附加配置（例如安全角色映射）的应用程序。如果将应用程序放入“dropins”目录中，那么不得在服务器配置中包含该应用程序的条目。否则，服务器将尝试装入应用程序两次，并且可能发生错误。对于不在“dropins”目录中的应用程序，您在服务器配置中使用应用程序条目来指定位置。该位置可以位于文件系统，也可以位于 URL。

应用程序可打包为归档文件、目录或松散应用程序（文件位于多个位置）。有关松散应用程序的更多信息，请参阅松散应用程序（见第 1605 页）。

对于“dropins”目录中的应用程序，应用程序监视器使用文件名和文件扩展名来确定应用程序类型并生成应用程序标识和应用程序名称。例如，如果归档文件或目录名称为 snoop.war，那么应用程序监视器会假定应用程序是 Web 应用程序，该应用程序标识和名称为“snoop”。对于所配置应用程序，应用程序类型和名称已指定。

有关与目录相关联的缺省目录结构和属性（例如 `server.config.dir`）的更多信息，请参阅目录位置和属性（见第 1101 页）。

注：在生产环境中使用“dropins”目录时，有一些限制。请参阅不能对 *“dropins”* 目录中的应用程序进行版本控制（见第 1671 页）。

- 通过xigemaAS 管理中心部署应用程序。

  例如，使用缺省目录结构来部署应用程序，您将它拖放到 `${server.config.dir}/dropins` 目录（即，`wlp/usr/servers/`*`server_name`*`/dropins`）。

  可以使用下列任一方式部署应用程序：

  - 将带有识别后缀（.ear、.war 等）的归档文件直接放到 /dropins 目录中。例如，`${server.config.dir}/dropins/myApp.war`
  - 将归档文件解压到以应用程序名称和识别后缀命名的目录。例如，`${server.config.dir}/dropins/myApp.war/WEB-INF/...`
  - 将归档文件或解压的归档放入以识别后缀命名的子目录中。例如，`${server.config.dir}/dropins/war/myApp/WEB-INF/...`
- 通过将应用程序拖放至 apps 目录中，并在服务器配置文件（server.xml）添加相关配置描述来部署应用程序。

  在 server.xml 配置文件中配置 `application` 元素。必须为应用程序配置下列属性：

id

必须唯一，供服务器内部使用。

name

必须唯一，取决于应用程序。name 的值可用作应用程序的上下文根。有关如何为应用程序设置上下文根的更多信息，请参阅将 *Web* 应用程序部署到 *xigemaAS*（见第 1507 页）。

type

指定应用程序的类型。

- 对于 Web 应用程序，受支持的类型为 war。
- 对于企业应用程序，受支持的类型为 ear。

location

指定应用程序的位置。它可能是绝对路径，或者是可供您下载应用程序的 URL。它也可能是应用程序的文件名（包含可能具有的文件扩展名）。

如果应用程序位于文件系统上，那么位置可以是完整路径名或简单文件名。如果位置不含完整路径，那么应用程序管理器会在 `${server.config.dir}/apps` 和 `${shared.app.dir}` 中查找应用程序。如果应用程序位于 URL 上，那么应用程序管理器会将应用程序下载到服务器工作区内部的临时文件夹，然后启动应用程序。

注：对已配置应用程序指定的位置不应该在“dropins”目录中。如果将应用程序拖放到“dropins”目录，而且也在 server.xml 文件中指定该位置，那么您是在告知服务器部署应用程序两次。

在下列两个示例中，位置是文件系统。如果位置是 URL，请在位置字段中输入 URL。

```
<osgiApplication location="D:/apps/ImpactEBA.eba"/>
<webApplication location="ImpactWeb.war"/>
```

第二个示例不含完整路径。在这种情况下，必须将应用程序放入下列其中一个位置：

- `${server.config.dir}/apps`（即，*server_directory*/usr/servers/*server_name*/apps）
- ${shared.app.dir}（即，*xigemaas_install_location*/usr/shared/apps）

可使用下列任一方式将应用程序部署至文件系统：

- 将带有识别后缀（.ear、.war 等）的归档文件直接放到所选位置。例如，*application_directory_path*/myApp.war
- 将归档文件解压到所选位置的子目录（此子目录以应用程序名称和识别后缀命名）。例如，*application_directory_path*/myApp.war/WEB-INF/...

注：

- 必须创建服务器级 apps 目录，而共享 apps 目录在缺省情况下是存在的。有关与服务器目录相关联的属性的更多信息，请参阅目录位置和属性（见第 1101 页）。
- 可以在启动服务器之前或之后设置 application 元素。如果在启动服务器之后设置该元素，那么会动态获得更改。

- 在 xigemaAS 中部署上下文和依赖性注入 (CDI) 应用程序

  通过对 xigemaAS 服务器配置 CDI 1.2 xigemaAS 功能部件，可使用该服务器来部署 CDI 应用程序。请参阅配置 *xigemaAS* 以使用上下文和依赖性注入 *1.2*（见第 1222 页），以了解更多信息。

  使用上下文和依赖性注入的应用程序必须已启用 CDI。对于 CDI 1.2 xigemaAS 功能部件，符合以下条件时将启用 CDI：

- 已有 bean 发现方式为 `all` 的 beans.xml 文件。
- 没有 beans.xml 文件，或者已有一个空白 beans.xml 文件和类带有 bean 定义注解。在此情况下，必须有 bean 部署归档。

有关 CDI 1.2 功能部件识别的不同类型的 bean 部署归档的更多信息，请参阅上下文和依赖性注入 *1.2* 行为更改（见第 1223 页）。

- 移除应用程序。

  对于已包含在服务器配置中的应用程序，请从 server.xml 文件中移除对应用程序的引用。应用程序随后会自动从服务器中移除。

  对于已部署到“dropins”目录中的应用程序，请将应用程序从该目录中删除。应用程序随后会自动从服务器中移除。

  要卸载“dropins”目录中的所有应用程序，请将应用程序监视器 dropinsEnabled 属性设置为 `false`，如控制动态更新（见第 1135 页）中所述。

对于所有已部署的应用程序，您都可以配置是否启用应用程序监视，以及配置检查应用程序更新的频率。对于“dropins”目录，您也可以配置目录的名称和位置，以及选择是否要部署该目录中的应用程序。请参阅控制动态更新（见第 1135 页）。

2.8.1 从命令行打包 xigemaAS 服务器

可从命令行创建压缩文件，该文件包含 xigemaAS 运行时环境、共享资源目录中的文件、特定服务器以及服务器中嵌入的应用程序。您也可以选择将运行时二进制文件从压缩文件中排除。

xigemaAS 服务器属于轻量级，因此您很容易将服务器安装打包在压缩文件中。可以存储此包，将包分发给同事，使用包来将安装部署到不同位置或另一台机器，甚至将安装嵌入产品分发中。

注：通过将 UTF-8 编码用于条目名称来创建结果文件，因此用来打开该文件的工具必须能够将 UTF-8 编码用于条目名称。Java™ SDK 中的 `jar` 命令使用此格式。

1. 要从命令行打包 xigemaAS 服务器，请完成以下步骤：
1. 打开命令行，然后将目录切换至 `wlp/bin` 目录。
2. 停止服务器。
3. 运行 `package` 命令以创建软件包。

   可打包xigemaAS 服务器或运行时。

   - 打包 xigemaAS 服务器。

     缺省归档格式为 .zip（在所有平台上）。您还可生成 .jar 归档。

     如果未指定服务器名称，那么会使用 defaultServer。如果未指定 --archive 参数，那么将 *server_name* 的值用于 *package_file_name*，而且会在 *${server.output.dir}* 目录中创建压缩文件。

     为环境选择正确命令。

     - 使用此命令生成 .zip 归档。

       ```
       server package server_name --archive=package_file_name.zip --include=all
       ```

       其中 *package_file_name*.zip 是您选择的文件名。此文件名可以包含完整路径名。如果省略完整路径，那么会在 *${server.output.dir}* 目录中创建一个称为 *package_file_name*.zip 的压缩文件。

- 使用此命令生成 .jar 归档。.jar 归档的优势在于：bin 目录中的脚本保持其许可权，以便在安装软件包时可执行这些脚本。

  ```
  server package server_name --archive=package_file_name.jar --include=all
  ```

  其中，*package_file_name*.jar 是所选的文件名。

  有关针对此归档文件的解压选项的更多信息，请参阅*JAR* 文件解压选项（见第 1089 页）。

您也可以将 --include 选项与此命令配合使用。例如，--include=all 选项会对 ${WLP_USER_DIR} 目录中的运行时二进制文件及相关文件进行打包；--include=usr 选项仅对 ${WLP_USER_DIR} 目录中的相关文件进行打包，从而有效地将运行时二进制文件从压缩文件中排除。

--include=usr 选项不适用于 .jar 归档格式。

如果您使用 --include=minify 选项，那么 server 命令将仅打包在运行服务器时所需要的运行时环境的那些部件以及 ${WLP_USER_DIR} 目录中的文件。此选项将显著减小最终获得归档的大小。

minify 操作所保留的运行时环境的部件取决于您正在打包的服务器中配置的功能部件。只会保留在运行该服务器时所需要的那些功能部件，其余功能部件都将移除。因此，随后您将无法启用已移除的功能部件。例如，如果仅保留了 servlet-3.0 功能部件，那么您随后将无法启用 jpa-2.0 功能部件。

如果配置已更改，那么您可以重复执行 minify 操作以进一步减小此归档的大小。但是，minify 操作不存在逆向操作，因此，如果您稍后需要已移除的一个或多个功能部件，那么必须从已完成的 xigemaAS 服务器重新开始。

在 minify 操作正在运行时，服务器暂时已启动，并且您将看到相关联的消息。正因为如此，对于无法启动的服务器，您将无法使用 --include=minify 选项，但是可以使用 --include=all 或 --include=usr 选项来将此服务器打包。

可以通过将 --os 选项与 --include=minify 选项配合使用来指定您希望所打包的服务器支持的操作系统。

要打包仅支持 Linux™ 的服务器，请使用下列命令：

```
server package --archive="linux.zip" --include=minify --os=Linux
```

- 打包 xigemaAS 运行时。

  创建包含 wlp 目录但不包含 usr 目录的运行时归档。服务器软件包的命名约定为 *package_name*.zip；例如，CustomerPortalApp.zip。要创建运行时归档，运行不带服务器名称但带 --include=wlp 选项的 package 命令：

  ```
  server package --include=wlp
  ```

  要指定软件包文件名和目标位置，请添加 --archive=*package_path_name* 选项；例如：

  ```
  server package --include=wlp --archive=c:\temp\myPackage.zip
  ```

  如果未使用 --archive 选项指定有效软件包名称或目标位置，那么该命令将在 *$WLP_OUTPUT_DIR* 位置（缺省情况下为 *${wlp.install.dir}*/usr/servers 目录）创建 wlp.zip 运行时归档。运行该命令前，目标位置必须存在。因此，如果目标位置为 c:\temp，那么 C:\temp 目录必须存在并且必须具有写许可权，该命令才能将归档写至 C:\temp 目录。

2.8.2 从服务器配置文件将 JNDI 绑定用于常量

要从服务器配置文件将常量绑定到缺省 Java™ 命名和目录接口 (JNDI) 命名空间，您可以使用 xigemaAS 上的 `<jndiEntry>` 元素。

可以使用 xigemaAS 中的缺省 JNDI 命名空间，提供到应用程序所需的其他对象的绑定。服务器配置文件中声明的任何数据源都可以在缺省 JNDI 命名空间中使用。此外，还可以将配置文件中的 Java™ 字符串和基本数据类型绑定到 JNDI 命名空间。随后在运行时应用程序可以使用这些常量，这提供了一种简单的可移植方式来将配置值传递到应用程序。

1. 通过在 xigemaAS 服务器的 server.xml 文件中指定 xigemaAS 功能部件 `jndi-1.0` 来将常量添加到缺省 JNDI 命名空间中。

```
<featureManager>
   <feature>jndi-1.0</feature>
</featureManager>
```

2. 通过在 server.xml 文件中指定具有 jndiName 和 value 属性的 `<jndiEntry>` 元素来将常量绑定到 JNDI 命名空间。

```
<jndiEntry jndiName="schoolOfAthens/defaultAdminUserName" value='"plato"' />
<jndiEntry jndiName="schoolOfAthens/defaultAdminPassword" value='"republic"' />
```

   如果要将 `java.net.URL` 实例绑定至 JNDI 命名空间，请使用 `jndiURLEntry` 配置：

```
<jndiURLEntry jndiName="urls/VsettanxigemaAS" value="http://www.vsettan.com/
support/xigemaas/" />
```

3. 使用 JNDI 上下文并利用以下代码从应用程序中查找常量：

```
  Object jndiConstant = new InitialContext().lookup("schoolOfAthens/defaultAdminUserName");
  String defaultAdmin = (String) jndiConstant;
```

   注：

   - `lookup()` 方法会将对象返回给应用程序。通过将 `jndiEntry` 元素中存储的值解释为 Java™ 文字串或基本数据类型来确定对象的类型。如果解析失败，那么将精确值提供为未修改的字符串。
   - `jndiEntry` 元素支持整数、浮点、布尔、字符和字符串字面值，如 *Java™ 语言规范 Java™ SE 7 Edition 3.10 小节*中所述。字符串和字符字面值可能包含 Unicode 转义序列（请参阅 *section 3.3: Unicode Escaped Sequences*) 以及八进制和字符转义序列（请参阅 *section 3.10.6: Escape Sequences for Character and String Literals*)。不支持空的字面值和类字面值；有关更多信息，请参阅 *section 3.10.7: The null literal* 和 *section 15.8.2: Class Literals* 。

   请参阅以下 Java™ 字面值示例：

   - 后跟换行符的字符串 `"Hello, world"`：

```
<jndiEntry jndiName="a" value='"Hello, world.\n"' />
```

   - 包含二进制值 `1010101` 的整数：

```
  <jndiEntry jndiName="b" value="0b1010101" />
```

- 单个字符 'X'：

```
<jndiEntry jndiName="c" value="'X'" />
```

- 双精度浮点数 1.0：

```
<jndiEntry jndiName="d" value="1.0D" />
```

2.8.3 对服务器配置文件中的动态值使用 JNDI 绑定

通过在 xigemaAS 概要文件上使用 jndiReferenceEntry 元素，可将对服务器配置文件中的动态值的引用绑定至缺省 Java™ 命名和目录接口 (JNDI) 命名空间。

xigemaAS 中提供了缺省 JNDI 命名空间，此命名空间用于提供对应用程序所需的其他对象的绑定。根据服务器中启用的功能部件，可将一组预先确定的对象绑定至缺省 JNDI 命名空间。此外，还可绑定对对象工厂的引用，这会动态确定它返回的值。还可使用此对象工厂将定制对象类型返回至应用程序。

1. 将 jndi-1.0 xigemaAS 功能部件添加至 server.xml 文件。

```
<featureManager>
   <feature>jndi-1.0</feature>
</featureManager>
```

2. 创建 ObjectFactory 类，此类返回通过程序定义的值。

```
import javax.naming.spi.ObjectFactory;

public class MyObjectFactory implements ObjectFactory {
  @Override
  public Object getObjectInstance(Object o, Name n, Context c, Hashtable<?, ?> envmt) throws
 Exception {
    Properties p = new Properties();
    p.put("abc", 123);
    return p;
  }
}
```

3. 在 server.xml 文件的 library 元素中包含 ObjectFactory。

```
<library id="objectFactoryLib">
   <fileset dir="${server.config.dir}/lib" includes="factory.jar"/>
</library>
```

4. 在 server.xml 文件的 jndiObjectFactory 元素中声明工厂并引用先前声明的库。

```
<jndiObjectFactory id="objectFactory" libraryRef="objectFactoryLib"

          className="com.ibm.example.factory.MyObjectFactory"/>
```

还可声明该工厂返回的对象类型。此类型由 javax.naming.Context.list() 方法返回。

```
<jndiObjectFactory id="objectFactory" libraryRef="objectFactoryLib"

          className="com.ibm.example.factory.MyObjectFactory"

          objectClassName="java.util.Properties"/>
```

5. 在 server.xml 文件的 jndiReferenceEntry 元素中声明条目并引用先前声明的工厂。

```
<jndiReferenceEntry id="refEntry" jndiName="ref/entry"
 factoryRef="objectFactory"/>
```

6. 对 server.xml 文件中的 jndiReferenceEntry 元素声明更多属性：

```
<jndiReferenceEntry id="refEntry" jndiName="ref/entry"
 factoryRef="objectFactory">
   <properties abc="123"/>
</jndiReferenceEntry>
```

还有其他属性在传递至工厂的 javax.naming.Reference 上表示为 javax.naming.StringRefAddr：

```
import javax.naming.spi.ObjectFactory;

public class MyObjectFactory implements ObjectFactory {
  @Override
  public Object getObjectInstance(Object o, Name n, Context c,
 Hashtable<?, ?> envmt) throws Exception {
    Properties p = new Properties();
    Reference ref = (Reference) o;
    RefAddr refAddr = ref.get("abc");
    p.put("abc", refAddr == null ? 123 : refAddr.getContent());
    return p;
  }
}
```

7. 可使用资源环境引用将产生的对象插入至应用程序：

```
@Resource(name="ref/entry")
private Properties properties;
```

2.8.4 将 OSGi 应用程序部署到 xigemaAS

可以通过在 server.xml 文件中启用服务器功能部件列表，将 OSGi 应用程序部署到 xigemaAS。

通过提供特定于 OSGi 的服务器功能部件列表，xigemaAS 为应用程序提供 OSGi 支持。这些功能部件为如下所示：

- blueprint-1.0
- osgi.jpa-1.0
- wab-1.0

有关 xigemaAS 中的服务器功能部件的完整列表，请参阅 *xigemaAS* 功能部件（见第 906 页）。

共享 xigemaAS 的公共 OSGi 捆绑软件

通过将 OSGi 捆绑软件放入目录并为服务器配置 server.xml 文件来共享公共 OSGi 捆绑软件，以便这些公共 OSGi 捆绑软件适用于 OSGi 应用程序。

- 在文件系统中创建目录并将所有公共 OSGi 捆绑软件放入该目录。
- 将下列行添加到 server.xml 文件。

```
  <bundleRepository>
```

```
    <fileset dir="directory_path" include="*.jar"/>
  </bundleRepository>
```

其中 *directory_path* 是包含公共 OSGi 捆绑软件的目录的路径。

- 在 OSGi 应用程序的 manifest.mf 中使用 `import` 短语来定义对公共捆绑软件的依赖性。

2.8.5 将数据访问应用程序部署到 xigemaAS

部署数据访问应用程序不只是将 Web 应用程序归档 (WAR) 文件或企业归档 (EAR) 文件安装到 xigemaAS。部署可以包含配置服务器和总体运行时环境的数据访问资源的任务。

此部分阐述了下列主题：

- 在 *xigemaAS* 概要文件中为数据库连接配置数据源和 *JDBC* 驱动程序
- 将 *JDBC* 应用程序部署到 *xigemaAS* 概要文件
- 可选： 在 *xigemaAS* 概要文件中配置连接池
- 可选： 在 xigemaAS 上开发应用程序定义的数据源
- 可选： 在 *xigemaAS* 上为数据源配置事务恢复
- 将数据访问应用程序迁移至 xigemaAS

将现有 JDBC 应用程序部署到 xigemaAS

您可以采用某一使用 Java™ 数据库连接 (JDBC) 和数据源的现有应用程序，然后将该应用程序部署到服务器。

可以获取现有 JDBC 应用程序并将其部署到 xigemaAS。要完成此部署，请将 `jdbc-4.0` xigemaAS 功能部件添加到 server.xml 文件。还必须添加代码，用以将 JDBC 驱动程序位置告知给服务器，以及指定 JDBC 驱动程序用于连接至数据库的属性。

在此示例中，您可以扩展您的 servlet 应用程序，或使用此处提供的 servlet 应用程序来测试通过 JDBC 驱动程序使用的互操作性是否按预期工作。

1. 创建服务器。
2. 启动服务器。
3. 将 *jdbc-4.0* 和 *servlet-3.0* *xigemaAS* 功能部件添加到 *server.xml* 文件。

```
<server>
   <featureManager>
       <feature>jdbc-4.0</feature>
       <feature>servlet-3.0</feature>
   </featureManager>
</server>
```

要检查服务器是否正常运行并且已成功启用功能部件，请参阅 `console.log` 文件，该文件存储在服务器的 `logs` 目录中。可以使用任何文本编辑器来查看该文件。您应该看到与以下示例相似的消息：

```
[AUDIT   ] CWWKF0012I: The server installed the following features:
 [jdbc-4.0, jndi-1.0].
[AUDIT   ] CWWKF0008I: Feature update completed in 0.326 seconds.
```

4. 在 server.xml 文件中指定数据库类型和数据源位置。

   其中 *path_to_derby* 是 derby 在操作系统上的安装位置，`lib` 是 `derby.jar` 所在的文件夹，`data/exampleDB` 是所创建的目录（如果不存在该目录）。

例如：

```
<jdbcDriver id="DerbyEmbedded" libraryRef="DerbyLib"/>

<library id="DerbyLib">
  <fileset dir="C:/path_to_derby/lib" includes="derby.jar"/>
</library>

<dataSource id="ds1" jndiName="jdbc/exampleDS" jdbcDriverRef="DerbyEmbedded">
  <properties.derby.embedded
   databaseName="C:/path_to_derby/data/exampleDB"
   createDatabase="create"
  />
</dataSource>
```

有关用于编写数据源定义代码的其他选项的信息，请参阅在配置文件中使用 *Ref* 标记（见第 1133 页）。

5. 将一些 SQL create、read、update 和 delete 语句添加至 JDBC 应用程序以测试与数据库的交互性。

```
package wasdev;

import java.io.*;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.annotation.Resource;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import javax.sql.DataSource;

@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {

  @Resource(name = "jdbc/exampleDS")
  private DataSource ds1;
  private Connection con = null;
  private static final long serialVersionUID = 1L;

  public HelloWorld() {
    super();
  }
  public void doGet (HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<H1>Hello World xigemaAS Profile</H1>\n");
          try {
       con = ds1.getConnection();
       Statement stmt = null;
       stmt = con.createStatement();
       // create a table
       stmt.executeUpdate("create table cities (name varchar(50) not null primary key,
 population int, county varchar(30))");
       // insert a test record
       stmt.executeUpdate("insert into cities values ('myHomeCity', 106769,
 'myHomeCounty')");
       // select a record
       ResultSet result = stmt.executeQuery("select county from cities where
 name='myHomeCity'");
       result.next();
       // display the county information for the city.
       out.println("The county for myHomeCity is " + result.getString(1));
       // drop the table to clean up and to be able to rerun the test.
       stmt.executeUpdate("drop table cities");
       }
    catch (SQLException e) {
       e.printStackTrace();
       }
    finally {
```

```
        if (con != null) {
            try{
               con.close();
               }
            catch (SQLException e) {
              e.printStackTrace();
              }
            }
        }
    }
}
```

6. 编译应用程序。

   其中，*path_to_xigemaas* 是 xigemaAS 在操作系统上的安装位置，*path_to_app* 是要编译的应用程序的 Java 文件位置。

   Windows 上的示例：

   ```
   C:\> javac -cp
   path_to_xigemaas\wlp\dev\api\spec\com.ibm.ws.javaee.servlet.3.0_1.0.1.jar
       path_to_App\HelloWorld.java
   ```

   Linux 上的示例：

   ```
   mo@machine01:~> javac -cp
            path_to_xigemaas/wlp/dev/api/spec/
   com.ibm.ws.javaee.servlet.3.0_1.0.1.jar
            path_to_App/HelloWorld.java
   ```

   如果无法识别 `javac` 命令，请确保在操作系统的 PATH 环境变量中具有 Java bin 目录。

7. 将应用程序添加到服务器。

   在此示例中，JDBC 应用程序放入服务器的 dropins 目录中：

   ```
   ...\dropins\HelloWorldApp.war\WEB-INF\classes\xigemaas\HelloWorld
   ```

   wasdev 目录使用 HelloWorld.java 中所使用的同一软件包名称。

8. 请检查 JDBC 应用程序是否正在运行。

   针对此示例，转至以下 URL：

   ```
   http://localhost:9080/HelloWorldApp/HelloWorld
   ```

   端口 9080 是 xigemaAS 服务器使用的缺省 HTTP 端口。 您可通过查看 server.xml 文件检查服务器设置为使用哪个 HTTP 端口。

   对于此示例，浏览器上的输出类似于：

   ```
   Hello World xigemaAS
   The county for myHomeCity is myHomeCounty
   ```

对 xigemaAS 启用 JDBC 跟踪

通过特定于驱动程序的定制跟踪设置，或者使用应用程序服务器补充 JDBC 跟踪选项，启用对 xigemaAS 的 JDBC 跟踪。

使用特定于驱动程序的定制跟踪工具有两种方式：

- 使用 Java™ 内置日志记录机制 `java.util.logging`（如果驱动程序支持的话）。
- 将定制跟踪设置配置为供应商属性。

如果 JDBC 驱动程序并不提供它自己的定制跟踪或日志记录工具，或者它提供的工具是最少的，那么可以使用应用程序服务器的补充 JDBC 跟踪。

如果使用定制供应商属性或补充 JDBC 跟踪来启用跟踪，那么必须在 bootstrap.properties 文件的跟踪规范中添加日志编写器名称。可以使用下列任一日志编写器：

DB2®

com.ibm.ws.db2.logwriter

Derby

com.ibm.ws.derby.logwriter

Informix® JCC（使用和 **DB2®** 相同的驱动程序）

com.ibm.ws.db2.logwriter

Informix® JDBC

com.ibm.ws.informix.logwriter

Microsoft™ SQL Server JDBC 驱动程序

com.ibm.ws.sqlserver.logwriter

DataDirect Connect for JDBC for Microsoft™ SQL Server

com.ibm.ws.sqlserver.logwriter

Sybase

com.ibm.ws.sybase.logwriter

其他数据库（例如 **solidDB®** 和 **MySQL**）

com.ibm.ws.database.logwriter

因为更改跟踪启用会涉及变更 bootstrap.properties 文件，所以必须重新启动服务器才能使更改生效。

以下示例说明如何使用各种 JDBC 跟踪方法。

- 使用 `java.util.logging`。

  如果使用的驱动程序支持 `java.util.logging`，那么它可以通过在 bootstrap.properties 文件的 `com.ibm.ws.logging.trace.specification` 中附加驱动程序的跟踪级别来启用。

  下面举例说明了 Microsoft™ SQL Server JDBC 驱动程序：

  - bootstrap.properties 文件的示例代码：

    ```
    com.ibm.ws.logging.trace.specification=*=audit=enabled:com.microsoft.sqlserver.jdbc=FINE
    ```

  下面举例说明了 Oracle JDBC：

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:oracle=FINE
```

- 对于 Oracle，您也必须使用下列两个选项中的一个，利用系统属性 `oracle.jdbc.Trace` 来启用跟踪：
  - 在 bootstrap.properties 文件中，添加设置 `oracle.jdbc.Trace=true`。
  - 在 Java™ 程序，添加设置 `System.setProperty("oracle.jdbc.Trace","true");`。

- 使用定制跟踪设置。

  如果使用的驱动程序具有定制跟踪设置，那么在 server.xml 文件中将它们设为 JDBC 驱动程序供应商属性。此外，还在 bootstrap.properties 文件的跟踪规范中添加日志编写器名称。

  下面举例说明了使用定制属性 `traceLevel` 的 DB2® JCC：

  - server.xml 文件的示例代码：

```
<dataSource id="db2" jndiName="jdbc/db2" jdbcDriverRef="DB2Driver" >
    <properties.db2.jcc databaseName="myDB" traceLevel="-1"/>
</dataSource>
```

  - bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.db2.logwriter=all=enabled
```

  下面举例说明了 Derby Network Client：

  - server.xml 文件的示例代码：

```
<dataSource id="derbyNC" jndiName="jdbc/derbyNC" jdbcDriverRef="DerbyNC" >
    <properties.derby.client databaseName="myDB" createDatabase="create" traceLevel="1"/>
</dataSource>
```

  - bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.derby.logwriter=all=enabled
```

  下面举例说明了 Informix® JCC。此数据库对 JCC 连接使用 DB2® 驱动程序。

  - server.xml 文件的示例代码：

```
<dataSource id="informixJCC" jndiName="jdbc/informixJCC" jdbcDriverRef="InformixDriverJCC"
 >
    <properties.informix.jcc databaseName="myDB" traceLevel="-1"/>
</dataSource>
```

  - bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.db2.logwriter=all=enabled
```

- 使用补充 JDBC 跟踪。

  如果 JDBC 驱动程序未提供合适的跟踪或日志记录工具，那么可以使用应用程序服务器的补充 JDBC 跟踪。应用程序服务器会自动根据使用的 JDBC 驱动程序来确定是否要启用补充 JDBC 跟踪。要对此进行覆盖，请将数据源属性 `supplementalJDBCTrace` 设为 `true` 或 `false`。

  1. 启用补充跟踪。

下面举例说明了对嵌入式 Derby 数据库启用补充跟踪。缺省情况下会对此数据库启用补充 JDBC 跟踪，因此您只需在 bootstrap.properties 文件中设置日志编写器：

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.derby.logwriter=all=enabled
```

下面举例说明了对 Informix® JDBC 启用补充跟踪。缺省情况下会对此数据库启用补充 JDBC 跟踪。

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.informix.logwriter=all=enabled
```

下面举例说明了对 Microsoft™ SQL Server JDBC 驱动程序启用补充跟踪和 `java.util.logging`：

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.sqlserver.logwriter=all=enabled:
    com.microsoft.sqlserver.jdbc=all
```

下面举例说明了对 DataDirect Connect for JDBC for Microsoft™ SQL Server 启用补充跟踪：

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.microsoft.sqlserver.jdbc=all
```

下面举例说明了对 solidDB® 启用补充跟踪。缺省情况下会对此数据库启用补充 JDBC 跟踪。

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.database.logwriter=all=enabled
```

下面举例说明了对 Sybase 启用补充跟踪。缺省情况下会对此数据库启用补充 JDBC 跟踪。

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.sybase.logwriter=all=enabled
```

下面举例说明了对其他数据库启用补充跟踪：

- bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled:com.ibm.ws.database.logwriter=all=enabled
```

2. 禁用补充跟踪

要禁用补充 JDBC 跟踪，请在 server.xml 文件中将 `supplementalJDBCTrace` 数据源属性设为 `false`，或者从 bootstrap.properties 文件中的 `com.ibm.ws.logging.trace.specification` 属性移除日志编写器名称。

- solidDB® 的 server.xml 文件的示例代码：

```
<dataSource id="soliddb" jndiName="jdbc/soliddb"
            jdbcDriverRef="solidDBDriver" supplementalJDBCTrace="false">
    <properties databaseName="dba" URL="jdbc:solid://localhost:2315/dba/dba" />
</dataSource>
```

- solidDB® 的 bootstrap.properties 文件的示例代码：

```
com.ibm.ws.logging.trace.specification=*=audit=enabled
```

👉 注：如果未见到 JDBC 跟踪，那么某个功能部件可能立即激活 JDBC。请检查 `bootstrapping.properties` 并编辑它以添加 JDBC 跟踪规范。

2.8.6 将 Web 应用程序部署到 xigemaAS

通过部署 helloworld.war 应用程序，可以了解服务器配置在 xigemaAS 概要文件中如何更改。

helloworld.war 应用程序使用简单的 servlet 在浏览器上显示一则消息。可以创建任何其他要显示的消息。xigemaAS 概要文件文档中未描述如何编写应用程序代码。

如果使用开发者工具将 Web 应用程序部署到 xigemaAS，那么会自动在 server.xml 文件中启用与应用程序相关的所有配置。但是，您也可以通过完成下列步骤来手动配置 server.xml 文件。

此示例使用 helloworld.war 应用程序，并且可以通过 `http://localhost:9090/helloworld` 来访问。在此示例中，已创建 xigemaAS 概要文件服务器实例，其缺省 HTTP 端口已更改为 9090，应用程序已部署至该端口。

1. 使用命令 `server create hwserver` 来创建服务器（名称为 `hwserver`）。
2. 将 helloworld.war 应用程序复制到 /usr/servers/hwserver/apps 目录；在步骤 1 中使用 `server create` 命令创建了此目录。
3. 在使用 `server create` 命令创建的 server.xml 文件中，如果要将服务器 `hwserver` 的缺省 HTTP 端口更改为 9090，请将属性值 `httpPort="9080"` 替换为 `httpPort="9090"`：

```
<server description="new server">

  <!-- Enable features -->
    <featureManager>
      <feature>jsp-2.2</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
      host="localhost"
      httpPort="9090"
      httpsPort="9443" />
</server>
```

4. 通过以下方式来更新 server.xml 以配置应用程序：
   - 通过使用 `webApplication` 元素来定义应用程序：

```
<server description="Hello World Server">

 <featureManager>
   <feature>servlet-3.0</feature>
 </featureManager>

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9090" />

<webApplication contextRoot="helloworld" location="helloworld.war" />

</server>
```

   - 通过使用 `application` 元素来定义应用程序：

```
<server description="Hello World Server">

 <featureManager>
```

```
    <feature>servlet-3.0</feature>
  </featureManager>

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9090" />

<application context-root="helloworld" type="war" id="helloworld"
    location="helloworld.war" name="helloworld"/>

</server>
```

webApplication 元素可将相同子元素用作 application 元素，但 context-root 和 type 除外。这两个元素不会一起对 context-root 生效，如果 application 和 webApplication 元素定义相同 context-root，那么只会使用一个，并且显示错误。

context-root 属性指定所部署应用程序的入口点。按下列优先顺序确定所部署应用程序的入口点：

- server.xml 文件中的 context-root
- application.xml（如果是 EAR 应用程序）
- ibm-web-ext.xml（如果是 Web 应用程序）
- server.xml 文件中应用程序的 name（如果是 Web 应用程序）
- Manifest.MF（如果是 WAB 应用程序）
- 目录名称或与 xigemaAS 的“dropins”目录相对的文件名

> 注：在应用程序服务器 server.xml 配置中，application 元素可包含 context-root 标记。此 context-root 标记适合与标记 type="war" 一起使用。对于所有其他应用程序类型，context-root 元素不起作用。
>
> 不能覆盖 EAR 应用程序或 EBA 应用程序的 context-root。不能对独立 WAR 文件或 webApplication 执行覆盖。

5. 使用命令 server run hwserver 在前台启动服务器。
6. 在 http://localhost:9090/helloworld 处测试应用程序。
7. 可选： 如果不需要服务器，请停止服务器。

2.8.7 将 SIP 应用程序部署到 xigemaAS

会话启动协议 (SIP) 应用程序是包含至少一个 SIP servlet 的 Java 程序。SIP 应用程序与其他 Web 应用程序使用相同方式进行部署。

在 xigemaAS 服务器中安装 sipServlet-1.1 功能部件。有关更多信息，请参阅添加和移除 *xigemaAS* 功能部件（见第 1129 页）。

配置 SIP 容器。有关更多信息，请参阅在 *xigemaAS* 上管理会话启动协议 *(SIP)*（见第 1225 页）。

要部署 SIP 应用程序，该应用程序必须包括在 Web 归档 (WAR) 文件、servlet 归档 (SAR) 文件或包含 WAR 或 SAR 文件的企业归档 (EAR) 文件中。

本任务描述如何手动部署 SIP 应用程序。

1. 通过下列其中一种方式，将 SIP 应用程序 WAR、SAR 或 EAR 文件添加至 xigemaAS 服务器：
   - 将归档文件移至服务器配置目录中对应混入工件的文件夹，即 wlp/usr/servers/*server_name*/dropins。xigemaAS 服务器监视 dropins 文件夹以查找新应用程序并使用缺省配置自动安装该应用程序。

- 将归档文件移至服务器配置目录中对应应用程序的文件夹，即 wlp/usr/servers/*server_name*/apps。然后，通过配置 server.xml 文件中的 `application` 元素，在 xigemaAS 服务器上安装 SIP 应用程序。

  以下以下示例安装 appName.ear 文件。`context-root` 属性指定所部署应用程序的入口点。

```
<application id="appId" name="appName" type="ear" location="appName.ear"
 context-root="/sip289/"/>
```

2.8.8 将 JPA 应用程序部署到 xigemaAS

为允许 xigemaAS 支持使用 Java™ Persistence API (JPA) 的应用程序，应根据您需要的规范将 `jpa-2.0` 或 `jpa-2.1` 功能部件添加至 server.xml 文件。您还需要定义持久性上下文和持久性单元，以及配置对实体管理器和实体管理器工厂的访问权。

此任务假定您已创建要在其中部署使用 JPA 的应用程序的 xigemaAS 服务器。请参阅手动创建应用服务器（见第 1101 页）。

xigemaAS 概要文件上有两个可用 JPA 功能部件：

- `jpa-2.0` 功能部件为应用程序（使用依照 JPA 2.0 规范编写的应用程序管理及容器管理的 JPA）提供支持。该支持是以 Apache OpenJPA 为基础构建的，提供扩展来支持容器管理的编程模型。
- `jpa-2.1` 功能部件为应用程序（使用依照 JPA 2.1 规范编写的应用程序管理及容器管理的 JPA）提供支持。该支持是以 EclipseLink 为基础构建的。
- 将 *`jpa-2.0`* 或 *`jpa-2.1`* 功能部件添加至 *server.xml* 文件。
- 将持久性上下文和持久性单元定义添加到 web.xml 文件。

  例如：

```
<persistence-context-ref>
    <persistence-context-ref-name>example/em</persistence-context-ref-name>
    <persistence-unit-name>ExamplePersistenceUnit</persistence-unit-name>
</persistence-context-ref>

<persistence-unit-ref>
    <persistence-unit-ref-name>example/emf</persistence-unit-ref-name>
    <persistence-unit-name>ExamplePersistenceUnit</persistence-unit-name>
</persistence-unit-ref>
```

- 配置对实体管理器的访问权。

  例如：

```
Context ctx = new InitialContext();
UserTransaction tran = (UserTransaction) ctx.lookup("java:comp/UserTransaction");
tran.begin();
EntityManager em = (EntityManager) ctx.lookup(java:comp/env/example/em");
Thing thing = new Thing();
em.persist(thing);
tran.commit();
```

- 配置对实体管理器工厂的访问权。

  例如：

```
Context ctx = new InitialContext();
EntityManagerFactory emf = (EntityManager) ctx.lookup(java:comp/env/example/emf");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Thing thing = new Thing();
```

```
em.persist(thing);
tx.commit();
int id = thing.getId();
em.close();
```

JPA 2.0 的 JPA 实体增强功能

xigemaAS 概要文件中包含的 JPA 2.0 规范提供程序基于 Apache OpenJPA。OpenJPA 使用 JPA 持久性类型（Entity、Embeddable 和 MappedSuperclass）的 Java™ 字节码增强功能来添加状态跟踪以及其他必需信息，以在 JPA 类中启用持久性及其他优化功能部件。在应用程序服务器环境中，当 xigemaAS 概要文件服务器载入应用程序时，系统会自动地增强 JPA 实体。

在应用程序服务器环境及非应用程序服务器环境中使用持久性 JAR 时，必须预先增强 JPA 类（或进行构建时增强）。执行构建时增强的最常用方式是执行 OpenJPA 增强程序 Ant 任务和 PCEnhancer。这些构建时增强选项要求 OpenJPA 库和从属库在 `classpath` 上。

👉 注：xigemaAS 概要文件的 JPA 2.1 规范提供程序为 EclipseLink。EclipseLink 不需要实体增强功能。

2.8.9 将 Web Service 应用程序部署到 xigemaAS

通过在 server.xml 文件中配置 xigemaAS 功能部件，您可以将 Web Service 应用程序部署到 xigemaAS。

将 JAX-RS 2.0 应用程序部署至 xigemaAS

可以使用 Java API for RESTful Web Services (JAX-RS) 来开发遵循表象化状态转变 (REST) 原理的服务。RESTful 服务依赖于处理资源。资源可以包含静态数据或动态更新的数据。通过在应用程序中标识资源，可以使服务更有用更易于开发。xigemaAS 提供两个 xigemaAS 功能部件（`jaxrs-1.1` 和 `jaxrs-2.0`）以支持 JAX-RS 编程模型。

异步处理

可在 JAX-RS 2.0 中使用异步处理方法来处理线程。异步处理在客户机 API 和服务器 API 中都是受支持的。有关在客户机和服务器 API 中进行异步处理的更多信息，请参阅 *JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services*（“规范”）的第 8 章。

以下两个示例显示客户机 API 和服务器 API 中的异步处理：

- 客户机 API 中的异步处理：

```
Client client = ClientBuilder.newClient();
WebTarget target = client.target("http://example.org/customers/{id}");
target.resolveTemplate("id", 123).request().async().get(
   new InvocationCallback «Customer» () {
          @Override
           public void completed(Customer customer) {
              // Do something
           }
          @Override
           public void failed(Throwable throwable) {
              // Process error
           }
});
```

- 服务器 API 中的异步处理：

```
@Path("/async")
public class MyResource {

          @GET
           public void getAsync(@Suspended final AsyncResponse
 asyncResponse){
              CompletionCallback callBack = new CompletionCallback(){
                     @Override
                     public void onComplete(Throwable throwable) {
                            ...
                     }
               };
             asyncResponse.register(callBack);
             asyncResponse.resume("some Response");
          }
}
```

xigemaAS 中的 JAX-RS 2.0 实现支持 EJB，并支持将无状态 bean 和单独会话 bean 用作根资源类。如果 EJB 方法是使用 `@Asynchronous` 注释的，那么 EJB 容器自动分配执行时所需的资源。因此，在此场景中，不必使用执行程序来生成异步响应。例如，

```
@Stateless
@Path("/")
class EJBResource {

 @GET @Asynchronous
 public void longRunningOp(@Suspended AsyncResponse ar) {
  executeLongRunningOp();
  ar.resume("Hello async world!");
 }
}
```

在此情况下，不需要显式线程管理，因为它在 EJB 容器的控制之下。响应是在通过对已插入 `AsyncResponse` 调用复原而产生的。因此，`longRunningOp` 的返回类型为空。

配置资源以通过 JAX-RS 2.0 中的 HTML 表单提交接收 multipart/form-data 部分

传输文件数据的 HTML 表单必须使用 POST 方法和“multipart/form-data”操作进行配置。JAX-RS 资源方法可通过两种方式的其中之一接收此数据，此方法使用 Java™ API for RESTful Web Services (JAX-RS) 实现接受此数据。

此任务提供有关配置 JAX-RS 方法以使用和产生 multipart/form-data 的指示信息。以下示例演示 HTML 表单：

```
<form action="http://www.example.com/" method="POST" enctype="multipart/form-data">
    <input type="text" name="fileid" />
    <br />
    <input type="text" name="description" />
    <br />
    <input type="file" name="thefile" />
    <br />
    <input type="submit" name="submit" value="submit"/>
</form>
```

可实现 JAX-RS 以接收部件中的数据，所以必要时您可自己处理这些部件。

1. 创建资源方法。

必须声明下列其中一个资源方法以从 HTTP POST 接收和回应 multipart/form-data 内容：

```
package com.example.jaxrs;
@POST
@Consumes("multipart/form-data")
@Produces("multipart/form-data")

public Response postFormData(IMultipartBody multipartBody) {
  List <IAttachment> attachments = multipartBody.getAllAttachments();
         String formElementValue = null;
         InputStream stream = null;
         for (Iterator<IAttachment> it = attachments.iterator(); it.hasNext();) {
              IAttachment attachment = it.next();
              if (attachment == null) {
                  continue;
              }
              DataHandler dataHandler = attachment.getDataHandler();
              stream = dataHandler.getInputStream();
              MultivaluedMap<String, String> map = attachment.getHeaders();
              String fileName = null;
              String formElementName = null;
              String[] contentDisposition = map.getFirst("Content-Disposition").split(";");
              for (String tempName : contentDisposition) {
                  String[] names = tempName.split("=");
                  formElementName = names[1].trim().replaceAll("\"", "");
                  if ((tempName.trim().startsWith("filename"))) {
                      fileName = formElementName;
                  }
              }
              if (fileName == null) {
                  StringBuffer sb = new StringBuffer();
                  BufferedReader br = new BufferedReader(new InputStreamReader(stream));
                  String line = null;
                  try {
                      while ((line = br.readLine()) != null) {
                          sb.append(line);
                      }
                  } catch (IOException e) {
                      e.printStackTrace();
                  } finally {
                      if (br != null) {
                          try {
                              br.close();
                          } catch (IOException e) {
                              e.printStackTrace();
                          }
                      }
                  }
                  formElementValue = sb.toString();
                  System.out.println(formElementName + ":" + formElementValue);
                      } else {
               //handle the file as you want
               File tempFile = new File(fileName);
               ...
              }
         }
         if (stream != null) {
             stream.close();
         }
         return Response.ok("test").build();
}
```

或者，

```
package com.example.jaxrs;
@POST
@Consumes("multipart/form-data")
@Produces("multipart/form-data")

public Response postFormData(List<IAttachment>attachments) {
  List <IAttachment> attachments = multipartBody.getAllAttachments();
         String formElementValue = null;
         InputStream stream = null;
         for (Iterator<IAttachment> it = attachments.iterator(); it.hasNext();) {
              IAttachment attachment = it.next();
              if (attachment == null) {
                  continue;
              }
              DataHandler dataHandler = attachment.getDataHandler();
              stream = dataHandler.getInputStream();
              MultivaluedMap<String, String> map = attachment.getHeaders();
              String fileName = null;
              String formElementName = null;
```

```
                String[] contentDisposition = map.getFirst("Content-Disposition").split(";");
                for (String tempName : contentDisposition) {
                    String[] names = tempName.split("=");
                    formElementName = names[1].trim().replaceAll("\"", "");
                    if ((tempName.trim().startsWith("filename"))) {
                        fileName = formElementName;
                    }
                }
                if (fileName == null) {
                    StringBuffer sb = new StringBuffer();
                    BufferedReader br = new BufferedReader(new InputStreamReader(stream));
                    String line = null;
                    try {
                        while ((line = br.readLine()) != null) {
                            sb.append(line);
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    } finally {
                        if (br != null) {
                            try {
                                br.close();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }
                    }
                    formElementValue = sb.toString();
                    System.out.println(formElementName + ":" + formElementValue);
                        } else {
                 //handle the file as you want
                 File tempFile = new File(fileName);
                 ...
                }
            }
            if (stream != null) {
                stream.close();
            }
            return Response.ok("test").build();
    }
```

表单 POST 提交的发起方可对 multipart 消息的一个或多个部件生成 Content-Transfer-Encoding 头。如果头为 `base64` 或 `quoted-printable` 编码类型，那么 IBM JAX-RS 实现尝试根据此头对部件的有效内容进行自动解码。

通过允许 JAX-RS 实现拆分部件及对其自动解码，并通过接收仍处于编码状态的部件以便自己处理，您从 Content-Type 为 multipart/form-data 的 HTTP POST 接收并回应数据。

配置 JAX-RS 2.0 客户机

对于 Java API for XML RESTful Web Services 2.0，可配置客户机以访问 REST 端点。JAX-RS 2.0 引入了新的标准化客户机 API 以便您可对远程 RESTful Web Service 发出 HTTP 请求。

要使用客户机 API 访问 Web 资源，需要**客户机**实例。缺省客户机实例可通过对 `ClientBuilder` 调用 `newClient` 或 `build` 获取。

1. 在 server.xml 文件中启用 jaxrsClient-2.0 或 jaxrs-2.0 功能部件。

```
<featureManager>
    <feature>jaxrs-2.0</feature>// If you only need the JAX-RS 2.0 client
 feature, you can enable jaxrsClient-2.0 instead of  jaxrs-2.0
</featureManager>
```

2. 创建 JAX-RS 2.0 客户机并向服务器发送请求：

```
javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();

javax.ws.rs.client.Client c = cb.build();
String res = null;

try {
```

```
res = c.target("<Resource_URL>")
            .path("<PATH>")
            .request()
            .get(String.class);
} catch (Exception e) {
      res = "[Error]:" + e.toString();
       } finally {
           c.close();
     }
```

有关异步 JAX-RS 2.0 客户机的更多信息，请参阅异步处理（见第 1510 页）。

- ◦ 使用 *com.ibm.ws.jaxrs.client.timeout* 客户机属性以设置超时值。

```
javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
         cb.property("com.ibm.ws.jaxrs.client.timeout", "1000");
         Client c = cb.build();
```

  👉 提示：超时属性的值为毫秒，类型必须为长整型或整型。如果值类型无效，那么将显示以下消息：

```
CWWKW0700E: 您在 JAX-RS 客户端上的属性
 com.ibm.ws.jaxrs.client.timeout 中指定的超时值 {0} 无效。此值设置为缺省值
 30000。{3}
```

  ◦ 使用 *com.ibm.ws.jaxrs.client.connection.timeout* 客户机属性和 *com.ibm.ws.jaxrs.client.receive.timeout* 客户机属性以设置超时值。

    ◦ com.ibm.ws.jaxrs.client.connection.timeout

```
javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
        cb.property("com.ibm.ws.jaxrs.client.connection.timeout",
 "1000");
        Client c = cb.build();
```

    ◦ com.ibm.ws.jaxrs.client.receive.timeout

```
javax.ws.rs.client.ClientBuilder cb = ClientBuilder.newBuilder();
        cb.property("com.ibm.ws.jaxrs.client.receive.timeout",
 "1000");
        Client c = cb.build();
```

  👉 提示：超时属性的值为毫秒，类型必须为长整型或整型。如果值类型无效，那么将显示以下消息：

```
CWWKW0700E: 您在 JAX-RS 客户端上的属性
 com.ibm.ws.jaxrs.client.receive.timeout 中指定的超时值 {0} 无效。此值设
置为缺省值 30000。{3}
```

- 使用以下客户机属性以获取客户机代理支持：

```
ClientBuilder cb = ClientBuilder.newBuilder();
cb.property("com.ibm.ws.jaxrs.client.proxy.host", "hostname");
cb.property("com.ibm.ws.jaxrs.client.proxy.port", "8888";);
cb.property("com.ibm.ws.jaxrs.client.proxy.type", "HTTP");

Client c = cb.build();
```

  ◦ com.ibm.ws.jaxrs.client.proxy.host

- com.ibm.ws.jaxrs.client.proxy.port

  提示：代理服务器端口值的类型必须为整型。缺省值为 80。如果值类型无效，那么将显示以下消息：

  ```
  CWWKW0701E: 您在 JAX-RS 客户端上的属性
   com.ibm.ws.jaxrs.client.proxy.port 中指定的代理服务器端口值 {0} 无效。此值设置为缺省值 80。{3}
  ```

- com.ibm.ws.jaxrs.client.proxy.type

  提示：代理服务器类型值必须为 HTTP 或 SOCKS。缺省值为 HTTP。如果代理服务器类型无效，那么将显示以下消息：

  ```
  CWWKW0702E: 您在 JAX-RS 客户端上的属性
   com.ibm.ws.jaxrs.client.proxy.type 中指定的代理服务器类型值 {0} 无效。此值设置为缺省值 HTTP。{3}
  ```

- 使用 *com.ibm.ws.jaxrs.client.ltpa.handler* 客户机属性以设置 SSO cookie 并将值设置为 true。

  ```
  ClientBuilder cb = ClientBuilder.newBuilder();
          Client c = cb.build();
          c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");
  ```

  如果要在 JAX-RS 2.0 中使用安全套接字层 (SSL) 功能，那么您需要启用 ssl-1.0 或 appSecurity-2.0 功能部件。对于 LTPA 令牌功能，必须启用 appSecurity-2.0 功能部件。

  注：ssl-1.0 功能部件是 appSecurity-2.0 功能部件的子功能部件。如果您启用 jaxrsClient-2.0 功能部件和 ssl-1.0 功能部件，那么系统会自动启用 appSecurity-2.0 功能部件。

- 使用 *com.ibm.ws.jaxrs.client.ssl.config* 客户机属性设置 server.xml 文件的 SSL 引用标识。

  ```
  ClientBuilder cb = ClientBuilder.newBuilder();
          cb.property("com.ibm.ws.jaxrs.client.ssl.config", "mySSLRefId");
          Client c = cb.build();
  ```

  注：server.xml 文件中的配置如下所示：

  ```
  <ssl id="mySSLRefId" keyStoreRef="clientKeyStore"
   trustStoreRef="clientTrustStore" />
  ```

- 使用 *com.ibm.ws.jaxrs.client.disableCNCheck* 客户机属性来禁用公共名称检查。

  ```
  ClientBuilder cb = ClientBuilder.newBuilder();
  cb.property("com.ibm.ws.jaxrs.client.disableCNCheck", true);
  ```

针对 JAX-RS 2.0 在 EAR 文件中部署 EJB

在 xigemaAS 中，JAX-RS 2.0 支持在必须包含在 EAR 文件内的 EJB JAR 文件中使用 EJB JAX-RS。

1. 将新 myearfile.ear 文件部署至 xigemaAS。
2. 使用以下 URL 模式以访问 JAX-RS 服务：

   ```
   http://<host>:<port>/<context root>/<path of jaxrs resource>
   ```

例如，可在 <myejbjaxrs.jar> 中访问 EJB JAX-RS：

```
http://<host>:<port>/myejbjaxrs/<path of jaxrs resource>
```

注：如果同一 EJB-jar 中有 EJB JAX-WS 类并且已启用 JAX-WS 2.2 功能部件（这意味着 JAX-WS 路由器模块也存在），那么缺省上下文根应该为短文件名 EJB jar+".jaxrs"，例如，"myejbjaxrs.jaxrs" 对应 <myejbjaxrs.jar>。

JAX-RS Web 应用程序的实现

您可以使用 Java™ API for RESTful Web Services (JAX-RS) 来开发遵循具象状态传输 (REST) 原理的服务。通过使用 JAX-RS，可以简化 RESTful 服务的开发。

JAX-RS 是用于快速开发 REST 应用程序的 Java™ API。这个标准 API 继续在整个 Java™ 群体中受支持。虽然 JAX-RS 提供了比 Servlet 更快速的 Web 应用程序开发方法，但 JAX-RS 的主要目标是构建 RESTful 服务。`jaxrs-1.1` 和 `jaxrs-2.0` 定义服务器端组件 API 以构建 REST 应用程序。xigemaAS JAX-RS 提供了 JAX-RS (JSR 311) 规范的实现。

通过使用 REST 的原理，业务应用程序可以受益于多项优势。RESTful 服务通常更易于开发和使用。大多数 RESTful 服务使用完善的传递标准，例如 HTTP。由于 HTTP 是具有 RESTful 属性的协议，因此 RESTful 服务具有可伸缩性优势，这使服务能够为不同的客户机提供服务并与多个服务进行互操作，从而使将来的成长变为可能。另外，由于大多数 RESTful 服务使用公共数据表示（例如 XML 和 JSON），因此 RESTful 服务的客户机通常易于开发，这将产生互操作性方面的优势。

通过使用 JAX-RS 技术，REST 应用程序比其他类型的分布式系统更易于开发、使用和伸缩。许多流行并广泛使用的互联网服务已成功地为它们的应用程序提供了 RESTful API。第三方已使用各种 REST API 来构建他们自己的业务和应用程序。

通过使用 Servlet 或过滤器提供了 JAX-RS 功能。在配置 Web 应用程序的 web.xml 文件并将基于 Apache Wink 框架的 xigemaAS JAX-RS 实现组装成 Web 应用程序的库目录之后，您的业务应用程序就能够使用 JAX-RS 功能。

安全 JAX-RS 应用程序的实现

xigemaAS JAX-RS 1.1 运行时环境由从 Apache Wink 项目派生的 servlet 驱动。JAX-RS 2.0 运行时环境由从 Apache CXF 3.0.2 派生的 servlet 驱动。在 xigemaAS 环境中，servlet 的生命周期是在 Web 容器中管理。因此，Web 容器所提供的安全服务适用于 xigemaAS 中部署的 REST 资源。

您可以使用用于组装 REST 应用程序的同一工具在 REST 资源上定义并添加安全性约束。在与您应用程序关联的 J2EE web 部署描述符中捕获这些约束。以下列表描述您可以包含在部署描述符中的安全性定义：

- 调用应用程序中表现的 REST 资源时的用户认证，包括
  - HTTP 基本认证。
  - 表单登录认证。
- 对 REST 资源的授权控制（由资源的 URL 模式定义）。
- 调用 REST 资源时使用 SSL 进行传输。
- 以编程方式使用 `SecurityContext` 对象以确定用户身份和角色。

Web 容器支持的所有安全性机制都适用于 REST 资源，包括使用基于 Kerberos 的 SPNEGO 认证机制。

保护下游 JAX-RS 资源

您可以通过配置 BasicAuth 方法进行认证以及使用 LTPA JAX-RS 安全处理程序来利用单点登录进行用户认证以保护下游 Java™ API for RESTful Web Services (JAX-RS) 资源。

该任务假设您完成了以下步骤：

- 将 JAX-RS 应用程序安装到应用服务器中。
- 为 JAX-RS 应用程序启用安全性。
- 通过将下游 JAX-RS 应用程序配置为将基本认证 (BasicAuth) 方法用于用户认证来保护 Web 容器内的 JAX-RS 应用程序。

编写 JAX-RS 资源时，可以将新的 LTPA JAX-RS 安全处理程序用于对下游资源调用进行无缝认证。

调用下游安全 JAX-RS 资源时，需要调用应用程序对目标资源进行认证。如果下游服务器上的目标资源使用 BasicAuth 方法实现安全，那么调用的应用程序可以利用 JAX-RS 资源的单点登录 (SSO)。使用单点登录，将认证的上下文沿着下游调用传播。您可以使用基于 LTPA 的安全客户机处理程序对的服务器中分布的下游资源进行认证。

为了说明此场景，假设您在单元中有两个服务器且您在这两个服务器上都部署了 JAX-RS 资源。假设您需要从 `server1` 上的一个资源调用部署在 `server2` 上的另一资源。 使用 BasicAuth 认证方法来保护 `server2` 资源时，使用 LTPA JAX-RS 安全处理程序来利用单点登录并在下游调用上无缝地传播用户认证，而不必在应用程序中提供或管理用户身份和密码。

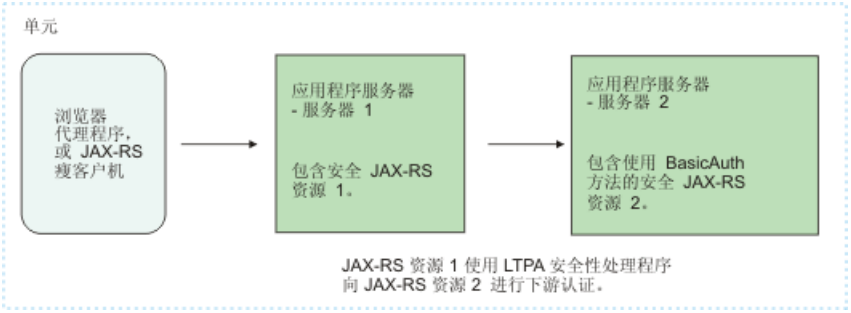


图 34: 保护 JAX-RS 下游资源

在应用程序构建时，利用 JAX-RS 安全处理程序，使用以下步骤配置下游服务器的用户认证。

1. 在应用程序构建时，使用基于 LTPA 的安全客户机处理程序 `LtpaAuthSecurityHandler` 对的服务器中分布的下游资源进行认证。
   - 对于 JAX-RS 1.1，使用 `LtpaAuthSecurityHandler` 类时，确保您使用 URL 的 https 方案定位资源，且目标应用程序已启用 SSL。极力建议您在发送用户凭证（包括 LTPA cookie）时使用 SSL 连接。您可以通过在安全处理程序上以 `false` 值调用 `setSSLRequired` 方法，在 `LtpaAuthSecurityHandler` 类中显式关闭对 SSL 的需求。缺省值为 `true`。

     ```
     yourLtpaAuthSecHandler.setSSLRequired(false);
     ```

   - 对于 JAX-RS 2.0，可使用 `com.ibm.ws.jaxrs.client.ltpa.handler` 客户机属性来获取 SSO cookie 并将值设置为 `true`：

     ```
     ClientBuilder cb = ClientBuilder.newBuilder();

             Client c = cb.build();
             c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");
     ```

```
        WebTarget t = c.target("http://" + serverIP + ":" + serverPort +
"/" + moduleName + "/ComplexClientTest/ComplexResource");
        String res =
t.path("echo1").path("test1").request().get(String.class);
        c.close();
        ret.append(res);
```

如果要在 JAX-RS 2.0 中使用安全套接字层 (SSL) 功能，那么您需要启用 ssl-1.0 或 appSecurity-2.0 功能部件。对于 LTPA 令牌功能，appSecurity-2.0 功能部件是必需的。

👉 注：ssl-1.0 功能部件是 appSecurity-2.0 功能部件的子功能部件。如果您启用 jaxrsClient-2.0 功能部件和 ssl-1.0 功能部件，那么系统会自动启用 appSecurity-2.0 功能部件。

2. 将安全处理程序添加到处理程序链。
3. 创建 REST 客户机实例。
4. 创建您要交互的资源实例。
5. 替换表示您资源地址的值。

您在定义了安全 JAX-RS 资源，以便在调用下游资源时，您可以使用单点登录并在下游调用上无缝地传播用户认证，而不必在应用程序中提供或管理用户身份和密码。

对于 JAX-RS 1.1，以下代码片段演示如何使用封装为 JAX-RS 客户机一部分的此安全处理程序。

```
 import org.apache.wink.client.Resource;
 import org.apache.wink.client.RestClient;
 import org.apache.wink.client.ClientConfig;
 import org.apache.wink.client.handlers.LtpaAuthSecurityHandler;

 ClientConfig config = new ClientConfig();
 LtpaAuthSecurityHandler secHandler = new LtpaAuthSecurityHandler();

 // Add this security handler to the handlers chain.
 config.handlers(secHandler);

 // Create the REST client instance.
 RestClient client = new RestClient(config);

 // Create the resource instance that you want to interact with.
 // Substitute a value representing your resource address
 resource =
  client.resource("http://localhost:8080/path/to/resource");

// Now you are ready to begin calling your resource.
```

对于 JAX-RS 2.0，以下代码片段演示如何使用封装为 JAX-RS 客户机一部分的此安全处理程序。

```
ClientBuilder cb = ClientBuilder.newBuilder();
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ltpa.handler", "true");

String res = "";
res = c.target("http://" + serverIP + ":" + serverPort + "/" + moduleName + "/rest/
ltpa").request();
c.close();
return res;
```

JAX-RS 2.0 行为更改

JAX-RS 2.0 实现包含一些行为更改。这些更改可能导致从 JAX-RS 1.1 升级的应用程序在 JAX-RS 2.0 上的行为方式不同或失效。

以下列表描述 JAX-RS 1.1 与 JAX-RS 2.0 之间的差别：

- 在 JAX-RS 1.1 和 Jersey 中，如果 EJB 或 CDI 类创建新实例并且此实例由 JAX-RS application.getSingletons() 方法返回，那么引擎使用所返回实例并且不会尝试从 EJB 或 CDI 容器访问该实例。在 JAX-RS 2.0 中，对于同一场景，引擎尝试从 EJB 或 CDI 容器访问该实例。如果可访问该实例，那么会使用检索到的实例。但是，如果无法访问该实例，那么会使用从 getSingletons() 方法返回的实例。例如：

```
@Override
public SetObject getSingletons() {
    SetObject objs = new HashSetObject();
    objs.add(new CDIInjectResource());
    objs.add(new EJBInjectResource());
    return objs;
}
```

- JAX-RS 2.0 处理多部件文件时包含许多 API 更改。例如，在 JAX-RS 1.1 中，可使用 `@FormParam` 来处理多部件文件，但在 JAX-RS 2.0 中，只能使用 `@IMultipartBody` 或 `@IAttachment` 来处理多部件文件。有关更多信息，请参阅配置资源以通过 *JAX-RS 2.0* 中的 *HTML* 表单提交接收 *multipart/form-data* 部分。
- 在 JAX-RS 1.1 中显示为第三方 API 的 jackson 包不再显示在 JAX-RS 2.0 中。如果您要在应用程序中使用任何 `org.codehaus.jackson` API，您需要在应用程序中压缩 jackson 包。
- 如果在 web.xml 文件中指定 `javax.ws.rs.core.Application` 作为 servlet 名称，那么 `@Context` 在 Application 对象中插入的 `getClasses` 方法不会返回资源类。

```
      <servlet>
  <servlet-name>javax.ws.rs.core.Application</servlet-name>
  </servlet>
 <servlet-mapping>
  <servlet-name>javax.ws.rs.core.Application</servlet-name>
  <url-pattern>/*</url-pattern>
 </servlet-mapping>
```

- JAX-RS 2.0 规范声明提供程序是一个类，该类实现一个或多个 JAX-RS 接口并可使用 `@Provider` 进行注解以执行自动发现。如果某个类具有引用提供程序接口的 `@Local` 注解，但它未实现任何 POJO 提供程序接口，那么它是无效提供程序。例如：

```
@Stateless
@Local(OneLocalInterfaceMyOtherStuffMessageBodyWriter.class)
public class OneLocalInterfaceMyOtherStuffProvide
```

- 如果使用 MessageBodyReader 和 MessageBodyWriter `@Consumes` 及 `@Produces` 注解，那么一些受支持介质类型可能受到限制。请使用 `isReadable` 或 `isWriteable` 方法来检查介质类型。例如：

```
@Provider
@Consumes("<custom/type>")
@Produces("<custom/type>")
@Singleton
```

```
public class MyMessageBodyReaderAndWriter implements
 MessageBodyReader,MessageBodyWriter {

    public boolean isReadable(Class<?> type,
                              Type genericType,
                              Annotation[] annotations,
                              MediaType mediaType) {
        if (mediaType.toString().equals("<custom/type>"))
            return true;
        return false;
    }

     public boolean isWriteable(Class<?> type,
                              Type genericType,
                              Annotation[] annotations,
                              MediaType mediaType) {
        if (mediaType.toString().equals("<custom/type>"))
            return true;
        return false;
    }
...
}
```

- 可在 JAX-RS 2.0 中使用异步处理来处理线程。有关更多信息，请参阅异步处理（见第 1510 页）。
- 在 JAX-RS 1.1 中显示为第三方 API 的 Wink API 在 JAX-RS 2.0 中都不受支持。以下是部分列表：
  - `org.apache.wink.common.model.atom.AtomEntry`。有关将 JAX-RS 2.0 与 Atom 集成的更多信息，请参阅*JAX-RS 2.0 与 Atom 的集成*（见第 1522 页）。
  - `org.apache.wink.client.handlers.BasicAuthSecurityHandler`。如果要在 JAX-RS 2.0 中使用基本认证，请查看以下代码片段：

    1. 如代码示例中所示，通过 JAX-RS 2.0 标准客户机 API 使用 ClientRequestFilter：

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import javax.ws.rs.client.ClientRequestContext;
import javax.ws.rs.client.ClientRequestFilter;
import javax.ws.rs.core.MultivaluedMap;
import javax.xml.bind.DatatypeConverter;

public class BasicAuthFilter implements ClientRequestFilter {

    private final String usr;
    private final String pwd;

    public BasicAuthFilter(String usr, String pwd) {
        this.usr = user;
        this.pwd = pwd;
    }

    public void filter(ClientRequestContext requestContext) throws
 IOException {
        MultivaluedMap<String, Object> headers =
 requestContext.getHeaders();

     String token = this.usr + ":" + this.pwd;
     final String basicAuthentication ="Basic " +
 DatatypeConverter.printBase64Binary(token.getBytes("UTF-8"));
     headers.add("Authorization", basicAuthentication);
```

```
        }
    }
```

2. 注册至 ClientBuilder：

```
ClientBuilder cb = ClientBuilder.newBuilder();
cb.register(new BasicAuthFilter("user","password"));
```

- `org.apache.wink.client.handlers.LtpaAuthSecurityHandler`。如果要使用基于 LTPA 的安全客户机保护下游资源，请参阅保护下游 *JAX-RS* 资源（见第 1517 页）。
- `org.apache.wink.server.internal.providers.exception.EJBAccessExceptionMapper`。此 API 不再受支持，因为它是 Wink 指定的 ExceptionMapper。可定义您自己的 ExceptionMapper 以映射 EJBAccessException。
- `com.ibm.websphere.jaxrs.server.IBMRestFilter`。此 API 不再受支持，因为它基于 Wink 过滤器。

👉 注：检测应用程序中是否有 Wink JAR 包。如果应用程序中有任何 Wink 包，那么必须执行以下步骤：

1. 确保已定义应用程序子类。
2. `getClasses` 和 `getSingletons` 的至少其中之一不得返回空值。

- 有关可在 JAX-RS 2.0 客户机中使用的受支持客户机属性的更多信息，请参阅配置 *JAX-RS 2.0* 客户机。
- 如果要在 JAX-RS 2.0 中使用安全套接字层 (SSL) 功能，请执行以下步骤：

1. 启用 `ssl-1.0` 功能部件或 `appSecurity-2.0` 功能部件。对于 LTPA 令牌功能，`appSecurity-2.0` 功能部件是必需的。

   👉 注：`ssl-1.0` 功能部件是 `appSecurity-2.0` 功能部件的子代。如果启用 `jaxrsClient-2.0` 功能部件和 `ssl-1.0` 功能部件，那么将自动启用 `appSecurity-2.0` 功能部件。

2. 按如下所示在 JAX-RS 2.0 客户机代码中启用 `com.ibm.ws.jaxrs.client.ssl.config` 属性：

```
ClientBuilder cb = ClientBuilder.newBuilder();
Client c = cb.build();
c.property("com.ibm.ws.jaxrs.client.ssl.config", "mySSLConfig"); //
mySSLConfig is the ssl ref id in xigemaAS server.xml
```

   👉 注：此属性可将 xigemaAS SSL 配置绑定至 ClientBuilder、客户机和 WebTarget 作用域。

- 如果要在 JAX-RS 2.0 服务器运行时中使用 Wink 客户机，请执行以下步骤：

1. 下载以下文件，它们可在 JAX-RS 2.0 服务器运行时中启用 Wink 客户机。
   - 从 *http://wink.apache.org/downloads.html* 下载 Apache Wink 及相关 JAR 文件。
   - 从 *http://hc.apache.org/* 下载 Apache HTTP 及相关 JAR 文件。

   👉 注：如果未启用该 JAX-RS 2.0 功能部件，那么还必须下载 JAX-RS API 并将其添加至第三方库。从 *https://jax-rs-spec.java.net/nonav/* 下载 JAX-RS API。

2. 将所有 JAR 文件保存到 <third-party lib> 目录中。
3. 将 <third-party lib> 的位置添加至 server.xml 文件：

```
<library id="thirdPartyLib">
    <fileset dir=" <third-party lib>" includes="*.jar"
 scanInterval="5s"/>
```

```
</library>
<enterpriseApplication id="<Your Ear ID>" location="<Your Ear Name>"
 name="<Your Ear Name>">
    <classloader commonLibraryRef="thirdPartyLib"/>
</enterpriseApplication>
```

注：有关在客户机和服务器 API 中进行异步处理的更多信息，请参阅 *JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services*（“规范”）的第 8 章。

JAX-RS 2.0 与 Atom 的集成

JAX-RS 2.0 可使用 Apache Abdera 添加 Atom 支持。

可向 JAX-RS 端点注册以下基于 Apache Abdera 的提供程序并使用资源方法显式处理 Abdera Feed 或 Entry 类：

- 基于 Apache Abdera 的订阅源提供程序：`net.wasdev.wlp.sample.abdera.jaxrs.atom.AtomFeedProvider`
- 基于 Apache Abdera 的条目提供程序：`net.wasdev.wlp.sample.abdera.jaxrs.atom.AtomEntryProvider`

注：`AtomFeedProvider` 和 `AtomEntryProvider` 都支持带格式的 Output 属性。

注：Apache CXF 提供用于将 JAX-RS 2.0 与 Atom 集成的其他方法。有关更多信息，请参阅*https://cxf.apache.org/docs/jax-rs-data-bindings.html#JAX-RSDataBindings-Atom*。

将 JAX-RS 2.0 与 EJB 和 CDI 集成

xigemaAS 中的 JAX-RS 2.0 与 Enterprise JavaBeans™ (EJB) 及上下文和依赖关系注入 (CDI) 集成。

为使 JAX-RS 2.0 使用企业 bean，您需要使用 `@Path` 注释 bean 的类并将其转换为根资源类。

通过与 EJB 集成，您可注释 EJB bean 以将它们展示为 REST 端点。还可使用 EJB 的 JTA 和安全功能。xigemaAS 中的 JAX-RS 2.0 支持使用无状态 bean 和单独会话 bean 作为根资源类、提供者和应用程序子类。通过与 CDI 集成，您可注释 CDI bean 或受管 bean 作为 REST 端点并对 Web Service 使用 CDI 注入。xigemaAS 中的 JAX-RS 2.0 支持 CDI 样式的 bean 作为根资源类、提供者和应用程序子类。提供者和应用程序子类必须是单例或使用应用程序作用域。CDI 规范简化了集成不同类型的 Java™ EE 组件的过程。它提供可将组件（例如 EJB 组件或受管 bean）注入其他组件（例如 JSP 或其他 EJB）的一种常用机制。

对于 EJB，可将注释与无状态会话 bean 和单例 POJO bean 配合使用。

- 对于无状态会话 bean，请按以下示例中所示使用 `@Stateless` 注释：

```
@Stateless
@Path("stateless-bean")
public class StatelessResource {...}
```

- 对于单例 bean，请按以下示例中所示使用 `@Singleton` 注释：

```
@Singleton
@Path("singleton-bean")
public class SingletonResource {...}
```

对于 CDI，可将 `@ApplicationScoped` 和 `@Inject` 注释与应用程序作用域的 bean 配合使用。

👉 提示：如果 CDI 功能部件被禁用，那么 JAX-RS 不会报告错误，但会通过使用 POJO 获取实例。

```
@ApplicationScoped
@Path("/ApplicationScopedResource")
public class ApplicationScopedResource {

    private @Inject
    SimpleBean injected;

...

}
```

对 JAX-RS 2.0 与 EJB 和 CDI 的限制

请参阅以下各项以了解 xigemaAS 中的 JAX-RS 2.0 的限制：

- 如果使用 EJB 作为 JAX-RS 资源、提供者或应用程序，那么不能对 EJB bean 的构造函数使用 `@Context` 注入。原因在于根据 EJB 和 JAX-RS 规范，只能对 JAX-RS 使用带有缺省构造函数的 EJB。
- 如果在 Java™ 类中使用 EJB 或 CDI 注释，但未在 server.xml 文件中配置 EJB（例如，ejbLite-3.2）或 CDI（例如，cdi-1.0）的 xigemaAS 功能部件（这意味着 xigemaAS 运行时中没有 EJB 或 CDI 支持），那么 JAX-RS 2.0 引擎使用 Java™ 类作为 POJO 类。
- 对于应用程序类，如果它未实现任何接口或者它具有 `@Localbean` 注释，那么它被视为 EJB；如果它实现本地接口或 POJO 接口，那么它不会被视为 EJB。
  - 对于提供者：
    - 如果类仅实现 POJO 提供者接口而没有 `@Local` 注释，那么它被视为有效 EJB 提供者。
    - 如果类有 `@LocalBean` 注释并实现 POJO 提供者接口，那么它被视为有效 EJB 提供者。
    - 如果类有带 `@Local` 注释的本地接口，那么本地接口是提供者接口。如果此类实现提供者接口，那么它是有效 EJB 提供者。
    - 如果类有带 `@Local` 注释的本地接口，并且该本地接口不是提供者接口，那么它不是有效提供者。

      原因是在此情况下，EJB 容器只能为本地接口而不是 POJO 提供者接口生成 EJB 存根。
    - 如果类只有引用提供者接口的 `@Local` 注释，而未实现此提供者接口，那么它不是有效提供者；根据 JAX-RS 2.0 规范：提供者是一个类，该类实现此规范中引入的一个或多个 JAX-RS 接口，并可使用 `@Provider` 进行注释以执行自动发现。
  - 对于资源：
    - 如果基于 EJB 的资源未实现任何接口，那么此类中声明的所有方法可作为 JAX-RS 资源提供。
    - 如果基于 EJB 的资源实现一个接口（本地或 POJO），那么此接口中声明的所有方法以 JAX-RS 资源形式提供。
    - 如果基于 EJB 的资源实现多个接口，那么：
      1. 所有接口是不带 `@Local` 注释的 POJO 接口时，接口中声明的所有方法以 JAX-RS 资源形式提供。
      2. 所有接口是带 `@Local` 注释的本地接口时，接口中声明的所有方法以 JAX-RS 资源形式提供。
      3. 某些接口是带 `@Local` 注释的本地接口而其他接口不是本地接口时，只有本地接口中声明的方法才以 JAX-RS 资源形式提供。原因在于，EJB 容器只能为本场景中的本地接口生成 EJB 存根。

4. 如果基于 EJB 的资源具有 `@LocalBean` 注释，那么类中声明的所有方法以 JAX-RS 资源形式提供。
5. 如果基于 EJB 的资源实现接口，那么必须在该接口中声明 JAX-RS 资源方法。如果该接口是不能修改的提供者，那么您必须为资源类创建新接口以添加资源方法。否则，它不会被视为 EJB 资源。

- 如果带 `@Path` 注释的资源类实现 JAX-RS 提供者接口或者它使用 `@Provider` 注释进行声明，那么此类同时充当资源和提供者。在此情况下，JAX-RS 2.0 引擎仅使用此类的一个实例（由资源和提供者共享），此实例的生命周期为单例，这是缺省情况。
- 如果某个类已在应用程序类的 getClasses 和 getSingletons 方法中进行了注册，那么缺省情况下，JAX-RS 2.0 引擎使用 getSingletons 方法中的实例并忽略 getClasses 方法中的注册。
- 如果 RESTful 资源也是 CDI 管理的 bean 并且其作用域为 `javax.enterprise.context.Dependent`，那么会因为 CDI 限制而不能调用 PreDestroy 方法。

JAX-RS 2.0 bean 和 EJB bean 生命周期

JAX-RS bean 与 EJB bean 具有不同生命周期。如果该 JAX-RS 和 EJB 两者的 bean 生命周期冲突，那么生命周期由 xigemaAS 中的 EJB 容器管理。所以，JAX-RS 生命周期不起作用时，系统会应用 EJB 实例。有关更多信息，请参阅下表：

表 40: JAX-RS 2.0 bean 和 EJB bean 生命周期

应用程序	**JAX-RS 2.0**	**EJB**	结果
资源	perRequest	无状态	无状态
	perRequest	单例	单例
	单例	无状态	无状态
	单例	单例	单例
提供者	单例	无状态	无状态
	单例	单例	单例

JAX-RS 2.0 作用域和 CDI 作用域生命周期

Bean 的作用域确定其实例的生命周期。JAX-RS 和 CDI 两者的作用域稍有不同。如果 JAX-RS 和 CDI 的范围生命周期冲突，请参阅下表以获取结果：

表 41: JAX-RS 2.0 作用域和 CDI 作用域生命周期

应用程序	**JAX-RS 2.0** 作用域	**CDI** 范围注释	结果
资源	perRequest	@ApplicationScoped	单例
	perRequest	@RequestScoped	perRequest
	perRequest	@Dependent	perRequest
	perRequest	@SessionScoped	Session
	perRequest		perRequest

应用程序	**JAX-RS 2.0** 作用域	**CDI** 范围注释	结果
	单例	@ApplicationScoped	单例
	单例	@RequestScoped	perRequest
	单例	@Dependent	单例
	单例	@SessionScoped	Session
	单例		单例
提供者	单例	@ApplicationScoped	单例
	单例	@RequestScoped	单例
	单例	@Dependent	单例
	单例	@SessionScoped	单例
	单例		单例

JAX-RS 2.0 作用域和 CDI 作用域生命周期冲突消息

如果 JAX-RS 2.0 与 CDI 两者的作用域生命周期冲突，那么会显示以下警告消息。它们只是警告消息，不需要执行任何操作。

- `CWWKW1001W: JAXRS-2.0 资源 {0} 的作用域 {1} 与 CDI 作用域 {2} 不匹配。xigemaAS 从 {3} 获取资源实例。`

  如果 JAXRS-2.0 资源作用域与 CDI 作用域不匹配并且 CDI 中存在该资源实例（导致 xigemaAS 从 CDI 获取该资源实例），那么会显示此消息。如果实例来自 JAX-RS，那么它未包含 CDI 注入。
- `CWWKW1002W: JAXRS-2.0 提供者 {0} 的 CDI 作用域为 {1}。xigemaAS 从 {2} 获取该提供者实例。`

  显示此消息的原因是该提供者实例仅为“单例”。如果该提供者的 CDI 作用域是 Dependent 或 ApplicationScoped，那么 xigemaAS 从 CDI 获取该提供者实例。如果实例来自 JAX-RS，那么它未包含 CDI 注入。

JAX-RS 2.0 与受管 bean 的集成

xigemaAS 中的 JAX-RS 2.0 支持使用受管 bean 作为根资源类、提供程序和应用程序子类。

- 要将 JAX-RS 2.0 与受管 bean 集成，请在 server.xml 文件的 featureManager 元素内添加 `<feature>managedBeans-1.0</feature>` 条目。
- 要将受管 bean 用作 JAX-RS 资源、提供程序或应用程序，请使用 `@ManagedBean` 来注释这些类。

  例如，按如下所示使用 Interceptors 受管 bean 功能部件：

```
@ManagedBean ("JaxrsManagedBean" )
@Path ("/managedbean" )
public class ManagedBeanResource {

    public static class MyInterceptor {
        @AroundInvoke
        public Object around(InvocationContext ctx) throws Exception {
              System. out .println("around() called" );
            return ctx.proceed();
```

```
        }
    }

    @GET
    @Produces( "text/plain")
    @Interceptors(MyInterceptor. class )
    public String getIt() {
        return "Hi managed bean!" ;
    }
}
```

对带有受管 bean 的 JAX-RS 2.0 的限制

由上下文和依赖性注入 (CDI) 管理的以下 JAX-RS 组件类才支持资源注入：

- 应用程序子类
- 提供程序
- 根资源类

具体地说，要将受管 bean 实例插入到某个 JAX-RS 组件类中，必须确保此组件类可识别为 CDI bean 并进行管理。

例如，要按如下所示将 `printMyName` 受管 bean 实例插入至 JAX-RS 根资源类，必须在 .WAR file/WEB-INF 存储库中添加空 beans.xml 文件：

```
@Path ("/managedbean" )
public class ManagedBeanResource {

    @Resource(name = "printMyName" )
    private PrintMyName printMyName ;

    @GET
    @Produces( "text/plain")
    public String getIt() {
        printMyName .print();
        return "Hi managed bean!" ;
    }
}


@ManagedBean ("printmyname" )
public class PrintMyName {

    public void print() {
        // TODO Auto-generated method stub
        System. out .println("Injection of ManagedBean is successful" );
    }

}
```

从客户机发送多个查询参数 - 级联或迭代编程

如果要将客户端中的多个查询参数发送至服务器，那么您可查看以下样本。

👉 注：通常，将多个查询参数放置在一个 WebTarget 对象中的方法是使用以下级联编程方式：

```
javax.ws.rs.core.Response response = client.target(...).queryParam(key,
 value).queryParam(key, value).queryParam(key, value).request.get();
```

但是，在某些情况下，级联编程方式不适用，因为键值对数是可变的，无法预测。对于这些情况，可使用以下基于迭代的编程方式：

```
Map<String, String> queryStrings;
...
javax.ws.rs.client.WebTarget  target = client.target(....);
 for (String key: queryStrings.keySet()){
    String value = queryStrings.get(key);
   target = target.queryParam(key, value);  //It is important to know
 queryParam method won't update current WebTarget object, but return a
 new one.
  }
 }
 javax.ws.rs.core.Response response = target.request().get();
```

使用 JAX-RS 2.0 上下文对象以获取有关请求的更多信息

Java API for RESTful Web Services (JAX-RS) 2.0 为应用程序子类、根资源类和提供者提供不同类型的上下文。可使用 `@Context` 注解将 HttpHeaders、UriInfo、HttpServletRequest 之类的上下文对象插入至应用程序子类、根资源类和提供者中的类字段或方法参数。

可使用对提供程序（客户机和服务器）、资源类（仅服务器）和应用程序子类（仅服务器）可用的以下上下文对象：

上下文对象	类型	描述
`Application`	类	可使用 `@Context` 注解将应用程序提供的`Application` 子类的实例插入到类字段或方法参数中。对 `Application` 子类实例的访问允许将配置信息集中到该类中。 👉 注：此 `Application` 子类不能插入到 `Application` 子类自身中，因为这会导致循环依赖性。
`UriInfo`	接口	`UriInfo` 接口提供有关请求 URI 组件的静态和动态信息（针对每个请求）。
`HttpHeaders`	接口	`HttpHeaders` 接口提供对映射表单中的请求头信息的访问或通过强类型便利方法进行的访问。

上下文对象	类型	描述
`Request`	接口	`Request` 接口允许调用者确定最佳匹配表示变体并评估资源的当前状态是否与请求中的任何前置条件匹配。
`SecurityContext`	接口	`SecurityContext` 接口提供对有关当前请求的安全上下文的信息的访问。
`Providers`	接口	`Providers` 接口允许根据一组搜索条件查找提供程序实例。
`ResourceContext`	接口	`ResourceContext` 接口提供对缺省范围（针对每个请求）中的资源或子资源类的实例化和初始化的访问。
`Configuration`	接口	客户机和服务器运行时配置都可通过 `@Context` 注入。这些配置可注入到提供程序（客户机或服务器）及资源类（仅服务器）中。

WADL2JAVA 命令

`wadl2java` 命令行工具处理现有 Web 应用程序描述语言 (WADL) 文件并生成用于开发 Java™ API for RESTful Web Services (JAX-RS) Web Service 应用程序的必要工件。`wadl2java` 命令行工具支持使用自顶向下方法开发 JAX-RS Web Service。从现有 WADL 文件着手时，使用 wadl2java 命令行工具生成必要的 JAX-RS 工件。

Web 应用程序描述语言 (WADL)

WADL 是一种以资源为中心的描述语言，旨在促进 RESTful Web 应用程序的建模、描述和测试。有关更多信息，请参阅 *Web* 应用程序描述语言。

语法

命令语法如下所示：

```
wadl2java --[options]
```

```
wadl2java -wadlns wadl-namespace -p  package-name -sp  [schema-namespace =]package-
name -tMap  schema-type=java-type * -repMap  media-type=class-name *  -resource
  resource-name -b  binding-file-name * -catalog  catalog-file-name -d  output-directory
 -interface -impl -async methodNames * -generateEnums -inheritResourceParams -noTypes -
noVoidForEmptyResponses -noAddressBinding -supportMultipleXmlReps -generateResponseIfHeadersSet
 -generateResponseForMethods methodNames * -async methodNames * -xjc xjc-argumentsv * -encoding
  encoding -h|-?|-help -version|-v -verbose|-V -quiet|-q|-Q  wadl
```

参数

以下 *options* 值对 `wadl2java` 命令可用：

-wadlns *wadl-namespace*

指定 WADL 命名空间。

-p *package-name*

指定要用于表示 WADL 资源元素的已生成代码的 Java 包名。

-sp */schema-namespace =/package-name*

指定要用于表示 WADL 语法元素的已生成代码的 Java 包名。（可选）指定命名空间至 Java™ 包名的映射。

-tMap *schema-type=java-type* *

指定 WADL 参数或表示模式类型与定制 Java™ 类型之间的可选映射。

-repMap *media-type=class-name* *

指定不带 wadl:element 属性的 WADL 表示与 Java™ 类之间的可选映射。

-resource *resource-name*

指定要用于表示不带 id 属性的 WADL 资源的已生成代码的简单类名。

-b *binding-file-name* *

指定外部 jaxb 绑定文件。对每个绑定文件使用一个 -b 标记。

-catalog *catalog-file-name*

指定要映射所导入 WADL 或模式的目录文件。

-d *output-directory*

指定用于放置代码的目录。

-interface

指定生成接口。

-impl

指定生成哑元服务实现。

-async *methodNames* *

指定支持暂挂异步调用时所需的方法名或标识的逗号分隔列表。

-generateEnums

指定可生成枚举类以表示带有多个选项的参数。

-inheritResourceParams

指定子资源可继承资源层次（路径或矩阵）参数。

-noTypes

禁止生成类型

-noVoidForEmptyResponses

对不带响应表示的方法使用 JAX-RS 响应返回类型。

-noAddressBinding

指定生成器可能未使用地址 jaxb 绑定文件将 wsa:EndpointReferenceType 或 wsa:EndpointReference 映射至 javax.xml.ws.wsaddressing.W3CEndpointReference。

-supportMultipleXmlReps

指定某个方法包含多个请求 XML 表示时，将针对每个此类表示生成单独方法。生成服务器端 JAX-RS 代码时，请不要启用此选项。包含 javax.xml.transform 的单个方法。在这类情况下会缺省生成源输入参数。

-generateResponseIfHeadersSet

如果 WADL 响应元素具有“header”参数，请使用 JAX-RS 响应返回类型。

-generateResponseForMethods *methodNames* *

指定生成 JAXRS 响应返回类型时所需的方法名或标识的逗号分隔列表。

-async *methodNames* *

指定支持暂挂异步调用时所需的方法名或标识的逗号分隔列表。

-xjc *xjc-argumentsv* *

指定使用 JAXB 数据绑定时直接传递至 XJC 的自变量的逗号分隔列表。此选项导致 XJC 装入用于扩充代码生成的额外插件。 例如，要装入用于将 toString() 方法添加至所有已生成类型的 toString(ts) 插件，应使用以下自变量：-xjc-Xts 可使用 -xjc-X 获取可用 XJC 插件列表。

-encoding *encoding*

指定生成 Java 源时要使用的字符集编码。

-h|-?|-help

显示选项的详细信息。

-version|-v

显示工具的版本。

-verbose|-V

指定生成器以详细方式运行。

-quiet|-q|-Q

-quiet|-q|-Q

wadl

wadl-url

将 JAX-WS 应用程序部署到 xigemaAS

可以使用 Java API for XML-based Web Services (JAX-WS) 来开发服务并将 JAX-WS 应用程序部署到 xigemaAS。xigemaAS 功能部件 `jaxws-2.2` 支持 JAX-WS 编程模型。

定制 Web Service 端点

可以使用 ibm-ws-bnd.xml 文件来定制应用程序中的服务器提供程序和客户机的 Web Service 端点。

ibm-ws-bnd.xml 文件必须位于基于 Web 的 Web Service 应用程序（WAR 文件）的 /WEB-INF 目录中，或基于 EJB 的 Web Service 应用程序（JAR 文件）的 /META-INF 目录中。如果客户机正在运行 `clientContainer`，那么 ibm-ws-bnd.xml 文件必须在客户机 EAR 的 JAR 文件的 /META-INF 目录中。

通过配置 ibm-ws-bnd.xml 文件，您可以覆盖服务提供程序和使用者的 Web Service 地址配置，使用部署在 xigemaAS 概要文件上的应用程序中的特定 URL 来导出 Web Service，然后从 JAX-WS 客户机应用程序导入这些 Web Service。

对于您可以在 ibm-ws-bnd.xml 文件中配置的所有可用元素，请参阅 *ibm-ws-bnd.xml 文件*（见第 1538 页）。

1. 覆盖服务提供程序的 Web Service 绑定。
   a. 覆盖 Web Service 端点地址。

      在 `webservice-endpoint` 元素中，您可以覆盖由 port-component-name 属性标识的 Web Service 端点的端点地址。address 属性指定应用程序的上下文根的相对路径。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
 <!-- 0 to many endpoint descriptions -->
  <webservice-endpoint port-component-name="Hello" address="/hiService" />
</webservices-bnd>
```

   b. 覆盖基于 EJB 的 Web Service 的上下文根。

      如果 Web Service 是基于 EJB 的 Web Service，并且是在 EJB 应用程序（JAR 归档）中进行定义，那么缺省上下文根为 EJB JAR 名称。但是，您可以使用 `http-publishing` 元素的 context-root 属性来覆盖缺省值。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
 <!-- optional http publishing module overrides -->
  <http-publishing context-root="/HiServer" />
</webservices-bnd>
```

      注：仅当 Web Service 是基于 EJB 的 Web Service，并且是在 EJB JAR 归档中进行定义时，context-root 属性才有效。如果 Web Service 是在 Web 应用程序（WAR 归档）中进行定义，那么无论它是否为基于 EJB 的 Web Service，都会忽略 `http-publishing` 元素的 context-root 属性，并且上下文根始终将为该 Web 应用程序的上下文根。

2. 覆盖客户机应用程序的 Web Service 绑定。
   a. 覆盖 WSDL 文档所在的位置。

在 service-ref 元素中，您可以覆盖 Web Service 客户机应用程序中的 Web Service 引用的 WSDL 位置。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
  <service-ref name="services/hello" wsdl-location="http://localhost:9080/HiServer/
hiService?wsdl" />
</webservices-bnd>
```

wsdl-location 属性指定绝对 URI（使用 HTTP 或文件协议），并且它还可以是指向该客户机应用程序的根目录的相对 URI。例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
  <service-ref name="services/hello" wsdl-location="WEB-INF/wsdl/hiService.wsdl" />
</webservices-bnd>
```

b. 覆盖端口地址。

在 port 元素中，您可以覆盖 Web Service 客户机中的 Web Service 引用的端口地址。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://www.vsettan.com.cn/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
  <service-ref name="services/hello" wsdl-location="WEB-INF/wsdl/hiService.wsdl">
    <port name="HelloPort" namespace="http://server.ejb.hello.sample.jaxws.ws.ibm.com/"
        address="http://localhost:9080/HiServer/hiService" />
 </service-ref>
</webservices-bnd>
```

port 元素的 address 属性指定 Web Service 端口的绝对 URI。如果所引用的 Web Service 只有一个端口，那么您可以使用 service-ref 元素的 port-address 属性。例如：

```
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.vsettan.com.cn/xml/ns/javaee http://www.vsettan.com.cn/
xml/ns/javaee/ibm-ws-bnd_1_0.xsd"
   version="1.0">
  <service-ref name="services/hello"
      port-address="http://localhost:9080/HiServer/hiService" />
</webservices-bnd>
```

👉 注：如果同时指定了 service-ref 元素的 port-address 属性以及 port 元素的 address 属性，那么 port 元素的 address 属性生效。

通过策略附件定义 Web Service 策略

在 xigemaAS 中，可以在 Web Service 应用程序的策略附件或 Web 服务描述语言 (WSDL) 文件中定义 Web Service 策略 (WS-Policy)。

WSDL 文件未打包在应用程序中时，可使用策略附件，为 Web Service 应用程序配置 WS-Policy。通常，在启用 Java API for XML Web Services (JAX-WS) 2.2 功能部件时，会在 Web Service 应用程序软件包中自动启用策略附件功能部件。仅需要配置 Web Service 客户机和提供者端以定义应用程序的 WS-Policy 附件文件。

WS-Policy 支持是下列规范在应用程序服务器中的实现。

- WS-Policy 是一种规范，在此规范中，Web Service 可使用 XML 声明其有关安全性和服务质量的策略，以及声明供 Web Service 使用者指定其策略需求的策略。 有关更多信息，请参阅 *Web Services Policy 1.5- 框架*。
- WS-Policy 附件使用 Web Services Policy 框架中定义的策略适用于的主体集，定义用于关联这些策略的两种通用机制。有关更多信息，请参阅 *Web Services Policy 1.2 - 附件*。

1. 创建 policy-attachments-client.xml 文件或 policy-attachments-server.xml 文件。

   以下示例向您说明如何在 policy-attachments-client.xml 文件或 policy-attachments-server.xml 文件中定义属性：

```
<attachments
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:sp13="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802"
xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<wsp:PolicyAttachment wsdlNamespace="http://tempuri.org/AreaService/">
 <wsp:AppliesTo>
    <wsp:URI>http://tempuri.org/AreaService/#wsdl11.service(AreaService)</
wsp:URI>
 </wsp:AppliesTo>
 <wsp:Policy wsu:Id="UsernameTokenwithPasswordHashoverSSL">
    <wsp:ExactlyOne>
        <wsp:All>
     <sp:SupportingTokens>
   <wsp:Policy>
            <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/
ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
         <wsp:Policy>
      <sp:WssUsernameToken10 />
      <sp:HashPassword />
         </wsp:Policy>
      </sp:UsernameToken>
   </wsp:Policy>
     </sp:SupportingTokens>
  </wsp:All>
    </wsp:ExactlyOne>
 </wsp:Policy>
</wsp:PolicyAttachment>
</attachments>
```

   可通过两种方式来绑定策略：

- 按如下所示通过使用端点 DomainExpression <wsa:EndpointReference> 直接定义 Web Service URL。

```
<wsp:AppliesTo>
   <wsa:EndpointReference>
       <wsa:Address>http://localhost:8091/wsatApp/HelloImplService</
wsa:Address>
   </wsa:EndpointReference>
</wsp:AppliesTo>
```

- 按如下所示通过使用 URI DomainExpression <wsp:URI> 绑定策略。

```
<wsp:AppliesTo>
    <wsp:URI>http://server.test.ws.ibm.com/
#wsdl11.service(HelloImplService)</wsp:URI>
</wsp:AppliesTo>
```

下表显示策略附件文件中 WSDL 元素及其标识表达式：

表 42: <*wsp:URI*> 表达式列表

WSDL 元素	标识表达式（忽略 **targetNamespace**）
Definitions	wsdl11.definitions()
Message	wsdl11.message(message)
Message/part	wsdl11.messagePart(message / part)
portType	wsdl11.portType(portType)
portType/operation	wsdl11.portTypeOperation(portType/operation)
portType/operation/input	wsdl11.portTypeOperation.input(portType/operation)
portType/operation/output	wsdl11.portTypeOperation.output(portType/operation)
portType/operation/fault	wsdl11.portTypeOperation.fault(portType/operation/fault)
Binding	wsdl11.binding(binding)
Binding/operation	wsdl11.bindingOperation(binding/operation)
Binding/operation/input	wsdl11.bindingOperation.input(binding/operation)
Binding/operation/output	wsdl11.bindingOperation.output(binding/operation)
Binding/operation/fault	wsdl11.bindingOperation.fault(binding/operation/fault)
Service	wsdl11.service(service)
port	wsdl11.port(service/port)

2. 将 policy-attachments-client.xml 文件或 policy-attachments-server.xml 文件放入应用程序的 WEB-INF 或 META-INF 文件夹。

- 将 policy-attachments-client.xml 放入应用程序的 WEB-INF 或 META-INF 文件夹（对于 Web Service 客户机的 WS-Policy）。
- 将 policy-attachments-server.xml 放入应用程序的 WEB-INF 或 META-INF 文件夹（对于 Web Service 提供程序的 WS-Policy）。

您已完成通过策略附件定义 WS-Policy，且未在应用程序中打包 WSDL。

启用 HTTP 同步管道客户机属性和用户定制属性

可通过对 xigemaAS 上的 JAX-WS 应用程序使用 ibm-ws-bnd.xml 文件来定义 HTTP 客户机属性和用户定制属性。

ibm-ws-bnd.xml 文件必须位于基于 Web 的 Web Service 应用程序（WAR 文件）的 /WEB-INF 目录中，或基于 EJB 的 Web Service 应用程序（JAR 文件）的 /META-INF 目录中。如果客户机正在运行 `clientContainer`，那么 ibm-ws-bnd.xml 文件必须在客户机 EAR 的 JAR 文件的 /META-INF 目录中。

可以使用 ibm-ws-bnd.xml 文件中的 `service-ref` 和 `port` 元素来定义由 `@WebServiceRef` 注解插入的特定服务客户机或端口的 HTTP 客户端属性和用户定制属性。`port` 元素中的属性将覆盖 `service-ref` 元素中的相同属性。

HTTP 客户机属性

xigemaAS 概要文件支持下列 HTTP 客户机属性；这些属性必须具有前缀 http.conduit.client.。例如：http.conduit.client.ConnectionTimeout。仅当发送或接收 SOAP 消息时，这些 HTTP 客户机属性才有效，并且它们不适用于连接至 WSDL URL。

- ConnectionTimeout
- ReceiveTimeout
- AsyncExecuteTimeout
- AsyncExecuteTimeoutRejection
- AutoRedirect
- MaxRetransmits
- AllowChunking
- ChunkingThreshold
- Connection
- DecoupledEndpoint
- ProxyServer
- ProxyServerPort
- ProxyServerType
- NonProxyHosts

有关这些属性的更多信息，请参阅 *Apache CXF 中的 HTTP 配置模式*和*客户机 HTTP 传输 (包括 SSL 支持)*。

用户定制属性

除了 xigemaAS 中受支持的 HTTP 客户端属性以外，您还可以定义可能在应用程序中使用的用户定制属性，并从客户机请求上下文中检索这些属性。在 `properties` 元素中定义的所有属性都将放入服务客户机请求上下文。

对于您可以在 ibm-ws-bnd.xml 文件中配置的所有可用元素，请参阅 *ibm-ws-bnd.xml 文件*（见第 1538 页）。

1. 配置 HTTP 同步管道属性。

下列示例说明了如何在 ibm-ws-bnd.xml 文件中配置 HTTP 客户端属性 ConnectionTimeout 和 ReceiveTimeout。以下示例说明如何对使用 `@WebServiceRef(name="service/SimpleEchoService")` 插入的服务客户机的所有端口应用 HTTP 客户端属性 ConnectionTimeout 和 ReceiveTimeout；对于 `SimpleEchoPort` 端口，还会应用用户定制属性 vendor 的值。

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/
ns/javaee/ibm-ws-bnd_1_0.xsd"
    version="1.0">
 <service-ref name="service/SimpleEchoService">
  <port name="SimpleEchoPort">
   <properties vendor="xigemaAS" />
  </port>
  <properties http.conduit.client.ConnectionTimeout="10000"
 http.conduit.client.ReceiveTimeout="15000" />
 </service-ref>
 ...
</webservices-bnd>
```

2. 然后按如下所示从 `@WebServiceRef(name="service/SimpleEchoService")` 注解插入的客户机的请求上下文中检索 ibm-ws-bnd.xml 文件中定义的属性。

```
@WebServiceRef(name="service/SimpleEchoService")
private EchoService echoService;
   ...
Echo echo = echoService.getEchoPort();
BindingProvider bp = (BindingProvider)echo;
String connTimeout =
 bp.getRequestContext().get("http.conduit.client.ConnectionTimeout").toString();
String recTimeout =
 bp.getRequestContext().get("http.conduit.client.ReceiveTimeout").toString();
String vendor = bp.getRequestContext().get("vendor").toString();
  ...
```

在 JAX-WS Web Service 中强制遵循 WSDL 绑定

在 WSDL 文档中添加绑定扩展，然后将扩展应用到 xigemaAS 概要文件上具有 JAX-WS 支持的应用程序。

xigemaAS 支持使用 JAX-WS 2.2 规范中所需的 `wsdl:binding` 扩展。本主题描述如何定义 WSDL 扩展，并使这些扩展与 xigemaAS 概要文件中的定制可扩展性元素相绑定。

1. 定义可扩展性元素并将其映射至 WSDL 扩展中定义的用户。

   以下示例显示如何与 WSDL 扩展 `{http://server.respectbinding.jaxws22/}goodBinding` 绑定。

```
@XmlRootElement(name = "goodBinding", namespace = "http://server.respectbinding.jaxws22/")
public class GoodBindingElement implements ExtensibilityElement, Serializable {
    private String uri;
    private QName elementType = null;
    private Boolean required = null;

    @XmlAttribute(name = "uri")
    public String getUri() {
        return this.uri;
    }

    @Override
    public QName getElementType() {
        return this.elementType;
    }

    @XmlAttribute(name = "required", namespace="http://schemas.xmlsoap.org/wsdl/")
    public Boolean getRequired() {
        return this.required;
    }
```

```
    public void setUri(String uri){
        this.uri = uri;
    }

    @Override
    public void setElementType(QName elementType) {
        this.elementType = elementType;
    }

    @Override
    public void setRequired(Boolean required) {
        this.required = required;
    }
}
```

2. 将 extensions.xml 文件添加到 Web 或 EJB 应用程序的 /META-INF 目录。此文件定义 WSDL 文档中可以使用扩展的位置。

   在以下示例中，extensions.xml 文件中添加了一个用来将 javax.wsdl.Binding 属性设置为 `goodBindingElement` 类的父类型的条目。该条目也意味着扩展 `{http://server.respectbinding.jaxws22/}goodBinding` 只能添加到 WSDL 文档中的 `binding` 元素。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>
    <!-- XML Binding -->
    <entry
 key="org.apache.cxf.bindings.xml-1">javax.wsdl.Binding=jaxws22.respectbinding.server.common.GoodBindingEle
entry>
</properties>
```

3. 将扩展 `{http://server.respectbinding.jaxws22/}goodBinding` 添加到 `binding` 元素下方，如下所示：

```
...
  <binding name="EchoPortBinding" type="tns:Echo">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <tns:goodBinding wsdl:required="true" uri="http://good/good" xmlns:wsdl="http://
schemas.xmlsoap.org/wsdl/" />
    <operation name="echo">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
      <fault name="Exception">
        <soap:fault name="Exception" use="literal"/>
      </fault>
    </operation>
  </binding>
  ...
```

JAX-WS 客户机应用程序的实现

您可使用 xigemaAS 中的 Java™ API for XML-Based Web Services (JAX-WS) 客户端编程模型支持。

实现静态 **JAX-WS Web Service** 客户机

可以根据 Web Services for Java™ Platform, Enterprise Edition (Java™ EE) 规范和 Java™ API for XML-Based Web Services (JAX-WS) 编程模型来开发静态 Web Service 客户机。

实现动态 **JAX-WS Web Service** 客户机

可以根据 Web Services for Java™ Platform, Enterprise Edition (Java™ EE) 规范和 Java™ API for XML-Based Web Services (JAX-WS) 编程模型来开发动态 Web Service 客户机。

运行非受管 **Web Service JAX-WS** 客户机

可以将带有 xigemaAS 的 JAX-WS 瘦客户机用作纯粹的 Java™ SE 环境或 OSGi 环境内的独立客户机运行时。如果您正在对 xigemaAS 运行具有非 IBM JDK 的瘦客户机，请在 Java™ 启动命令行上添加 `-Dcom.ibm.websphere.thinclient=true` 属性。

运行受管 **Web Service JAX-WS** 客户机

可以使用 `@WebServiceRef` 或 `@Resource` 注解来插入 JAX-WS 服务或端点的实例，也可以使用 Java™ 命名和目录接口 (JNDI) 查找功能来执行服务查找操作。

👉 注：下列各节以及任何子主题都与 xigemaAS 无关：

- 设置 Web Service 的开发环境
- xigemaAS 缺省目录
- 将 Web Service 应用程序部署到应用程序服务器上

有关 `jaxws-2.2` 功能部件的更多信息，请参阅 *xigemaAS* 功能部件（见第 906 页）。

JAX-WS Web Service 应用程序的实现

您可使用 xigemaAS 概要文件中的 Java™ API for XML-Based Web Services (JAX-WS) 服务器端编程模型支持。

使用 **JAX-WS** 来实现 **Web Service** 应用程序

从现有的 JavaBeans™ 或企业 Bean 着手时，您可以根据 Java™ API for XML-Based Web Service (JAX-WS) 编程模型使用自底向上方法来开发 Web Service。

使用 **JAX-WS** 通过现有 **WSDL** 文件来实现 **Web Service** 应用程序

以现有 Web 服务描述语言 (WSDL) 文件开始时，可以使用根据 Java™ API for XML-Based Web Services (JAX-WS) 编程模型来开发 Web Service 的自顶向下方法。

👉 注：下列各节以及任何子主题都与 xigemaAS 无关：

- 设置 Web Service 的开发环境
- xigemaAS 缺省目录
- 将 Web Service 应用程序部署到应用程序服务器上

有关 `jaxws-2.2` 功能部件的更多信息，请参阅 *xigemaAS* 功能部件（见第 906 页）。

ibm-ws-bnd.xml 文件

您可使用 xigemaAS 概要文件中的 ibm-ws-bnd.xml 文件来定制 Web Service 端点以及为 Web Service 提供程序和 Web Service 客户机配置安全设置。必须将此文件存储在 Web 应用程序的 /WEB-INF 目录中，或者存储在 EJB 模块的 /META-INF 目录中。

ibm-ws-bnd.xml 文件提供了以下元素层次结构：

- `<webservices-bnd>`

- *<webservice-endpoint-properties>*
- *<webservice-endpoint>*
  - *<properties>*
- *<http-publishing>*
  - *<webservice-security>*
- *<service-ref>*
  - *<properties>*
  - *<port>*
    - *<properties>*

对于 ibm-ws-bnd.xml 文件中每个元素的描述为如下所示：

<webservice-endpoint-properties>

此元素用来定义同一模块中的所有 Web Service 端点的缺省属性。为一个模块只能指定一个 webservice-endpoint-properties 元素。可以通过 webservice-endpoint-properties 元素的属性来指定适合于 Web Service 端点的任何“名称/值”对。

<webservice-endpoint>

此元素用来指定所指定服务实例的绑定。它有以下属性：

- port-component-name：此属性是必需属性。此属性用来指定端口组件的名称。
- address：此属性是可选属性。此属性用来指定服务端点的被覆盖地址。

<http-publishing>

对所有 Web Service 端点使用 HTTP 协议时，此元素用来指定发布配置。每个模块中只能指定一个 <http-publishing> 元素。它具有以下属性：

- context-root：此属性是可选属性。此属性用来指定基于 EJB 的 JAX-WS 应用程序的 EJB 模块的上下文根。

<webservice-security>

此元素用来为 POJO Web Service 及基于 EJB 的 Web Service 配置基于角色的授权。它有以下属性：

- security-constraint：此属性是可选属性。此属性用来使安全性约束与一个或多个 Web 资源集合相关联，并且仅用作 Web 应用程序中部署描述符或注解的补充配置。
- security-role：此属性是可选属性。此属性包含安全角色的定义，并且仅用作 Web 应用程序中部署描述符或注解的补充配置。
- login-config：此属性是可选属性。此属性用来配置认证方法和域名，并且仅对 JAR 文件中基于 EJB 的 Web Service 起作用。如果在部署描述符文件中指定了同一属性，那么将使用该部署描述符中的值。此属性的值必须为 *BASIC* 或 *CLIENT_CERT*。

<service-ref>

此元素用来定义 Web Service 客户机的 Web Service 引用配置。它有以下属性：

- name：此属性是必需属性。此属性用来指定 Web Service 引用的名称。

- component-name：此属性是可选属性。如果在 EJB 模块中使用了服务引用，那么此属性用来指定 EJB Bean 名称。
- port-address：此属性是可选属性。如果所引用的 Web Service 只有一个端口，那么此属性用来指定 Web Service 端口的地址。
- wsdl-location：此属性是可选属性。此属性用来指定要覆盖的 WSDL 的 URL。

<port>

此元素用来定义与 Web Service 引用相关联的端口配置。可以在 `service-ref` 元素中定义多个 `port` 元素。它有以下属性：

- name：此属性是必需属性。此属性用来指定 Web Service 端口的名称。
- namespace：此属性是可选属性。此属性用来指定 Web Service 端口的命名空间。如果存在 namespace 属性，那么会将绑定应用于具有相同名称和命名空间的端口。否则，会将绑定应用于具有相同名称的端口。
- address：此属性是可选属性。此属性用来指定 Web Service 端口的地址。如果存在此属性，那么它将覆盖在 `service-ref` 元素中定义的 port-address 属性的值。
- username：此属性是可选属性。此属性用来指定用于基本认证的用户名。
- password：此属性是可选属性。此属性用来指定用于基本认证的密码。可以对密码进行编码。
- ssl-ref：此属性是可选属性。它使用 id 属性来引用在 server.xml 文件中配置的 `ssl` 元素。如果未指定此属性，但是服务器通过启用 `ssl-1.0` 功能部件而支持传输级别安全性，那么服务客户机将使用 xigemaAS 概要文件的缺省 SSL 配置。
- key-alias：此属性是可选属性。此属性用来指定客户机证书的别名。如果未指定此属性，并且 Web Service 提供程序支持客户机证书，那么密钥库中的第一个证书将用作此属性的值。此属性还可以覆盖在 server.xml 文件的 `ssl` 元素中定义的 clientKeyAlias 属性。

<properties>

此元素用来定义 Web Service 端点或客户机的属性。属性可以是任意名称和任意值。

- 对于 Web Service 客户机，您可以指定 client.ConnectionTimeout、authorization.UserName 和 tlsClientParameters.disableCNcheck 以及其他属性。
- 对于 Web Service 端点，您可以指定 publishedEndpointUrl、org.apache.cxf.wsdl.create.imports 和其他属性。

以下示例说明了 ibm-ws-bnd.xml 文件的典型配置：

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/
javaee/ibm-ws-bnd_1_0.xsd" version="1.0">

    <!-- optional Web service endpoint default properties -->
    <webservice-endpoint-properties org.apache.cxf.wsdl.create.imports="true" ... />

    <!-- 0 to many endpoint descriptions -->
    <webservice-endpoint port-component-name="..." address="optional override address">
        <!-- optional Web service endpoint properties -->
        <properties publishedEndpointUrl="http://www.example.com/services/hello"
        userPassword="security" />
    </webservice-endpoint>
    ...

    <!-- optional http publishing module overrides -->
```

```
    <http-publishing context-root="optional override of module's context root">
        <!-- optional web.xml fragment for security -->
        <webservice-security>
            <!-- SECURITY CONSTRAINTS -->
            <security-constraint>
                <web-resource-collection>
                    <web-resource-name>SayHelloServiceName</web-resource-name>
                    <url-pattern>/SayHelloService</url-pattern>
                </web-resource-collection>
                <auth-constraint>
                    <description>SayHelloServiceAuth</description>
                    <role-name>group123</role-name>
                </auth-constraint>
                <user-data-constraint>
                    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
                </user-data-constraint>
            </security-constraint>
            <!-- SECURITY CONSTRAINTS -->
            <security-constraint>
                <web-resource-collection>
                    <web-resource-name>SecuredSayHelloServiceName</web-resource-name>
                    <url-pattern>/Secured*</url-pattern>
                </web-resource-collection>
                <auth-constraint>
                    <description>SecuredSayHelloServiceAuth</description>
                    <role-name>group123</role-name>
                </auth-constraint>
                <user-data-constraint>
                    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
                </user-data-constraint>
            </security-constraint>
            <!-- SECURITY ROLES -->
            <security-role id="group123_id">
                <role-name>group123</role-name>
            </security-role>
            <!-- AUTHENTICATION METHOD: Client certificate or basic authentication -->
            <login-config>
                <auth-method>BASIC</auth-method>
            </login-config>
        </webservice-security>
    </http-publishing>

    <!-- 0 to many client refs -->
    <service-ref name="services/sayHelloService" port-address="http://productServer:productPort/
context-root/SayHelloService"/>
    <service-ref name="services/securedSayHelloService" port-address="http://
productServer:productPort/context-root/SecuredSayHelloService" wsdl-location="optional override of
 WSDL document" />
</webservices-bnd>
```

Web Service 命令

xigemaAS 提供一些命令实用程序以使用 Java™ Architecture for XML Binding (JAXB) 和 Java™ API for XML Web Services (JAX-WS) 应用程序。

JAXB 应用程序的 schemagen 命令

借助使用 `schemagen` 模式生成器工具来创建 XML 模式，通过 Java™ 类生成模式文件。在 XML 模式与 Java™ 类之间创建映射后，通过使用 JAXB 绑定运行时 API，可在 XML 实例文档与 Java™ 对象之间相互转换。生成的 Java™ 类包含 JAXB 运行时解析 XML 以进行编组和取消编组所需的所有必需信息。可在 JAX-WS 应用程序内使用 JAXB 类，或在非 JAX-WS Java™ 应用程序中使用 JAXB 类以处理 XML 数据。

JAXB 应用程序的 xjc 命令

借助使用 JAXB 模式编译器提供的 `xjc` 命令行工具，可通过 XML 模式文件生成完全注解的 Java™ 类。使用 `xjc` 模式编译器工具从 XML 模式定义 (XSD) 开始创建一组 JavaBeans™（映射到在 XML 模式中定义的元素和类型）。在 XML 模式与 Java™ 类之间创建映射后，通过使用 JAXB 绑定运行时 API，可在 XML 实例文档

与 Java™ 对象之间相互转换。生成的带注解 Java™ 类包含 JAXB 运行时解析 XML 以进行编组和取消编组所需的所有必要信息。可在 JAX-WS 应用程序内使用生成的 JAXB 类，或在非 JAX-WS Java™ 应用程序中使用生成的 JAXB 类以处理 XML 数据。

用于 JAX-WS 应用程序的 wsgen 命令

从用于 Web Service 编程的 Java™ 代码着手时，`wsgen` 命令行工具生成 JAX-WS 应用程序所需的必要工件。从服务端点实现着手时，使用 `wsgen` 工具生成必要的 JAX-WS 工件。

用于 JAX-WS 应用程序的 wsimport 命令

`wsimport` 命令行工具处理现有 Web 服务描述语言 (WSDL) 文件并生成用于开发 JAX-WS Web Service 应用程序的必要工件。从现有 WSDL 文件着手时，使用 `wsimport` 命令行工具生成必要的 JAX-WS 工件。

注意：如果要更改生成的类的包名称，请在 `wsimport` 命令行中包含 `-p <Expected_Java_Package_Name>`。之后，请不要手动更改 Java™ 包名称。

xigemaAS 中的 Web Service 原子事务

Web Service 原子事务 (WS-AT) 是一项 OASIS 标准。现在，xigemaAS 中支持该功能部件。在本节中，您可以学习如何在 xigemaAS 中启用和配置 WS-AT 功能部件。

Web Service 原子事务概述

应用程序服务器中的 Web Service 原子事务 (WS-AT) 支持为 Web Service 环境提供事务服务质量。分布式 Web Service 应用程序以及它们使用的资源可以参与分布式全局事务。现在，xigemaAS 中也支持 WS-AT。

Web Service 协议提供了定义 Web Service 应用程序的标准方法，不论使用哪种产品、平台或编程语言，应用程序都可以使用它来操作。WS-AT 支持是下列规范在应用程序服务器中的实现。这些规范定义了一组 Web Service，它们使 Web Service 应用程序可以参与分布在异构 Web Service 环境中的全局事务。

- WS-AT 是为原子事务定义协议的特定协调类型。xigemaAS 只能使用 *Web Service 原子事务 V1.2* 规范。
- Web Services Coordination (WS-COOR) 指定 CoordinationContext 和 Registration 服务，参与者 Web Service 可以征调它们来参与特定协调类型提供的协议。xigemaAS 只能使用 *Web Services Coordination V1.2* 规范。

WS-AT 支持是互操作性协议，它没有为事务支持引入新的编程接口。全局事务定界由标准企业应用程序使用 Java™ 事务 API (JTA) UserTransaction 接口提供。如果在全局事务下运行的应用程序组件发出 Web Service 请求，那么 WS-AT CoordinationContext 将隐式传播到目标 Web Service，但是仅当如有关配置事务部署属性的主题中所述设置了相应的应用程序部署描述符时，才会进行这样的传播，

如果应用程序服务器是为包含 WS-AT Coordination 上下文的 Web Service 请求托管目标端点的系统，那么该应用程序服务器将在成为事务上下文（目标 Web Service 应用程序将在该上下文中运行）的目标运行时环境中自动建立下级 JTA 事务。

下图显示包含 WS-AT CoordinationContext 的 Web Service 请求在两个应用程序服务器之间共享的事务上下文。

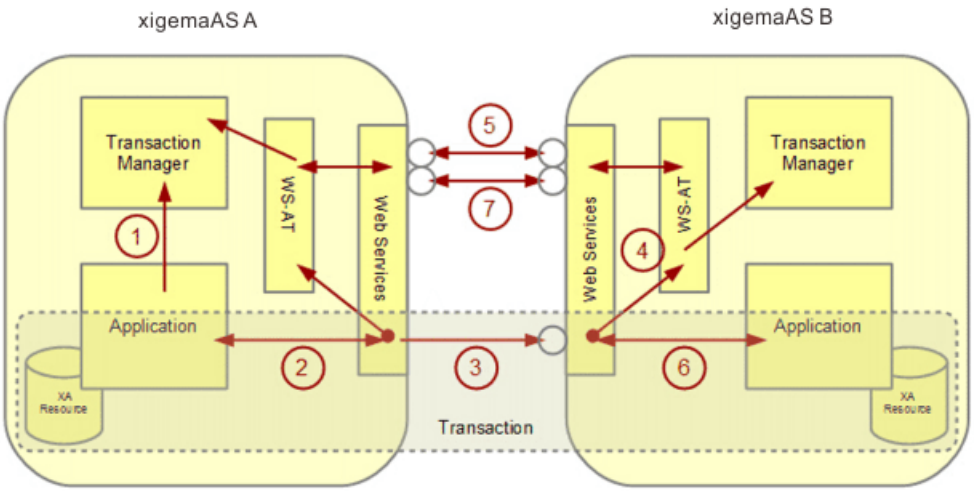


图 35: 在两个应用程序服务器之间共享的事务上下文

缺省情况下，如果您启用 WS-AT 功能并在 xigemaAS 中的客户端开始事务，那么客户机事务中包含的所有 Web Service 操作都将会添加到全局事务。

表 43: 用于确定是否能够在 xigemaAS 中将 WS-AT 全局事务用于出站客户机的条件

下表列出在 xigemaAS 中为出站客户机启用或禁用 WS-AT 全局事务所需的所有条件。如果 WS-AT 功能已禁用，那么不论事务是否存在，或者 WSDL 中是否存在策略断言，可能都无法启用 WS-AT 全局事务。但是，如果 WS-AT 功能已启用，那么只有在事务存在时，才可以启用 WS-AT 全局事务。

WS-AT 功能是否已启用	事务是否存在	**WSDL** 中是否存在策略断言	**WS-AT** 全局事务是否已启用
否	否	否	否
否	否	是	否，同时 xigemaAS 抛出异常
否	是	否	否
否	是	是	否，同时 xigemaAS 抛出异常
是	否	否	否
是	否	是 (wsp: Optional="false")	否，同时 xigemaAS 抛出异常
		是 (wsp: Optional="true")	否
是	是	否	是
是	是	是	是

表 44: 用于确定是否能够在 xigemaAS 中将 WS-AT 全局事务用于入站服务器的条件

下表列出在 xigemaAS 中为入站服务器启用或禁用 WS-AT 全局事务所需的所有条件。如果 WS-AT 功能已禁用，那么不论 SOAP 头中是否存在协调上下文，或者 WSDL 中是否存在策略断言，可能都无法启用 WS-AT 全局事务。但是，如果 WS-AT 功能已启用，那么只有在 SOAP 头中存在协调上下文时，才可以启用 WS-AT 全局事务。

WS-AT 功能是否已启用	**SOAP** 头中是否存在协调上下文	**WSDL** 中是否存在策略断言	**WS-AT** 全局事务是否已启用
否	否	否	否
否	否	是	否
否	是	否	否，同时 xigemaAS 抛出异常
否	是	是	否，同时 xigemaAS 抛出异常
是	否	否	否
是	否	是 (wsp: Optional="false")	否，同时 xigemaAS 抛出异常
		是 (wsp: Optional="true")	否
是	是	否	是
是	是	是	是

应用程序设计

WS-AT 是两阶段落实事务协议，并且仅适合持续时间短的事务。

WS-AT 最适合将事务上下文分布在部署于单个企业内的 Web Service 之间。只有请求/响应消息交换模式才会传递事务上下文，这是因为事务的发起方（应用程序或容器）在请求完成事务之前必须确保该事务中运行的所有业务任务均已完成。单向请求所调用的 Web Service 从来不会在请求客户机的事务中运行。

服务故障对 WS-AT 事务的影响类似于 Enterprise JavaBeans™ (EJB) 应用程序异常对事务的影响，如 EJB 规范所述。即使请求者 WS-AT 事务中运行的服务返回了故障，应用程序服务器也不会自动将该事务标记为“仅回滚”。请求者的异常处理程序选择该事务能否继续进行并选择是否将该事务标记为“仅回滚”。如果请求者在应用程序服务器中运行，那么可以使用标准 JTA 或 EJB API 将该事务标记为“仅回滚”。生成故障的服务组件在返回故障前可以自行将该事务标记为“仅回滚”。如果服务组件的实现遇到系统异常，那么它通常会允许它的容器处理该异常。应用程序服务器容器在处理由服务实现生成的系统异常时，会自动将接收到的所有事务上下文标记为“仅回滚”。

配置 Web Service 原子事务

Web Service 原子事务 (WS-AT) 是一项 OASIS 标准。现在，xigemaAS 中支持此功能部件。

必须先启用 Web Service 原子事务 (WS-AT)，然后才能在 xigemaAS 中对其进行配置。此任务描述如何在 xigemaAS 中启用 WS-AT 功能部件，并且还指示如何向 WS-AT 功能部件添加不同的配置。

如果未在 xigemaAS 中添加 WS-AT 功能，但应用程序中有任何以下信息，那么将显示异常消息。该消息表示 xigemaAS 日志中未安装 WS-AT 功能部件：

- WSDL 中的 WS-Transaction 策略信息
- Web Service SOAP 头 中的 Web Service 原子事务协调上下文信息

要在 xigemaAS 中启用和配置 WS-AT 功能部件，请执行以下操作：

1. 要在 xigemaAS 中启用 WS-AT 功能部件，请在 server.xml 文件中的 featureManager 元素内添加以下条目：

```
<feature>wsAtomicTransaction-1.2</feature>
```

2. 可选： 为 WS-AT 功能部件添加一些特殊配置。

   可以按如下所示为 WS-AT 内部 2PC 协议通信配置 SSL 或代理端点：

```
<wsAtomicTransaction SSLEnabled="false" SSLRef="defaultSSLConfig"
 externalURLPrefix="" clientAuth="false"/>
```

1. 有关更多信息，请参阅在 *server.xml* 文件中配置 *SSL* 或代理端点参数。
3. 可选： 要在当前全局事务的作用域以外运行事务，请向客户机端添加以下代码：

```
UOWManager uowManager = (UOWManager) ctx
    .lookup("java:comp/websphere/UOWManager");
uowManager
    .runUnderUOW(
        UOWSynchronizationRegistry.UOW_TYPE_LOCAL_TRANSACTION,
  false, new UOWAction() {
          public void run() throws Exception {
      ...
      // Example code
      callService1(service1, wsTransactionEnd1);
      ...
    }
   });
```

Web Service 原子事务安全性

Web Service 原子事务 (WS-AT) 定义自己的内部 Web Service，这些 Web Service 用作该协议的一部分。这些内部 Web Service 将在事务启动期间以及落实或回滚处理期间调用。同时，WS-AT 需要双向通信。WS-AT 需要 DMZ 代理以使 xigemaAS 服务器可在典型防火墙环境中运行，还需要 HTTPS 端点以获取授权。以下内容描述如何在 xigemaAS 中为 WS-AT 配置代理和安全套接字层 (SSL)。

防火墙代理

当 xigemaAS 服务器位于典型防火墙环境中时，这些服务器无法通过防火墙与彼此通信。在这种情况下，服务器需要 DMZ 代理才能正常运行。可以通过配置 server.xml 文件来定义代理端点。

例如，将 IBM HTTP Server (IHS) 设置为 DMZ 代理。在图 *36: 通过 IHS 在两个 xigemaAS 服务器之间通信*（见第 1546 页） 中，要通过 IHS 在两个 xigemaAS 服务器之间通信，请执行以下步骤：

1. 配置 IHS 插件以将请求转发至 xigemaAS 服务器。
   - 对于 IHS 1，配置 IHS 1 插件以将请求转发至 xigemaAS A，并在 `<UriGroup>` 项中添加 `<Uri Name="/xigemaas/wsatservice/*" />`。

- 对于 IHS 2，配置 IHS 2 插件以将请求转发至 xigemaAS B，并在 `<UriGroup>` 项中添加 `<Uri Name="/xigemaas/wsatservice/*" />`。

2. 将 `externalURLPrefix` 参数添加到 xigemaAS 中的 server.xml 文件：

   - 在 xigemaAS A 中，对于 IHS 1，将 `externalURLPrefix` 值设置为 `http://proxyserver1:80`，如下所示：

     ```
     <wsAtomicTransaction externalURLPrefix="http://proxyserver1:80"/>
     ```

   - 在 xigemaAS B 中，对于 IHS 2，将 `externalURLPrefix` 值设置为 `http://proxyserver2:81`，如下所示：

     ```
     <wsAtomicTransaction externalURLPrefix="http://proxyserver2:81"/>
     ```

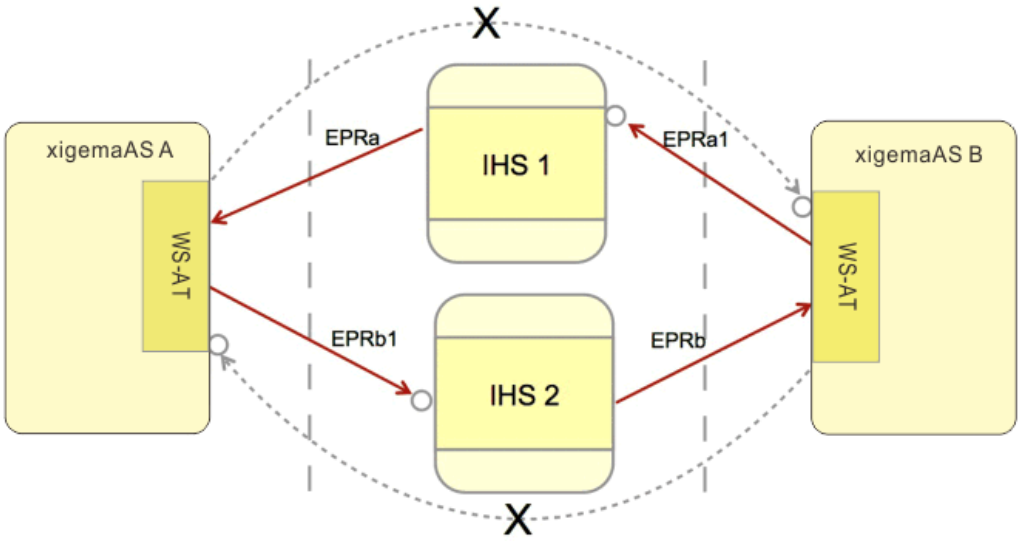


图 36: 通过 IHS 在两个 xigemaAS 服务器之间通信

安全套接字层

缺省情况下，xigemaAS 服务器不使用 SSL 通信。如果需要 WS-AT 安全性并要求将安全性信息重定向到安全端口，那么必须将 `appSecurity-2.0` xigemaAS 功能添加到 server.xml 文件。

可以根据以下项为 WS-AT 安全性配置 SSL：

```
<wsAtomicTransaction SSLEnabled="false" SSLRef="defaultSSLConfig"
 clientAuth="false"/>
```

其中：

SSLEnabled

指定是否为 WS-AT 启用 SSL。有效值为 true 和 false。缺省值为 false。设置为 true 可启用 SSL。

SSLRef

定义 server.xml 文件中的 SSL 配置。WS-AT 需要此 SSL 配置才能通信。

clientAuth

指定是否为 WS-AT 启用 clientAuthentication。有效值为 true 和 false。缺省值为 false。在将 `clientAuth` 值设置为 true 以启用 clientAuthentication 之前，请确保在 SSL 配置中将 `clientAuthenticationSupported` 值设置为 true。

👉 重要：SSL 配置仅在 xigemaAS 服务器级别适用。

👉 提示：在以下代码片段中，缺省 `SSLEnabled` 值为 false。如果要同时启用 SSL 和代理，请将 `SSLEnabled` 值设置为 true，并将 `externalURLPrefix` 值设置为以 `https://` 开头的代理地址。

```
<wsAtomicTransaction SSLEnabled="false" SSLRef="defaultSSLConfig"
 externalURLPrefix="" clientAuth="false"/>
```

Web Service 原子事务互操作

xigemaAS 支持与 xigemaAS 经典版进行 WS-AT 互操作。

在 xigemaAS 经典版中，可以使用策略集控制事务策略。

2.8.10 将消息传递应用程序部署到 xigemaAS

部署在 xigemaAS 中的 JMS 应用程序可以利用内部嵌入式消息传递提供程序或者外部 xigemaMQ 作为其 JMS 消息传递提供程序。

部署 JMS 应用程序以连接至嵌入式消息传递服务器

要部署使用 Java™ 消息传递服务 (JMS) 的消息传递应用程序，您必须将 `wasJmsServer-1.0` 和 `wasJmsClient-2.0` 功能部件添加至 server.xml 文件，然后定义连接工厂和目的地属性。如果要执行 JNDI 查找，那么必须将 `jndi-1.0` 功能部件与其他两个功能部件一起添加。

确保您已创建要在其中部署使用 JMS 的消息传递应用程序的 xigemaAS服务器。有关更多信息，请参阅手动创建应用服务器（见第 1101 页）。

👉 重要：`wasJmsClient-2.0` 功能部件支持 JMS 1.1 和 JMS 2.0 规范的功能部件。但是，如果只想使用符合 JMS 1.1 规范的功能部件，那么可选择使用 `wasJmsClient-1.1` 功能部件。

`wasJmsServer-1.0` 功能部件支持使用 Java™ 消息传递服务 1.1 规范的应用程序。

1. 将 `wasJmsServer-1.0`、`wasJmsClient-2.0` 和 `jndi-1.0` 功能部件添加到 server.xml 文件。

```
<featureManager>
    <feature>wasJmsServer-1.0</feature>
    <feature>wasJmsClient-2.0</feature>
    <feature>jndi-1.0</feature>
</featureManager>
```

2. 将目标定义添加至 server.xml 文件。

```
<messagingEngine>
      <queue id="QUEUE1"> </queue>
</messagingEngine>
```

3. 可选： 添加 <wasJmsEndpoint> 元素，以使 JMS 消息传递引擎能够接受来自 TCP/IP 的远程入局消息传递连接（使用 SSL 和不使用 SSL）。

```
<wasJmsEndpoint id="InboundJmsEndpoint"
```

```
    host="*"
    wasJmsPort="7276"
    wasJmsSSLPort="9101">
</wasJmsEndpoint>
```

注：是否添加 <wasJmsEndpoint> 是可选的。缺省情况下，xigemaAS 使消息传递引擎能够侦听 7276 端口（不安全）和 7286（安全）。如果要指定另一端口，那么可以配置 <wasJmsEndpoint>。

4. 将连接工厂定义添加至 server.xml 文件。

   - 对于点到点域：

```
<jmsQueueConnectionFactory jndiName="jndi_JMS_BASE_QCF">
    <properties.wasJms
    remoteServerAddress="localhost:7276:BootStrapBasicMessaging" />
</jmsQueueConnectionFactory>

<jmsQueue jndiName="jndi_INPUT_Q">
    <properties.wasJms queueName="QUEUE1" />
</jmsQueue>
```

   - 对于发布/预订域：

```
<jmsTopicConnectionFactory jndiName="eis/tcf">
    <properties.wasJms
    clientID="defaultID" />
</jmsTopicConnectionFactory>

<jmsTopic jndiName="eis/topic1">
    <properties.wasJms topicName="Football" />
</jmsTopic>
```

   现在，已将 JMS 应用程序连接至嵌入式消息传递服务器。

将 JMS 应用程序部署到 xigemaAS 以使用 xigemaAS 消息传递提供程序

通过 xigemaAS 中的 xigemaMQ 消息传递提供程序，Java™ 消息服务 (JMS) 消息传递应用程序可以使用 xigemaMQ 系统作为 JMS 消息传递资源的外部提供程序。

必须确保您已创建要在其中部署使用 JMS 的消息传递应用程序的 xigemaAS 服务器。有关更多信息，请参阅手动创建应用服务器（见第 1101 页）。

`wmqJmsClient-2.0` 功能部件提供对使用 JMS 2.0 规范的应用程序的支持。

注：`wmqJmsClient-2.0` 功能部件支持 JMS 1.1 和 JMS 2.0 规范的功能部件。但是，如果只想使用符合 JMS 1.1 规范的功能部件，那么可选择使用 `wmqJmsClient-1.1` 功能部件。

1. 将 `wmqJmsClient-2.0` 功能部件添加至 server.xml 文件。如果要执行 JNDI 查找，那么还必须添加 `jndi-1.0` 功能部件。

```
<featureManager>
     <feature>wmqJmsClient-2.0</feature>
     <feature>jndi-1.0</feature>
</featureManager>
```

   如果添加 `wmqJmsClient-2.0` 功能部件，那么将使 xigemaAS 服务器能够装入必需的 xigemaMQ 捆绑软件，从而可让您定义 xigemaMQ JMS 资源。例如，连接工厂和激活规范属性会提供客户机库以连接至 xigemaMQ 网络。

2. 可以通过在 server.xml 文件中添加以下项目来指定 xigemaMQ 资源适配器的位置：

```
<variable name="wmqJmsClient.rar.location" value="/path/to/wmq/rar/
wmq.jmsra.rar"/>
```

其中，value 属性指定 xigemaMQ 资源适配器文件的绝对路径 wmq.jmsra.rar。

有关受支持的版本和获取 wmq.jmsra.rar 文件的详细信息，请参阅 xigemaMQ 技术说明 1633761。

3. 将连接工厂定义添加至 server.xml 文件。

```
<jmsConnectionFactory jndiName="jms/wmqCF" connectionManagerRef="ConMgr6">
     <properties.wmqJms
        transportType="CLIENT"
        hostName="localhost"
        port="1414"
        channel="SYSTEM.DEF.SVRCONN"
        queueManager="QM1"/>
</jmsConnectionFactory>
<connectionManager id="ConMgr6" maxPoolSize="2"/>

<jmsQueue id="jms/queue1" jndiName="jms/wmqQ1">
     <properties.wmqJms
        baseQueueName="MDBQ"
        baseQueueManagerName="QM1"/>
    </jmsQueue>
```

4. 配置 JMS 应用程序以采用“绑定”方式进行连接。

为允许 JMS 应用程序使用共享内存或者采用“绑定”方式连接至 xigemaMQ，必须将 xigemaAS 和 xigemaMQ 部署在同一个服务器上。要允许 JMS 应用程序以“绑定”方式进行连接，请使用 server.xml 文件中的 nativeLibraryPath 元素来指定 xigemaMQ 本机库的位置。

```
 <wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
```

在 xigemaAS 应用程序客户机容器上部署 JMS 应用程序

可配置使用 wasJmsClient-2.0 功能部件的消息传递应用程序以在 xigemaAS 应用程序客户机容器上运行。

必须先创建客户机并在 client.xml 文件中为客户机应用程序添加配置，才能运行应用程序客户机。有关如何创建应用程序客户机的示例，请参阅手动创建 *xigemaAS* 应用程序客户机（见第 1110 页）主题。

1. 要在该客户机上运行消息传递应用程序，请将 javaeeClient-7.0 功能部件添加至 client.xml 文件。

```
<featureManager>
    <feature>javaeeClient-7.0</feature>
</featureManager>
```

2. 将连接工厂定义添加至 client.xml 文件。

   - 对于点到点域：

```
<jmsQueueConnectionFactory jndiName="jndi_JMS_BASE_QCF" connectionManagerRef="ConMgr6">
      <properties.wasJms remoteServerAddress="localhost:7276:BootstrapBasicMessaging" />
</jmsQueueConnectionFactory>
<connectionManager id="ConMgr6" maxPoolSize="10"/>

<jmsQueue jndiName="jndi_INPUT_Q">
    <properties.wasJms queueName="QUEUE1" />
</jmsQueue>
```

- 对于发布/预订域：

```
<jmsTopicConnectionFactory jndiName="eis/tcf">
    <properties.wasJms
    clientID="defaultID" />
</jmsTopicConnectionFactory>

<jmsTopic jndiName="eis/topic1">
    <properties.wasJms topicName="Football" />
</jmsTopic>
```

该 JMS 应用程序已准备好在 xigemaAS 应用程序客户机容器上运行。

启用 JMS 跟踪以进行 xigemaAS 嵌入式消息传递

可在 xigemaAS 中使用 JMS 跟踪以确定问题和进行故障诊断。

设置 JMS 跟踪

可在 server.xml 文件中设置以下跟踪字符串以获取必需信息。

连接问题

使用以下跟踪字符串以收集有关连接问题的信息。

```
SIBTrm=all
```

通信和 **TCP/IP**

使用以下跟踪字符串以收集有关应用程序服务器通道框架和 TCP/IP 网络通信的信息。

```
SIBCommunications=all:SIBJFapChannel=all:TCPChannel=fine:com.ibm.io.async.*=all
```

JMS 客户机应用程序

使用以下跟踪字符串以收集有关来自 JMS 应用程序的请求的信息。

```
SIBJms*=all:SIBCommunications=all:SIBJFapChannel=all:
SIBMessageTrace=all:SIBTrm=all:SIBJmsRa=all:SIBRa=all
```

锁定的消息

如果要交付至 MDB 和应用程序的消息的锁定时间超过预期，请使用以下跟踪字符串。

```
SIBProcessor=all:SIBMessageTrace=all
```

消息驱动的 **Bean**

使用以下跟踪字符串以收集有关使用激活规范针对消息目标配置的 MDB 的信息。

```
SIBMessageTrace=all:SIBJmsRa=all:SIBRa=all
```

消息格式和模式

如果解析或处理消息数据时存在问题，请使用以下跟踪字符串。

```
SIBMfp=all:SIBCommunications=all
```

消息处理器

使用以下跟踪字符串以收集有关消息传递引擎的核心功能的信息。收集此数据很有帮助，但数据规模可能很大。

```
SIBProcessor=all:SIBMessageTrace=all
```

消息存储

使用以下跟踪字符串以收集有关持久消息数据的信息，这些数据被写至存储器以便在发生故障时可以恢复。

```
SIBMessageStore=all
```

性能和消息跟踪

使用以下跟踪字符串以确定消息传递延迟或是否需要查找消息的去处。

```
SIBMessageTrace=all
```

发布/预订

如果订户未获得其主题的相应发布，请使用以下跟踪字符串。

```
SIBMatchSpace=all:SIBProcessor=all
```

安全性

如果未正确认证用户或未正确允许或禁止其对资源的访问，请使用以下跟踪字符串。

```
SIBSecurity=all
```

在 xigemaAS 中部署消息驱动 Bean

xigemaAS 支持配置和部署消息驱动 Bean (MDB)，以从各种目标异步处理 JMS 消息。

部署消息驱动 Bean 以连接至 xigemaMQ

可以使用消息驱动 Bean (MDB) 来连接至 xigemaMQ。

1. 配置 MDB 功能部件：

   通过在 server.xml 文件中配置 `jmsMdb-3.2` 和 `wmqJmsClient-2.0` 功能部件，在 xigemaAS 服务器中启用 MDB 支持。如果要执行 JNDI 查找，那么还必须将 `jndi-1.0` 功能部件与其他两个功能部件一起添加。

要在 xigemaAS 中启用 MDB 支持，请在 server.xml 文件中配置 `jmsMdb-3.2` 和 `wmqJmsClient-2.0` 功能部件。如果要执行 JNDI 查找，那么还必须将 `jndi-1.0` 功能部件与其他两个功能部件一起添加。

```
<featureManager>
     <feature>jmsMdb-3.2</feature>
     <feature>wmqJmsClient-2.0</feature>
     <feature>jndi-1.0</feature>
</featureManager>
```

配置 `wmqJmsClient-2.0` 功能部件使用户能够定义必需的 JMS 资源，并且使 MDB 能够与消息传递引擎进行交互。

👉 注：`wmqJmsClient-2.0` 功能部件支持 JMS 1.1 和 JMS 2.0 规范的功能部件。但是，如果只想使用符合 JMS 1.1 规范的功能部件，那么可选择使用 `wmqJmsClient-1.1` 功能部件。

2. 可以通过在 server.xml 文件中添加以下项目来指定 xigemaMQ 资源适配器的位置：

```
<variable name="wmqJmsClient.rar.location" value="/path/to/wmq/rar/
wmq.jmsra.rar"/>
```

其中，`value` 属性指定 xigemaMQ 资源适配器文件的绝对路径 wmq.jmsra.rar。

3. 配置使用 JMS 资源适配器的 JCA 激活规范，以便 MDB 充当特定 JMS 目标的侦听器。下面采用不同的方法来配置 MDB 以与消息传递引擎进行交互。

   1. 使用激活规范属性来配置 MDB。

      可以在 server.xml 文件中定义此属性，以便 MDB 可以使用此属性来侦听特定 JMS 目标。

      ```
      <jmsActivationSpec id="JMSSample/JMSSampleMDB">
           <properties.wmqJms destinationRef="jndi/MDBQ" transportType="CLIENT" queueManager="myQM"
       hostName="myHost" port="1414"/>
      </jmsActivationSpec>

      <jmsQueue id="jndi/MDBQ" jndiName="jndi/MDBQ">
           <properties.wmqJms baseQueueName="MYQ" baseQueueManagerName="myQM"/>
      </jmsQueue>
      ```

      <destinationRef> 是指 <jmsQueue> 的标识。如果未提到 <jmsQueue> 中的标识，那么 <destinationRef> 必须指向 <jmsQueue> 的 <jndiName>。

      如果 resourceAdapterID 为非 wasJms 和 wmqJms 时，那么 <destinationRef> 指向的必须是队列名。

      ```
      <jmsActivationSpec id="JMSSample/JMSSampleMDB">
       <properties.myJms destinationRef="Q1" transportType="CLIENT" queueManager="myQM" hostName="myHost"
       port="1414"/>
      </jmsActivationSpec>
      <jmsQueue id="Q1" jndiName="jndi/MDBQ">
       <properties.myJms queueName="Q1"/>
      </jmsQueue>
      ```

      👉 注：标识值的格式必须为 `application name/module name/bean name`，其中：

      - *myJms* 为 resourceAdapterID。
      - *application name* 是已部署的应用程序的名称（例如，JMSSample）。仅当将 Bean 打包在 EAR 文件中时，应用程序名称才适用。应用程序名称缺省设置为 EAR 文件的基本名称，不带文件扩展名，除非 application.xml 部署描述符指定了应用程序名称。
      - *module name* 是用来打包该 Bean 的模块的名称。在独立 ejb-jar 文件或者 WAR 文件中，<module-name> 缺省设置为模块的基本名称，移除了所有文件扩展名。在 WAR 文件中，<module-name> 缺省设置为模块的路径名，并且移除了所有文件扩展名，但是包括了所

有目录名称。可以使用 ejb-jar.xml（对于 ejb-jar 文件）或者 web.xml（对于 WAR 文件）的 module-name 元素来覆盖缺省 <module-name>。

- *bean name* 是企业 Bean 的 ejb-name。对于通过注解定义的企业 Bean，Bean 名称缺省设置为会话 Bean 类的非限定名称，除非在 `MessageDriven` 注解的 name() 属性的内容中指定。对于通过 ejb-jar.xml 定义的企业 Bean，在 <ejb-name> 部署描述符元素中指定了 Bean 名称。

2. 使用注解以及 server.xml 文件中定义的激活规范属性。

xigemaAS 概要文件支持为 MDB 定义注解，可以将注解与 server.xml 文件中定义的激活规范属性配合使用。为了使用注解，请如前一步骤中提到的那样首先定义激活规范属性。对于每个 MDB，用户可以如以下示例中所示定义注解：

```
@MessageDriven(
        name = "JMSSampleMDB",
        activationConfig = {
           @ActivationConfigProperty(propertyName  = "destinationType", propertyValue =
 "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName  = "userName", propertyValue = "user1"),
    @ActivationConfigProperty(propertyName  = "password",propertyValue = "user1pwd"),
    @ActivationConfigProperty(propertyName  = "destination", propertyValue =
 "jndi_INPUT_Q")
  }
)
public class JMSSampleMDB implements MessageListener{

 @TransactionAttribute(value = TransactionAttributeType.REQUIRED)
 public void onMessage(Message message) {
  }
}
```

3. 使用 EJB 绑定文件 (ibm-ejb-jar-bnd.xml)。

您还可以使用此绑定文件来定义 MDB 在连接至消息传递引擎时所需要的资源信息。当您使用 EJB 绑定文件时，激活规范属性标识不一定需要采用步骤 1 中所提到的 `application name/module name/bean name` 格式。

在 server.xml 中添加激活规范属性信息。

```
<jmsActivationSpec id="PriceChangeAS">
     <properties.wmqJms destinationRef="jms/TriggerQ" transportType="CLIENT"
 queueManager="myQM" hostName="myHost" port="1414"/>
</jmsActivationSpec>

<jmsQueue id="jms/TriggerQ" jndiName="jms/TriggerQ">
     <properties.wmqJms baseQueueName="Q1"/>
 </jmsQueue>
```

在 ibm-ejb-jar-bnd.xml 文件中添加以下 MDB 绑定信息。

```
<ejb-jar-bnd
        xmlns="http://websphere.ibm.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/
javaee/ibm-ejb-jar-bnd_1_1.xsd"
  (http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-bnd_1_1.xsd%27) version="1.1">

    <message-driven name="PriceChangeMDBBean">
        <jca-adapter activation-spec-binding-name="PriceChangeAS" destination-binding-name="jms/
TriggerQ" />
    </message-driven>
</ejb-jar-bnd>
```

👉 注：使用 EJB 绑定文件时，ibm-ejb-jar-bnd.xml 文件中的 activation-spec-binding-name 属性必须指向 server.xml 文件中所指定的激活规范属性标识值。

部署消息驱动的 Bean 以连接至 JCA 资源适配器

可以使用消息驱动的 Bean (MDB) 连接至 JCA 资源适配器。

当您将消息驱动的 Bean (MDB) 应用程序部署到 xigemaAS 时，server.xml 文件中需要某些配置以使每个 MDB 能够从资源适配器接收消息。本主题中描述的示例配置适用于仅包含消息驱动的 Bean 和 JMS 资源适配器的应用程序。如果该应用程序还将使用会话 Bean，那么必须包括 `ejbLite-3.1` 功能部件。

配置属性将随所使用的特定资源适配器不同而不同。

1. 配置 MDB 功能部件：

   要在 xigemaAS 中启用 MDB 支持，请在 server.xml 文件中配置 `mdb-3.1` 和 `jca-1.6` 功能部件。如果要执行 JNDI 查找，那么还必须将 `jndi-1.0` 功能部件与其他两个功能部件一起添加。

   ```
   <featureManager>
        <feature>mdb-3.1</feature>
        <feature>jca-1.6</feature>
        <feature>jndi-1.0</feature>
   </featureManager>
   ```

2. 配置 JCA 资源适配器所需要的任何其他功能部件。
3. 配置使用该资源适配器的 JCA 激活规范，以便 MDB 充当侦听器。下列示例说明您可以采用不同的方式来配置 MDB 以与消息传递引擎进行交互。这些示例特定于 JMS 资源适配器。

   1. 使用激活规范属性来配置 MDB。

      可以在 server.xml 文件中定义此属性，以便 MDB 可以使用此属性来侦听特定 JMS 目标。

      ```
      <activationSpec id="JMSSample/JMSSampleMDB">
           <properties.wmqJms destinationRef="jndi/MDBQ" transportType="CLIENT" queueManager="myQM"
       hostName="myHost" port="1414"/>
      </activationSpec>

      <jmsQueue id=”jndi/MDBQ” jndiName="jndi/MDBQ">
           <properties.wmqJms baseQueueName="MYQ" baseQueueManagerName="myQM"/>
      </jmsQueue>
      ```

      <destinationRef> 是指 <jmsQueue> 的标识。如果未提到 <jmsQueue> 中的标识，那么 <destinationRef> 元素必须指向 <jmsQueue> 的 <jndiName>。

      如果 resourceAdapterID 为非 wasJms 和 wmqJms 时，那么 <destinationRef> 指向的必须是队列名。

      ```
      <jmsActivationSpec id="JMSSample/JMSSampleMDB">
       <properties.myJms destinationRef="Q1" transportType="CLIENT" queueManager="myQM" hostName="myHost"
       port="1414"/>
      </jmsActivationSpec>
      <jmsQueue id="Q1" jndiName="jndi/MDBQ">
       <properties.myJms queueName="Q1"/>
      </jmsQueue>
      ```

      👉 注：标识值的格式必须为 `application name/module name/bean name`，其中：

      - *myJms* 为 resourceAdapterID。
      - *application name* 是已部署的应用程序的名称（例如，JMSSample）。仅当将 Bean 打包在 EAR 文件中时，应用程序名称才适用。应用程序名称缺省设置为 EAR 文件的基本名称，不带文件扩展名，除非 application.xml 部署描述符指定了应用程序名称。
      - *module name* 是用来打包该 Bean 的模块的名称。在独立 ejb-jar 文件或者 WAR 文件中，<module-name> 缺省设置为模块的基本名称，移除了所有文件扩展名。在 WAR 文件中，<module-name> 缺省设置为模块的路径名，并且移除了所有文件扩展名，但是包括了所

有目录名称。可以使用 ejb-jar.xml（对于 ejb-jar 文件）或者 web.xml（对于 WAR 文件）的 module-name 元素来覆盖缺省 <module-name>。

- *bean name* 是企业 Bean 的 ejb-name。对于通过注解定义的企业 Bean，Bean 名称缺省设置为会话 Bean 类的非限定名称，除非在 `MessageDriven` 注解的 name() 属性的内容中指定。对于通过 ejb-jar.xml 定义的企业 Bean，在 <ejb-name> 部署描述符元素中指定了 Bean 名称。

2. 通过将注解与 server.xml 文件中所定义的激活规范属性配合使用来配置 MDB。

   xigemaAS 概要文件支持为消息驱动的 Bean 定义注解，可以将注解与 server.xml 文件中定义的激活规范属性配合使用。要使用注解，请如前一步骤中提到的那样首先定义激活规范属性。对于每个消息驱动的 Bean，您可以如以下示例中所示定义注解：

```
@MessageDriven(
        name = "JMSSampleMDB",
        activationConfig = {
            @ActivationConfigProperty(propertyName  = "destinationType",
                                                                  propertyValue =
 "javax.jms.Queue"),
            @ActivationConfigProperty(propertyName  = "userName",
                                                                  propertyValue = "user1"),
            @ActivationConfigProperty(propertyName  = "password",
                                                                  propertyValue =
 "user1pwd"),
            @ActivationConfigProperty(propertyName  = "destination",
                                                                  propertyValue =
 "jndi_INPUT_Q")
    }
  )
public class JMSSampleMDB implements MessageListener{

    @TransactionAttribute(value = TransactionAttributeType.REQUIRED)
   public void onMessage(Message message) {
  }
}
```

3. 使用 EJB 绑定文件 ibm-ejb-jar-bnd.xml 来配置 MDB。

   您还可以使用此绑定文件来定义 MDB 在连接至消息传递引擎时所需要的资源信息。当您使用 EJB 绑定文件时，激活规范属性标识不是一定需要采用步骤 a 中所提到的 `application name/module name/bean name` 格式。

   在 server.xml 文件中添加激活规范属性信息。

```
<activationSpec id="PriceChangeAS">
     <properties.wmqJms destinationRef="jms/TriggerQ" transportType="CLIENT" queueManager="myQM"
 hostName="myHost" port="1414"/>
</activationSpec>

<jmsQueue id=”jms/TriggerQ” jndiName="jms/TriggerQ">
     <properties.wmqJms baseQueueName="Q1"/>
 </jmsQueue>
```

   在 ibm-ejb-jar-bnd.xml 文件中添加以下 MDB 绑定信息。

```
<ejb-jar-bnd
        xmlns="http://websphere.ibm.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/
javaee/ibm-ejb-jar-bnd_1_1.xsd"
  (http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-bnd_1_1.xsd%27) version="1.1">

  <message-driven name="PriceChangeMDBBean">
          <jca-adapter activation-spec-binding-name="PriceChangeAS" destination-binding-name="jms/
TriggerQ" />
    </message-driven>
</ejb-jar-bnd>
```

👉 注：使用 EJB 绑定文件时，ibm-ejb-jar-bnd.xml 文件中的 activation-spec-binding-name 属性必须指向 server.xml 文件中所指定的激活规范属性标识值。

部署消息驱动的 Bean 以连接至嵌入式消息传递服务器

使用消息驱动 Bean (MDB) 来连接至嵌入式消息传递服务器。

在 xigemaAS 中可以采用以下两种方案来配置 MDB：

- 当 MDB 和消息传递引擎在同一 xigemaAS 服务器上时。
- 当 MDB 连接至正在另一 xigemaAS 服务器上运行的远程消息传递引擎时。

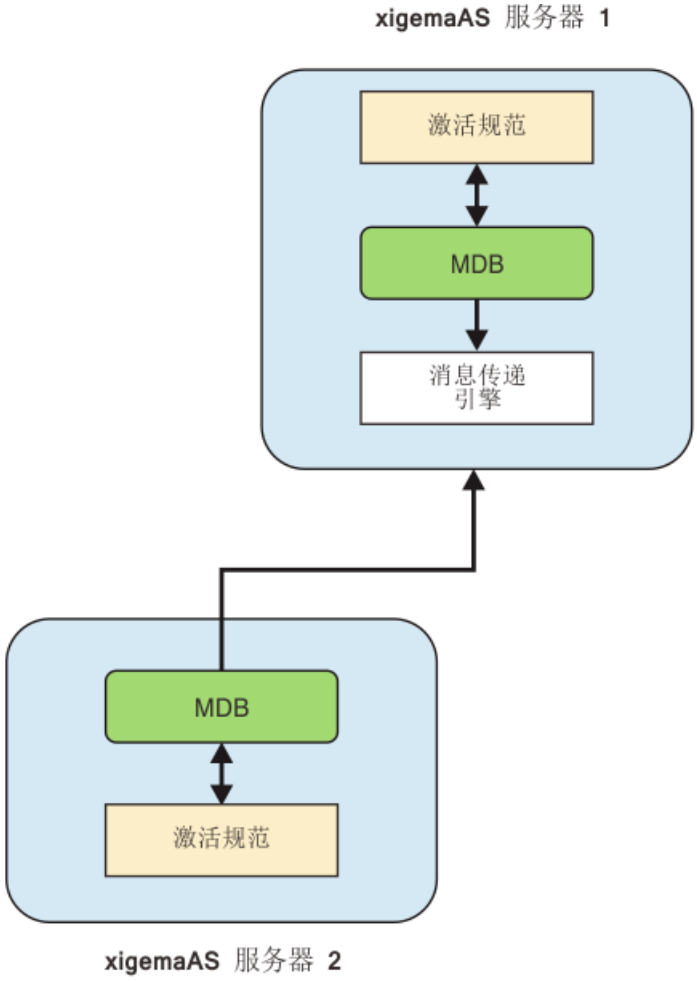


图 37: MDB 的两种配置方案

1. 配置 MDB 功能部件：

   要在 xigemaAS 概要文件中启用 MDB 支持，请在 server.xml 文件中配置 `jmsMdb-3.1` 和 `wasJmsClient-2.0` 功能部件。如果要执行 JNDI 查找，那么还必须将 `jndi-1.0` 功能部件与其他两个功能部件一起添加。

```
<featureManager>
     <feature>jmsMdb-3.1</feature>
     <feature>wasJmsClient-2.0</feature>
     <feature>jndi-1.0</feature>
```

```
</featureManager>
```

配置 wasJmsClient-2.0 功能部件使用户能够定义必需的 JMS 资源，并且使 MDB 能够与消息传递引擎进行交互。

👉 重要：wasJmsClient-2.0 功能部件支持 JMS 1.1 和 JMS 2.0 规范的功能部件。但是，如果只想使用符合 JMS 1.1 规范的功能部件，那么可选择使用 wasJmsClient-1.1 功能部件。

2. 配置使用 JMS 资源适配器的 JCA 激活规范，以便 MDB 充当特定 JMS 目标的侦听器。下面采用不同的方法来配置 MDB 以与消息传递引擎进行交互。

   1. 使用激活规范属性来配置 MDB。

      可以在 server.xml 文件中定义此属性，以便 MDB 可以使用此属性来侦听特定 JMS 目标。

      ```
      <jmsActivationSpec id="JMSSample/JMSSampleMDB">
           <properties.wasJms destinationRef="jndi/MDBQ"  />
      </jmsActivationSpec>

      <jmsQueue id=”jndi/MDBQ” jndiName="jndi/MDBQ">
           <properties.wasJms queueName="Q1"/>
      </jmsQueue>
      ```

      <destinationRef> 是指 <jmsQueue> 的标识。如果未提到 <jmsQueue> 中的标识，那么 <destinationRef> 必须指向 <jmsQueue> 的 <jndiName>。

      如果 resourceAdapterID 为非 wasJms 和 wmqJms 时，那么 <destinationRef> 指向的必须是队列名。

      ```
      <jmsActivationSpec id="JMSSample/JMSSampleMDB">
       <properties.myJms destinationRef="Q1" transportType="CLIENT" queueManager="myQM" hostName="myHost"
       port="1414"/>
      </jmsActivationSpec>
      <jmsQueue id="Q1" jndiName="jndi/MDBQ">
       <properties.myJms queueName="Q1"/>
      </jmsQueue>
      ```

      👉 注：标识值的格式必须为 application name/module name/bean name，其中：

      - *myJms* 为 resourceAdapterID。
      - *application name* 是已部署的应用程序的名称（例如，JMSSample）。仅当将 Bean 打包在 EAR 文件中时，应用程序名称才适用。应用程序名称缺省设置为 EAR 文件的基本名称，不带文件扩展名，除非 application.xml 部署描述符指定了应用程序名称。
      - *module name* 是用来打包该 Bean 的模块的名称。在独立 ejb-jar 文件或者 WAR 文件中，<module-name> 缺省设置为模块的基本名称，移除了所有文件扩展名。在 WAR 文件中，<module-name> 缺省设置为模块的路径名，并且移除了所有文件扩展名，但是包括了所有目录名称。可以使用 ejb-jar.xml（对于 ejb-jar 文件）或者 web.xml（对于 WAR 文件）的 module-name 元素来覆盖缺省 <module-name>。
      - *bean name* 是企业 Bean 的 ejb-name。对于通过注解定义的企业 Bean，Bean 名称缺省设置为会话 Bean 类的非限定名称，除非在 MessageDriven 注解的 name() 属性的内容中指定。对于通过 ejb-jar.xml 定义的企业 Bean，在 <ejb-name> 部署描述符元素中指定了 Bean 名称。

   2. 使用注解以及 server.xml 文件中定义的激活规范属性。

      xigemaAS 概要文件支持为 MDB 定义注解，可以将注解与 server.xml 文件中定义的激活规范属性配合使用。为了使用注解，请如前一步骤中提到的那样首先定义激活规范属性。对于每个 MDB，用户可以如以下示例中所示定义注解：

      ```
      @MessageDriven(
      ```

```
        name = "JMSSampleMDB",
        activationConfig = {
            @ActivationConfigProperty(propertyName  = "destinationType",
                                                                propertyValue =
 "javax.jms.Queue"),
            @ActivationConfigProperty(propertyName  = "userName",
                                                                propertyValue = "user1"),
            @ActivationConfigProperty(propertyName  = "password",
                                                                propertyValue =
 "user1pwd"),
            @ActivationConfigProperty(propertyName  = "destination",
                                                                propertyValue =
 "jndi_INPUT_Q")
    }
  )
public class JMSSampleMDB implements MessageListener{

    @TransactionAttribute(value = TransactionAttributeType.REQUIRED)
      public void onMessage(Message message) {
  }
}
```

3. 使用 EJB 绑定文件 (ibm-ejb-jar-bnd.xml)。

   您还可以使用此绑定文件来定义 MDB 在连接至消息传递引擎时所需要的资源信息。当您使用 EJB 绑定文件时，激活规范属性标识不一定需要采用步骤 1 中所提到的 `application name/module name/bean name` 格式。

   在 server.xml 中添加激活规范属性信息。

```
<jmsActivationSpec id="PriceChangeAS">
     <properties.wasJms destinationRef="jms/TriggerQ" />
</jmsActivationSpec>

<jmsQueue id=”jms/TriggerQ” jndiName="jms/TriggerQ">
     <properties.wasJms queueName="Q1"/>
 </jmsQueue>
```

   在 ibm-ejb-jar-bnd.xml 文件中添加以下 MDB 绑定信息。

```
<ejb-jar-bnd
        xmlns="http://websphere.ibm.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/
javaee/ibm-ejb-jar-bnd_1_1.xsd"
  (http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-bnd_1_1.xsd%27) version="1.1">

  <message-driven name="PriceChangeMDBBean">
          <jca-adapter activation-spec-binding-name="PriceChangeAS" destination-binding-name="jms/
TriggerQ" />
    </message-driven>
</ejb-jar-bnd>
```

   注：使用 EJB 绑定文件时，ibm-ejb-jar-bnd.xml 文件中的 activation-spec-binding-name 属性必须指向 server.xml 文件中所指定的激活规范属性标识值。

配置连接池以进行 JMS 连接

可定制可通过 JMS 连接工厂创建的并行连接数（缺省值为 50）。通过定义 <connectionManager> 元素，可为特定 JMS 连接工厂配置连接池，然后在 server.xml 文件中使该元素与 JMS 连接工厂相关联。

server.xml 文件中的 `connectionManager` 元素用于为 JMS 连接工厂属性定义连接池，如以下示例中所示。连接池大小也是在 `connectionManager` 元素中定义的。

```
<jmsConnectionFactory id="jmscf1"
  jndiName="jndi/example"
```

```
  connectionManagerRef="cm1">
 <properties.wasJms/>
</jmsConnectionFactory>
<connectionManager id="cm1" maxPoolSize="10" minPoolSize="2" />
```

服务器对 connectionManager 元素上未定义的所有连接管理设置使用缺省值。如果未对连接工厂定义连接管理器，那么服务器会对所有设置使用缺省值。

可以定义多个连接工厂，并使每个连接工厂与不同连接管理器相关联。但是，不能使多个连接工厂与单个连接管理器相关联。

2.8.11 为 xigemaAS 部署 Java 批处理应用程序

可开发基于 Java™ 规范请求 (JSR) 352 的 Java™ 批处理应用程序，然后提交 Java™ 批处理作业以在 xigemaAS 服务器上运行。

Java 批处理和受管批处理概述

Java 批处理功能扩展应用程序服务器以满足必须执行批处理工作的应用程序和事务性应用程序。批处理工作可能需要数小时（甚至数天）的时间才能完成并且在运行时使用大量内存或处理能力。

通常由产品托管的 Java Platform Enterprise Edition V7 (Java EE 7) 应用程序执行简短的轻量级事务性工作单元。在大多数情况下，单个请求只需要使用几秒处理器时间和相对较少的内存就可以完成。但是，许多应用程序必须完成需要大量计算和资源的批处理工作。

xigemaAS 概要文件支持以下批处理功能部件：

- Java 批处理

  batch-1.0 功能部件允许使用 JSR-352 编程模型。
- 受管批处理

  batchManagement-1.0 功能部件提供以下功能：

  - 用于远程提交作业的 REST 接口
  - batchManager 命令行实用程序
  - 作业日志记录支持
  - 多服务器支持（通过使用 JMS）

batch-1.0 和 batchManagement-1.0 功能部件支持 Java SE 7 及更高版本。

👉 注：batchManagement-1.0 功能部件还会启用 batch-1.0 功能部件。

为批处理 REST API 配置 xigemaAS

xigemaAS 包含用于管理 Java 批处理作业的 RESTful 管理界面。受管批处理启用安全 HTTPS REST 接口，以便您可在外部管理 Java 批处理作业。

1. 将 batchManagement-1.0 功能部件添加至 server.xml 文件。

```
<featureManager>
 <feature>batchManagement-1.0</feature>
</featureManager>
```

2. 通过配置 Java 批处理功能部件使用的 databaseStore 来配置批处理持久性。使用 jobStoreRef 元素在 server.xml 文件中引用 databaseStore。

   以下示例演示 server.xml 文件的外观。

```
<batchPersistence jobStoreRef="BatchDatabaseStore" />

<databaseStore id="BatchDatabaseStore" dataSourceRef="batchDB" />
```

   有关数据库持久性（包括表的自动创建与手动创建）的更多信息，请参阅 Java 批处理持久性配置。

3. 在 server.xml 文件中创建 SSL 证书和用户注册表，以便 batchManagement-1.0 自动启用 SSL 功能部件。

```
<keyStore id="defaultKeyStore" password="xigemaAS"/>

<basicRegistry id="basic" realm="ibm/api">
 <user name="bob" password="bobpwd" />
 <user name="jane" password="janepwd" />
</basicRegistry>
```

   👉 重要：此示例中的缺省自签名 SSL 证书仅适用于开发，不适用于生产。

   有关配置批处理环境的基于角色的管理及为用户分配角色的信息，请参阅“保护 xigemaAS 批处理环境”。

现在已为 xigemaAS 服务器配置 RESTful 接口。

Java 批处理持久性配置

Java 批处理使用持久性存储以在作业实例的多次运行间保存状态、检查点和应用程序持久数据。持久性存储允许作业实例重新启动（如果之前运行失败或必须停止），方法是为重新启动的作业提供相应数据。

Java 批处理的基于内存的持久性配置

批处理持久性允许作业实例以“失败”或“停止”状态结束后重新启动。如果缺少批处理持久性配置，那么 Java 批处理使用缺省功能（基于内存的持久性）在作业实例的多次运行间跟踪状态、检查点和应用程序持久数据。

如果 server.xml 文件中没有 batchPersistence 和 databaseStore 元素，那么底层批处理容器将对 Java 批处理使用缺省的基于内存的持久性实现。

👉 注：作为对 Java 批处理的基于内存的持久性的限制，缺省情况下，Java 批处理中的持久性基于内存。如果批处理容器运行时或服务器 JVM 崩溃或重新启动，那么持久性将丢失。此功能仅适用于开发用途，不适用于生产系统或关键批处理支持。

Java 批处理的数据库持久性配置

缺省情况下，批处理运行时根据 databaseStore 元素中定义的服务器配置自动创建不存在的表。表定义是根据数据库存储的 schema 和 tablePrefix 属性定制的。

或者，可使用 ddlGen 脚本根据服务器配置生成 DDL。必要时，可在手动创建表前定制该 DDL。此 DDL 还合并了 schema 和 tablePrefix 等服务器配置，并且包含与 databaseStore 所引用数据源的数据库类型相应的 SQL。

注：定制 DDL 必须使用正整数主键标识。作为数据库持久性的限制，Java 批处理不会接受在主键标识列中持久存储的负整数或为零的标识。Java 批处理容器运行时仅运行使用主键标识列中持久存储的正整数作业标识的作业。

可对 databaseStore 使用 createTables="false" 属性来禁止自动创建表。此选项可用于确保批处理运行时意外地找不到您手动创建的表时您使用手动创建的表而不是自动创建的表。

此页面上显示的样本使用缺省的自动创建行为。此行为相当于 createTables="true"。

Java 批处理的基于内存的持久性配置

批处理持久性允许作业实例以“失败”或“停止”状态结束后重新启动。如果缺少批处理持久性配置，那么 Java 批处理使用缺省功能（基于内存的持久性）在作业实例的多次运行间跟踪状态、检查点和应用程序持久数据。

如果 server.xml 文件中没有 `batchPersistence` 和 `databaseStore` 元素，那么底层批处理容器将对 Java 批处理使用缺省的基于内存的持久性实现。

注：作为对 Java 批处理的基于内存的持久性的限制，缺省情况下，Java 批处理中的持久性基于内存。如果批处理容器运行时或服务器 JVM 崩溃或重新启动，那么持久性将丢失。此功能仅适用于开发用途，不适用于生产系统或关键批处理支持。

Java 批处理的数据库持久性配置

缺省情况下，批处理运行时根据 databaseStore 元素中定义的服务器配置自动创建不存在的表。表定义是根据数据库存储的 schema 和 tablePrefix 属性定制的。

或者，可使用 ddlGen 脚本根据服务器配置生成 DDL。必要时，可在手动创建表前定制该 DDL。此 DDL 还合并了 schema 和 tablePrefix 等服务器配置，并且包含与 databaseStore 所引用数据源的数据库类型相应的 SQL。

可对 databaseStore 使用 createTables="false" 属性来禁止自动创建表。此选项可用于确保批处理运行时意外地找不到您手动创建的表时使用手动创建的表而不是自动创建的表。

此页面上显示的样本使用缺省的自动创建行为。此行为相当于 createTables="true"。

注：要避免因隔离级别低于 REPEATABLE_READ 而导致的数据完整性问题，请将数据源的隔离级别设置为 TRANSACTION_REPEATABLE_READ。如果不指定隔离级别，那么缺省值取决于数据库。在大多数情况下，缺省值为 TRANSACTION_REPEATABLE_READ。

持久性配置样本

以下样本配置对 Derby 的自动创建的目标数据库表 RUNTIMEDB 的批处理访问。

```
<!-- Batch persistence config.  References a databaseStore. -->
   <batchPersistence jobStoreRef="BatchDatabaseStore" />

   <!-- The database store for the batch tables.  -->
   <!-- Note this database store is referenced by the batchPersistence
element. -->
   <databaseStore id="BatchDatabaseStore" dataSourceRef="batchDB"
schema="JBATCH" tablePrefix="" />

   <!-- Derby JDBC driver -->
   <!-- Note this library is referenced by the dataSource element -->
```

```
    <library id="DerbyLib">
        <fileset dir="${server.config.dir}/resources/derby" />
    </library>

    <!-- Data source for the batch tables. -->
    <!-- Note this data source is referenced by databaseStore element -->
    <dataSource id="batchDB" isolationLevel="TRANSACTION_REPEATABLE_READ" >
        <jdbcDriver libraryRef="DerbyLib" />
        <properties.derby.embedded
            databaseName="${server.config.dir}/resources/RUNTIMEDB"
            createDatabase="create"
            user="user"
            password="pass" />
    </dataSource>
```

保护 xigemaAS 批处理环境

xigemaAS 批处理框架允许您配置基于角色的对所有批处理管理操作的访问权及查看与批处理作业相关联的元数据和日志。

批处理容器定义了三个批处理角色。单个用户可以具有多个批处理角色。

batchAdmin

batchAdmin 具有对所有批处理操作的非受限访问权。这包括以下许可权：提交新作业、停止并重新启动任何用户的作业、查看批处理域中的任何用户提交的任何作业元数据和作业日志以及清除任何作业。batchAdmin 不一定是 xigemaAS 管理员。

batchSubmitter

batchSubmitter 有权提交新作业，但只能对他们自己的作业执行停止、停止重新或清除之类的批处理操作。batchSubmitter 不能查看或修改其他用户的作业。例如，如果 user1 和 user2 都定义为 batchSubmitter，并且 user1 提交了作业，那么 user2 不能查看与 user1 的作业相关联的作业实例数据。

batchMonitor

batchMonitor 具有对所有作业的只读许可权。此角色的用户可以查看所有作业实例和执行，并有权访问所有作业日志。batchMonitor 不能提交自己的作业，也不能停止、重新启动或清除任何作业。

👉 注：启用批处理安全性后，系统将根据授予当前用户的批处理角色来过滤返回列表的任何 JobOperator API 方法或 REST 操作。例如，如果一个仅具有 batchSubmitter 许可权的用户请求所有作业实例的列表，那么系统仅返回当前用户提交的作业实例。

1. 缺省情况下，`batch-1.0` 功能部件不会启用任何安全性。对于所有用户（不管他们是否经过认证），JobOperator 方法保持开放状态。这些方法保持开放状态只是用于开发目的，不需要任何安全配置。

   `batchManagement-1.0` 功能部件启用批处理 REST API。该 REST API 始终要求用户进行认证，即使未启用 `appSecurity-2.0` 功能部件也是如此，但所有用户将被视为批处理管理员并可对任何作业实例执行所有批处理操作。启用 `appSecurity-2.0` 后，系统将执行基于批处理角色的安全授权，用户被限制为只能执行他们被给定的批处理角色所定义的批处理操作。

   a. 通过 JobOperator API 启用基于批处理角色的安全性

```
<featureManager>
 <feature>batch-1.0</feature>
 <feature>appSecurity-2.0</feature>
```

```
</featureManager>
```

b. 通过 REST API 启用基于批处理角色的安全性。

👉 注：batchManagement-1.0 功能部件包含 batch-1.0 功能部件。

```
<featureManager>
 <feature>batchManagement-1.0</feature>
 <feature>appSecurity-2.0</feature>
</featureManager>
```

2. 配置 server.xml 文件以支持基于角色的安全性。

以下示例说明用于定义用户列表的基本用户注册表。此注册表将由基于批处理角色的安全配置样本使用。

```
<basicRegistry id="basic" realm="ibm/api">
 <user name="alice" password="alicepwd" />
 <user name="bob" password="bobpwd" />
 <user name="jane" password="janepwd" />
 <user name="joe" password="joepwd" />
 <user name="phyllis" password="phyllispwd" />
 <user name="kai" password="kaipwd" />
</basicRegistry>
```

在此示例中，一个用户占用多个角色。

```
<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
  <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
  <user name="jane" />
  <user name="phyllis" />
  <user name="bob"/>
 </security-role>
 <security-role name="batchMonitor" >
  <user name="joe" />
  <user name="bob"/>
 </security-role>
</authorization-roles>
```

在此示例中，一个用户占用多个角色。所有用户具有 batchSubmitter 角色。

```
<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
  <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
  <special-subject type="ALL_AUTHENTICATED_USERS"/>
 </security-role>
 <security-role name="batchMonitor" >
  <user name="joe" />
  <user name="bob"/>
 </security-role>
</authorization-roles>
```

在此示例中，一个用户占用多个角色。所有用户（包括那些未认证用户）被允许具有 batchMonitor 角色。

```
<authorization-roles id="com.ibm.ws.batch">
 <security-role name="batchAdmin" >
  <user name="alice" />
 </security-role>
 <security-role name="batchSubmitter" >
  <user name="joe" />
  <user name="bob"/>
 </security-role>
 <security-role name="batchMonitor" >
  <special-subject type="EVERYONE"/>
 </security-role>
</authorization-roles>
```

Java 批处理关闭和恢复

如果服务器关闭时作业仍在运行，那么 Java 批处理功能部件的行为方式有所不同。

取消激活 Java 批处理

服务器停止时或 Java 批处理功能部件被移除时，Java 批处理会取消激活。它也会取消激活然后重新激活以处理动态配置更新。取消激活 Java 批处理会导致向所有活动作业发出停止请求并针对每个停止记录消息。停止作业需要 2 秒时间。如果 2 秒后仍有 Java 批处理作业运行，那么系统会记录一条消息，指示哪些作业执行标识仍处于活动状态。然后 Java 批处理关闭。如果 Java 批处理关闭后作业仍在运行，那么它们可能会遇到不可预测的行为。

👉 重要：如果服务器意外停止，那么在服务器重新启动时，Java 批处理通过将服务器上运行但未完成的所有作业标记为“失败”来进行恢复。

批处理 REST API

xigemaAS 包含用于管理批处理作业的 RESTful 管理接口。

与批处理作业相关联的基本操作包括提交（启动）、停止、重新启动和查看状态。可使用任何 HTTP REST 客户机执行这些操作。在请求中提交的或在响应中返回的任何数据为 JSON 格式。

以下示例显示您可以使用 REST API 执行的功能。

👉 注：批处理 REST API 针对每个 URI 确定版本。没有版本号的 URL 被视为 API V1。批处理 REST 接口的 Web 地址全部以根 Web 地址 `https://{host}:{port}/vsettan/api/batch/{version}/` 开头。

👉 切记：批处理 REST 接口的 Web 地址全部以根 Web 地址 `https://{host}:{port}/vsettan/api/batch` 开头。

- 分布式服务器批处理环境中的 *JOBLOGS* 请求（见第 1579 页）
- 分布式服务器批处理环境中的清除请求（见第 1579 页）

当用户标识只是提交者时的 API

当仅向通过 HTTPS 调用 REST API 的用户标识授予了 **batchSubmitter** 角色的访问权时，将对 GET ("read") URL 返回的结果进行过滤。提交者只能看到与提交者本人所提交作业实例相关联的实例和执行数据。与此相反，如果向用户标识授予了 **batchAdmin** 和 **batchMonitor** 角色的访问权，那么这些用户标识将能够查看（由使用任何给定参数集的任何给定 URL 返回的）所有实例和执行数据。

如果用户标识只有 **batchSubmitter** 角色的访问权，那么该用户标识所看到的结果将首先由给定 URL 的文档中所述的标准参数过滤，然后再由“仅返回与提交者本人所提交作业实例相关联的实例和执行数据”过滤。

有关更多信息，请参阅“保护 xigemaAS 批处理环境” «» 。

作业实例

GET /vsettan/api/batch/jobinstances/

此 URI 返回作业实例的列表。查询参数包括：

- page=[page number]：指示要返回的页面（记录子集）。缺省值为 0。
- pageSize=[number of records per page]：指示每页记录数。缺省值为 50。

样本请求：

```
https://localhost:9443/vsettan/api/batch/jobinstances
```

```
https://localhost:9443/vsettan/api/batch/jobinstances?page=13&pagesize=20
```

样本响应：

```
[
    {
        "jobName":"test_sleepyBatchlet",
        "instanceId":7,
        "appName":"SimpleBatchJob#SimpleBatchJob.war",
        "submitter":"bob",
        "batchStatus":"COMPLETED",
        "jobXMLName":"test_sleepyBatchlet",
        "instanceState":"COMPLETED",
        "_links":[
            {
                "rel":"self",
                "href":"https://localhost:9443/vsettan/api/batch/jobinstan
ces/7"
            },
            {
                "rel":"job logs",
                "href":"https://localhost:9443/vsettan/api/batch/jobinstan
ces/7/joblogs"
            }
        ]
    },
    {
        "jobName":"test_sleepyBatchlet",
        "instanceId":6,
        "appName":"SimpleBatchJob#SimpleBatchJob.war",
        "submitter":"bob",
```

```
        "batchStatus":"COMPLETED",
        "jobXMLName":"test_sleepyBatchlet",
        "instanceState":"COMPLETED",
        "_links":[
            {
                "rel":"self",
                "href":"https://localhost:9443/vsettan/api/batch/jobinstan
ces/6"
            },
            {
                "rel":"job logs",
                "href":"https://localhost:9443/vsettan/api/batch/jobinstan
ces/6/joblogs"
            }
        ]
    }
]
```

GET /vsettan/api/batch/v2/jobinstances/

此 URI 返回通过以下查询参数过滤的作业实例的列表：

- jobInstanceId=[instanceId]:[instanceId]：返回等于和包含于 instanceId 范围中的作业实例。
- jobInstanceId=>[instanceId]：返回等于和大于所提供 instanceId 的作业实例。
- jobInstanceId=<[instanceId]：返回等于和小于所提供 instanceId 的作业实例。
- jobinstanceId=[instanceId],[instanceId],[instanceId]：返回所指定作业实例。
- createTime=[yyyy-MM-dd]:[yyy-MM-dd]：返回此日期范围内（包含上限和下限）的作业实例。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=[yyyy-MM-dd]：返回给定日期的作业实例。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=>3d：返回三天之前的日期或之后创建的作业实例。例如，createTime 晚于或等于三天之前的日期开始时间。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=<3d：返回三天之前的日期或之前创建的作业实例。例如，createTime 早于或等于三天之前的日期结束时间。有关更多信息，请参阅关于此主题中内容的进一步注释。
- instanceState=[state],[state]：返回具有所提供状态的作业实例。有效实例状态为：SUBMITTED、JMS_QUEUED、JMS_CONSUMED、DISPATCHED、FAILED、STOPPED、COMPLETED 和 ABANDONED。
- exitStatus=[string]：返回与退出状态字符串相匹配的作业实例。字符串条件可在任一端使用通配符 (*) 运算符。
- page=[page number]：指示要返回的页面（记录子集）。缺省值为 0。
- pageSize=[number of records per page]：指示每页记录数。缺省值为 50。

注：涉及 **createTime** 的查询中的服务器缺省时区的作用

提交作业时，作业实例的 createTime 存储在作业存储库中并规范化为 UTC。如果通过 yyyy-MM-dd 指定日期（例如，在 `createTime=[yyyy-MM-dd]` 或 `createTime=[yyyy-MM-dd]:[yyy-MM-dd]:` 中），那么必须将 yyyy-MM-dd 日期字符串转换为 UTC 时间的特定范围，以与作业实例表记录中的 createTime 值匹配。为此，此应用程序使用处理 REST 请求的服务器的缺省时区。系统使用此服务器的时区将日期字符串转换为要匹配 UTC 时间范围。

下表演示查询参数 jobInstanceId=10:13 返回的数据。

JOBINSTANCEID	CREATETIME（server1 的时区中的 *）	INSTANCESTATE	EXITSTATUS
10	11-05-2015.01:10:00	COMPLETED	SUCCESS
11	11-08-2014.02:20:00	COMPLETED	SUCCESS
12	11-10-2015.03:30:00	FAILED	FAILURE
13	11-11-2015:04:40:00	COMPLETED	SUCCESS

* 因为作业存储库将 createTime 存储为 UTC 格式，所以了解以上表数据显示的 createTime 的格式使用特定服务器（例如，“server1”）的服务器缺省时区很重要。如果已从具有不同缺省时区的另一服务器（非“server1”）的角度构造类似表，那么应显示另一组 CREATETIME 列值，这些值具有对应的基于时区的时差。它是处理 REST 请求的服务器的缺省时区，它用于将 yyyy-MM-dd 日期字符串参数映射至数据库中的 UTC createTime 值（如以下示例中所示）。

以下命令返回相同结果，不管针对哪个服务器发出这些命令都是如此：

- jobInstanceId=11:13 将返回 JOBINSTANCEID 11、12 和 13。
- jobInstanceId=>12 将返回 JOBINSTANCEID 12 和 13。
- jobInstanceId=<12 将返回 JOBINSTANCEID 11 和 12。
- jobInstanceId=11,12 将返回 JOBINSTANCEID 11 和 12。
- instanceState=FAILED 将返回 JOBINSTANCEID 12。
- exitStatus=SUCCESS 将返回 JOBINSTANCEID 10、11 和 13。

对于具有不同缺省时区的服务器，以下命令可能返回不同结果。在这些示例中，它们是针对“server1”（用于填充上表的服务器）在 11-11-2015:07:00:00（使用该服务器的缺省时区）发出的。

- createTime=>2d 将返回 JOBINSTANCEID 12 和 13（在两天前的日期（即 11-09-2015）或之后）
- createTime=<2d 将返回 JOBINSTANCEID 10 和 11（在两天前的日期（即 11-09-2015）或之前）
- createTime=2015-11-08:2015-11-11 将返回 JOBINSTANCEID 11、12 和 13。
- createTime=2015-11-10 将返回 JOBINSTANCEID 为 12 的记录。

样本请求：

```
https://localhost:9443/vsettanapi/batch/v2/jobinstances?instanceId=20:50
https://localhost:9443/vsettan/api/batch/jobinstances?createTime=>2d
https://localhost:9443/vsettan/api/batch/v2/jobinstances?instanceId=4,12,1
7&instanceState=COMPLETED
https://localhost:9443/vsettan/api/batch/v2/jobinstances?instanceId=4500:4
600&createTime=2015-11-27&instanceState=COMPLETED&exitStatus=*JOB*&page=0&
pageSize=1000
```

GET /vsettan/api/batch/jobinstances/*job instance id*

此 URI 返回有关指定作业实例的详细信息，例如，与指定作业实例相关联的所有执行。结果按从最新到最旧的顺序返回。最新结果在列表中显示在最前。

样本响应：

```
{
    "jobName":"test_sleepyBatchlet",
```

```
    "instanceId":7,
    "appName":"SimpleBatchJob#SimpleBatchJob.war",
    "submitter":"bob",
    "batchStatus":"COMPLETED",
    "jobXMLName":"test_sleepyBatchlet",
    "instanceState":"COMPLETED",
    "_links":[
        {
            "rel":"self",
            "href":"https://localhost:9443/vsettan/api/batch/jobinstances/
7"
        },
        {
            "rel":"job logs",
            "href":"https://localhost:9443/vsettan/api/batch/jobinstances/
7/joblogs"
        },
        {
            "rel":"job execution",
            "href":"https://localhost:9443/vsettan/api/batch/jobinstances/
7/jobexecutions/7"
        }
    ]
}
```

POST /vsettan/api/batch/jobinstances/

使用此 URI 提交（启动）新作业。

以下示例说明请求主体，该请求主体以 JSON 格式提交 WAR 模块中打包的作业：

```
{
  "applicationName" : "SimpleBatchJob",
  "moduleName" : "SimpleBatchJob.war",
  "jobXMLName" : "test_batchlet_jsl",
  "jobParameters" : { "prop1" : "prop1value", "prop2" : "prop2value"}
}
```

以下示例说明请求主体，该请求主体以 JSON 格式提交 EJB 模块中打包的作业：

```
{
  "applicationName" : "SimpleBatchJob",
  "moduleName" : "SimpleBatchJobEJB.jar",
  "componentName" : "MyEJB",
  "jobXMLName" : "test_batchlet_jsl",
  "jobParameters" : { "prop1" : "prop1value", "prop2" : "prop2value"}
}
```

applicationName 标识批处理应用程序。它是必需的，除非指定了 *moduleName*（在此情况下，*applicationName* 是从 *moduleName* 派生的，方法是删除 *moduleName* 的 .war 或 .jar 后缀。）例如，如果未提供 *applicationName* 并且 *moduleName*=`SimpleBatchJob.war`，那么 *applicationName* 缺省为 `SimpleBatchJob`。

moduleName 标识批处理应用程序中包含作业工件（例如，JSL）的模块。作业是在该模块的组件上下文中提交的。*moduleName* 是必需的，除非指定了 *applicationName*（在此情况下，*moduleName* 是从 *applicationName* 派生的，方法是在 *applicationName* 后附加 .war）。例如，如果 *applicationName*=`SimpleBatchJob` 并且未提供 *moduleName*，那么 *moduleName* 缺省为 `SimpleBatchJob.war`。

componentName 标识批处理应用程序 EJB 模块内的 EJB 组件。如果指定，那么该作业在 EJB 的组件上下文中提交。

👉 注：仅当模块为 EJB 模块时，才需要 *componentName*。如果模块为 WAR 模块，那么不需要 *componentName*。

必须输入 *jobXMLName* 的值。*jobParameters* 的值是可选的。

作为使用 JSL 作业（在批处理应用程序的 META-INF/batch-jobs 下打包）定义的替代项，可在 REST 作业提交请求中传递 JSL 内嵌项。作为内嵌项提交的 JSL 始终覆盖随批处理应用程序打包的任何 JSL。有两种方法在 HTTP 请求中提交 JSL 内嵌项。

1. 以 JSON 属性形式将 JSL 包含在作业提交请求中。将 jobXML 属性添加至 JSON 对象并将 JSL 文件的完整内容作为该属性的值添加至 JSON 字符串。

   👉 注：必须将 XML 字符串正确转义以便 JSON 解析器可对其进行解析。大部分 JSON 库将完成此任务。

   以下示例演示用于提交作业的请求主体，此作业使用的单部件 HTTP 请求带有 JSON 以内嵌项方式传递的 JSL。

```
{
  "applicationName" : "SimpleBatchJob",
  "jobXMLName":"test_multiPartition_3Steps",
  "jobXML":"<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"yes\"?>
  \r\n<job id=\"test_multiPartition_3Steps\"
  xmlns=\"http://xmlns.jcp.org/xml/ns/javaee\"r\n\tversion=\"1.0>
  \r\n\t<step id=\"step1"\" next=\"step2\">
  \r\n\t\t<batchlet ref=\"simpleBatchlet\"/>
  \r\n\t\t<partition>\r\n\t\t\t<plan  partitions=\"3\"/>
  \r\n\t\t</partition>\r\n\t</step>
  \r\n\t<step  id=\"step2\" next=\"step3\">
  \r\n\t\t<batchlet ref=\"simpleBatchlet\" />
  \r\n\t\t<partition>\r\n\t\t\t<plan  partitions=\"3\" />
  \r\n\t\t</partition>\r\n\t</step>\r\n\t<step  id=\"step3\">
  \r\n\t\t<batchlet ref=\"simpleBatchlet\" />\r\n\t\t<partition>
  \r\n\t\t\t<plan partitions=\"3\"/>
  \r\n\t\t</partition>\r\n\t</step>\r\n</job>"
}
```

   👉 注：jobXML 字段必须由 JSON 解析器解析并编组为有效 JSON 对象。jobXMLName 字段是可选项，因为内嵌 JSL 的作业标识信息用于表示作业名。

2. 使用 HTTP 多重部件表单。如果使用 HTTP 多重部件表单，那么 JSON 作业提交数据和 XML 作业定义需要作为 HTTP 主体的两个独立部分提交。多重部件表单的 JSON 部件必须命名为 *jobdata*，该表单的 XML 部件必须命名为 *jsl*。使用多重部件表单时，XML 不必编组为 JSON 字符串。

   以下示例演示用于提交作业的请求主体，此作业使用的多重部件 HTTP 请求带有通过 jsl 消息部件以内嵌项方式传递的 JSL。

```
Content-Type: multipart/form-data;boundary=--------------------------49
424d5f4a4241544348

-----------------------------49424d5f4a4241544348
Content-Disposition: form-data; name="jobdata"
Content-Type: application/json; charset=UTF-8
{
```

```
  "applicationName" : "SimpleBatchJob",
  "moduleName" : "SimpleBatchJob.war",
  "jobXMLName" : "test_multiPartition"
}

-----------------------------49424d5f4a4241544348
Content-Disposition: form-data; name="jsl"
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<job id="test_multiPartition" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
<step id="step1">
<batchlet ref="simpleBatchlet" />
<partition>
  <plan partitions="3" />
  </partition>
</step>
</job>

-----------------------------49424d5f4a4241544348--
```

👉 注：jobXMLName 字段是可选项，因为内嵌 JSL 的作业标识信息用于表示作业名。

以下样本响应演示成功的作业提交：

```
{
    "jobName": "test_sleepyBatchlet",
    "instanceId": 10,
    "appName": "SimpleBatchJob#SimpleBatchJob.war",
    "submitter": "bob",
    "batchStatus": "STARTING",
    "jobXMLName": "test_sleepyBatchlet",
    "instanceState": "SUBMITTED",
    "_links": [
        {
            "rel": "self",
            "href": "https://localhost:9443/vsettan/api/batch/jobinstances
/10"
        },
        {
            "rel": "job logs",
            "href": "https://localhost:9443/vsettan/api/batch/jobinstances
/10/joblogs"
        }
    ]
}
```

PUT /vsettan/api/batch/jobinstances/*job instance id*?action=stop

使用此 URI 停止与此作业实例相关联的最新作业执行（如果它正在运行）。如果它未在运行，那么 API 返回错误。

PUT /vsettan/api/batch/jobinstances/*job instance id*?action=restart

使用此 URI 重新启动与此作业实例相关联的最新作业执行（仅当它处于“停止”或“失败”状态时）。如果没有与此实例相关联的作业执行，或最新作业执行为“完成”状态，那么 API 将返回错误。

PUT /vsettan/api/batch/jobinstances/*job instance id*?action=restart&reusePreviousParams=true

使用此 URI 重新启动最新作业执行，并复用与此作业实例相关联的先前执行中的作业参数。先前执行必须为“停止”或“失败”状态。如果没有与此实例相关联的作业执行，或最新作业执行为“完成”状态，那么 API 将返回错误。请注意，reusePreviousParams 是可选设置。缺省设置为 reusePreviousParams=false。

👉 注：如果 reusePreviousParams=true，那么在当前重新启动请求中提交的任何作业参数优先于任何先前作业参数。当前参数覆盖具有相同作业参数键名的先前参数。

DELETE /vsettan/api/batch/jobinstances/*job instance id*

此 URI 清除与此作业实例相关联的所有数据库条目和作业日志。如果作业实例具有活动作业执行，那么此 API 将返回错误。如果删除作业日志时存在错误，那么系统不会尝试从作业存储数据库删除作业实例数据。查询参数包括：

- `purgeJobStoreOnly=true|false`：如果 `purgeJobStoreOnly=true`，那么系统不会尝试清除与此作业实例相关联的作业日志。缺省设置为 `purgeJobStoreOnly=false`。如果作业实例具有活动作业执行，那么此 API 将返回错误。

  👉 注：使用单个清除 API 时，不会返回任何清除响应消息。

DELETE /vsettan/api/batch/v2/jobinstances/

此 URI 清除与以下清除过滤参数返回的作业实例相关联的所有数据库条目和作业日志。

👉 注：建议您使用 GET 接口列示这些作业并在执行 DELETE 接口清除它们之前验证它们是否为要清除的相应作业。

- page=[page number]：指示要返回的页面（记录子集）。缺省值为 0。
- pageSize=[number of records per page]：指示每页记录数。缺省值为 50。
- purgeJobStoreOnly=true|false：如果 purgeJobStoreOnly=true，那么系统不会尝试清除与此作业实例相关联的作业日志。缺省设置为 purgeJobStoreOnly=false。如果作业实例具有活动作业执行，那么此 API 将返回错误。
- jobInstanceId=[instanceId]:[instanceId]：清除等于和包含于 instanceId 范围中的作业实例。
- jobInstanceId=>[instanceId]：清除等于和大于所提供 instanceId 的作业实例。
- jobInstanceId=<[instanceId]：清除等于和小于所提供 instanceId 的作业实例。
- jobinstanceId=[instanceId],[instanceId],[instanceId]：清除所指定作业实例。
- createTime=[yyyy-MM-dd]:[yyy-MM-dd]：返回此日期范围内（包含上限和下限）的作业实例。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=[yyyy-MM-dd]：返回给定日期的作业实例。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=>3d：返回三天之前的日期或之后创建的作业实例。例如，createTime 晚于或等于三天之前的日期开始时间。有关更多信息，请参阅关于此主题中内容的进一步注释。
- createTime=<3d：返回三天之前的日期或之前创建的作业实例。例如，createTime 早于或等于三天之前的日期结束时间。有关更多信息，请参阅关于此主题中内容的进一步注释。
- instanceState=[state],[state]：清除具有所提供状态的作业实例。有效实例状态为：SUBMITTED、JMS_QUEUED、JMS_CONSUMED、DISPATCHED、FAILED、STOPPED、COMPLETED 和 ABANDONED。
- exitStatus=[string]：返回与退出状态字符串相匹配的作业实例。字符串条件可在任一端使用通配符 (*) 运算符。

👉 注：缺省情况下，除非指定了 page 和 pageSize 参数，否则最多清除 50 个记录。

👉 注：涉及 **createTime** 的查询中的服务器缺省时区的作用

提交作业时，作业实例的 createTime 存储在作业存储库中并规范化为 UTC。如果通过 yyyy-MM-dd 指定日期（例如，在 `createTime=[yyyy-MM-dd]` 或 `createTime=[yyyy-MM-dd]:[yyy-MM-dd]:` 中），那么必须将 yyyy-MM-dd 日期字符串转换为 UTC 时间的特定范围，以与作业实例表记录中的 createTime 值匹配。为此，此应用程序使用处理 REST 请求的服务器的缺省时区。系统使用此服务器的时区将日期字符串转换为要匹配 UTC 时间范围。

下表演示查询参数 jobInstanceId=10:13 返回的数据。

JOBINSTANCEID	**CREATETIME（server1 的时区中的 *）**	**INSTANCESTATE**	**EXITSTATUS**
10	11-05-2015.01:10:00	COMPLETED	SUCCESS
11	11-08-2014.02:20:00	COMPLETED	SUCCESS
12	11-10-2015.03:30:00	FAILED	FAILURE
13	11-11-2015:04:40:00	COMPLETED	SUCCESS

* 因为作业存储库将 createTime 存储为 UTC 格式，所以了解以上表数据显示的 createTime 的格式使用特定服务器（例如，“server1”）的服务器缺省时区很重要。如果已从具有不同缺省时区的另一服务器（非“server1”）的角度构造类似表，那么应显示另一组 CREATETIME 列值，这些值具有对应的基于时区的时差。它是处理 REST 请求的服务器的缺省时区，它用于将 yyyy-MM-dd 日期字符串参数映射至数据库中的 UTC createTime 值（如以下示例中所示）。

以下命令返回相同结果，不管针对哪个服务器发出这些命令都是如此：

- jobInstanceId=11:13 将返回 JOBINSTANCEID 11、12 和 13。
- jobInstanceId=>12 将返回 JOBINSTANCEID 12 和 13。
- jobInstanceId=<12 将返回 JOBINSTANCEID 11 和 12。
- jobInstanceId=11,12 将返回 JOBINSTANCEID 11 和 12。
- instanceState=FAILED 将返回 JOBINSTANCEID 12。
- exitStatus=SUCCESS 将返回 JOBINSTANCEID 10、11 和 13。

对于具有不同缺省时区的服务器，以下命令可能返回不同结果。在这些示例中，它们是针对“server1”（用于填充上表的服务器）在 11-11-2015:07:00:00（使用该服务器的缺省时区）发出的。

- createTime=>2d 将返回 JOBINSTANCEID 12 和 13（在两天前的日期（即 11-09-2015）或之后）
- createTime=<2d 将返回 JOBINSTANCEID 10 和 11（在两天前的日期（即 11-09-2015）或之前）
- createTime=2015-11-08:2015-11-11 将返回 JOBINSTANCEID 11、12 和 13。
- createTime=2015-11-10 将返回 JOBINSTANCEID 为 12 的记录。

样本响应：

```
[{"instanceId":394,"purgeStatus":"COMPLETED","message":"Successful purge."
,"redirectUrl":""},
{"instanceId":395,"purgeStatus":"COMPLETED","message":"Successful purge.",
"redirectUrl":""},
```

```
{"instanceId":396,"purgeStatus":"COMPLETED","message":"Successful purge.",
"redirectUrl":""},
{"instanceId":397,"purgeStatus":"COMPLETED","message":"Successful purge.",
"redirectUrl":""},
{"instanceId":398,"purgeStatus":"COMPLETED","message":"Successful purge.",
"redirectUrl":""}]
```


可返回以下 purgeStatus 值：

COMPLETED

指示作业清除已成功完成。

FAILED

指示作业清除失败。

STILL_ACTIVE

指示作业清除失败，因为它仍处于活动状态。

JOBLOGS_ONLY

指示数据库清除失败，但作业日志已成功清除。

NOT_LOCAL

指示作业清除失败，因为作业不在本地。

作业执行

GET /vsettan/api/batch/jobexecutions/*job execution id*

此 URI 返回有关指定作业执行的详细信息并包含关联步骤执行和作业日志的链接。

样本请求：

```
https://localhost:9443/vsettan/api/batch/jobexecutions/9
```

样本响应：


```
{
    "jobName":"test_sleepyBatchlet",
    "executionId":9,
    "instanceId":9,
    "batchStatus":"COMPLETED",
    "exitStatus":"COMPLETED",
    "createTime":"2015/05/07 16:09:41.025 -0400",
    "endTime":"2015/05/07 16:09:52.127 -0400",
    "lastUpdatedTime":"2015/05/07 16:09:52.127 -0400",
    "startTime":"2015/05/07 16:09:41.327 -0400",
    "jobParameters":{
    },
    "restUrl":"https://localhost:9443/vsettan/api/batch",
    "serverId":"localhost/C:/vsettan/RAD_workspaces/xigemaAS7/build.image/
wlp/usr/server1",
    "logpath":"C:\\vsettan\\xigemaAS\\wlp\\usr\\servers\\server1\\logs\\jo
blogs\\test_sleepyBatchlet\\2015-05-07\\instance.9\\execution.9\\",
    "stepExecutions":[
        {
            "stepExecutionId":9,
            "stepName":"step1",
```

```
            "batchStatus":"COMPLETED",
            "exitStatus":"SleepyBatchlet:i=10;stopRequested=false",
            "stepExecution":"https://localhost:9443/vsettan/api/batch/jobe
xecutions/9/stepexecutions/step1"
        }
    ],
    "_links":[
        {
            "rel":"self",
            "href":"https://localhost:9443/vsettan/api/batch/jobexecutions
/9"
        },
        {
            "rel":"job instance",
            "href":"https://localhost:9443/vsettan/api/batch/jobinstances/
9"
        },
        {
            "rel":"step executions",
            "href":"https://localhost:9443/vsettan/api/batch/jobexecutions
/9/stepexecutions"
        },
        {
            "rel":"job logs",
            "href":"https://localhost:9443/vsettan/api/batch/jobexecutions
/9/joblogs"
        },
        {
            "rel":"stop url",
            "href":"https://localhost:9443/vsettan/api/batch/jobexecutions
/9?action=stop"
        }
    ]
}
```


GET /vsettan/api/batch/jobexecutions/*job execution id*/jobinstance

此 URI 返回有关与指定作业执行相关的作业实例的详细信息。

GET /vsettan/api/batch/jobinstances/*job instance id*/jobexecutions

此 URI 返回有关指定作业实例的作业执行的详细信息。这包括指向关联的步骤执行和作业日志的链接。

GET /vsettan/api/batch/jobinstances/*job instance id*/jobexecutions/*job execution sequence number*

此 URI 返回有关与指定作业实例标识相关的指定作业执行的详细信息。这包括指向关联的步骤执行和作业日志的链接。

👉 注：作业执行序列号表示与指定作业实例相关的第 0 个、第 1 个、第 2 个（以此类推）作业执行。

PUT /vsettan/api/batch/jobexecutions/*job execution id*?action=stop

使用此 URI 停止正在运行的指定作业执行。必需参数包含 action = *stop, restart*。

PUT /vsettan/api/batch/jobexecutions/*job execution id*?action=restart

使用此 URI 重新启动指定作业执行。必需参数包含 action = *stop, restart*。

步骤执行

GET /vsettan/api/batch/jobexecutions/*job execution id*/stepexecutions

此 URI 返回指定作业执行的所有步骤执行详细信息的 JSON 数组。如果作业包含分区步骤，那么将返回每个步骤中列示的分区信息。

样本请求：

```
https://localhost:8020/vsettan/api/batch/jobexecutions/40/stepexecutions
```

以下样本演示针对分区步骤的响应。


```
[
   {
      "stepExecutionId":47,
      "executionId":39,
      "instanceId":35,
      "stepName":"step1",
      "batchStatus":"COMPLETED",
      "startTime":"2015/03/30 11:10:08.652 -0400",
      "endTime":"2015/03/30 11:10:09.817 -0400",
      "exitStatus":"COMPLETED",
      "metrics":{
         "READ_COUNT":"0",
         "WRITE_COUNT":"0",
         "COMMIT_COUNT":"0",
         "ROLLBACK_COUNT":"0",
         "READ_SKIP_COUNT":"0",
         "PROCESS_SKIP_COUNT":"0",
         "FILTER_COUNT":"0",
         "WRITE_SKIP_COUNT":"0"
      },
      "partitions":[
         {
            "partitionNumber":0,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:09.579 -0400",
            "endTime":"2015/03/30 11:10:09.706 -0400",
            "exitStatus":"step1",
            "metrics":{
               "READ_COUNT":"0",
               "WRITE_COUNT":"0",
               "COMMIT_COUNT":"0",
               "ROLLBACK_COUNT":"0",
               "READ_SKIP_COUNT":"0",
               "PROCESS_SKIP_COUNT":"0",
               "FILTER_COUNT":"0",
               "WRITE_SKIP_COUNT":"0"
            }
         },
         {
            "partitionNumber":1,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:09.257 -0400",
            "endTime":"2015/03/30 11:10:09.302 -0400",
            "exitStatus":"step1",
            "metrics":{
               "READ_COUNT":"0",
               "WRITE_COUNT":"0",
               "COMMIT_COUNT":"0",
```

```
                "ROLLBACK_COUNT":"0",
                "READ_SKIP_COUNT":"0",
                "PROCESS_SKIP_COUNT":"0",
                "FILTER_COUNT":"0",
                "WRITE_SKIP_COUNT":"0"
            }
         },
         {
            "partitionNumber":2,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:09.469 -0400",
            "endTime":"2015/03/30 11:10:09.548 -0400",
            "exitStatus":"step1",
            "metrics":{
                "READ_COUNT":"0",
                "WRITE_COUNT":"0",
                "COMMIT_COUNT":"0",
                "ROLLBACK_COUNT":"0",
                "READ_SKIP_COUNT":"0",
                "PROCESS_SKIP_COUNT":"0",
                "FILTER_COUNT":"0",
                "WRITE_SKIP_COUNT":"0"
            }
         }
      ]
   },
   {
      "stepExecutionId":51,
      "executionId":39,
      "instanceId":35,
      "stepName":"step2",
      "batchStatus":"COMPLETED",
      "startTime":"2015/03/30 11:10:09.915 -0400",
      "endTime":"2015/03/30 11:10:10.648 -0400",
      "exitStatus":"COMPLETED",
      "metrics":{
         "READ_COUNT":"0",
         "WRITE_COUNT":"0",
         "COMMIT_COUNT":"0",
         "ROLLBACK_COUNT":"0",
         "READ_SKIP_COUNT":"0",
         "PROCESS_SKIP_COUNT":"0",
         "FILTER_COUNT":"0",
         "WRITE_SKIP_COUNT":"0"
      },
      "partitions":[
         {
            "partitionNumber":0,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:10.324 -0400",
            "endTime":"2015/03/30 11:10:10.417 -0400",
            "exitStatus":"step2",
            "metrics":{
                "READ_COUNT":"0",
                "WRITE_COUNT":"0",
                "COMMIT_COUNT":"0",
                "ROLLBACK_COUNT":"0",
                "READ_SKIP_COUNT":"0",
                "PROCESS_SKIP_COUNT":"0",
                "FILTER_COUNT":"0",
```

```
                "WRITE_SKIP_COUNT":"0"
            }
         },
         {
            "partitionNumber":1,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:10.260 -0400",
            "endTime":"2015/03/30 11:10:10.347 -0400",
            "exitStatus":"step2",
            "metrics":{
               "READ_COUNT":"0",
               "WRITE_COUNT":"0",
               "COMMIT_COUNT":"0",
               "ROLLBACK_COUNT":"0",
               "READ_SKIP_COUNT":"0",
               "PROCESS_SKIP_COUNT":"0",
               "FILTER_COUNT":"0",
               "WRITE_SKIP_COUNT":"0"
            }
         },
         {
            "partitionNumber":2,
            "batchStatus":"COMPLETED",
            "startTime":"2015/03/30 11:10:10.507 -0400",
            "endTime":"2015/03/30 11:10:10.557 -0400",
            "exitStatus":"step2",
            "metrics":{
               "READ_COUNT":"0",
               "WRITE_COUNT":"0",
               "COMMIT_COUNT":"0",
               "ROLLBACK_COUNT":"0",
               "READ_SKIP_COUNT":"0",
               "PROCESS_SKIP_COUNT":"0",
               "FILTER_COUNT":"0",
               "WRITE_SKIP_COUNT":"0"
            }
         },
         {
            "_links":[
            {
               "rel":"job execution",
               "href":"https://localhost:9443/vsettan/api/batch/jobexecuti
ons/9"
            },
            {
               "rel":"job instance",
               "href":"https://localhost:9443/vsettan/api/batch/jobinstanc
es/9"
            }
         ]
    }
]
```


GET /vsettan/api/batch/jobexecutions/*job execution id*/stepexecutions/*step name*

此 URI 返回 JSON 数组，此数组包含指定作业执行的步骤执行详细信息及步骤名称。

GET /vsettan/api/batch/jobinstances/*job instance id*/jobexecutions/*job execution sequence number*/stepexecutions/*step name*

此 URI 返回 JSON 数组，此数组包含指定作业实例的步骤执行详细信息、作业执行及步骤名称。

GET /vsettan/api/batch/stepexecutions/*step execution id*

此 URI 返回 JSON 数组，此数组包含指定步骤执行的步骤执行详细信息。

作业日志

GET /vsettan/api/batch/jobinstances/*job instance id*/joblogs

此 URI 返回 JSON 数组，此数组带有指定作业实例的所有作业日志部分的 REST 链接。

GET /vsettan/api/batch/jobexecutions/*job execution id*/joblogs

此 URI 返回 JSON 数组，此数组带有指定作业执行的所有作业日志部分的 REST 链接。

👉 重要：以下示例显示 REST 链接的格式。

如果您的作业执行包含以下作业日志部分：

```
joblogs/instance.inst-id/execution.exec-id/part.1.log
joblogs/instance.inst-id/execution.exec-id/part.2.log
joblogs/instance.inst-id/execution.exec-id/step.step-name/partition.
0/part.1.log
joblogs/instance.inst-id/execution.exec-id/step.step-name/partition.
1/part.1.log
```

，那么对应 REST 链接为：

```
/vsettan/api/batch/jobexecutionsexec-id/joblogs?part=part.1.log
/vsettan/api/batch/jobexecutionsexec-id/joblogs?part=part.2.log
/vsettan/api/batch/jobexecutionsexec-id/joblogs?part=step.step-name/
partition.0/part.1.log
/vsettan/api/batch/jobexecutionsexec-id/joblogs?part=step.step-name/
partition.1/part.1.log
```

GET /vsettan/api/batch/jobexecutions/*job execution id*/joblogs

可选参数包括：

type = text

如果指定 text，那么将以纯文本形式返回所有作业日志。所有作业日志部分将汇总到一起。系统会在流中插入用于定界头记录和脚记录的部分以在汇总不同部分时对它们进行定界。

type = zip

如果指定 zip，那么将以压缩文件格式返回指定作业实例或作业执行的所有作业日志。压缩文件中将保留作业日志的目录结构。

GET /vsettan/api/batch/jobinstances/*job instance id*/joblogs?type=text|zip

GET /vsettan/api/batch/jobexecutions/*job execution id*/joblogs?type=text|zip

已指定 type 参数的这两个 URI 的行为根据值而有所不同。

type = text

如果指定 text，那么将以纯文本形式返回所有作业日志。所有作业日志部分将汇总到一起。系统会在流中插入用于定界头记录和脚记录的部分以在汇总不同部分时对它们进行定界。

type = zip

如果指定 zip，那么将以压缩文件格式返回指定作业实例或作业执行的所有作业日志。压缩文件中将保留作业日志的目录结构。

GET /vsettan/api/batch/jobexecutions/*job execution id*/joblogs?part=*path to part*&type=text|zip

指定了 part 参数后，此 URI 将作业日志部件返回为纯文本 (type=text) 或压缩文件 (type=zip)。缺省设置为 type=text。

HTTP 返回码

以下 HTTP 返回码适用于 REST API。

- HTTP 200 正常
- HTTP 201 已成功创建新响应。
- HTTP 202 已接受请求，但处理未完成。
- HTTP 400 错误请求，带有无效参数。请参阅所返回消息以了解详细信息。
- HTTP 401 无权访问此资源。
- HTTP 403 认证失败。
- HTTP 404 找不到所请求资源或该资源不存在。
- HTTP 409 该请求与该资源的当前状态冲突。请参阅所返回消息以了解详细信息。
- HTTP 500 内部服务器错误。

分布式服务器批处理环境中的 STOP 请求

发送至批处理 REST API 的停止请求必须直接发送至正在运行该作业的执行程序。如果停止请求被发送至未在运行该作业的分派器或执行程序，那么系统通过 HTTP 302 重定向响应消息将该请求将重定向至正确执行程序。HTTP 302 重定向响应中的**位置**字段指示要用于该停止请求的正确 URL。

分布式服务器批处理环境中的 JOBLOGS 请求

发送至批处理 REST API 的作业日志请求必须直接发送至正在运行该作业的执行程序。如果作业日志请求被发送至未在运行该作业的分派器或执行程序，那么系统通过 HTTP 302 重定向响应消息将该请求将重定向至正确执行程序。HTTP 302 重定向响应中的**位置**字段指示要用于该作业日志请求的正确 URL。

注：仅当作业实例的所有作业执行在同一执行程序上运行时，针对整个作业实例发送至批处理 REST API 的作业日志请求才会生效。如果作业执行在不同执行程序上运行，那么针对作业实例的作业日志请求将失败。在此情况下，必须单独访存每个作业执行的作业日志。

分布式服务器批处理环境中的清除请求

发送至批处理 REST API 的清除请求必须直接发送至正在运行该作业的执行程序。如果清除请求被发送至未在运行该作业的分派器或执行程序，那么系统通过 HTTP 302 重定向响应消息将该请求将重定向至正确执行程序。HTTP 302 重定向响应中的**位置**字段指示要用于该清除请求的正确 URL。

注：仅当作业实例的所有作业执行在同一执行程序上运行时，针对整个作业实例发送至批处理 REST API 的清除请求才会生效。如果作业执行在不同执行程序上运行，那么针对作业实例的清除请求将失败。

使用 xigemaAS 嵌入式消息传递提供程序启用多服务器支持

可设置批处理环境以便让一些服务器充当批处理分派器，而让其他服务器充当批处理执行程序。批处理分派器接受来自外部客户机的请求，并将它们提供给批处理执行程序。批处理执行程序接收与其定义功能相匹配的请求，然后执行这些请求。批处理分派器和批处理执行程序使用 Java 消息传递服务 (JMS) 进行通信。

确定嵌入式消息传递引擎所在的位置。它可以在批处理分派服务器、批处理执行程序服务器或单独服务器上。完成此任务前，必须配置此服务器。JMS 连接工厂和激活规范在其配置中引用消息引擎服务器。要配置消息引擎，请执行以下操作：

1. 将 wasJmsServer-1.0 功能部件添加至 server.xml。
2. 通过添加 messageEngine 元素定义消息引擎。定义用于批处理分派器和批处理执行程序的队列。以下示例说明 server.xml 中的消息引擎配置。

```
<!-specify the ports for the message engine.
The ports in this example are the default ports.
This element is not needed when the default ports are used. -->
<wasJmsEndpoint host="*"
    wasJmsPort="7280"
    wasJmsSSLPort="7290"
    enabled="true">
 </wasJmsEndpoint>
<messagingEngine>
 <!- queue for batch jms message. -->
 <queue id="batchxigemaASQueue"
  forceReliability="ReliablePersistent"
  receiveAllowed="true"/>
</messagingEngine>
```

此任务帮助您使用 xigemaAS 嵌入式消息传递提供程序配置批处理分派服务器和批处理执行程序。

1. 要配置使用 xigemaAS 嵌入式消息传递提供程序的批处理分派器和执行程序，请执行以下操作：
1. 配置批处理 JMS 分派器。
   a. 通过将 wasJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。
   b. 将 batchJmsDispatcher 元素添加至托管批处理分派器的服务器上的 server.xml。

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection
 factory}
 queueRef={reference to a configured JMS queue} />
```

注：如果未指定 connectionFactoryRef 和 queueRef 属性，那么 connectionFactoryRef 的缺省值为 batchConnectionFactory，queueRef 的缺省值为 batchJobSubmissionQueue。可将 batchJmsDispatcher 元素指定为 <batchJmsDispatcher/>。在 server.xml 文件中，您仍然必须配置 batchConnectionFactory JMS 连接工厂和 JMS batchJobSubmissionQueue 队列。

   c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

   以下示例说明批处理 JMS 分派器配置及其 JMS 配置。

注：remoteServerAddress 属性指向托管 xigemaAS 消息引擎的服务器的 host:port。

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />
```

```
<jmsConnectionFactory id="batchConnectionFactory" jndiName="jms/batch/connectionFactory">
  <properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
  </properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id='batchJobSubmissionQueue" jndiName="jms/batch/jobSubmissionQueue">
  <properties.wasJms deliveryMode="Persistent"
        queuename="batchxigemaASQueue">
  </properties.wasJms>
</jmsQueue>
```

2. 配置批处理 JMS 执行程序。

   a. 通过将 wasJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

   b. 将 batchJmsExecutor 元素添加至托管批处理执行程序的服务器上的 server.xml 文件。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue} />
```

   👉 注：如果未指定 `activationSpecRef` 和 `queueRef` 属性，那么 `activationSpecRef` 的缺省值为 batchActivationSpec，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将 `batchJmsExecutor` 元素指定为 `<batchJmsExecutor/>`。在 server.xml 文件中，您仍然必须对 `batchActivationSpec` 配置 JMS 激活规范，并配置 `batchJobSubmissionQueue` JMS 队列。

   c. 将相应 JMS 激活规范和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

   d. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

   - 根据系统定义的属性进行过滤：

     这是批处理 JMS 消息上的一组可用批处理分派器属性，批处理执行程序可使用它们来进行过滤以获取入站消息。

     - com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
     - com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
     - com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称

     👉 注：建议使用至少一个 com_ibm_ws_batch_applicationName 属性来指定消息选择器，以确保该执行程序仅接收它可处理的作业。

     以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 `SimpleBatchJob` 和 `BonusPayout` 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'">
```

     以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 `SimpleBatchJob` 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob'">
```

   - 根据用户定义的属性进行过滤：

     批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 `true` 的任意字符，并包含“_”和“$”。字母或数字是方法

Character.isJavaLetterOrDigit 对其返回 `true` 的任意字符。请查看 JMS Javadoc 以了解有关 JMS 消息选择器的更多信息。

以下示例说明使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 的可行消息选择器。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 specialCapability = 'superCapability'">
```

以下示例演示批处理 JMS 执行程序配置及其 JMS 配置。

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
      queueRef="batchJobSubmissionQueue"/>

<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"
      messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayoutCDI'"
      destinationType="javax.jms.Queue"
      remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent" queueName="batchxigemaASQueue">
 </properties.wasJms>
</jmsQueue>
```

3. 在服务器上安装批处理应用程序。有关更多信息，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

使用 xigemaMQ 消息传递提供程序启用多服务器支持

可设置批处理环境以便让一些服务器充当批处理分派器，而让其他服务器充当批处理执行程序。批处理分派器接受来自外部客户机的请求，并将它们提供给批处理执行程序。批处理执行程序接收与其定义功能相匹配的请求，然后执行这些请求。批处理分派器和批处理执行程序使用 Java 消息传递服务 (JMS) 进行通信。

此任务帮助您使用 xigemaMQ 消息传递提供程序配置批处理分派服务器和批处理执行程序。

1. 配置批处理 JMS 分派器。
   a. 通过将 wmqJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。
   b. 将 batchJmsDispatcher 元素添加至托管批处理分派器的服务器上的 server.xml 文件。

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection
 factory}
 queueRef={reference to a configured JMS queue} />
```

> 👉 注：如果未指定 `connectionFactoryRef` 和 `queueRef` 属性，那么 `connectionFactoryRef` 的缺省值为 batchConnectionFactory，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将 `batchJmsDispatcher` 元素指定为 `<batchJmsDispatcher/>`。在 server.xml 文件中，您仍然必须配置 batchConnectionFactory JMS 连接工厂和 JMS batchJobSubmissionQueue 队列。

   c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

以下示例说明使用 WMQ 绑定方式的批处理 JMS 分派器配置及其 JMS 配置。

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
        queueRef="batchJobSubmissionQueue" />
```

```
<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
    value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
  startupRetryCount=999
  startupRetryInterval="1000ms"
  reconnectionRetryCount=10
  reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
 <properties.wmqJms transportType="BINDINGS"
      queueManager="WMQX">
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
     priority="QDEF"
     baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

以下示例说明使用 WMQ 客户机方式的批处理 JMS 分派器配置及其 JMS 配置。

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
        queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
    value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
   startupRetryInterval="1000ms"
   reconnectionRetryCount=10
   reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
 <properties.wmqJms
      hostName="webs24.pok.stglabs.vsettan.com"
      transportType="CLIENT"
      channel="WAS.JMS.SVRCONN"
      port="1414"
      queueManager="WMQX">
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
     priority="QDEF"
     baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

2. 配置批处理 JMS 执行程序。

   a. 通过将 wmqJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

   b. 将 batchJmsExecutor 元素添加至托管批处理执行程序的服务器上的 server.xml 文件。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue} />
```

   👉 注：如果未指定 activationSpecRef 和 queueRef 属性，那么 activationSpecRef 的缺省值为 batchActivationSpec，queueRef 的缺省值为 batchJobSubmissionQueue。可将 batchJmsExecutor 元素指定为 <batchJmsExecutor/>。在 server.xml 文件中，您仍然必须对 batchActivationSpec 配置 JMS 激活规范，并配置 batchJobSubmissionQueue JMS 队列。

   c. 将相应 JMS 激活规范和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

   d. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

   - 根据系统定义的属性进行过滤：

     这是批处理 JMS 消息上的一组可用批处理分派器属性，批处理执行程序可使用它们来进行过滤以获取入站消息。

     - com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
     - com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
     - com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称

     👉 注：建议使用至少一个 com_ibm_ws_batch_applicationName 属性来指定消息选择器，以确保该执行程序仅接收它可处理的作业。

     以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 SimpleBatchJob 和 BonusPayout 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'">
```

     以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 SimpleBatchJob 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob'">
```

   - 根据用户定义的属性进行过滤：

     批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 true 的任意字符，并包含“_”和“$”。字母或数字是方法 Character.isJavaLetterOrDigit 对其返回 true 的任意字符。请查看 JMS Javadoc 以了解有关 JMS 消息选择器的更多信息。

     以下示例使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 来演示可行消息选择器。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 specialCapability = 'superCapability'">
```

以下示例说明使用 WMQ 绑定方式的批处理 JMS 执行程序配置及其 JMS 配置。

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
  startupRetryCount=999
  startupRetryInterval="1000ms"
  reconnectionRetryCount=10
  reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
          destinationType="javax.jms.Queue"
          messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'"
          transportType="BINDINGS"
          queueManager="WMQX">
    </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
          priority="QDEF"
          baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

以下示例说明使用 WMQ 客户机方式的批处理 JMS 执行程序配置及其 JMS 配置。

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
   startupRetryInterval="1000ms"
   reconnectionRetryCount=10
   reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
      messageSelector="com_ibm_ws_batch_applicationName = SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'"
   transportType="CLIENT"
   channel="WAS.JMS.SVRCONN"
   destinationType="javax.jms.Queue"
   queueManager="WMQX"
   hostName="webs24.pok.stglabs.vsettan.com"
   port="1414">
    </properties.wmqJms>
</jmsActivationSpec>
```

```
<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
    baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

3. 在服务器上安装批处理应用程序。有关更多信息，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

使用 xigemaMQ 消息传递提供者启用多服务器分区支持

可设置批处理环境以便让服务器充当批处理分派器，并让其他服务器充当批处理执行程序。

批处理分派器接受来自外部客户机的请求，并将它们提供给批处理执行程序。批处理执行程序接收与其定义功能相匹配的请求，然后执行这些请求。如果作业配置为运行分区，那么批处理执行程序将它们提供给其他执行程序以运行。批处理执行程序通过使用 Java 消息传递服务 (JMS) 进行通信。此任务帮助您使用 xigemaMQ 消息传递提供者配置批处理分派服务器和批处理执行程序。

1. 配置批处理 JMS 分派器。
   a. 通过将 wmqJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。
   b. 将 batchJmsDispatcher 元素添加至托管批处理分派器的服务器上的 server.xml 文件；例如：

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection
 factory}
 queueRef={reference to a configured JMS queue} />
```

   👉 注：如果未指定 `connectionFactoryRef` 和 `queueRef` 属性，那么 `connectionFactoryRef` 的缺省值为 batchConnectionFactory，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将 `batchJmsDispatcher` 元素指定为 `<batchJmsDispatcher/>`。在 server.xml 文件中，您仍然必须配置 batchConnectionFactory JMS 连接工厂和 JMS batchJobSubmissionQueue 队列。

   c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

   以下示例说明使用 xigemaMQ 绑定方式的批处理 JMS 分派器配置及其 JMS 配置：

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
        queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
    value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
  startupRetryCount=999
  startupRetryInterval="1000ms"
  reconnectionRetryCount=10
  reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
 <properties.wmqJms transportType="BINDINGS"
      queueManager="WMQX">
 </properties.wmqJms>
```

```
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
     priority="QDEF"
     baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

以下示例说明使用 xigemaMQ 客户机方式的批处理 JMS 分派器配置及其 JMS 配置：

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
        queueRef="batchJobSubmissionQueue" />

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
    value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
   startupRetryInterval="1000ms"
   reconnectionRetryCount=10
   reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
 <properties.wmqJms
 hostName="webs24.pok.stglabs.vsettan.com.cn"
      transportType="CLIENT"
      channel="WAS.JMS.SVRCONN"
      port="1414"
      queueManager="WMQX"/>>
 </properties.wmqJms>
</jmsConnectionFactory>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
     priority="QDEF"
     baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

2. 配置批处理 JMS 执行程序以执行作业

   a. 通过将 wmqJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

   b. 将 batchJmsDispatcher 元素添加至托管批处理执行程序的服务器上的 server.xml 文件。

   > 👉 注：如果未将 batchJmsDispatcher 元素添加至 server.xml 文件，那么服务器不会分派分区以在多个服务器上运行，而是在分区本地运行。

   c. 将 batchJmsExecutor 元素和 connectionFactory 添加至托管批处理执行程序的服务器上的 server.xml 文件。有关更多信息，请参阅步骤 1b 和 1c。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue} />
```

> 👉 注：如果未指定 `activationSpecRef` 和 `queueRef` 属性，那么 `activationSpecRef` 的缺省值为 batchActivationSpec，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将

batchJmsExecutor 元素指定为 <batchJmsExecutor/>。在 server.xml 文件中，您仍然必须对 batchActivationSpec 配置 JMS 激活规范，并配置 batchJobSubmissionQueue JMS 队列。

d. 将相应 JMS 激活规范和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

e. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

- 根据系统定义的属性进行过滤：

  以下批处理分派器属性在批处理 JMS 消息上可用，批处理执行程序可使用它们来进行过滤以获取入站消息。

  - com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
  - com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
  - com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称
  - com_ibm_ws_batch_work_type：工作类型：“作业”

  👉 注：指定带有至少一个 com_ibm_ws_batch_applicationName 属性的消息选择器以确保执行程序仅接收它可以处理的作业。

  以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 SimpleBatchJob 和 BonusPayout 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'">
```

- 根据用户定义的属性进行过滤：

  批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 true 的任意字符，并包含“_”和“$”。字母或数字是方法 Character.isJavaLetterOrDigit 对其返回 true 的任意字符。查看 JMS API 文档以了解有关 JMS 消息选择器的更多信息。

  以下示例使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 来演示可行消息选择器：

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 specialCapability = 'superCapability'">
```

以下示例说明使用 xigemaMQ 绑定方式的批处理 JMS 执行程序配置及其 JMS 配置：

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
  startupRetryCount=999
  startupRetryInterval="1000ms"
  reconnectionRetryCount=10
  reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
```

```
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
          destinationType="javax.jms.Queue"
          messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_ws_batch_work_type='Job'""
          transportType="BINDINGS"
          queueManager="WMQX">
    </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
          priority="QDEF"
          baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

以下示例说明使用 xigemaMQ 客户机方式的批处理 JMS 执行程序配置及其 JMS 配置：

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
   startupRetryInterval="1000ms"
   reconnectionRetryCount=10
   reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
      messageSelector="com_ibm_ws_batch_applicationName = SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'"
   transportType="CLIENT"
   channel="WAS.JMS.SVRCONN"
   destinationType="javax.jms.Queue"
   queueManager="WMQX"
   hostName="webs24.pok.stglabs.vsettan.com.cn"
   port="1414">
    </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
    baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

3. 配置批处理 JMS 执行程序以仅执行分区。

   a. 通过将 wmqJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

   b. 将 batchJmsExecutor 元素和 connectionFactory 添加至托管批处理执行程序的服务器上的 server.xml 文件。有关更多信息，请参阅步骤 1b 和 1c。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue}
```

```
        replyConnectionFactoryRef={reference to the configured JMS connection factory} />
```

👉 注：如果未指定 activationSpecRef 和 queueRef 属性，那么 activationSpecRef 的缺省值为 batchActivationSpec，queueRef 的缺省值为 batchJobSubmissionQueue。可将 batchJmsExecutor 元素指定为 <batchJmsExecutor/>。在 server.xml 文件中，您仍然必须对 batchActivationSpec 配置 JMS 激活规范，并配置 batchJobSubmissionQueue JMS 队列。

c. 将相应 JMS 激活规范和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

d. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

- 根据系统定义的属性进行过滤：

  以下批处理分派器属性在批处理 JMS 消息上可用，批处理执行程序可使用它们来进行过滤以获取入站消息。

  - com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
  - com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
  - com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称
  - com_ibm_ws_batch_work_type：工作类型：“分区”
  - com_ibm_ws_batch_partitionNum(Type=Integer)：要分派的分区的分区号
  - com_ibm_ws_batch_stepName：JSL 中定义的步骤的名称

  👉 注：指定带有至少一个 com_ibm_ws_batch_applicationName 属性的消息选择器以确保执行程序仅接收它可以处理的作业。

  以下示例指示执行程序的 messageSelector 属性，用于接受对应应用程序 SimpleBatchJob 和 BonusPayout 的作业。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout'">
```

- 根据用户定义的属性进行过滤：

  批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 true 的任意字符，并包含“_”和“$”。字母或数字是方法 Character.isJavaLetterOrDigit 对其返回 true 的任意字符。查看 JMS API 文档以了解有关 JMS 消息选择器的更多信息。

  以下示例使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 来演示可行消息选择器：

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 specialCapability = 'superCapability'">
```

以下示例说明使用 xigemaMQ 绑定方式的批处理 JMS 执行程序配置及其 JMS 配置：

```
<jmsConnectionFactory id="batchConnectionFactory"
                      jndiName="jms/batch/connectionFactory"/>

<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"
                  replyConnectionFactoryRef="batchConnectionFactory"/>

<!-- wmq resource adapter -->
```

```
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- nativeLibraryPath is required for BINDING mode -->
<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"
  startupRetryCount=999
  startupRetryInterval="1000ms"
  reconnectionRetryCount=10
  reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
          destinationType="javax.jms.Queue"
          messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_batch_work_type='Job'"
          transportType="BINDINGS"
          queueManager="WMQX">
    </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
          priority="QDEF"
          baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

以下示例说明使用 xigemaMQ 客户机方式的批处理 JMS 执行程序配置及其 JMS 配置：

```
<jmsConnectionFactory id="batchConnectionFactory"
                      jndiName="jms/batch/connectionFactory"/>

<batchJmsExecutor activationSpecRef="batchActivationSpec"
    queueRef="batchJobSubmissionQueue"
                  replyConnectionFactoryRef="batchConnectionFactory"/>

<!-- wmq resource adapter -->
<variable name="wmqJmsClient.rar.location"
          value="${server.config.dir}/wmq.wlp.rar"/>

<!-- the startup retry and reconnect retry properties are
     recommended to ensure robustness of the system.-->
<wmqJmsClient startupRetryCount=999
   startupRetryInterval="1000ms"
   reconnectionRetryCount=10
   reconnectionRetryInterval="5m">
</wmqJmsClient>

<authData password="pwd" user="user">
</authData>

<JmsActivationSpec id="batchActivationSpec" >
    <properties.wmqJms destinationRef="batchJobSubmissionQueue"
      messageSelector="(com_ibm_ws_batch_applicationName = SimpleBatchJob' OR
 com_ibm_ws_batch_applicationName = 'BonusPayout') AND com_ibm_batch_work_type='Job'"
   transportType="CLIENT"
   channel="WAS.JMS.SVRCONN"
   destinationType="javax.jms.Queue"
   queueManager="WMQX"
   hostName="webs24.pok.stglabs.vsettan.com.cn"
   port="1414">
    </properties.wmqJms>
</jmsActivationSpec>

<!-- baseQueueName is the queue defined on WMQ system -->
<jmsQueue id="batchJobSubmissionQueue"
```

```
   jndiName="jms/batch/jobSubmissionQueue">
    <properties.wmqJms baseQueueName="BATCHQ"
    baseQueueManagerName="WMQX">
    </properties.wmqJms>
</jmsQueue>
```

4. 在服务器上安装批处理应用程序。有关更多信息，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

   要禁用多服务器分区执行：如果您想要多服务器支持但不想要多服务器分区执行，那么可在 jsl 作业 XML 文件中将作业属性 come.ibm.websphere.batch.partition.multiJVM 设置为 false。以下示例说明 JSL 作业如何禁用多服务器分区：

```
<property name="com.ibm.websphere.batch.partition.multiJVM" value="false"/
>
```

   注：如果要在远程执行程序上运行分区，那么系统不会为该分区创建任何作业日志。

使用 xigemaAS 嵌入式消息传递提供者启用多服务器分区支持

可设置批处理环境以便让服务器充当批处理分派器，并让其他服务器充当批处理执行程序。

1. 确定嵌入式消息传递引擎所在的位置。它可以在批处理分派服务器、批处理执行程序服务器或单独服务器上。完成此任务前，必须配置此服务器。JMS 连接工厂和激活规范在其配置中引用消息引擎服务器。
2. 要配置消息引擎，请执行以下操作：
   a. 将 wasJmsServer-1.0 功能部件添加至 server.xml。
   b. 通过添加 messageEngine 元素定义消息引擎。定义用于批处理分派器和批处理执行程序的队列。以下示例说明 server.xml 文件中的消息引擎配置：

```
<!-specify the ports for the message engine.
The ports in this example are the default ports.
This element is not needed when the default ports are used. -->
<wasJmsEndpoint host="*"
    wasJmsPort="7280"
    wasJmsSSLPort="7290"
    enabled="true">
 </wasJmsEndpoint>
<messagingEngine>
 <!- queue for batch jms message. -->
 <queue id="batchxigemaASQueue"
  forceReliability="ReliablePersistent"
  receiveAllowed="true"/>
</messagingEngine>
```

批处理分派器接受来自外部客户机的请求，并将它们提供给批处理执行程序。批处理执行程序接收与其定义功能相匹配的请求，然后执行这些请求。如果作业配置为运行分区，那么批处理执行程序将它们提供给其他执行程序以运行。批处理执行程序通过使用 Java 消息传递服务 (JMS) 相互通信。此任务帮助您使用 xigemaAS 概要文件嵌入式消息传递提供者配置批处理分派服务器和批处理执行程序。

1. 配置批处理 JMS 分派器。
   a. 通过将 wasJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。
   b. 将 batchJmsDispatcher 元素添加至托管批处理分派器的服务器上的 server.xml 文件；例如：

```
<batchJmsDispatcher connectionFactoryRef={reference to a configured JMS connection
 factory}
 queueRef={reference to a configured JMS queue} />
```

👉 注：如果未指定 connectionFactoryRef 和 queueRef 属性，那么 connectionFactoryRef 的缺省值为 batchConnectionFactory，queueRef 的缺省值为 batchJobSubmissionQueue。可将 batchJmsDispatcher 元素指定为 <batchJmsDispatcher/>。在 server.xml 文件中，您仍然必须配置 batchConnectionFactory JMS 连接工厂和 JMS batchJobSubmissionQueue 队列。

c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

以下示例说明批处理 JMS 分派器配置及其 JMS 配置：

👉 注：remoteServerAddress 属性指向托管 xigemaAS 概要文件消息引擎的服务器的 host:port。

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
  <properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
  </properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
  <properties.wasJms deliveryMode="Persistent"
       queuename="batchxigemaASQueue">
  </properties.wasJms>
</jmsQueue>
```

2. 配置批处理 JMS 执行程序以执行批处理作业。

a. 通过将 wasJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

b. 将 batchJmsDispatcher 元素添加至托管批处理执行程序的服务器上的 server.xml 文件。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue} />
```

👉 注：如果未将 batchJmsDispatcher 元素添加至 server.xml 文件，那么服务器不会分派分区以在多个服务器上运行，而是在分区本地运行。

👉 注：如果未指定 connectionFactoryRef 和 queueRef 属性，那么 connectionFactoryRef 的缺省值为 batchConnectionFactory，queueRef 的缺省值为 batchJobSubmissionQueue。可将 batchJmsDispatcher 元素指定为 <batchJmsDispatcher/>。您仍必须在 server.xml 文件中配置 batchConnectionFactory JMS 连接工厂和 batchJobSubmissionQueue。

c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。

这不是特定于批处理配置的操作。以下示例说明批处理 JMS 分派器配置及其 JMS 配置：

👉 注：remoteServerAddress 属性指向托管 xigemaAS 概要文件消息引擎的服务器的 host:port。

```
<batchJmsDispatcher connectionFactoryRef="batchConnectionFactory"
      queueRef="batchJobSubmissionQueue" />

<jmsConnectionFactory id="batchConnectionFactory"
        jndiName="jms/batch/connectionFactory">
  <properties.wasJms remoteServerAddress="host:7280:BootstrapBasicMessaging">
  </properties.wasJms>
</jmsConnectionFactory>

<jmsQueue id='batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
```

```
  <properties.wasJms deliveryMode="Persistent"
       queuename="batchxigemaASQueue">
  </properties.wasJms>
</jmsQueue>
```

d. 将 batchJmsExecutor 元素添加至托管批处理执行程序的服务器上的 server.xml；例如：

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue} />
```

👉 注：如果未指定 `activationSpecRef` 和 `queueRef` 属性，那么 `activationSpecRef` 的缺省值为 batchConnectionFactory，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将 `batchJmsExecutor` 元素指定为 `<batchJmsExecutor/>`。在 server.xml 文件中，您仍然必须对 batchActivationSpec 配置 JMS 激活规范，并配置 batchJobSubmissionQueue 队列。

e. 将相应 JMS 激活规范和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

f. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

- 根据系统定义的属性进行过滤：

  以下批处理分派器属性在批处理 JMS 消息上可用，批处理执行程序可使用它们来进行过滤以获取入站消息。

  - com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
  - com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
  - com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称
  - com_ibm_ws_batch_work_type：工作类型：“作业”

  👉 注：指定带有至少一个 com_ibm_ws_batch_applicationName 属性的消息选择器以确保执行程序仅接收它可以处理的作业。

  以下示例指示执行程序的 messageSelector 属性，以仅接受针对应用程序 `SimpleBatchJob` 的批处理作业。

  ```
  messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
   com_ibm_ws_batch_work_type = 'Job'">
  ```

- 根据用户定义的属性进行过滤：

  批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 `true` 的任意字符，并包含“_”和“$”。字母或数字是方法 Character.isJavaLetterOrDigit 对其返回 `true` 的任意字符。查看 JMS API 文档以了解有关 JMS 消息选择器的更多信息。

  以下示例使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 来演示可行消息选择器：

  ```
  messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
   specialCapability = 'superCapability'">
  ```

以下示例演示批处理 JMS 执行程序配置及其 JMS 配置：

```
<batchJmsExecutor activationSpecRef="batchActivationSpec"
      queueRef="batchJobSubmissionQueue"/>
```

```
<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"
      messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob'
 OR com_ibm_ws_batch_applicationName = 'BonusPayoutCDI') AND
 com_ibm_ws_batch_work_type='Job'"
      destinationType="javax.jms.Queue"
      remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
         queueName="batchxigemaASQueue">
 </properties.wasJms>
</jmsQueue>
```

3. 配置批处理 JMS 执行程序以仅执行分区。

   a. 通过将 wasJmsClient-2.0 功能部件添加至 server.xml 文件中的功能部件管理器可启用 JMS 支持。

   b. 将 batchJmsExecutor 元素添加至服务器上的 server.xml 文件，该服务器托管正运行批处理作业的批处理执行程序。

```
<batchJmsExecutor activationSpecRef={configured activation specification or batch
 executor}
 queueRef={reference to the configured JMS queue}
       replyConnectionFactoryRef={reference to the configured JMS connection factory} />
```

   👉 注：如果未指定 `connectionFactoryRef` 和 `queueRef` 属性，那么 `connectionFactoryRef` 的缺省值为 batchConnectionFactory，`queueRef` 的缺省值为 batchJobSubmissionQueue。可将 `batchJmsDispatcher` 元素指定为 `<batchJmsDispatcher/>`。您仍必须在 server.xml 文件中配置 batchConnectionFactory JMS 连接工厂和 `batchJobSubmissionQueue`。

   c. 将相应 JMS 连接工厂和 JMS 队列添加至服务器配置。这不是特定于批处理配置的操作。

   d. 通过在激活规范中包含 JMS 消息选择器来定义批处理执行程序服务器功能。

      ◦ 根据系统定义的属性进行过滤：

      以下批处理分派器属性在批处理 JMS 消息上可用，批处理执行程序可使用它们来进行过滤以获取入站消息。

      ◦ com_ibm_ws_batch_applicationName：用于获取作业请求的批处理应用程序的名称
      ◦ com_ibm_ws_batch_moduleName：用于获取作业请求的批处理应用程序的模块名称
      ◦ com_ibm_ws_batch_componentName：用于获取作业请求的批处理应用程序的组件名称
      ◦ com_ibm_ws_batch_work_type：工作类型：“分区”
      ◦ com_ibm_ws_batch_partitionNum(Type=Integer)：要分派的分区的分区号
      ◦ com_ibm_ws_batch_stepName：JSL 中定义的步骤的名称

      👉 注：指定带有至少一个 com_ibm_ws_batch_applicationName 属性的消息选择器以确保执行程序仅接收它可以处理的作业。

      以下示例指示执行程序的 messageSelector 属性，以仅接受针对应用程序 `SimpleBatchJob` 的分区。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 com_ibm_ws_batch_work_type = 'Partition'">
```

以下示例指示执行程序的 messageSelector 属性，以仅接受针对应用程序 SimpleBatchJob 的分区。

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 com_ibm_ws_batch_work_type = 'Partition' AND come_ibm_ws_batch_stepName = 'step1'">
```

- 根据用户定义的属性进行过滤：

  批处理分派器设置符合批处理分派器请求消息上的相应 JMS 消息属性的所有作业参数。消息选择器还可使用这些属性将额外过滤条件添加至消息选择器。属性名称或标识必须符合 JMS 消息属性约束。例如，属性是长度不受限制的字母和数字组成的序列，第一位必须为字母。字母是方法 Character.isJavaLetter 对其返回 true 的任意字符，并包含“_”和“$”。字母或数字是方法 Character.isJavaLetterOrDigit 对其返回 true 的任意字符。查看 JMS API 文档以了解有关 JMS 消息选择器的更多信息。

  以下示例使用 com_ibm_ws_batch_applicationName 属性和作业参数 specialCapability 来演示可行消息选择器：

```
messageSelector="com_ibm_ws_batch_applicationName = 'SimpleBatchJob' AND
 specialCapability = 'superCapability'">
```

以下示例演示批处理 JMS 执行程序配置及其 JMS 配置：

```
<jmsConnectionFactory id="batchConnectionFactory"
                      jndiName="jms/batch/connectionFactory"/>

<batchJmsExecutor activationSpecRef="batchActivationSpec"
      queueRef="batchJobSubmissionQueue"
                  replyConnectionFactoryRef="batchConnectionFactory"/>

<jmsActivationSpec id="batchActivationSpec" >
 <properties.wasJms destinationRef="batchJobSubmissionQueue"
      messageSelector="(com_ibm_ws_batch_applicationName = 'SimpleBatchJob'
 OR com_ibm_ws_batch_applicationName = 'BonusPayoutCDI') AND
 com_ibm_ws_batch_work_type='Partition'"
      destinationRef="batchJobSubmissionQueue"
                      destinationType="javax.jms.Queue"
      remoteServerAddress="host:7280:BootstrapBasicMessaging">
 </properties.wasJms>
</jmsActivationSpec>

<jmsQueue id="batchJobSubmissionQueue"
   jndiName="jms/batch/jobSubmissionQueue">
 <properties.wasJms deliveryMode="Persistent"
         queueName="batchxigemaASQueue">
 </properties.wasJms>
</jmsQueue>
```

4. 在服务器上安装批处理应用程序。有关更多信息，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

   要禁用多服务器分区执行：如果您想要多服务器支持但不想要多服务器分区执行，那么可在 jsl 作业 XML 文件中将作业属性 come.ibm.websphere.batch.partition.multiJVM 设置为 false。以下示例说明 JSL 作业如何禁用多服务器分区：

```
<property name="com.ibm.websphere.batch.partition.multiJVM" value="false"/
>
```

   注：如果要在远程执行程序上运行分区，那么系统不会为该分区创建任何作业日志。

启用批处理作业事件发布

通过使用 Java 消息传递系统 (JMS)，批处理服务器可将与作业相关的事件发布至外部客户机。

批处理服务器将与作业相关的事件发布至外部客户的功能允许监视器查看与作业相关的事件并报告故障。批处理分派器服务器可对分派阶段中的作业发布事件。批处理执行程序服务器可在经历执行的不同阶段时对作业发布事件。这些事件是在以下结构的主题树中发布的：

表 45: 用于发布事件的主题树的文件结构

事件发布主题树文件结构

结构	描述
batch	主题树的根。
batch/jobs	所有与作业相关的事件的主题树。
batch/jobs/instance	与作业实例相关的所有事件的主题树。
batch/jobs/instance/submitted	主题树节点。批处理服务器针对新的作业提交请求创建作业实例时，将发布消息。
batch/jobs/instance/jms_queued	主题树节点。批处理 JMS 分派器将作业提交请求放置在作业提交队列中时，将发布消息。
batch/jobs/instance/jms_consumed	主题树节点。批处理执行程序从作业提交队列接收到作业提供请求时，将发布消息。
batch/jobs/instance/dispatched	主题树节点。批处理执行程序接受作业实例以执行时，将发布消息。
batch/jobs/instance/completed	主题树节点。作业实例已完成时，将发布消息。
batch/jobs/instance/stopped	主题树节点。作业实例已停止时，将发布消息。
batch/jobs/instance/stopping	主题树节点。作业实例正在停止时，将发布消息。
batch/jobs/instance/failed	主题树节点。作业实例失败时，将发布消息。
batch/jobs/instance/purged	主题树节点。作业实例已成功清除时，将发布消息。
batch/jobs/execution	与作业执行相关的所有事件的主题树。
batch/jobs/execution/restarting	主题树节点。批处理执行程序正在重新启动执行时，将发布消息。
batch/jobs/execution/starting	主题树节点。作业执行正在启动时，将发布消息。
batch/jobs/execution/completed	主题树节点。作业执行成功结束时，将发布消息。
batch/jobs/execution/failed	主题树节点。作业执行因为失败而结束时，将发布消息。
batch/jobs/execution/stopped	主题树节点。作业执行已停止时，将发布消息。

结构	描述
batch/jobs/execution/jobLogPart	主题树节点。创建新作业日志部分、作业停止或作业结束时，会发布消息。
batch/jobs/execution/step/started	主题树节点。步骤执行已启动时，将发布消息。
batch/jobs/execution/step/completed	主题树节点。步骤执行已成功完成时，将发布消息。
batch/jobs/execution/step/failed	主题树节点。步骤执行失败时，将发布消息。
batch/jobs/execution/step/stopped	主题树节点。步骤执行已停止时，将发布消息。
batch/jobs/execution/step/checkpoint	主题树节点。获取检查点时，将发布消息。
batch/jobs/execution/partition/started	主题树节点。分区已启动时，将发布消息。
batch/jobs/execution/partition/completed	主题树节点。分区已成功完成时，将发布消息。
batch/jobs/execution/partition/failed	主题树节点。分区失败时，将发布消息。
batch/jobs/execution/partition/stopped	主题树节点。分区已停止时，将发布消息。
batch/jobs/execution/split-flow/started	主题树节点。拆分流程已启动时，将发布消息。
batch/jobs/execution/split-flow/ended	主题树节点。拆分流程已完成时，将发布消息。

每个主题的已发布消息为 JMS TextMessage。此消息的内容为 JSON 格式的字符串，表示主题的对象，例如，作业实例、作业执行、步骤执行或分区。此外，此消息还包括以下 JMS 消息属性集：

- `com_ibm_ws_batch_internal_jobInstanceId`：作业实例标识（如果可用）。
- `com_ibm_ws_batch_internal_jobExecutionId`：作业执行标识（如果可用）。
- `com_ibm_ws_batch_internal_stepExecutionId`：作业步骤执行标识（如果可用）。

必须配置批处理服务器以允许发布与作业相关的事件。批处理分派器和批处理执行程序具有相同配置。以下步骤允许对批处理服务器发布与作业相关的事件。

1. 通过将相应 JMS 功能部件添加至 server.xml 文件中的功能部件管理器，启用 JMS 支持。

   如果要使用 xigemaAS 缺省消息传递提供程序，请添加 wasJmsClient-2.0 功能部件和消息传递引擎的相关 JMS 配置。如果要使用 xigemaMQ 消息传递提供程序，请添加 wmqJmsClient-2.0 功能部件。
2. 将 batchJmsEvents 元素添加至 server.xml 文件。

   ```
   <batchJmsEvents connectionFactoryRef="batchConnectionFactory" />
   ```

   注：如果未指定 connectionFactoryRef 属性，那么 connectionFactoryRef 的缺省值为 batchConnectionFactory。您仍然必须在 server.xml 文件中配置 batchConnectionFactory JMS 连接工厂。
3. 将对应 JMS 连接工厂添加至服务器配置。这不是特定于批处理配置的操作。

   以下示例使用 xigemaMQ 消息传递提供程序来演示批处理事件配置及其 JMS 配置。

   ```
   <!-- wmq resource adapter -->
   <variable name="wmqJmsClient.rar.location" value="${server.config.dir}/
   wmq.wlp.rar"/>
   ```

```
<!-- require for BINDING mode -->
<wmqJmsClient nativeLibraryPath="/mqm/jms/java/lib"/>

<batchJmsEvents connectionFactoryRef="batchConnectionFactory" />

<jmsConnectionFactory id="batchConnectionFactory" jndiName="jms/batch/
connectionFactory">
  <properties.wmq.Jms
    transportType="BINDINGS"
    queueManager="WMQX" />
</jmsConnectionFactory>
```

以下示例使用 xigemaAS 缺省消息传递提供程序来演示批处理事件配置及其 JMS 配置。

```
<batchJmsEvents connectionFactoryRef="batchConnectionFactory" />
<jmsConnectionFactory id="batchConnectionFactory" jndiName="jms/batch/
connectionFactory">
  <properties.wasJms></properties.wasJms>
</jmsConnectionFactory>
```

以下示例演示基本执行流程的事件序列。

- 提交并运行带有检查点的单步骤作业。

```
batch/jobs/instance/submitted
batch/jobs/instance/jms_queued
batch/jobs/instance/jms_consumed
batch/jobs/execution/starting
batch/jobs/instance/dispatched
batch/jobs/execution/started
batch/jobs/execution/step/started
batch/jobs/execution/step/checkpoint
batch/jobs/execution/step/checkpoint
...
batch/jobs/execution/step/checkpoint
batch/jobs/execution/step/completed
batch/jobs/execution/completed
batch/jobs/instance/completed
```

- 提交并运行带有分区的单步骤作业。

```
batch/jobs/instance/submitted
batch/jobs/instance/jms_queued
batch/jobs/instance/jms_consumed
batch/jobs/execution/starting
batch/jobs/instance/dispatched
batch/jobs/execution/started
batch/jobs/execution/step/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/started
batch/jobs/execution/partition/completed
batch/jobs/execution/partition/completed
batch/jobs/execution/partition/completed
batch/jobs/execution/step/completed
batch/jobs/execution/completed
```

```
batch/jobs/instance/completed
```

batchManager 命令行客户机实用程序

batchManager 命令行客户机实用程序提供命令行界面以管理在 xigemaAS 上运行的批处理作业。

batchManager 命令行客户机实用程序通过批处理管理器的 REST API 与批处理管理器交互。要使用 batchManager 命令行客户机实用程序，批处理管理器必须正在 xigemaAS 服务器上运行。使用批处理管理功能部件安装并启用 xigemaAS 批处理管理器。

SSL 配置

batchManager 命令行客户机实用程序通过 SSL 连接与批处理管理器通信。为方便与正在 xigemaAS 服务器上运行的批处理管理器进行 SSL 通信，该实用程序必须能够验证 xigemaAS 服务器的 SSL 证书。

如果 SSL 证书由知名认证中心 (CA) 签署，那么该实用程序可让 CA 验证该证书。不必进行更多配置。

如果 SSL 证书并非由 CA 签署，那么您必须执行下列其中一个操作来配置该实用程序以信任服务器的 SSL 证书。

- 指定选项 `--trustSslCertificates`，这会将该实用程序配置为信任所有 SSL 证书。
- 将服务器的 SSL 证书包含在实用程序的信任库中。

如果选择指定选项 `--trustSslCertificates`，那么该实用程序信任它接收的所有 SSL 证书，并且不必进行更多配置。

如果选择要将服务器的 SSL 证书包含在实用程序的信任库中的选项，那么还必须配置该实用程序以便该实用程序可找到其信任库。该实用程序是独立 Java main。通过使用 `javax.net.ssl.truststore` 之类的系统属性配置 SSL。

如果批处理管理器与该实用程序在同一机器上运行，那么可让该实用程序直接指向服务器密钥库：

```
   $ export JVM_ARGS="-Djavax.net.ssl.trustStore=/path/to/server/
keystore.jks"
   $ batchManager submit ...
```

注意：-D 属性之类的 JVM 自变量通过 *JVM-ARGS* 环境变量传递至 batchManager 命令行客户机实用程序。

如果无法直接使用服务器密钥库，那么必须从服务器密钥库导出服务器证书，然后将其导入至客户机信任库。使用 JDK 密钥工具实用程序导出和导入证书。在以下示例中，服务器证书存储在 [server-dir]/resources/security/key.jks 密钥库文件中的 *default* 别名下，密码为 xigemaAS。

```
    $ keytool -export -alias default -file server.crt -keystore [server-
dir]/resources/security/key.jks -storepass xigemaAS
    $ keytool -import -alias server_crt -file server.crt -keystore /path/to/
truststore.jks -storepass passw0rd
```

注意：`import` 命令创建 truststore.jks 文件（如果该文件不存在）。

```
    $ export JVM_ARGS="-Djavax.net.ssl.trustStore=/path/to/truststore.jks"
    $ batchManager submit ...
```

命令和用法

batchManager 命令行客户机实用程序提供用于提交、停止、重新启动作业和检查作业状态的命令。

以通用方式使用实用程序：

```
$ batchManager [command] [options]
```

查看可用命令列表：

```
$ batchManager help
```

查看特定命令的描述和选项：

```
$ batchManager help [command]
```

以下示例说明如何提交作业并等待其完成：

```
 $ batchManager submit \
       --batchManager=<host>:<port>
       --user=[credentials for logging into the batch manager]
       --password=[credentials for logging into the batch manager]
       --applicationName=[application name used when packaging the batch
app]
       --jobXMLName=[job XML file basename in the app's batch-jobs dir]
       --wait
```

返回码

batchManager 命令行客户机实用程序输出以下返回码：

代码	描述
0	任务正常完成。
20	未指定必需的自变量。
21	指定了无法识别的自变量。
22	指定了无效自变量值。
255	发生未知错误。

注：如果您指定 `--wait` 自变量，那么该实用程序输出以下返回码以指示您正在等待的作业的状态。

代码	描述
33	该作业已停止。
34	该作业未成功完成。
35	该作业已成功完成。
36	该作业已被放弃。

查看 Java 批处理作业日志

在 xigemaAS 概要文件中运行 Java 批处理作业时，将为每个作业编写一个日志。

日志按以下目录结构创建：

log directory/joblogs/*job name*/*date*/instance.*job instance ID*/execution.*execution ID*

⚠ 注意：变量 *job name* 为 JSL (XML) 文档内的 job 元素的 id 属性。它不一定与 JSL 文件的文件名相关。

日志命名从 part.1.log 开始，并根据需要转至新日志部分。缺省情况下，作业日志包含执行该作业执行的线程在服务器中记录的所有消息和跟踪信息。系统不会收集未记录在 java.util.logging 框架内的输出。

👉 重要：如果无法创建新的日志部分，那么批处理将尝试在没有日志或使用当前日志部分的情况下继续处理。

有关使用 REST API 检索或删除作业日志的更多信息，请参阅“REST API 管理” «» 文档。

👉 重要：如果使用 log4j API，使用 log4j 框架的应用程序可使用 org.apache.log4j.jul.JULAppender 参与批处理作业日志记录。JULAppender 将 log4j 日志记录转发至 java.util.logging 框架，系统可在此处收集它们以进行作业日志记录。

配置作业日志记录

可使用 <batchJobLogging> 配置元素 `<batchJobLogging enabled="true" maxRecords="1000" />` 来配置批处理作业日志记录。

属性 `maxRecords` 指示记录滚动到下一部分前写至作业日志部分的记录数。

此批处理功能部件使用名为 com.ibm.ws.batch.JobLogger 的记录器以将某些批处理消息仅记录至作业日志。示例包括作业生命周期消息和检查点消息。记录器不会写至服务器日志。缺省情况下，记录器支持 `Level.FINE` 消息。可通过在服务器的跟踪规范中指定记录器级别来配置记录器级别。例如，`<logging traceSpecification="*=info:com.ibm.ws.batch.JobLogger=all" />`。

作业线程编写的任何日志消息（包括运行时和应用程序代码编写的消息）将写至作业日志和服务器日志。

System.out 和 System.err 文件仅写至服务器日志，而不会写至作业日志。

分区步骤

对于每个分区，分区步骤具有更多子目录。*execution ID* 目录中的日志文件包含运行顶级作业的线程中的条目。这些分区的作业日志是使用以下结构存储的：

log directory/joblogs/*job name*/*date*/instance.*job instance ID*/execution.*execution ID*/*name of partitioned step*/*partition number*

👉 重要：系统将在 *name of partitioned step* 目录中为每个分区创建一个目录。

拆分流程

如果作业中出现拆分流程，那么系统会创建更多子目录以从每个流程的线程捕获输出。直接位于 *execution ID* 目录下的日志文件包含运行顶级作业的线程中的条目。各个流程线程的作业日志是使用以下结构存储的：

log directory/joblogs/*job name*/*date*/instance.*job instance ID*/execution.*execution ID*/*split ID*/*flow ID*

👉 重要：系统将在 *split ID* 目录中为每个流程创建一个目录。

作业日志事件

如果启用了批处理作业事件，那么在完成作业日志部分且作业转为已结束状态（例如，**已停止**、**已失败**或**已完成**）时，会发布作业日志事件。作业日志事件消息包含多个 JSON 属性，可帮助识别消息以及实际作业日志文件内容。

👉 重要：JSON 属性属于 JMS 消息体，可通过将消息体文本转换为 JsonObject 并从此对象拉取特定 JSON 属性来检索。

以下示例说明如何检索作业日志内容属性。

```
//The retrieved job log event message
Message msg
//Convert the Message to a TextMessage
TextMessage txtMsg = (TextMessage) msg;
//Convert the text in the message to a JsonObject
JsonObject jobLogEventObject = Json.createReader(new
 StringReader(txtMsg.getText())).readObject();
//Pull the job log text content from the JsonObject as an example
JsonArray logContentArray = jobLogEventObject.getJsonArray("contents");
```

有关启用批处理作业事件的更多信息，请参阅“启用批处理作业事件发布” «» 。

2.8.12 共享库

共享库是多个应用程序使用的文件。可以使用共享库和全局库来减少系统上重复库文件的数目。

库元素

xigemaAS 库具有三个元素：<folder>、<file> 和 <fileset>。例如：

```
<library>
 <folder dir="..." />
 <file name="..." />
 <fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>
```

所指定的文件必须是资源（例如，JAR 文件）的容器，而不是资源本身。

如果列表中的元素是文件，那么会搜索 JAR 文件或压缩 .zip 文件的内容。如果指定了文件夹，那么会从该目录中载入资源。

全局库

全局库可供任何应用程序使用。JAR 文件放入全局库目录，然后在每个应用程序的类加载器配置中加以指定。

可以将全局库放在以下两个位置：

- ${shared.config.dir}/lib/global
- ${server.config.dir}/lib/global

如果启动应用程序时这些位置中存在文件，并且该应用程序未配置 <classloader> 元素，那么应用程序会使用这些库。如果存在类加载器配置，那么不会使用这些库，除非显式引用全局库。

有关更多信息，请参阅为所有 *Java EE* 应用程序提供全局库（见第 1139 页）。

资源文件

在 xigemaAS 概要文件库内，可在 xigemaAS 元素内定义资源文件。例如，

```
<library>
<folder dir="..." />
<file name="..." />
<fileset dir="..." includes="*.jar" scanInterval="5s" />
<folder dir="${server.config.dir}/mylibs" />
<file name="${server.config.dir}/otherlibs/my.jar" />
```

以上文件夹设置允许 mylibs 目录下的所有文件在类路径上可用。可使用此条目样式来提供 xml 和 .properties。

库元素

xigemaAS 库有三个子元素：<folder>、<file> 和 <fileset>。例如，

```
<library>
<folder dir="..." />
<file name="..." />
<fileset dir="..." includes="*.jar" scanInterval="5s" />
</library>
```

- <folder>：每个已配置文件夹下的所有资源将是可装入的
- <file>：每个已配置文件应是资源的本机库或容器（例如，JAR 或 ZIP 文件）。容器中的所有资源是可装入的，所指定的任何其他文件类型不起作用。
- <fileset>：每个已配置文件集实际上是一组文件。文件集中的每个文件应是资源的本机库或容器（例如，JAR 或 ZIP 文件）。容器中的所有资源是可装入的，所指定的任何其他文件类型不起作用。

例如，

```
<library id="someLibrary">
<!-- Location of XML and .properties files in the file system for easy
 editing -->
<folder dir="${server.config.dir}/editableConfig" />

<!-- Location of some classes and resources in the file system -->
<folder dir="${server.config.dir}/extraStuff" />

<!-- A zip file containing some resources -->
<file name="${server.config.dir}/lib/someResources.zip" />

<!-- All the jar files in ther servers lib folder -->
<fileset dir="${server.config.dir}/lib" includes="*.jar" scanInterval="5s" /
>
</library>

<app location ="webStore.war">
<classloader commonLibraryRef="someLibrary" />
```

```
</app>
```

以上配置片段允许 webStore 应用程序装入 editableConfig 目录下的所有资源。

2.8.13 松散应用程序

松散应用程序来自多个物理位置，它们通过一个 XML 文件提供给运行时。Java™ EE 和 OSGi 应用程序支持松散应用程序，并且松散应用程序在开发环境中使用时特别有益。

常规应用程序

通常，应用程序包含在一个目录（或一个归档）中，其内容、模块、资源、类数据和元数据位于该目录内的已知位置。例如：Web 应用程序的资源的位置如下所示：

- 库 Java™ 归档 (JAR) 文件存储在 WEB-INF/lib 中
- 类在库 JAR 文件或 WEB-INF/classes 中
- 部署描述符在 WEB-INF/web.xml 中
- 要处理的内容位于该目录的根目录

松散应用程序

松散应用程序被描述为“表示应用程序的虚拟目录（信息可能位于任何位置）”。它允许开发工具运行直接从工作空间载入关联文件（而不是导出）的应用程序。关联文件的示例包括 Java™ 类、JavaServer Pages 或映像。如果直接从工作空间载入关联文件，那么会导致“构建 - 运行 - 调试”循环以更快速度运行。内容不在一个目录下，而是可能来自其他位置。这些位置是在 XML 配置文件中指定的。

可通过两种方式将该 XML 文件提供给运行时：

1. 通过将该 XML 文件放在应用程序配置元素的 location 属性中，并附加 .xml 后缀
2. 通过将该 XML 文件直接放置到应用程序 dropins 文件夹中

例如，如果指定 <application location="myapp.war" />，那么运行时查找名为 myapp.war.xml 的文件。对于应用程序目录或归档，搜索规则是相同的。如果同时找到应用程序文件和 .xml 松散应用程序配置文件，那么松散应用程序配置文件被忽略。例如，如果您有 myapp.war 和 myapp.war.xml，那么 xigemaAS 服务器使用 myapp.war 运行该应用程序。还可直接将松散应用程序部署到 dropins 文件夹中。要使用 dropins 文件夹，请遵循针对该文件夹定义的命名约定并将 .xml 附加至文件名结尾。

松散应用程序配置文件

xigemaAS 服务器使用松散应用程序配置文件获取应用程序内容，而不是从根目录或单个归档查找该应用程序。通过使用相应 XML，您可执行以下操作：

- 将任何物理目录映射至该应用程序内的任何位置
- 将任何物理文件映射至该应用程序内的任何位置
- 将任何物理 JAR 文件或目录以嵌套归档形式映射至任何位置
- 将多个物理源映射至单个目标位置（合并）

例如：

1. 将归档的根目录映射至磁盘上的某个位置，例如，Eclipse 项目中的某个文件夹。

2. 将不在“通常”位置的 Java™ bin/output 文件夹映射至 WEB-INF/classes 文件夹。由于工作空间首选项、公司准则、源控制项目布局准则等等，此位置可能在另一文件夹中。同一项目中可能有多个 Java™ 源位置和输出位置，您可能想要将它们都映射至 WEB-INF/classes。
3. 将“外部”JAR 文件映射至该应用程序。此“外部”JAR 文件可能为下列其中一项：
   - 一个独立 Java™ 项目，您想要将其视为 WEB-INF/lib 中的 JAR 文件
   - 硬盘驱动器上其他位置的一个实用程序 JAR 文件，您针对其构建了 .war 文件，并且需要在运行时包含在 WEB-INF/lib 中

松散应用程序配置文件示例

可在该松散应用程序配置文件中配置三个不同元素：

- <archive>（用于归档）
- <file>（用于文件）
- <dir>（用于目录）

归档

<archive> 元素始终用作松散应用程序配置文件的根目录。它也是 XML 中表示的虚拟文件系统的根目录。可在根 <archive> 元素下嵌套这三个元素中的任何元素。根 <archive> 元素没有任何属性。

归档元素可递归嵌套。对于嵌套在根 <archive> 元素下的 <archive> 元素，可设置 targetInArchive 属性。targetInArchive 属性定义该归档出现在松散定义的封闭归档内的路径。不能使用 <archive> 元素将文件系统上的归档映射为应用程序中的归档。要使用松散应用程序配置映射磁盘上的归档，请改用 <file> 元素。

👉 注：targetInArchive 属性值是带有前导正斜杠 (/) 的绝对路径。

以下代码是根 <archive> 元素（其下嵌套了另一 <archive> 元素）的示例：

```
<archive>
    <archive targetInArchive="/jarName.jar">
        <!-- more objects can be embedded here-->
    </archive>
</archive>
```

文件

可使用 <file> 元素将硬盘上的文件映射至松散应用程序配置中的文件。可在 <file> 元素上设置以下属性：

- targetInArchive 定义该归档出现在松散定义的封闭归档内的路径。
- sourceOnDisk 定义文件在文件系统上的实际位置。

👉 注：sourceOnDisk 属性值是绝对位置。可使用 ${xigemaas.server.dir} 之类已正确解析的 xigemaAS 变量。

以下代码是 C:/devFolder/myApplication.zip 中由松散应用程序配置表示为 /apps/webApplication.war 的文件示例：

```
<file targetInArchive="/apps/webApplication.war"
        sourceOnDisk="C:/devFolder/myApplication.zip" />
```

以下代码是在路径 /apps/webApplication.war 上定义归档的封闭归档的示例。在此归档内，webApplication.war 在路径 /applications/myApplications 上定义文件 jarName.jar。实际文件位置为 c:\devFolder\myApplication.zip：

```
<archive targetInArchive="/apps/webApplication.war">
    <file targetInArchive="/applications/myApplications/jarName.jar"
          sourceOnDisk="C:/devFolder/myApplication.zip" />
</archive>
```

目录

可使用 <dir> 元素将磁盘上的某个目录及其所有内容映射至松散应用程序配置上的某个目录位置。该元素具有与 <file> 元素相同的属性，您可以类似方式使用该元素。

以下代码是松散应用程序配置显示的目录的示例，如在 /META-INF 和文件系统上的 ${was.server.dir}/applicationData/myApplication 中：

```
<dir targetInArchive="/META-INF"
       sourceOnDisk="${was.server.dir}/applicationData/myApplication" />
```

要将该目录添加至归档以使其在 /apps/jarName.jar/META-INF 中显示，请按如下所示嵌入 <dir> 元素：

```
<archive targetInArchive="/apps/jarName.jar">
    <dir targetInArchive="/META-INF"
          sourceOnDisk="${was.server.dir}/applicationData/myApplication
" />
</archive>
```

在以上两个示例中，${was.server.dir}/applicationData/myApplication 中的所有文件已映射并显示在松散应用程序配置中由 targetInArchive 属性映射的目录下。

虚拟路径和文件名

如果将 <file> 或 <dir> 元素添加至归档，那么松散归档中的文件或目录的名称不必与磁盘上的实际名称相同。

以下代码是说明如何配置 ${was.server.dir}/applicationFiles/newfile.txt 以在归档中显示为 /application.txt 的示例：

```
<archive>
    <file targetInArchive="/application.txt"
          sourceOnDisk="${was.server.dir}/applicationFiles/newfile.txt"/>
</archive>
```

同一概念对任何已添加文件或目录的路径也适用。磁盘上的物理资源不必位于与要声明的对象对应的目录层次结构中。

以下代码是说明如何使归档中的 ${was.server.dir}/applicationFiles/newfile.txt 显示为 /only/available/in/application.txt 的示例：

```
<archive>
    <file targetInArchive="/only/available/in/application.txt"
          sourceOnDisk=""${was.server.dir}/applicationFiles/newfile.txt"/>
```

```
</archive>
```

在不同情况下，xigemaAS 服务器按 targetInArchive 属性声明的名称和路径查看资源。xigemaAS 服务器可浏览所声明的目录层次结构，即使该层次结构仅包含虚拟元素（就像以上示例中一样）。

```
<archive>
    <file targetInArchive="/only/available/in/red.txt"
           sourceOnDisk="${was.server.dir}/applicationFiles/newfile.txt" />
<archive targetInArchive="/apps/jarName.jar">
    <dir targetInArchive="/META-INF"
          sourceOnDisk="${was.server.dir}/applicationData/myApplication" />
</archive>
```

同名的文件夹和文件

如果在松散应用程序配置的同一虚拟位置中有两个同名文件夹，那么这些文件夹会合并，并且这两个文件夹的内容可用。如果有两个文件在松散归档中具有相同目标位置，那么系统使用第一次出现的文件。第一次出现以按自顶向下方法读取松散应用程序配置文件的元素时遇到的文件为准。

如果发现的第一个文件是错误文件，请对 XML 重新排序以便首先处理包含您需要的文件版本的元素。第一次出现适用于 <dir> 元素中定义的文件及 <file> 元素中定义的文件。具有相同名称和虚拟位置的文件的第一次出现是从虚拟文件系统返回的文件。

松散应用程序的注意事项

对于所有松散配置的应用程序，这些文件不在磁盘上它们被声明为应该位于其中的层次结构中。如果应用程序直接访问它们自己的资源，并期望这些资源在磁盘上按预期方式排列（具有扩展 war 或 ear 布局），那么它们可能展示意外行为。

可在应用程序中使用 ServletContext.getRealPath 来发现物理资源路径。ServletContext.getRealPath 可发现要打开以读/写入数据的文件路径并获取目录。但是，如果在 Web 应用程序中使用 ServletContext.getRealPath 以获取“/”的路径，那么不能使用此路径在磁盘上浏览该应用程序。

ServletContext.getRealPath 仅允许返回单个物理路径，松散应用程序可能已合并多个目录以形成对该应用程序可视的一个路径。

考虑以下配置：

```
<archive>
    <dir targetInArchive="/"
          sourceOnDisk="c:\myapplication" />
    <dir targetInArchive="/web/pages"
          sourceOnDisk="c:\webpagesforapplication" />
</archive>
```

直接访问 /web/pages 然后向上浏览目录层次结构的应用程序发现物理路径 /web/pages 的父代为 c:\ 而不是 /web。c:\ 没有 pages 目录，也没有父目录。

仅当应用程序尝试直接访问磁盘上的内容并根据磁盘上对应分层布局的假定执行它们自己的路径浏览时，这些注意事项才适用。相同应用程序部署为归档时也会遇到问题。这些应用程序通常会遇到可移植性问题。

复杂示例

以下代码是松散应用程序配置的较复杂示例。此示例使用所有元素并创建文件和目录的复杂映射：

```
<archive>
    <dir targetInArchive="/appResources"
           sourceOnDisk="${was.server.dir}/applicationFiles" />
    <archive targetInArchive="application.jar">
        <dir targetInArchive="/src"
               sourceOnDisk="${was.server.dir}/applicationCode/src" />
    </archive>
    <archive targetInArchive="webApp.war">
        <dir targetInArchive="/META-INF"
               sourceOnDisk="${was.server.dir}/manifestFiles/" />
        <dir targetInArchive="/WEB-INF"
               sourceOnDisk="c:/myWorkspace/webAppProject/web-inf" />
        <archive targetInArchive="/WEB-INF/lib/myUtility.jar">
            <dir targetInArchive="/"
                   sourceOnDisk="c:/myWorkspace/myUtilityProject/src" />
            <file targetInArchive="/someJar.jar"
                    sourceOnDisk="c:/myWorkspace/myUtilityProject/
aJar.jar" />
        </archive>
    </archive>
    <file targetInArchive="/myjar.jar"
            sourceOnDisk="${was.server.dir}/apps/application.zip" />
</archive>
```

2.8.14 在 xigemaAS 服务器上发现 REST API 文档

可发现 REST API 文档。使用 API 发现功能部件查找 xigemaAS 服务器上的可用 REST API，然后使用 Swagger 用户接口调用所发现的 REST 端点。

1. 在您要查找其可用 REST API 的 xigemaAS 服务器的 server.xml 文件中，将 `apiDiscovery-1.0` 功能部件添加至功能部件管理器。

   `apiDiscovery-1.0` 功能部件在产品中启用 REST API 发现捆绑软件。此功能部件还会展示 JMX 之类的 xigemaAS REST 端点（如果服务器配置使用 `restConnector-1.0` 功能部件）。

   确保服务器配置具有所部署应用程序需要的所有功能部件，例如，`servlet-3.0`、`jsp-2.2` 等等。还应确保端口和用户注册表设置对于所部署应用程序而言是正确的。

   以下 server.xml 文件具有 `apiDiscovery-1.0` 功能部件：

   ```
   <server>
       <featureManager>
           <feature>apiDiscovery-1.0</feature>
       </featureManager>

       <httpEndpoint id="defaultHttpEndpoint"
                     host="*"
                     httpPort="8010"
                     httpsPort="8020"/>

       <keyStore id="defaultKeyStore" password="xigemaAS"/>

       <basicRegistry id="basic" realm="ibm/api">
           <user name="bob" password="bobpwd" />
       </basicRegistry>
   </server>
   ```

2. 展示 xigemaAS REST 端点中的 Swagger 2.0 文档。

可通过以下两种方式中的任何一种配置 API 文档的位置：

- 使用 SPI `com.ibm.wsspi.rest.api.discovery.APIProvider` 接口的 `getDocumentation` 方法。

  `getDocument` 方法允许扩展功能部件中的 OSGi 捆绑软件将 REST API 文档添加至整体“主控项”文档。对于此发行版，唯一受支持 DocType 为 `DocType.Swagger_20_JSON` 和 `DocType.Swagger_20_YAML`。此接口的实现者可将序列化 JSON 或 YAML 文档返回为 `java.lang.String` 值，或者可以将文件引用（带有前缀 file:///）传递至 JSON 或 YAML 文件位置。
- 使用所部署 Web 应用程序。

  每个 Web 模块可继续自己的 REST API 文档。企业应用程序 (EAR) 文件内部的多个 WAR 文件可具有不同 Swagger 2.0 文档。

  公开 Web 模块文档的最简单方法是将 `swagger.json` 或 `swagger.yaml` 文件包含在对应的 META-INF 文件夹中。部署应用程序期间，API 发现功能部件会针对每个 Web 模块查找 `META-INF/swagger.json` 值。如果找不到 `META-INF/swagger.json` 值，那么 API 发现功能部件将查找 META-`INF/swagger.yaml` 值。

  展示 Web 模块的 REST API 文档的另一方法是在 server.xml 配置文件中进行。将每个 Web 模块的 `webModuleDoc` 元素放置在父 `apiDiscovery` 元素中；例如：

```
<apiDiscovery>
   <webModuleDoc contextRoot="/30ExampleServletInEar" enabled="true" docURL="/swagger.json" />
   <webModuleDoc contextRoot="/22ExampleServlet" enabled="false" />
   <webModuleDoc contextRoot="/custom" enabled="true" docURL="http://petstore.swagger.io/v2/swagger.json" />
</apiDiscovery>
```

  `webModuleDoc` 元素必须具有 `contextRoot` 属性，此属性唯一标识您要展示其文档的 Web 模块。

  可选属性 `enabled` 用于对 Web 模块开启和关闭 API 发现。此属性的缺省值为 `true`。

  `docURL` 属性指定用于查找 Web 模块文档的位置。`docURL` 值可以正斜杠 (/) 开头，以使 URL 相对于上下文根；例如，/30ExampleServletInEar/swagger.json。`docURL` 值也可以 http 或 https 开头，以表示用于标识文档完整位置的绝对 URL。

  如果 Web 应用程序未提供 swagger.json 或 swagger.yaml 文件，并且应用程序包含 JAX-RS 注释资源，那么可以自动生成 Swagger 文档。服务器配置必须具有 `apiDiscovery-1.0` 功能部件及 `jaxrs-1.1` 或 `jaxrs-2.0` 功能部件；例如：

```
<featureManager>
    <feature>apiDiscovery-1.0</feature>
    <feature>jaxrs-1.0</feature>
</featureManager>
```

  产品将扫描 Web 应用程序中的所有类以查找 JAX-RS 和 Swagger 注释（搜索带有 `@Path`、`@Api` 和 `@SwaggerDefinition` 注释的类）。产品还将在 Web 应用程序部署或启动期间自动生成相应的 Swagger 文档。

  还可以使用 `apiDiscovery-1.0` 功能部件来将先前生成的文档与注释扫描期间找到的文档进行合并。产品在 Web 模块中搜索 META-INF/stub/swagger.json 或 META-INF/stub/swagger.yaml 文件。如果功能部件找到这两个文件其中的任何一个，那么它将生成包含该文件内容以及 Web 模块中所有 JAX-RS 和 Swagger 注释的 Swagger 文档。可以使用此功能部件来记录非 JAX-RS servlet，因为该文档将自动与 JAX-RS 部分合并。

公开 Web 模块的 REST API 文档的另一方法是将 `swagger.json` 或 `swagger.yaml` 添加至 Web 应用程序的上下文根；例如：

```
http://host:http_port/context_root/swagger.json
```

或

```
http://host:http_port/context_root/swagger.yaml
```

对 *context_root*`/swagger.json` 或 *context_root*`/swagger.yaml` 的调用将以所请求 JSON 或 YAML 格式返回上下文根的文档。在 Web 模块启动期间，API 发现功能部件会将所有可用 *context_root*`/swagger.json` 放置到从 /ibm/api/docs 和 /ibm/api/explorer 汇总的文档中。如果 *context_root*`/swagger.json` 不可用，那么 API 发现功能部件会将所有可用 *context_root*`/swagger.yaml` 放置到从 /ibm/api/docs 和 /ibm/api/explorer 汇总的文档中。`apiDiscovery-1.0` 功能部件实际上可以处理 `swagger.json` 或 `swagger.yaml`，这表示，如果 Web 应用程序的 server.xml 文件配置了 `docURL` 属性、META-INF 文件夹中的 swagger.json 或 swagger.yaml 文件或 JAX-RS 和 Swagger 注释，那么对 *context_root*`/swagger.json` 或 *context_root*`/swagger.yaml` 的调用将以所请求 JSON 或 YAML 格式返回另一配置的文档。

3. 发现 API 文档。

   配置 API 文档的位置后，可通过以下方式发现该文档：

   - 使用 `GET https://`*host*`:`*https_port*`/ibm/api/docs` 端点。

     此端点提供有效 Swagger 2.0 文档，其所有可用 xigemaAS REST API 已合并至单个文档。对于要通过程序浏览可用 API 集的使用者应用程序（例如，API 管理解决方案），这很有用。包含带有 `application/yaml` 值的可选 `Accept` 头可以提供 YAML 格式的 Swagger 2.0 文档。此端点具有多基数可选查询参数（名为 `root`），此参数可过滤所发现上下文根。例如，对 `GET https://`*host*`:`*https_port*`/ibm/api/docs?root=/myApp` 的调用将检索仅具有 `myApp` 上下文根的文档的 Swagger 2.0 文档。
   - 使用 `GET https://`*host*`:`*https_port*`/ibm/api/explorer` 端点。

     此端点提供富吸引力的已渲染 HTTP 页面来显示 /ibm/api/docs URL 中的内容。此页面遵循标准在线样本 (*http://petstore.swagger.io/*) 的模式，所以用户可调用 REST API。此端点帮助您探查 xigemaAS 服务器上的可用 REST API，并且可能会从页面调用这些 API。过滤器输入框允许使用上下文根的逗号分隔列表来过滤内容。此过滤的工作方式与 `root` 查询参数类似。可以通过提供所需的输入值并单击“尝试”按钮来测试 API。

   启用 `apiDiscovery-1.0` 功能部件后，`ObjectName` 值为 `WebSphere:feature=apiDiscovery,name=APIDiscovery` 的管理 bean (MBean) 在 xigemaAS `MBeanServer` 中注册。此 MBean 提供以下属性：

   - `DocumentationURL` 属性是 /ibm/api/docs 端点的完整 URL，它显示原始 JSON 或 YAML 文档。
   - `ExplorerURL` 属性是 /ibm/api/explorer 端点的完整 URL，它显示文档的呈现 UI。

   可以使用此 MBean 来了解 REST API 发现是否已启用以及客户机可以在何处访问该功能部件。

预订 xigemaAS REST API 更新

xigemaAS REST API Discovery 功能部件现在展示新 REST API（即 /ibm/api/docs/subscription），它允许用户预订任何 REST API 更新，例如，可用的新 API 或要移除的旧 API。如果用户希望端点中发生由特定 xigemaAS 实例提供的任何更改时立即进行通知，那么这很有用。

启用预订

除基本 apiDiscovery-1.0 配置外，还需要在 server.xml 中配置 websocket-1.0 或 websocket-1.1。

```
<server>
    <featureManager>
        <feature>apiDiscovery-1.0</feature>
        <feature>websocket-1.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                 httpPort="8010"
                 httpsPort="8020"/>

    <keyStore id="defaultKeyStore" password="xigemaAS"/>

    <basicRegistry id="basic" realm="ibm/api">
        <user name="bob" password="bobpwd" />
    </basicRegistry>
</server>
```

/ibm/api/docs/subscription 端点允许格式如下的带有 JSON 有效内容的 POST 请求：

```
{ "docType" : String }
```

其中 String 可以是 Swagger_20_JSON 或 Swagger_20_YAML。返回的 JSON 有效内容概述预订订阅源的类型及其 URL。

示例请求和对应响应

请求：

```
{"docType":"Swagger_20_JSON"}
```

响应：

```
{
  "feedType": "websocket",
  "feedURL": "wss://myserver.com:8020/ibm/api/docs/subscription/
websocket/60db0d79-1863-48f5-a0f9-4fe22a27b82d"
}
```

现在可使用 WebSocket 客户机连接至订阅源 URL。连接后，xigemaAS 服务器中对 REST API 的任何进一步更新将通过 WebSocket 推送。更新为 JSON 或 YAML 格式（取决于预订）。

2.9 监视 xigemaAS 服务器运行时环境

可以使用 monitor-1.0 功能部件来监视服务器运行时环境。

要启用对 xigemaAS 服务器的监视，可以在 server.xml 文件中添加 monitor-1.0 xigemaAS 功能部件。

monitor-1.0 功能部件为用户运行时组件提供监视支持。

- JVM
- Web 应用程序
- 线程池
- JAX-WS 端点
- 会话管理
- 连接池

有关更多详细信息，请参阅 *xigemaAS 功能部件*（见第 906 页）。

1. 添加 monitor-1.0 功能部件，监视便会启动。

   注：如果您在 Java 虚拟机 (JVM) 上不使用服务器脚本 #server.sh # server.bat#来启动服务器，请确保如以下示例中为 JVM 配置了 JavaAgent：agentlib=-javaagent:*<path to xigemaas install>*/bin/tools/ws-javaagent.jar。

2. 监视数据以标准 MXBean 形式报告。
3. 可以使用 JConsole 来连接至 JVM，并通过单击 MXBean 的每个属性来查看性能数据。

   用于监视的 MXBean 如下所示：

   - *WebSphere:type=JvmStats*
   - *WebSphere:type=ServletStats,name=**
   - *WebSphere:type=ThreadPoolStats,name=Default Executor*
   - *org.apache.cxf:type=WebServiceStats,service=*,port=**
   - *WebSphere:type=SessionStats,name=**
   - *WebSphere:type=ConnectionPool,name=**

4. 可选： 传统 PMI MBean (Perf MBean) 附带相同数据。请注意，Perf MBean 已稳定化。

2.9.1 JVM 监视

您可使用 xigemaAS 中用于 JVM 监视的 JvmStats MXBean。

每个 xigemaAS 实例均具有一个 JvmStats MXBean。

用于确定 JVM MXBean 的 ObjectName 是：

```
WebSphere:type=JvmStats
```

可用实例数 = 1

此 MXBean 负责 JVM 的报告性能。下列属性适用于 JVM。

堆信息

- 可用堆大小（以字节计）
- 堆中 JVM 使用的内存总大小（以字节计）
- 堆大小（以字节计）

CPU 信息

- 此 JVM 耗用的 CPU 百分比

垃圾回收 **(GC)** 信息

- JVM 启动后进行的 GC 次数
- GC 活动所花的总时间

常规信息

- JVM 启动后经过的时间（以毫秒计）。

计数器定义（**MXBean** 的属性）

- `Heap`：用于当前 JVM 的堆大小。
- `FreeMemory`：适用于当前 JVM 的可用堆。
- `UsedMemory`：当前 JVM 所使用的堆。
- `ProcessCPU`：JVM 进程使用的 CPU 百分比。
- `GcCount`：JVM 启动后进行的 GC 次数。
- `GcTime`：GC 时间的总累计值。
- `UpTime`：JVM 启动后的时间（以毫秒计）。

管理接口

JVM 监视的管理接口是 `com.ibm.websphere.monitor.jmx.JvmMXBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.9.2 Web 应用程序监视

您可使用 xigemaAS 的用于 Web 应用程序监视的 servlet MXBean。

Web 应用程序中的每个 Servlet 都具有性能数据。每个 servlet 都具有它自己的 MXBean。

用于确定每个 servlet MXBean 的 `ObjectName` 是：

```
WebSphere:type=ServletStats,name=<AppName>.<ServletName>
```

例如：

```
WebSphere:type=ServletStats,name=snoop.Alpine Snoop Servlet
WebSphere:type=ServletStats,name=MyApp.MyServlet
```

此 MXBean 负责报告每个 servlet 的 `ServletStats`。`ServletStats` MXBean 具有下列键数据：

- 请求计数
- 响应时间
- Servlet 名称
- 应用程序名称

计数器定义（**MXBean** 的属性）

- `AppName`：应用程序的名称。
- `ServletName`：servlet 的名称。
- `RequestCount`：此 servlet 的命中次数。

- `ResponseTime`：平均响应时间（纳秒）。
- `Description`：计数器的描述。
- `RequestCountDetails`：`RequestCount` 详细信息，包括最近的时间戳记。
- `ResponseTimeDetails`：`ResponseTime` 详细信息，包括取得的快照数、最小值和最大值。

管理接口

Web 应用程序监视的管理接口是 `com.ibm.websphere.webcontainer.ServletStatsMXBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.9.3 ThreadPool 监视

您可使用 xigemaAS 中用于线程池监视的 `ThreadPool` MXBean。

所有 Web 请求都在线程池（名称为 **Default Executor** 的线程池）中执行。可以使用 `ThreadPoolMXBean` 来监视 **Default Executor** 线程池的使用情况。

用来确定线程池的 MXBean 的 `ObjectName` 为：

```
WebSphere:type=ThreadPoolStats,name=Default Executor
```

适用于 `ThreadPool` 的关键性能数据是：

- 池中的线程数，用以表示池大小。
- 服务请求的活动线程数。

`ThreadPool` 的属性

- `ActiveThreads`
- `PoolSize`
- `PoolName`（仅支持 Default Executor 线程池）

管理接口

`ThreadPool` 监视的管理接口是 `com.ibm.websphere.monitor.jmx.ThreadPoolMXBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.9.4 JAX-WS 监视

您可使用 xigemaAS 用于 JAX-WS 监视的端点 MXBean。

JAX-WS 应用程序中的每个端点和操作都具有性能数据。每个 Web Service 端点都具有其自己的 MXBean。

用于标识每个端点 MXBean 的 `ObjectName` 可能采用下列其中一种格式：

```
org.apache.cxf:bus.id=<bus.name>,type=Performance.Counter.Server,service="<NameSpace><ServiceName>",port="<Po
org.apache.cxf:bus.id=<bus.name>,type=Performance.Counter.Client,service="<NameSpace><ServiceName>",port="<Po
```

例如：

```
org.apache.cxf:bus.id=JaxWsxigemaASDemo-Server-Bus,type=Performance.Counter.Server,
```

```
    service="{http://jaxws.samples/}SimpleEchoService",port="SimpleEchoPort"
org.apache.cxf:bus.id=JaxWsxigemaASDemo-Server-Bus,type=Performance.Counter.Client,
    service="{http://jaxws.samples/}SimpleEchoService",port="SimpleEchoPort"
```

此 MXBean 负责报告每个端点和操作的 Web Service 状态。

计数器定义（**MXBean** 的属性）

- `AvgResponseTime`：平均响应时间（毫秒）。
- `MaxResponseTime`：最大响应时间（毫秒）。
- `MinResponseTime`：最小响应时间（毫秒）。
- `NumInvocations`：对此端点或操作的调用次数。
- `NumChekedApplicationFaults`：已检查应用程序故障数目。
- `NumLogicalRuntimFaluts`：逻辑运行时故障数目。
- `NumRuntimeFaults`：运行时故障数目。
- `NumUnCheckedApplicationFaults`：未检查应用程序故障数目。
- `TotalHandlingTime`：总响应处理时间（毫秒）。

.

管理接口

用于 Web Service 应用程序监视的管理接口是 `org.apache.cxf.management.counters.ResponseTimeCounterMBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关 `org.apache.cxf.management.counters.ResponseTimeCounterMBean` 的更多信息，请参阅 *Apache CXF Javadoc*。

2.9.5 SIP 应用程序监视

会话启动协议 (SIP) 性能监控基础结构 (PMI) 是一个组件，用于收集正在运行的应用程序服务器的 SIP 性能指标。要监视 SIP 指标，必须在服务器上启用 PMI。要为 SIP 启用监视，请将 `monitor-1.0` 和 `sipServlet-1.1` xigemaAS 功能部件添加到 server.xml 文件。

所有 xigemaAS SIP PMI 计数器都由标准 MXBean 显示。

SIP 容器为 SIP 计数器提供以下 MXBean 接口：

- WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic
- WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration
- WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest
- WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest
- WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse
- WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse
- WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor

每个接口都显示一组不同的 SIP PMI 指标。请参阅表以获取模块详细信息。查看 SIP PMI 计数器：

- 创建您自己的 JMX 客户机应用程序，以通过启动 MXBean 操作检查计数器。

SIP 在 PMI 中提供以下计数器以监视 SIP 性能。

表 46: SIP 容器基本计数器

可从中检索计数器的 MXBean 的对象名称为："WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic"。要检索属性，请使用 JMXConnection.getAttribute 方法。例如：_connection.getAttribute("WebSphere:type=SipContainerBasicCounters,name=SipContainer.Basic", "SipAppSessions")。

下表列出 SIP 容器基本计数器。

名称	属性	描述	详细程度
入局流量	ReceivedSipMsgs	容器处理的平均消息数，针对可配置的时间段计算。	服务器
新 SIP 应用程序会话数	NewSipApplications	容器中创建的新 SIP 应用程序会话平均数，针对可配置的时间段计算。	服务器
响应时间	SipRequestProcessing	从消息进入容器到从容器发出响应之间花费的平均时间量。	服务器
队列大小	InvokerSize	产品中调用队列的大小	服务器
Rejected SIP messages	RejectedMessages	已拒绝的 SIP 消息的数量	服务器
SIP 计时器调用	SipTimersInvocations	SIP 计时器（计时器 A、计时器 B、计时器 C、计时器 D、计时器 E、计时器 F、计时器 G、计时器 H）的调用数	服务器
活动 SIP 会话数	SipSessions	属于每个应用程序的 SIP 会话的数量	服务器
活动 SIP 应用程序会话数	SipAppSessions	属于每个应用程序的 SIP 应用程序会话的数量	服务器

表 47: SIP 容器入站请求

可从中检索计数器的 MXBean 的对象名称为："WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest"。要检索计数器，请使用 JMXConnection.invoke 方法。例如，_connection.invoke("WebSphere:type=InboundRequestCounters,name=SipContainer.InboundRequest", "getTotalInboundRequests", _appName, "INVITE")。

此表列出入站请求计数器。

名称	方法	描述	详细程度
入站 NOT SIP STANDARD 请求数	getTotalInboundRequests(appName, “NOTSIPSTANDARD”);	属于每个应用程序的入站 NOT SIP STANDARD 请求的数量	应用程序

名称	方法	描述	详细程度
入站 REGISTER 请求数	getTotalInboundRequests(appName, “REGISTER”);	属于每个应用程序的入站 REGISTER 请求的数量	应用程序
入站 INVITE 请求数	getTotalInboundRequests(appName, “INVITE”);	属于每个应用程序的入站 INVITE 请求的数量	应用程序
入站 ACK 请求数	getTotalInboundRequests(appName, “ACK”);	属于每个应用程序的入站 ACK 请求的数量	应用程序
入站 OPTIONS 请求数	getTotalInboundRequests(appName, “OPTIONS”);	属于每个应用程序的入站 OPTIONS 请求的数量	应用程序
入站 BYE 请求数	getTotalInboundRequests(appName, “BYE”);	属于每个应用程序的入站 BYE 请求的数量	应用程序
入站 CANCEL 请求数	getTotalInboundRequests(appName, “CANCEL”);	属于每个应用程序的入站 CANCEL 请求的数量	应用程序
入站 PRACK 请求数	getTotalInboundRequests(appName, “PRACK”);	属于每个应用程序的入站 PRACK 请求的数量	应用程序
入站 INFO 请求数	getTotalInboundRequests(appName, “INFO”);	属于每个应用程序的入站 INFO 请求的数量	应用程序
入站 SUBSCRIBE 请求数	getTotalInboundRequests(appName, “SUBSCRIBE”);	属于每个应用程序的入站 SUBSCRIBE 请求的数量	应用程序
入站 NOTIFY 请求数	getTotalInboundRequests(appName, “NOTIFY”);	属于每个应用程序的入站 NOTIFY 请求的数量	应用程序
入站 MESSAGE 请求数	getTotalInboundRequests(appName, “MESSAGE”);	属于每个应用程序的入站 MESSAGE 请求的数量	应用程序
入站 PUBLISH 请求数	getTotalInboundRequests(appName, “PUBLISH”);	属于每个应用程序的入站 PUBLISH 请求的数量	应用程序
入站 REFER 请求数	getTotalInboundRequests(appName, “REFER”);	属于每个应用程序的入站 REFER 请求的数量	应用程序
入站 UPDATE 请求数	getTotalInboundRequests(appName, “UPDATE”);	属于每个应用程序的入站 UPDATE 请求的数量	应用程序

表 48: SIP 容器入站响应

可从中检索计数器的 MXBean 的对象名称为：`"WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse"`。要检索计数器，请使用 JMXConnection.invoke 方法。例如，`_connection.invoke("WebSphere:type=InboundResponseCounters,name=SipContainer.InboundResponse", "getTotalInboundResponses", _appName, "100")`。

此表列出入站响应计数器。

名称	方法	描述	详细程度
入站 100 响应数	getTotalInboundResponses (appName, “100”);	属于每个应用程序的入站 100（正在尝试）响应的数量	应用程序
入站 180 响应数	getTotalInboundResponses (appName, “180”);	属于每个应用程序的入站 180（正在响铃）响应的数量	应用程序
入站 181 响应数	getTotalInboundResponses (appName, “181”);	属于每个应用程序的入站 181（呼叫正在转接）响应的数量	应用程序
入站 182 响应数	getTotalInboundResponses (appName, “182”);	属于每个应用程序的入站 182（呼叫已排队）响应的数量	应用程序
入站 183 响应数	getTotalInboundResponses (appName, “183”);	属于每个应用程序的入站 183（会话进行中）响应的数量	应用程序
入站 200 响应数	getTotalInboundResponses (appName, “200”);	属于每个应用程序的入站 200（正常）响应的数量	应用程序
入站 202 响应数	getTotalInboundResponses (appName, “202”);	属于每个应用程序的入站 202（已接受）响应的数量	应用程序
入站 300 响应数	getTotalInboundResponses (appName, “300”);	属于每个应用程序的入站 300（多项选择）响应的数量	应用程序
入站 301 响应数	getTotalInboundResponses (appName, “301”);	属于每个应用程序的入站 301（永久移动）响应的数量	应用程序
入站 302 响应数	getTotalInboundResponses (appName, “302”);	属于每个应用程序的入站 302（临时移动）响应的数量	应用程序
入站 305 响应数	getTotalInboundResponses (appName, “305”);	属于每个应用程序的入站 305（使用代理）响应的数量	应用程序
入站 380 响应数	getTotalInboundResponses (appName, “380”);	属于每个应用程序的入站 380（备用服务）响应的数量	应用程序
入站 400 响应数	getTotalInboundResponses (appName, “400”);	属于每个应用程序的入站 400（错误请求）响应的数量	应用程序

名称	方法	描述	详细程度
入站 401 响应数	getTotalInboundResponses (appName, “401”);	属于每个应用程序的入站 401（未经授权）响应的数量	应用程序
入站 402 响应数	getTotalInboundResponses (appName, “402”);	属于每个应用程序的入站 402（需要付款）响应的数量	应用程序
入站 403 响应数	getTotalInboundResponses (appName, “403”);	属于每个应用程序的入站 403（禁止）响应的数量	应用程序
入站 404 响应数	getTotalInboundResponses (appName, “404”);	属于每个应用程序的入站 404（未找到）响应的数量	应用程序
入站 405 响应数	getTotalInboundResponses (appName, “405”);	属于每个应用程序的入站 405（方法不受允许）响应的数量	应用程序
入站 406 响应数	getTotalInboundResponses (appName, “406”);	属于每个应用程序的入站 406（不可接受）响应的数量	应用程序
入站 407 响应数	getTotalInboundResponses (appName, “407”);	属于每个应用程序的入站 407（需要代理认证）响应的数量	应用程序
入站 408 响应数	getTotalInboundResponses (appName, “408”);	属于每个应用程序的入站 408（请求超时）响应的数量	应用程序
入站 410 响应数	getTotalInboundResponses (appName, “410”);	属于每个应用程序的入站 410（已消失）响应的数量	应用程序
入站 413 响应数	getTotalInboundResponses (appName, “413”);	属于每个应用程序的入站 413（请求实体过大）响应的数量	应用程序
入站 414 响应数	getTotalInboundResponses (appName, “414”);	属于每个应用程序的入站 414（请求 URI 过长）响应的数量	应用程序
入站 415 响应数	getTotalInboundResponses (appName, “415”);	属于每个应用程序的入站 415（不支持的媒体类型）响应的数量	应用程序
入站 416 响应数	getTotalInboundResponses (appName, “416”);	属于每个应用程序的入站 416（不支持的 URI 方案）响应的数量	应用程序

名称	方法	描述	详细程度
入站 420 响应数	getTotalInboundResponses (appName, “420”);	属于每个应用程序的入站 420（错误扩展）响应的数量	应用程序
入站 421 响应数	getTotalInboundResponses (appName, “421”);	属于每个应用程序的入站 421（需要扩展）响应的数量	应用程序
入站 423 响应数	getTotalInboundResponses (appName, “423”);	属于每个应用程序的入站 423（时间间隔过短）响应的数量	应用程序
入站 480 响应数	getTotalInboundResponses (appName, “480”);	属于每个应用程序的入站 480（暂时不可用）响应的数量	应用程序
入站 481 响应数	getTotalInboundResponses (appName, “481”);	属于每个应用程序的入站 481（呼叫支路完成）响应的数量	应用程序
入站 482 响应数	getTotalInboundResponses (appName, “482”);	属于每个应用程序的入站 482（检测到回路）响应的数量	应用程序
入站 483 响应数	getTotalInboundResponses (appName, “483”);	属于每个应用程序的入站 483（跳数过多）响应的数量	应用程序
入站 484 响应数	getTotalInboundResponses (appName, “484”);	属于每个应用程序的入站 484（地址不完整）响应的数量	应用程序
入站 485 响应数	getTotalInboundResponses (appName, “485”);	属于每个应用程序的入站 485（不明确）响应的数量	应用程序
入站 486 响应数	getTotalInboundResponses (appName, “486”);	属于每个应用程序的入站 486（此处忙）响应的数量	应用程序
入站 487 响应数	getTotalInboundResponses (appName, “487”);	属于每个应用程序的入站 487（请求已终止）响应的数量	应用程序
入站 488 响应数	getTotalInboundResponses (appName, “488”);	属于每个应用程序的入站 488（此处不可接受）响应的数量	应用程序
入站 491 响应数	getTotalInboundResponses (appName, “491”);	属于每个应用程序的入站 491（请求暂挂）响应的数量	应用程序

名称	方法	描述	详细程度
入站 493 响应数	getTotalInboundResponses (appName, “493”);	属于每个应用程序的入站 493（不可辨识）响应的数量	应用程序
入站 500 响应数	getTotalInboundResponses (appName, “500”);	属于每个应用程序的入站 500（服务器内部错误）响应的数量	应用程序
入站 501 响应数	getTotalInboundResponses (appName, “501”);	属于每个应用程序的入站 501（未实现）响应的数量	应用程序
入站 502 响应数	getTotalInboundResponses (appName, “502”);	属于每个应用程序的入站 502（错误网关）响应的数量	应用程序
入站 503 响应数	getTotalInboundResponses (appName, “503”);	属于每个应用程序的入站 503（服务不可用）响应的数量	应用程序
入站 504 响应数	getTotalInboundResponses (appName, “504”);	属于每个应用程序的入站 504（服务器超时）响应的数量	应用程序
入站 505 响应数	getTotalInboundResponses (appName, “505”);	属于每个应用程序的入站 505（版本不受支持）响应的数量	应用程序
入站 513 响应数	getTotalInboundResponses (appName, “513”);	属于每个应用程序的入站 513（消息过大）响应的数量	应用程序
入站 600 响应数	getTotalInboundResponses (appName, “600”);	属于每个应用程序的入站 600（全忙）响应的数量	应用程序
入站 603 响应数	getTotalInboundResponses (appName, “603”);	属于每个应用程序的入站 603（拒绝）响应的数量	应用程序
入站 604 响应数	getTotalInboundResponses (appName, “604”);	属于每个应用程序的入站 604（任何地方都不存在）响应的数量	应用程序
入站 606 响应数	getTotalInboundResponses (appName, “606”);	属于每个应用程序的入站 606（任何地方都不可接受）响应的数量	应用程序

表 49: SIP 容器出站请求

可从中检索计数器的 MXBean 的对象名称为：`"WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest"`。要检索计数器，请使用 JMXConnection.invoke 方法。例如，`_con`

nection.invoke("WebSphere:type=OutboundRequestCounters,name=SipContainer.OutboundRequest", "getTotalOutboundRequests", _appName, "INVITE")。

此表列出出站请求计数器。

名称	方法	描述	详细程度
出站 NOT SIP STANDARD 请求数	getTotalOutboundRequests(appName, “NOTSIPSTANDARD”);	属于每个应用程序的出站 NOT SIP STANDARD 请求的数量	应用程序
出站 REGISTER 请求数	getTotalOutboundRequests(appName, “REGISTER”);	属于每个应用程序的出站 REGISTER 请求的数量	应用程序
出站 INVITE 请求数	getTotalOutboundRequests(appName, “INVITE”);	属于每个应用程序的出站 INVITE 请求的数量	应用程序
出站 ACK 请求数	getTotalOutboundRequests(appName, “ACK”);	属于每个应用程序的出站 ACK 请求的数量	应用程序
出站 OPTIONS 请求数	getTotalOutboundRequests(appName, “OPTIONS”);	属于每个应用程序的出站 OPTIONS 请求的数量	应用程序
出站 BYE 请求数	getTotalOutboundRequests(appName, “BYE”);	属于每个应用程序的出站 BYE 请求的数量	应用程序
出站 CANCEL 请求数	getTotalOutboundRequests(appName, “CANCEL”);	属于每个应用程序的出站 CANCEL 请求的数量	应用程序
出站 PRACK 请求数	getTotalOutboundRequests(appName, “PRACK”);	属于每个应用程序的出站 PRACK 请求的数量	应用程序
出站 INFO 请求数	getTotalOutboundRequests(appName, “INFO”);	属于每个应用程序的出站 INFO 请求的数量	应用程序
出站 SUBSCRIBE 请求数	getTotalOutboundRequests(appName, “SUBSCRIBE”);	属于每个应用程序的出站 SUBSCRIBE 请求的数量	应用程序
出站 NOTIFY 请求数	getTotalOutboundRequests(appName, “NOTIFY”);	属于每个应用程序的出站 NOTIFY 请求的数量	应用程序
出站 MESSAGE 请求数	getTotalOutboundRequests(appName, “MESSAGE”);	属于每个应用程序的出站 MESSAGE 请求的数量	应用程序
出站 PUBLISH 请求数	getTotalOutboundRequests(appName, “PUBLISH”);	属于每个应用程序的出站 PUBLISH 请求的数量	应用程序
出站 REFER 请求数	getTotalOutboundRequests(appName, “REFER”);	属于每个应用程序的出站 REFER 请求的数量	应用程序
出站 UPDATE 请求数	getTotalOutboundRequests(appName, “UPDATE”);	属于每个应用程序的出站 UPDATE 请求的数量	应用程序

表 50: SIP 容器出站响应

可从中检索计数器的 MXBean 的对象名称为："WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse"。要检索计数器，请使用 JMXConnection.invoke 方法。例如，_connection.invoke("WebSphere:type=OutboundResponseCounters,name=SipContainer.OutboundResponse", "getTotalOutboundResponses", _appName, "100")。

此表列出出站响应计数器。

名称	方法	描述	详细程度
出站 100 响应数	getTotalOutboundResponses(appName, “100”);	属于每个应用程序的出站 100（正在尝试）响应的数量	应用程序
出站 180 响应数	getTotalOutboundResponses(appName, “180”);	属于每个应用程序的出站 180（正在响铃）响应的数量	应用程序
出站 181 响应数	getTotalOutboundResponses(appName, “181”);	属于每个应用程序的出站 181（呼叫正在转接）响应的数量	应用程序
出站 182 响应数	getTotalOutboundResponses(appName, “182”);	属于每个应用程序的出站 182（呼叫已排队）响应的数量	应用程序
出站 183 响应数	getTotalOutboundResponses(appName, “183”);	属于每个应用程序的出站 183（会话进行中）响应的数量	应用程序
出站 200 响应数	getTotalOutboundResponses(appName, “200”);	属于每个应用程序的出站 200（正常）响应的数量	应用程序
出站 202 响应数	getTotalOutboundResponses(appName, “202”);	属于每个应用程序的出站 202（已接受）响应的数量	应用程序
出站 300 响应数	getTotalOutboundResponses(appName, “300”);	属于每个应用程序的出站 300（多项选择）响应的数量	应用程序
出站 301 响应数	getTotalOutboundResponses(appName, “301”);	属于每个应用程序的出站 301（永久移动）响应的数量	应用程序
出站 302 响应数	getTotalOutboundResponses(appName, “302”);	属于每个应用程序的出站 302（临时移动）响应的数量	应用程序
出站 305 响应数	getTotalOutboundResponses(appName, “305”);	属于每个应用程序的出站 305（使用代理）响应的数量	应用程序

名称	方法	描述	详细程度
出站 380 响应数	getTotalOutboundResponses(appName, “380”);	属于每个应用程序的出站 380（备用服务）响应的数量	应用程序
出站 400 响应数	getTotalOutboundResponses(appName, “400”);	属于每个应用程序的出站 400（错误请求）响应的数量	应用程序
出站 401 响应数	getTotalOutboundResponses(appName, “401”);	属于每个应用程序的出站 401（未经授权）响应的数量	应用程序
出站 402 响应数	getTotalOutboundResponses(appName, “402”);	属于每个应用程序的出站 402（需要付款）响应的数量	应用程序
出站 403 响应数	getTotalOutboundResponses(appName, “403”);	属于每个应用程序的出站 403（禁止）响应的数量	应用程序
出站 404 响应数	getTotalOutboundResponses(appName, “404”);	属于每个应用程序的出站 404（未找到）响应的数量	应用程序
出站 405 响应数	getTotalOutboundResponses(appName, “405”);	属于每个应用程序的出站 405（方法不受允许）响应的数量	应用程序
出站 406 响应数	getTotalOutboundResponses(appName, “406”);	属于每个应用程序的出站 406（不可接受）响应的数量	应用程序
出站 407 响应数	getTotalOutboundResponses(appName, “407”);	属于每个应用程序的出站 407（需要代理认证）响应的数量	应用程序
出站 408 响应数	getTotalOutboundResponses(appName, “408”);	属于每个应用程序的出站 408（请求超时）响应的数量	应用程序
出站 410 响应数	getTotalOutboundResponses(appName, “410”);	属于每个应用程序的出站 410（已消失）响应的数量	应用程序
出站 413 响应数	getTotalOutboundResponses(appName, “413”);	属于每个应用程序的出站 413（请求实体过大）响应的数量	应用程序
出站 414 响应数	getTotalOutboundResponses(appName, “414”);	属于每个应用程序的出站 414（请求 URI 过长）响应的数量	应用程序

名称	方法	描述	详细程度
出站 415 响应数	getTotalOutboundResponses(appName, “415”);	属于每个应用程序的出站 415（不支持的媒体类型）响应的数量	应用程序
出站 416 响应数	getTotalOutboundResponses(appName, “416”);	属于每个应用程序的出站 416（不支持的 URI 方案）响应的数量	应用程序
出站 420 响应数	getTotalOutboundResponses(appName, “420”);	属于每个应用程序的出站 420（错误扩展）响应的数量	应用程序
出站 421 响应数	getTotalOutboundResponses(appName, “421”);	属于每个应用程序的出站 421（需要扩展）响应的数量	应用程序
出站 423 响应数	getTotalOutboundResponses(appName, “423”);	属于每个应用程序的出站 423（时间间隔过短）响应的数量	应用程序
出站 480 响应数	getTotalOutboundResponses(appName, “480”);	属于每个应用程序的出站 480（暂时不可用）响应的数量	应用程序
出站 481 响应数	getTotalOutboundResponses(appName, “481”);	属于每个应用程序的出站 481（呼叫支路完成）响应的数量	应用程序
出站 482 响应数	getTotalOutboundResponses(appName, “482”);	属于每个应用程序的出站 482（检测到回路）响应的数量	应用程序
出站 483 响应数	getTotalOutboundResponses(appName, “483”);	属于每个应用程序的出站 483（跳数过多）响应的数量	应用程序
出站 484 响应数	getTotalOutboundResponses(appName, “484”);	属于每个应用程序的出站 484（地址不完整）响应的数量	应用程序
出站 485 响应数	getTotalOutboundResponses(appName, “485”);	属于每个应用程序的出站 485（不明确）响应的数量	应用程序
出站 486 响应数	getTotalOutboundResponses(appName, “486”);	属于每个应用程序的出站 486（此处忙）响应的数量	应用程序
出站 487 响应数	getTotalOutboundResponses(appName, “487”);	属于每个应用程序的出站 487（请求已终止）响应的数量	应用程序

名称	方法	描述	详细程度
出站 488 响应数	getTotalOutboundResponses(appName, “488”);	属于每个应用程序的出站 488（此处不可接受）响应的数量	应用程序
出站 491 响应数	getTotalOutboundResponses(appName, “491”);	属于每个应用程序的出站 491（请求暂挂）响应的数量	应用程序
出站 493 响应数	getTotalOutboundResponses(appName, “493”);	属于每个应用程序的出站 493（不可辨识）响应的数量	应用程序
出站 500 响应数	getTotalOutboundResponses(appName, “500”);	属于每个应用程序的出站 500（服务器内部错误）响应的数量	应用程序
出站 501 响应数	getTotalOutboundResponses(appName, “501”);	属于每个应用程序的出站 501（未实现）响应的数量	应用程序
出站 502 响应数	getTotalOutboundResponses(appName, “502”);	属于每个应用程序的出站 502（错误网关）响应的数量	应用程序
出站 503 响应数	getTotalOutboundResponses(appName, “503”);	属于每个应用程序的出站 503（服务不可用）响应的数量	应用程序
出站 504 响应数	getTotalOutboundResponses(appName, “504”);	属于每个应用程序的出站 504（服务器超时）响应的数量	应用程序
出站 505 响应数	getTotalOutboundResponses(appName, “505”);	属于每个应用程序的出站 505（版本不受支持）响应的数量	应用程序
出站 513 响应数	getTotalOutboundResponses(appName, “513”);	属于每个应用程序的出站 513（消息过大）响应的数量	应用程序
出站 600 响应数	getTotalOutboundResponses(appName, “600”);	属于每个应用程序的出站 600（全忙）响应的数量	应用程序
出站 603 响应数	getTotalOutboundResponses(appName, “603”);	属于每个应用程序的出站 603（拒绝）响应的数量	应用程序
出站 604 响应数	getTotalOutboundResponses(appName, “604”);	属于每个应用程序的出站 604（任何地方都不存在）响应的数量	应用程序

名称	方法	描述	详细程度
出站 606 响应数	getTotalOutboundResponses(appName, “606”);	属于每个应用程序的出站 606（任何地方都不可接受）响应的数量	应用程序

表 51: SIP 容器任务持续时间计数器

可从中检索计数器的 MXBean 的对象名称为：`"WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration"`。要检索属性，请使用 JMXConnection.getAttribute 方法。例如：`_connection.getAttribute("WebSphere:type=TaskDurationCounters,name=SipContainer.TaskDuration", " AvgTaskDurationOutBoundQueue")`

此表列出任务持续时间模块计数器。

名称	属性/方法	描述	详细程度
出站队列中的平均任务持续时间	`AvgTaskDurationOutBoundQueue`	在所配置的时间窗口内，SIP 堆栈出站队列中的平均任务持续时间	服务器
出站队列中的最长任务持续时间	`MaxTaskDurationOutBoundQueue`	在所配置的时间窗口内，SIP 堆栈出站队列中的最长任务持续时间	服务器
出站队列中的最短任务持续时间	`MinTaskDurationOutBoundQueue`	在所配置的时间窗口内，SIP 堆栈出站队列中的最短任务持续时间	服务器
处理队列中的平均任务持续时间	`AvgTaskDurationInProcessingQueue`	在所配置的时间窗口内，SIP 容器处理队列中的平均任务持续时间	服务器
处理队列中的最长任务持续时间	`MaxTaskDurationInProcessingQueue`	在所配置的时间窗口内，SIP 容器处理队列中的最长任务持续时间	服务器
处理队列中的最短任务持续时间	`MinTaskDurationInProcessingQueue`	在所配置的时间窗口内，SIP 容器处理队列中的最短任务持续时间	服务器
应用程序代码中的平均任务持续时间	getAvgTaskDurationInApplication(String appName)	在所配置的时间段内，SIP 应用程序代码中的平均任务持续时间	应用程序
应用程序代码中的最长任务持续时间	getMaxTaskDurationInApplication(String appName)	在所配置的时间段内，SIP 应用程序代码中的最长任务持续时间	应用程序
应用程序代码中的最短任务持续时间	getMinTaskDurationInApplication(String appName)	在所配置的时间段内，SIP 应用程序代码中的最短任务持续时间	应用程序

表 52: SIP 容器队列监视计数器

可从中检索计数器的 MXBean 的对象名称为："WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor"。要检索属性，请使用 JMXConnection.getAttribute 方法。例如：_connection.getAttribute("WebSphere:type=QueueMonitoringModule,name=SipContainer.QueueMonitor", " TotalTasksCountInProcessingQueue")。

此表列出队列监视计数器。

名称	属性	描述	详细程度
流过处理 SIP 容器队列的任务总数	TotalTasksCountInProcessingQueue	在所配置的时间窗口内，流过处理 SIP 容器队列的任务（例如消息或 SIP 计时器事件）总数	服务器
处理 SIP 容器队列中的最大任务数	PeakTasksCountInProcessingQueue	在所配置的时间窗口内，处理 SIP 容器队列中的最大任务数	服务器
处理 SIP 容器队列中的最小任务数	MinTasksCountInProcessingQueue	在所配置的时间窗口内，处理 SIP 容器队列中的最小任务数	服务器
处理 SIP 容器队列的最大使用百分比	PercentageFullTasksCountInProcessingQueue	在所配置的时间窗口内，处理 SIP 容器队列的最大使用百分比。	服务器
流过出站 SIP 堆栈队列的任务总数	TotalTasksCountInOutboundQueue	在所配置的时间窗口内，流过出站 SIP 堆栈队列的任务总数	服务器
出站 SIP 堆栈队列中的最大任务数	PeakTasksCountInOutboundQueue	在所配置的时间窗口内，出站 SIP 堆栈队列中的最大任务数	服务器
出站 SIP 堆栈队列中的最小任务数	MinTasksCountInOutboundQueue	在所配置的时间窗口内，出站 SIP 堆栈队列中的最小任务数	服务器
出站 SIP 堆栈队列的最大使用百分比	PercentageFullTasksCountInOutboundQueue	在所配置的时间窗口内，出站 SIP 堆栈队列的最大使用百分比。	服务器

2.9.6 会话监视

可以使用 SessionStats MXBean 来监视 xigemaAS 中会话的性能数据。

每个应用程序的会话性能数据作为 MXBean 提供，可通 JMX 对其进行访问。

与单个 Web 应用程序相关联的会话都有自己的 SessionStats MXBean（即，每个 Web 应用程序都有一个 SessionStats MXBean）。

用于标识每个会话 MXBean 的 ObjectName 为：

```
WebSphere:type=SessionStats,name=*
```

例如：

```
WebSphere:type=SessionStats,name=default_host/trade_lite
WebSphere:type=SessionStats,name=default_host/moneybank
```

MXBean 负责报告单个 Web 应用程序的 SessionStats。启用监视之后，以下关键数据可用于 SessionStats MXBean：

CreateCount

创建的会话总数。

LiveCount

内存中当前已高速缓存的会话总数。

ActiveCount

同时处于活动状态的会话总数。如果 xigemaAS 正在处理使用某个会话的请求，那么该会话是活动的。

InvalidatedCount

失效的会话总数。

InvalidatedCountbyTimeout

由于超时而失效的会话总数。

2.9.7 ConnectionPool 监视

可以使用 ConnectionPool MXBean 对 xigemaAS 进行 ConnectionPool 监视。

系统为每个连接池提供性能数据。连接池管理来自数据源和连接工厂的连接。

每个连接管理器具有关联 ConnectionPool MXBean，每个连接管理器有一个 MXBean。

用于标识每个 ConnectionPool MXBean 的 ObjectName 为：

```
WebSphere:type=ConnectionPoolStats,name=<IDENTIFIER_OF_CONNECTION_MANAGER>
```

以下示例显示没有 JNDI 名称的连接池（对于数据源或连接工厂）。如果未指定 JNDI，那么数据源 [default-x] 名称被视为数据源对象。

```
WebSphere:type=ConnectionPoolStats,name=transaction/dataSource[default-0]/connectionManager

<transaction enableLoggingForHeuristicReporting="true" transactionLogSize="2048">
 <dataSource transactional="false">
  <jdbcDriver libraryRef="DerbyLib"/>
  <properties.derby.embedded databaseName="<DIR Path>/<DatabaseName>" createDatabase="create"/>
 </dataSource>
</transaction>
```

提供连接管理器后的示例配置

- 如果未指定显式标识，那么系统根据其父代生成标识

  ```
  WebSphere:type=ConnectionPoolStats,name=dataSource[MyDataSource]/connectionManager[default-0]
  ```

```
<dataSource id="MyDataSource">
 <connectionManager maxPoolSize="10"/>
 <jdbcDriver libraryRef="DB2JCC4LIB"/>
 <properties.db2.jcc .../>
</dataSource>
```

- 如果指定了标识，那么该标识变为标识符

```
WebSphere:type=ConnectionPoolStats,name=connectionManager[Pool2]

<dataSource id="DataSource2" jdbcDriverRef="DB2JCCDriver" connectionManagerRef="Pool2">
 <properties.db2.jcc .../>
</dataSource>
<connectionManager id="Pool2" maxPoolSize="20"/>
```

- 获取类型 2 驱动程序连接池的正确标识。
  - 请确保正在使用池的应用程序调用了 DB2，以便初始化该池。
  - 浏览至 REST 接口以确定要在配置中使用的正确标识。 例如：

    ```
    host:443/xigemaJMXConnectorREST/mbeans
    ```

  为类型 2 驱动程序的连接池指定正确的标识

```
{"objectName":"WebSphere:type=ConnectionPoolStats,name=jdbc/acp01","
    className":"com.ibm.ws.connectionpool.monitor.ConnectionPoolStats","

    URL":"/xigemaJMXConnectorREST/mbeans/WebSphere%3Aname%3Djdbc%2Facp01%

    2Ctype%3DConnectionPoolStats"}
```

ConnectionPool MXBean 负责报告单个连接管理器的 ConnectionPool 统计信息。启用监视之后，以下计数器属性可用于 ConnectionPool MXBean：

CreateCount

创建的连接总数。

DestroyCount

损坏的连接总数。

ManagedConnectionCount

正在使用的 ManagedConnection 对象数。

WaitTime

到授权连接为止的平均等待时间（以毫秒计）。

ConnectionHandleCount

正在使用的 Connection 对象数。

FreeConnectionCount

池中的空闲连接数。

2.9.8 多组件监视

要过滤您想要监视的组件，可以在 xigemaAS 中使用 monitor-1.0 功能部件。必须在 server.xml 文件中配置要过滤的组件。

1. 要指定您想要过滤的组件，请在 server.xml 文件中添加以下代码。

```
<server description="new server">

     <featureManager>
            <feature>jsp-2.2</feature>
            <feature>jdbc-4.0</feature>
            <feature>monitor-1.0</feature>
     </featureManager>

     <monitor  filter="JVM,ThreadPool,WebContainer,Session,ConnectionPool"/>
</server>
```

缺省情况下，如果在 <monitor> 标记中未提供过滤器，那么将监视当前作为 monitor-1.0 功能部件的一部分来监视的所有组件。您可以通过提供组名作为过滤器的一部分来指定您想要监视的组件。

例如：如果您想要仅监视 *JVM* 和 *WebContainer* 组件，请在 server.xml 文件中指定这些组件，如下所示：

```
<monitor filter="JVM,WebContainer"/>
```

2. 停止监视组件。

要停止监视组件，在运行时必须从过滤器组中移除该组件。

例如，以下过滤器配置监视 *JVM*、*ThreadPool*、*WebContainer*、*Session* 和 *ConnectionPool* 组件：

```
<monitor  filter="JVM,ThreadPool,WebContainer,Session,ConnectionPool"/>
```

要停止监视组件 *WebContainer* 和 *Session*，请将这些组件从过滤器配置中移除：

```
<monitor filter="JVM,ThreadPool,ConnectionPool" />
```

3. 在运行时启用对组件进行监视。

如果您想要在运行时启用对特定组件进行监视，那么可以在运行时在监视标记中指定这些组件。

由过滤组件收集的数据将作为 MXBean 提供。有关各种 MXBean 的更多信息，请参阅监视 *xigemaAS* 服务器运行时环境（见第 1612 页）。

> 注：当前，仅在组件级别（例如，*WebContainer*、*ThreadPool* 和 *JVM*）才支持细粒度监视，在计数器级别不支持。

2.9.9 HTTP 访问日志记录

可以为 HTTP 端点配置访问日志设置。

HTTP 访问日志设置

HTTP 访问日志包含由 HTTP 端点所处理的所有入站客户机请求的记录。可以在 HTTP 服务器中启用访问日志记录，或者使用下列两种方式在 xigemaAS 服务器中将其启用：多个端点共用一个日志，或者每个端点一个日志。

> 注：如果未指定属性，那么将使用缺省值。

- 使用公共日志

  要对多个使用公共设置的端点启用日志记录，请将 *httpAccessLogging* 作为 server.xml 文件中的一个顶级元素，然后从多个 httpEndpoint 元素引用该元素：

```
<httpAccessLogging id="accessLogging"/>
<httpEndpoint id="defaulHttpEndpoint" accessLoggingRef="accessLogging"/>
<httpEndpoint id="otherHttpEndpoint" accessLoggingRef="accessLogging" httpPort="9081"
 httpsPort="9444"/>
```

- 对每个端点使用不同的日志

  要对每个端点启用日志记录，请使用 *accessLogging* 子元素并指定一个与其他日志不冲突的文件路径：

```
<httpEndpoint id="defaultHttpEndpoint">
  <accessLogging filepath="${server.output.dir}/logs/http_defaultEndpoint_access.log"/>
</httpEndpoint>
```

HTTP 访问日志格式

在 server.xml 中使用 httpAcccessLogging 或 accessLogging 元素的 *logFormat* 属性来指定此日志格式字符串：

```
<httpAccessLogging logFormat='%h %u %{t}W "%r" %s %b'/>
```

或

```
<httpEndpoint id="defaultHttpEndpoint">
    <accessLogging filepath="${server.output.dir}/logs/http_defaultEndpoint_access.log"
                   logFormat='%h %i %u %t "%r" %s %b' />
</httpEndpoint>
```

2.10 调整 xigemaAS

可以调整 xigemaAS 的参数和属性。

xigemaAS 支持 server.xml 文件中影响应用程序性能的不同属性。可以使用这些参数和属性来实现更好的性能。要针对安全应用程序调整 xigemaAS，请参阅“针对安全应用程序调整 xigemaAS”。

- 调整 JVM。

  不管您是配置开发环境还是生产环境，调整 JVM 都是一个极其重要的调整步骤。针对 xigemaAS 调整 JVM 时，请使用 *`${server.config.dir}` 目录中的 jvm.options 文件*。可以指定要使用的每个 JVM 参数，每行一个选项。有关更多信息，请参阅定制 *xigemaAS* 环境（见第 1114 页）。jvm.options 文件的示例如下所示：

```
-Xms50m
-Xmx256m
```

  对于开发环境，您可能对更快速地启动服务器感兴趣，因此请考虑将最小堆大小设置为较小的值，而将最大堆大小设置为您的应用程序所需的任何值。对于生产环境，将最小堆大小和最大堆大小设置为相同的值时，可以通过避免堆扩展和收缩来提供最佳性能。

- 调整传输通道服务。

  传输通道服务会管理客户机连接、用于 HTTP 的 I/O 处理、线程池以及连接池。对于 xigemaAS 上的应用程序，下列属性适用于不同的元素，而这些元素可用于提高运行时性能和/或可伸缩性。。

`httpOptions` 的 maxKeepAliveRequests

此选项指定启用持续连接时，单个 HTTP 连接上可接受的最大持续请求数。值为 *-1* 时意味着不受限制。此选项支持等待时间较短或者吞吐量较大的应用程序，并且支持在构建新连接可能成本高昂的情况下使用 SSL 连接。以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<httpOptions maxKeepAliveRequests="-1" />
```

`connectionManager` 的 maxPoolSize

此选项指定连接池的最大物理连接数。缺省值为 `50`。这里的优化设置取决于应用程序特征。对于每个线程均获取与数据库的连接的应用程序，可开始于 1:1（映射到 `coreThreads` 属性）。以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<connectionManager ... maxPoolSize="40" />
```

`connectionManager` 的 purgePolicy

此选项指定在池中检测到旧连接时要销毁哪些连接。缺省值为整个池。最好是仅清除失败连接。以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<connectionManager ... purgePolicy="FailingConnectionOnly" />
```

`connectionManager` 的 numConnectionsPerThreadLocal

此选项指定每个执行程序线程要高速缓存的数据库连接数。此设置可以通过为每个线程保留指定数目的数据库连接，在多核 (8+) 机器上提供重大改进。

如果对连接使用线程本地存储器，那么可以提升多线程系统上应用程序的性能。如果将 numConnectionsPerThreadLocal 设置为 `1` 或以上，那么每个线程的这些连接都存储在线程本地存储器中。使用 numConnectionsPerThreadLocal 时，考虑另外两个值：

- 应用程序线程数
- 连接池最大连接数

为实现最佳性能，如果具有 `n` 个应用程序线程，那么必须将最大池连接数至少设为 `n` 乘以 numConnectionsPerThreadLocal 属性的值，且必须为所有连接请求使用相同凭证。例如，如果使用 20 个应用程序线程，请将最大池连接数设置为 20 或以上；如果将 numConnectionPerThreadLocal 属性的值设置为 `2` 并且有 20 个应用程序线程，那么将最大池连接数设置为 40 或以上。以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<connectionManager ... numConnectionsPerThreadLocal="1" />
```

`dataSource` 的 statementCacheSize

此选项指定每个连接的高速缓存预编译语句的最大数目。要设置此选项，请完成下列必备步骤：

- 查看应用程序代码（或您从数据库或数据库驱动程序收集的 SQL 跟踪）以找出所有独有预编译语句。
- 请确保高速缓存大小大于语句数。

以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<dataSource ... statementCacheSize="60" >
```

dataSource 的 isolationLevel

数据源隔离级别指定数据完整性和并发性的程度，这又控制数据库锁定的级别。提供四个不同的选项，顺序如下：最佳性能（最低完整性）到最低性能（最佳完整性）。

TRANSACTION_READ_UNCOMMITTED

可以进行脏读取、不可重复读取和幻象读取。

TRANSACTION_READ_COMMITTED

脏读取受到阻止；可以进行不可重复读取和幻象读取。

TRANSACTION_REPEATABLE_READ

脏读取和不可重复读取受到阻止；可以进行幻象读取。

TRANSACTION_SERIALIZABLE

脏读取、不可重复读取和幻象读取受到阻止。

以下示例说明了如何在 server.xml 文件中对此选项进行编码：

```
<dataSource ... isolationLevel="TRANSACTION_READ_COMMITTED">
```

- 调整缺省执行程序。

  xigemaAS缺省执行程序可自我调整，并通过动态添加或移除线程以适应当前工作负载。对于大部分工作负载，此执行程序不需要任何调整，建议您不要更改执行程序的任何设置，除非您遇到有关创建线程的特定问题。

  必要时，可在 server.xml 文件中配置 executor 元素的 coreThreads 和 maxThreads 参数以设置 xigemaAS自动调整代码的下限和上限。coreThreads 设置通常不是必需的，因为此执行程序包含积极的反死锁代码，此代码会添加线程以使此执行程序摆脱死锁情况。极少情况下，反死锁代码添加的线程数超过必需数目。在此情况下，可使用 executor 元素的 maxThreads 参数来阻止允许此执行程序创建的线程数。
- 缩短 servlet 的响应时间。

  要缩短 servlet 的响应时间，请将以下属性添加至 server.xml 文件：

  ```
  <webContainerskipMetaInfResourcesProcessing="true"/>
  ```
- 缩短空闲服务器 CPU 时间。

  要缩短空闲服务器 CPU 时间，请将以下属性添加至 server.xml 文件：

  ```
  <applicationMonitor dropinsEnabled="false" updateTrigger="disabled"/>
  <config updateTrigger="disabled"/>
  ```

  添加属性时，服务器不再监视配置或应用程序更新。
- 调整启动时间。

CDI 1.2

缺省情况下，CDI 1.2 功能部件会扫描所有应用程序归档。CDI 1.2 功能部件可能会大幅增加启动时间，并对较大应用程序有极大影响。可通过将 enableImplicitBeanArchives 值设置为 false 禁用针对注释的隐式归档扫描。此设置会跳过归档扫描，除非它们包含 beans.xml 文件。

```
<cdi12 enableImplicitBeanArchives="false"/>
```

👉 注：即使 cdi-1.2 功能部件不在 server.xml 文件的 <features> 部分中，也可能包括此功能部件，因为其他功能部件（例如，webProfile-7.0 和 javaee-7.0）包括 cdi-1.2 功能部件。在 messages.log 文件中查找“服务器已安装以下功能部件：”来了解是否安装了 cdi-1.2。

2.10.1 针对安全应用程序调整 xigemaAS

可以调整 xigemaAS 以最大程度地提高安全应用程序的性能。

如果要保护 xigemaAS 应用程序环境，那么了解安全性对性能的影响很重要。在应用程序服务器环境中，在应用安全设置的情况下运行应用程序往往会降低性能，因为诸如加密、认证和授权等安全任务都会增加处理器的使用率。这些服务往往会延长应用程序服务器请求的路径，这样每个请求都需要更多资源，并因此减小应用程序吞吐量。

在大多数情况下，可以通过性能调优来减少或消除这种与安全相关的性能损失。您可以调整安全服务所使用的资源，并选择仅使用特定应用程序或环境所需要的安全服务。如果要实现尽可能最好的性能，而不牺牲任何必要的安全性，那么需要了解网络拓扑以及应用程序的安全需求。

- 选择要进行加密的连接。

  在 xigemaAS 环境中，您可以对以下传输进行加密：

  - 通向 Web 服务器的 HTTP 流量
  - 从 Web 服务器到应用程序服务器的 HTTP 流量
  - SOAP/JMX 流量
  - 文件传输服务
  - HTTP 上的 Web Service

  当您确定要通过加密连接来传输的流量时，请考虑正在连接通信机器的网络是专用网络还是公用网络。设置安全连接以及对通过该连接的流量进行加密和解密涉及大量资源。例如，可以不要求在安全网络上进行加密，这样可以大幅提高性能。如果应用程序不要求对客户机到 HTTP 服务器之间以及 HTTP 服务器到应用程序服务器之间的流量进行加密，那么可以仅对客户机到 HTTP 服务器的通信使用 SSL，并因此减少安全所需的资源。
- 启用片上高级加密标准 (AES) 加密。

  如果使用的是 IBM® SDK Java™ Technology Edition V7 SR3 或更高版本，并且正在支持高级加密标准 (AES) 新指令 (AES-NI) 指令集的 Intel™ 处理器上运行，那么可以利用片上 AES 加密来实现性能提升。使用这些功能部件，您可以利用硬件指令（不需要额外软件）来运行 AES 加密和解密。

  要启用 AES-NI，请将以下属性添加到 JVM 命令行或 jvm.options 文件：

  ```
  com.ibm.crypto.provider.doAESInHardware=true
  ```

  将以下属性添加到 JVM 命令行或 jvm.options 文件以验证处理器是否支持 AES-NI 指令集：

  ```
  com.ibm.crypto.provider.AESNITrace=true
  ```
- 选择密码密钥长度。

  在某些情况下，由应用于特定数据类型传输的规则来控制密码密钥的位长度。在这些情况下，可以预先确定您为特定 SSL 连接选择的密码和密钥长度。对于未规定密钥长度的情况，必须选择相应的资源来分配给安全性，以免性能下降幅度超出必要的幅度。例如，256 位密码较之相应的 128 位密码提供更强的加密。但是，使用更强的密码来对消息进行解密时需要更多的处理时间。

选择加密强度时，请考虑通过网络传输的数据类型。例如，诸如财务或医疗记录等敏感信息需要最高级别的安全性。此外，也请考虑谁要访问网络。如果网络受防火墙保护，请考虑降低密码强度，或者可能的话通过已解密连接来传输数据。

有关在 xigemaAS 上配置 SSL 设置的更多信息，请参阅 *SSL* 配置属性（见第 1279 页）。

- 设置连接保持活动请求数。

  在安全套接字层 (SSL) 协议中，初始握手使用公用密钥密码来交换专用密钥以获取速度更快的专用密钥密码。这个速度更快的密码用来对初始握手以后的通信进行加密和解密。因为用于后续通信的密码在速度上快于初始握手的密码，所以从性能的角度，限制应用程序服务器中执行 SSL 握手的次数很重要。要获得此结果，可以延长 SSL 连接的长度，或者加强会话亲缘关系。

  延长单一 SSL 连接持续时间的一种方式是启用持久 HTTP 保持活动连接。延长持续时间可以阻止在连续请求上进行 SSL 握手。可以通过确认 server.xml 文件的 httpOptions 元素中的 keepAliveEnabled 属性已设为 `true` 来确保已启用持续连接。缺省值为 `true`。

  调整持续连接的另一种方式是在单一 HTTP 连接上设置最大的连续请求数目。如果客户机正在连续地发出 100 个以上的请求，请考虑在 server.xml 文件中增大 httpOptions 元素的 maxKeepAliveRequests 属性值。缺省值为 100。

- 设定认证高速缓存设置。

  因为创建认证主体可能会增大处理器使用率，所以 xigemaAS 会在成功认证用户之后提供认证高速缓存来存储主体。要充分利用此服务来加强性能，必须确保已根据用户和应用程序来打开和调整此服务。

  - 确保没有禁用认证高速缓存。缺省情况下，会启用认证高速缓存来帮助提高性能。
  - 考虑更改认证高速缓存超时值。增大超时值可让主体在认证高速缓存中保留更长时间，并减少所需的重新认证次数。但是，增大超时值会增加用户许可权相对于已修改的外部存储库（例如 LDAP）变过时的风险。设置认证高速缓存超时以反映客户机会话的估计长度。要指定高速缓存超时，可以将超时属性的值设为您在 server.xml 文件的 authCache 元素中选择的任何时间。缺省值为 600 秒。
  - 最后，如果您的认证时间长于预期时间，或者您注意到通向外部认证存储库的流量超过预期，那么认证高速缓存可能已满。如果认证高速缓存已满，那么会将主体收回。已认证的用户到认证高速缓存条目之间没有一对一的映射。高速缓存中每个用户的条目数目取决于其他安全性配置。最佳实践是认证高速缓存的最大大小大于同时对服务器进行访问的不同已认证用户数目。以此方式设置认证高速缓存的最大大小有助于防止在超时之前将主体从高速缓存中收回。要更改认证高速缓存的最大大小，您可以在 server.xml 文件的 authCache 元素中设置 maxSize 属性的值。缺省大小为 25000。

  有关更多信息，请参阅在 *xigemaAS* 上配置认证高速缓存（见第 1306 页）。

- 配置 HTTP 会话亲缘关系设置。

  安全应用程序环境中最耗性能的操作之一是初始设置，包括 SSL 握手和认证。要防止 SSL 握手和重新认证增加处理器使用率，确保配置 HTTP 会话亲缘关系很重要。

  HTTP 会话亲缘关系确保将连续的客户机请求路由到同一个应用程序服务器。HTTP 会话亲缘关系以各种方式提升性能，但确切地说，它会防止重新认证和 SSL 握手增加处理器使用率。有关设置 HTTP 会话亲缘关系的指示信息，请参阅 HTTP Server 或 Load Balancer 的文档。

  有关更多信息，请参阅为 *xigemaAS* 配置会话持久性（见第 1145 页）。

2.10.2 在 xigemaAS 中调整联合 LDAP 存储库

可通过监视并调整高速缓存及 server.xml 文件中的上下文池元素来改进联合 LDAP 存储库的性能。

LDAP 存储库的高速缓存查询结果可节省您的时间，因为每次执行 LDAP 操作时不必从后端服务器检索该数据。LDAP 高速缓存属性存储在 <ldapCache> 元素中以便更迅速的访问。必须监视高速缓存的状态并调整高速缓存控制参数以改进高速缓存性能。可调整上下文池参数以改进对 LDAP 服务器的并行访问的性能。

- 在 server.xml 文件中配置 <ldapCache> 元素。

  指定 LDAP 高速缓存控制参数以改进性能：

attributesCache

<size>：指定存储在高速缓存中的实体数。可根据业务需求增加高速缓存的大小，例如，如果业务方案中需要更多数目的实体，请增加高速缓存大小。

<timeout>：指定失效前结果可高速缓存的时间。如果后端 LDAP 数据频繁刷新以保持最新高速缓存，请设置更小的超时时段值。

<sizeLimit>：指定每个实体可存储在高速缓存中的最大 LDAP 属性数。如果一个实体与许多属性相关联，请增加 <sizeLimit> 值。

searchResultSizeLimit

指定可存储在高速缓存中的最大搜索结果数。使用 <searchResultSizeLimit> 元素中的参数来调整作为查询一部分返回的搜索结果。

- 配置 server.xml 文件中的 <contextPool> 元素参数以改进对 LDAP 服务器的并行访问的性能。

  可在 <contextPool> 元素中调整以下参数以控制高速缓存：

contextPool

<initialSize>：指定上下文池的初始大小。必须根据存储库上的负载来设置该值。如果预计对 LDAP 服务器的初始请求数很高，请增加初始大小的值。

<maxSize>：指定最大上下文池大小。必须根据存储库上的负载来设置该值。如果要限制与 LDAP 服务器的连接数，请将 <maxSize> 元素值设置为小于 LDAP 服务器可处理的最大连接数的一半。

<timeout>：指定上下文池超时之前的时段。指定较短超时值以便可在指定时段超时后对 LDAP 服务器进行新连接。例如，如果在所配置时间间隔后所建立连接超时，请设置短于防火墙超时时段的时段，以便重新建立连接。

<waitTime>：指定上下文池超时之前的等待时间。如果指定的值很高，那么建立与 LDAP 服务器的连接所花的时间会相应增加。

2.11 xigemaAS故障诊断提示

xigemaAS 故障诊断提示。

为帮助您确定并解决问题，本产品包括统一的记录组件。请参阅日志记录和跟踪（见第 1649 页）。

如果您收到异常消息，那么可以在 消息（见第 1675 页）中找到有关该消息的信息。

下列两个主题中提供了使用 xigemaAS 时适用的主要已知限制的详细信息：

- 运行时环境已知问题和限制（见第 1668 页）.

下面一组提示可帮助您对常见问题进行故障诊断。

- *JDK* 故障诊断
- 安全性故障诊断（见第 1639 页）
- *LDAP* 故障诊断（见第 1641 页）
- *SSL* 故障诊断（见第 1642 页）
- 对日志记录和跟踪进行故障诊断（见第 1642 页）
- 对 *JAX-WS* 进行故障诊断（见第 1643 页）
- 对 *WS-Security* 进行故障诊断（见第 1646 页）
- 诊断性能（见第 1647 页）
- 对 *SAML* 进行故障诊断

检查 JDK 是否处于支持的级别

如果您遇到尚不具有现成解释的问题，请检查您正在使用的 JDK 是否符合 Java™ V6 或更高版本的标准，以及是否处于当前服务级别。请参阅支持的最低 *Java* 级别（见第 1669 页）。

注：使用基于 Oracle 的 JVM（使用 Java™ V6）时，可能会发生死锁。如果是使用受影响的 JVM 或 JDK，那么下列设置可以帮助您防止发生死锁：

- 启用以下 VM 选项：`-XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass`
- 要设置 Equinox 框架选项以将类名锁定用于类装入，请设置以下 Equinox 配置选项：`-Dosgi.classloader.lock=classname`

启动 xigemaAS 服务器时，可以在 Java™ 属性文件（例如，jvm.options）中进行这些设置。

安全性故障诊断

此部分描述了一些常见安全问题以及您可以选择的解决方案。

SESN0008E: 认证为匿名的用户尝试访问以下用户所拥有的会话：**LdapRegistry/cn=steven,o=myCompany,c=CN**。

如果未认证的用户尝试访问由已认证的用户所创建的会话，那么会发生此错误。缺省情况下启用的会话安全性会阻止对会话进行未授权访问。仅创建会话的用户可以访问该会话。

如果使用 JSP（例如，login.jsp）进行表单登录，而且浏览器发送的 SSO 令牌到期，那么可能会发生此错误。因为 SSO 令牌到期，所以会提示用户使用针对表单登录而配置的 login.jsp 页面重新登录。缺省情况下，jsp 页面会尝试获取会话。此会话最初是由其令牌已到期的用户创建。但是，访问此会话时，令牌到期，用户未获认证以及未建立凭证，都会导致此错误。

要避免此错误，请重新启动那个启动新会话的浏览器，或者将 login.jsp 文件配置成缺省情况下不创建会话。如果选择更新 login.jsp 文件，请设置 `<%@ page session="false" %>`。

CWWKS9104A: 对 **{2}** 调用 **{1}** 时，授权用户 **{0}** 失败。未向用户授予访问任何必需角色的权限：**{3}**。

如果您对应用程序所需的角色没有权限，那么会发生此错误。请确保用户或用户所属的组已至少映射到错误消息中所提及的其中一个角色。ibm-application-bnd.xmi/xml 文件中所定义的用户到角色映射优先于 server.xml 文件中所定义的映射。检查两个资源以确保定义了正确的映射。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。

CWWKZ0013E: 无法启动名为 **{0}** 的两个应用程序，其后是异常安全行为和错误消息，例如 **CWWKS9104A**。

如果使用 `application` 元素在 server.xml 中和 dropins 文件夹中都指定您的应用程序，那么会发生此错误。因此，会尝试安装您的应用程序两次，这样 server.xml 文件中的安全性配置可能会生效，也可能不会生效。要修正此问题，必须将您的应用程序从 dropins 文件夹中移除，然后将其复制到另一个目录。如果必须将您的应用程序保留在 dropins 文件夹中，那么必须通过在 server.xml 文件中使用下列代码来禁用应用程序监视：

```
<applicationMonitor dropinsEnabled="false" />
```

未认证的用户可以访问我的 **servlet** 或 **JSP**。

如果认证失败或者您的 servlet 或 JSP 未受保护，那么将创建主体为 UNAUTHENTICATED 的用户。如果您未定义任何安全性约束或者您将 EVERYONE 特殊主体集映射到您的应用程序所需的角色，那么未认证的用户可以访问您的 servlet 或 JSP。复审 ibm-application-bnd.xmi/xml 和 server.xml 文件中用户到角色的映射。采用下列其中一个选项：

- 如果您的 servlet 或 JSP 未受保护，请在应用程序的部署描述符中添加安全性约束。请参阅认证（见第 1035 页）。
- 如果不想任何未认证的用户访问您的应用程序，请将 EVERYONE 特殊主体集从该角色的映射中移除。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。
- 如果不能授权某一用户访问您的 servlet 或 JSP，请通过检查 ibm-application-bnd.xmi/xml 和 server.xml 文件来复审映射到该角色的用户。您可以将角色映射到用户、组或特殊主体集。如果角色已映射到 EVERYONE 特殊主体集，那么会向所有用户授予访问权。如果角色已映射到 ALL_AUTHENTICATED 特殊主体集，那么会向所有已认证的用户授予访问权。如果要进一步限制可以访问您的 servlet 或 JSP 的用户，请移除这些特殊主体集。最后，检查用户所属的组。虽然用户可能没有显式访问权，但组可能已映射到该角色。在这种情况下，将用户从获授权的组中移除或者将组从角色映射中移除。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。

单点登录 **(SSO)** 不起作用。

如果 SSO 不起作用，请确保不同的 xigemaAS 概要文件服务器（使用 `webAppSecurity` 元素的相同 LTPA 密钥、密码以及 ssoCookieName 属性）具有相同的世界时间 (UTC) 且共享同一个用户注册表。此外，如果令牌到期或者如果在以不一致的方式更改用户注册表（例如，修改域或移除 cookie 所表示的用户）后从 Web 浏览器发送 cookie，那么 SSO 会失败并且用户会收到重新输入凭证信息的提示。请参阅针对 *xigemaAS* 使用 *LTPA cookie* 来定制 *SSO* 配置（见第 1316 页）。

调试安全性公用 **API**。

装入 `WSSecurityHelper` 和 `RegistryHelper`，即使未启用安全性亦如此，例如，即使未指定安全性功能部件 - `appSecurity-1.0`或`appSecurity-2.0`。如果未启用安全性，那么 `WSSecurityHelper.isServerSecurityEnabled()` 和 `WSSecurityHelper.isGlobalSecurityEnabled()` 方法都返回 false，而 `RegistryHelper.getUserRegistry()` 方法返回空值。

如果未启用安全性，那么可能不会装入其他安全性公用 API 类。如果尝试访问这些类，并调用其中一个类上的方法，那么您可能会收到 `java.lang.NoClassDefFoundError` 异常。

要避免收到 `java.lang.NoClassDefFoundError` 异常，必须先通过调用 `WSSecurityHelper.isServerSecurityEnabled()` 或 `WSSecurityHelper.isGlobalSecurityEnabled()` 类来测试以查

看是否启用了安全性，然后只有在启用了安全性的情况下才调用其他安全性公用 API 类。有关此编码方法的示例，请参阅 安全性公共 *API*（见第 1060 页）。

无法认证使用 **Unicode** 字符的用户

如果要对名称中包含 Unicode 字符的用户进行认证，必须通过将以下 jvm 选项添加至服务器启动命令以将 xigemaAS 服务器使用的字符编码类型设置为 UTF-8：

```
-Dclient.encoding.override=UTF-8
```

还必须在登录页面中指定 `charset` 和 `pageEncoding`。以下是在登录 JSP 页面上指定这些参数的示例：

```
<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

java.lang.annotation.AnnotationFormatError: java.lang.IllegalArgumentException:Wrong type at constant pool index at sun.reflect.annotation.AnnotationParser.parseAnnotations(AnnotationParser.java:87)

OpenID Connect 或 OAuth 提供者使用数据库存储器向某些版本的 JDK 7 注册客户机时，可能发生此错误。

为避免此问题，请升级至 JDK V7.1。

LDAP 故障诊断

此部分描述了一些常见 LDAP 问题以及您可以选择的解决方案。

FFDC1015I: 已创建 **FFDC** 事件：**javax.naming.ServiceUnavailableException**：**myldapserver.mycompany.com**：**636**；套接字已关闭 **com.ibm.ws.security.registry.ldap.internal.LdapRegistry 298**

messages.log 中的此消息指示无法访问所配置的 LDAP 服务器。请检查 LDAP 服务器以验证它是否处于运行状态，以及验证它的 IP 地址是否可以从 xigemaAS 服务器进行访问。

javax.naming.CommunicationException：简单绑定失败：**myldapserver.mycompany.com**：**636 [**根异常是 **javax.net.ssl.SSLHandshakeException**：**com.ibm.jsse2.util.g**：**PKIX** 路径构建失败：**java.security.cert.CertPathBuilderException**：找不到所请求目标的有效认证路径**]**

如果在 LDAP 服务器上启用 SSL，而未将 LDAP 服务器的签署者复制到 `LDAPSSLSettings` 元素中所引用的信任库，那么与 LDAP 服务器的连接会失败，且产生 SSL 握手错误。请确保将 LDAP 服务器的签署者复制到信任库。

javax.naming.CommunicationException：**myldapserver.mycompany.com**：**389 [**根异常是 **java.net.BindException**：地址已在使用中：连接**]**

此消息可能出现在 FFDC 日志中，并指示本地服务器上的可用套接字已耗尽，这可能会导致无法连接至 LDAP 服务器。请确保套接字未使用，然后再试一次。

CWWKS1100A: 用户标识 **xxxxx** 认证失败。指定的用户标识或密码无效

即使消息中提及的用户在高负载的 LDAP 服务器上是有效的，服务器上也可能会发生此 FFDC 异常。借助 LDAP 配置，您可以使用开发者工具来添加 `reuseConnection=false` 属性或者将其禁用。要修正问题，请将此属性设置为缺省值 `true`。

SSL 故障诊断

此部分描述了一些常见 SSL 问题以及您可以选择的解决方案。

CWWKS9105E: 无法确定用于自动重定向的 **SSL** 端口。

如果您尝试访问的应用程序会重定向至 SSL 端口，但 SSL 端口不可用，那么会发生此错误。该端口可能会由于缺少 SSL 配置或 SSL 配置定义发生某些错误而不可用。请在 server.xml 文件中检查 SSL 配置，以确保它存在并且是正确的。请参阅保护与 *xigemaAS* 的通信（见第 1278 页）。

FFDC1015I: 在 **ffdc_12.04.18_16.09.42.0.log** 中创建了 **FFDC** 事件：**"java.lang.IllegalArgumentException**：配置未知、不完整：缺少标识字段 **com.ibm.ws.config.internal.cm.ManagedServiceFactoryTracker aSyncReadNupdate**。尝试读取配置并更新 **ManagedServiceFactory** 时抛出异常。异常 **= java.lang.IllegalArgumentException**：配置未知、不完整：缺少标识字段**"**

如果配置中存在不含标识字段的 `keystore` 元素，那么会发生此错误。如果使用最低 SSL 配置，请将标识字段设置为 defaultKeyStore。

如果使用启用了 **SSL** 的 **LDAP** 用户注册表，但未指定 **sslRef** 值，那么可能会收到异常。

例如，配置将 `sslEnabled` 设置为 true 但没有 `sslRef` 值，如以下示例中所示：

```
<ltldapRegistry id="ldap" realm="SampleLdapIDSRealm"
                                          host="ccwin12.austin.vsettan.com.c
n" port="636" ignoreCase="true"
baseDN="o=vsettan,c=cn"
bindDN="cn=root"
bindPassword="rootpwd"
ldapType="IBM Tivoli Directory Server"
idsFilters="ibm_dir_server"
sslEnabled="true"
searchTimeout="8m" />
```

必须输入 `sslRef` 值。如果使用的是类似如下的最低 SSL 配置：

```
<ltkeyStore id="defaultKeyStore" location="key.jks"
password="mypassword" />
```

`sslRef` 字段应该设置为 `defaultSSLConfig`。

如果已设定定制 SSL 配置，那么应该将该配置的名称放入 `sslRef` 字段。

对日志记录和跟踪进行故障诊断

此部分描述了日志记录和跟踪的一些常见问题。

java.util.logging -- Java 日志记录编程接口。

xigemaAS 概要文件不支持使用 logging.properties 文件配置 java.util.logging。使用 java 代码（例如，在部署的应用程序或用户界面中使用），来创建和添加 java.util.logging 处理程序、过滤器或格式化程序。

由于 xigemaAS 概要文件服务器根据日志记录配置元素的 `traceSpecification` 属性管理 java.util.logging 记录器级别，因此应避免使用 Logger.setLevel 方法。

对 JAX-WS 进行故障诊断

本节描述了一些常见 JAX-WS 问题以及您可以选择的解决方案。

在 **xigemaAS** 概要文件上部署 **Web Service** 应用程序时，发生 **org.apache.cxf.bus.extension.ExtensionException**。

如果应用程序中已嵌入 CXF 库和配置，并且您想要将该应用程序部署到 xigemaAS，那么必须确保通过从 server.xml 文件中移除 `jaxws-2.2` 服务器功能部件来禁用此功能部件。

将二进制文件从 **MTOM** 服务取回到 **MTOM** 客户机时返回空文件。

如果 MTOM 客户机已将二进制文件成功发送至 MTOM 服务，但是从 MTOM 服务取回这些二进制文件时返回空文件，就会发生这种情况。

作为一种变通方法，您可以创建新的 `DataHandler`，然后通过填充二进制文件的内容来初始化该 `DataHandler`。例如，使用 `FileDataStore` 对象从 I/O 进行读取，返回新创建的 `DataHandler`，然后将该 `DataHandler` 传递到其他 Web Service。

```
...
File f = new File("received_image");
  if (f.exists()) {
      connection.start();
 }

  FileOutputStream fos = new FileOutputStream(f);
        callerSubject.getPublicCredentials(WSCredential.class);

  FileDataSource fos_out = new FileDataSource(f);
  DataHandler img_out = new DataHandler(fos_out);
  return img_out;
...
```

将采用 **xsd:gMonth** 格式的 **XMLGregorianCalendar** 与 **Oracle JVM** 配合使用时发生了 **javax.xml.ws.soap.SOAPFaultException**：取消编组错误。

如果您使用 `XMLGregorianCalendar` 类将 `gMonth` 格式的常量作为客户端的 SOAP 请求的一部分来存储，并且在具有 Oracle JVM 的 xigemaAS 上启用了 `jaxws-2.2` 功能部件，就会发生此错误。造成此错误的根本原因是，对于 `xsd:gMonth` 类型，Oracle JVM 支持新的标准格式 *--MM*，而最新的 JAXB RI（参考实现）仅支持 *--MM--* 格式。

要解决此问题，可以针对客户端应用程序和服务器端应用程序将 Oracle JVM 更改为 IBM® JVM。

解析由 **wsdlLocation** 属性指定的 **WSDL** 文件时发生了 **java.io.FileNotFoundException**。

如果您像在 `wsdlLocation = "file:/WEB-INF/wsdl/a.wsdl"` 代码中一样指定 WSDL 文件，并且该 WSDL 文件打包在应用程序中，就会发生此错误。如果您想要引用打包在应用程序中的 WSDL 文件，那么必须对 `@WebService`、`@WebServiceProvider`、`@WebServiceClient` 或 `@WebServieRef` 注释中定义的 wsdlLocation 属性使用相对 URI。

`wsdlLocation` 属性的相对 URI 必须为下列其中一个值：

- `wsdlLocation = "/WEB-INF/wsdl/a.wsdl"`
- `wsdlLocation = "WEB-INF/wsdl/a.wsdl"`

messages.log 文件中出现大量**"Creating service"**消息。

当您调用所生成客户机存根类中的 `getPort` 或相关方法时，会发生这种情况。xigemaAS 配置为使用缺省日志记录配置，并且所有参考消息都写入 messages.log 文件。出现了大量消息，可能类似如下：

```
00000037 org.apache.cxf.service.factory.ReflectionServiceFactoryBean I Creating Service {http
://www.echo.org/}EchoService from WSDL:
    wsjar:file:/wlp/usr/servers/default/workarea/org.eclipse.osgi/bundles/100/data/cache/
    com.ibm.ws.app.manager_gen_15a42559-f979-4ee6-b488-9fa1fb212c96/.cache/Echo.war!/WEB-INF/
wsdl/Echo.wsdl
```

要禁止这些参考消息，可以在 server.xml 文件中更改日志记录配置，如下所示：

```
<logging traceSpecification="org.apache.cxf.*=warning=enabled"/>
```

SOAPFaultException：给定的 **SOAPAction** ***test*** 与客户机应用程序发送 **SOAP Action** 时进行的操作不匹配。

👉 注：*test* 是请求 HTTP 头中 `soapAction` 属性的值。

可以使 SOAP Web Service 中的每个操作与一个 SOAP Action 字符串相关联，例如在 WSDL 绑定中或者通过注释。Web Service 客户机将 SOAP Action 字符串作为头，和请求一起发送，以与 Web Service 提供程序的操作匹配。在以下场景中显示错误消息：客户机发送的 SOAP Action 不匹配操作的 SOAP Action。

要解决此问题，请确保对 Web Service 客户机和 Web Service 提供程序指定完全相同的 SOAP 操作。

使用 **Service.create()** 方法来创建服务时，发生 **javax.xml.ws.WebServiceException**。

如果在未提供 WSDL 文档的情况下使用 `Service.create()` 方法来创建服务，那么会发生此错误。如果您想要使用 `Service.create()` 来创建服务以及使用 `getPort()` 方法来获取端口号，那么必须使用 `addPort()` 方法来提供绑定信息。

下面提供了样本示例代码，说明如何使用 `addPort()` 方法：

```
QName serviceName = new QName("http://test.com/wssvt/acme/InsBusiness/", "MTOM11Service");
QName portName = new QName("http://test.com/wssvt/acme/InsBusiness/", "MTOM11Port");

// Setup the necessary JAX-WS artifacts
Service svc = Service.create(serviceName);
// Add the port in the service instance
svc.addPort(portName, SOAPBinding.SOAP11HTTP_MTOM_BINDING, mtom11URL);

port = svc.getPort(portName, MTOMInterface.class);
```

如果 **@WebResult** 注释中的 **name** 属性与 **WSDL** 文档中的 **name** 元素不同，那么会返回 **NULL** 响应。

如果使用 SEI 类来定义 `@WebResult` 注释的 name 属性，并且 SEI 类已提供如下所示的 WSDL 位置，那么会发生此问题：

```
@WebService(wsdlLocation="WEB-INF/wsdl/WebServiceIfc.wsdl")
public interface WebServiceRuntimeIfc {
        @WebMethod
        @WebResult(name="nononoreturn")
        public String echo (String parm);
}
```

但是所提供 WSDL 文档中的 XML 元素声明如下所示：

```
<xs:element name="echoResponse">
 <xs:complexType>
  <xs:sequence>
```

```
          <xs:element name="return" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

调用 `echo()` 方法时，Web Service 客户机将获取 NULL 响应。

要解决此问题，必须确保 `@WebResult` 注释的 name 属性与 WSDL 文档中 `name` 元素的值匹配。

JAX-WS 客户机应用程序无法从服务器端获取 **WSDL** 文档更改。

如果 Web Service 客户机和 Web Service 提供程序位于 xigemaAS 概要文件上的两个不同应用程序中，并且客户机需要动态地从 Web Service 提供程序检索 WSDL 文档，那么会发生此问题。这是因为首次访问 WSDL 文档时 xigemaAS 概要文件会高速缓存 WSDL 定义，此行为与完整概要文件中的行为不同。通过在 xigemaAS 概要文件中高速缓存 WSDL 定义，应用程序可以避免频繁地连接和解析 WSDL 文档。

要解决此问题，必须重新启动客户机应用程序以便能重新装入所更新的 WSDL 定义。

如果 **Web Service** 引用其自身，那么 **xigemaAS** 概要文件中的 **Web** 应用程序会挂起。

如果将注释 `@WebService` 用于 Web Service，使用注释 `@WebServiceRef` 引用其自身，那么服务端点发布会失败。例如，

```
@WebService
public class Test {
      @WebServiceRef
      private Test selfservice;

      public String echo(String msg) {
            return selfservice.invoke();
      }

      public String invoke() {
            return "hello world";
      }
}
```

要避免此问题，请使用服务存根（在以下示例中显示为 `TestService`）来获取服务实例，而不是使用 `@WebServiceRef` 引用自助服务。 例如，

```
@WebService
public class Test {

      private Test selfservice;

      public String echo(String msg) {
            TestService ts=new TestService();
            selfservice=ts.getTestPort();
            return selfservice.invoke();
      }

      public String invoke() {
            return "hello world";
      }
}
```

对 WS-Security 进行故障诊断

本节描述了您可以用来解决 WS-Security 问题的一些常见解决方案。

1. 请检查 server.xml 文件以确保正确配置了必需的 JAX-WS (`jaxws-2.2`) 功能部件和安全性 (`appSecurity-2.0`) 功能部件。
2. 要使用 WS-Security 来保护 Web Service 应用程序，JAX-WS 应用程序必须包含一个具有嵌入式 WS-Security 策略的 WSDL 文件。在 `wsdl:binding` 和/或 `wsdl:operation` 部分，必须存在对于该嵌入式 WS-Security 策略的 PolicyReference。
3. 如果您使用回调处理程序来检索密码以生成 UsernameToken，或者用于从密钥库文件中检索专用密钥的密码，那么必须将密码回调处理程序作为 xigemaAS 中的用户功能部件进行打包和部署。

启用 WS-Security 跟踪

如果根据错误消息中的信息无法确定发生此问题的原因，那么可以使用 xigemaAS 的跟踪和日志记录工具来收集对于 WS-Security 组件的跟踪。请参阅 *xigemaAS:* 跟踪和日志记录。

可以使用以下跟踪字符串来收集跟踪以调试 WS-Security 故障：

```
org.apache.ws.security.*=all=enabled:
org.apache.cxf.ws.security.*=all=enabled:
org.apache.cxf.ws.policy.*=all=enabled
org.apache.xml.security.*=all=enabled:
com.ibm.ws.wssecurity.*=all=enabled
```

本节描述了一些常见 WS-Security 问题以及您可以选择的解决方案。

org.apache.cxf.ws.policy.PolicyException：不能满足任何替代策略。

当 server.xml 文件中未定义 WS-Security 功能部件 `wsSecurity-1.1` 时，通常会发生此错误。为了避免发生此错误，您必须在 server.xml 文件中按如下所示定义 `wsSecurity-1.1` 功能部件：

```
<featureManager>
  <feature>usr:wsseccbh-1.0</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
  <feature>jaxws-2.2</feature>
  <feature>wsSecurity-1.1</feature>
</featureManager>
```

org.apache.cxf.ws.policy.PolicyException：不存在回调处理程序，没有密码可用。

如果 WS-Security 运行时无法检索在生成 UsernameToken 时所需要的密码，或者无法访问密钥库中的专用密钥，就会发生此错误。为了避免发生此错误，请检查以下配置：

1. 请确保密码回调处理程序功能部件 `wsseccbh-1.0` 在 server.xml 文件中定义为用户功能部件：

```
<featureManager>
  <feature>usr:wsseccbh-1.0</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
  <feature>jaxws-2.2</feature>
  <feature>wsSecurity-1.1</feature>
</featureManager>
```

2. 请确保在 server.xml 文件的 <wsSecurityClient> 或 <wsSecurityProvider> 元素中定义了回调处理程序属性 `ws-security.callback-handler`：

```
ws-security.callback-handler="<full class name of the callback handler>"

example:
```

```
ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.CommonPasswordCallback"
```

3. 请确保密码回调处理程序返回了在 server.xml 文件中指定的用户名或密钥别名的正确密码。该密码必须为明文，并且未编码，也未加密。

org.apache.ws.security.WSSecurityException：签名未涵盖必需的元素 **(soap:Body)**。

如果 Web Service 提供程序应用程序期望指定请求消息中的 SOAP 主体，但是接收到的 SOAP 消息未指定 SOAP 主体，就会发生此错误。为了避免发生此错误，请确保为 Web Service 客户机所配置的正确 WS-Security 策略与 Web Service 提供程序的策略相匹配。

org.apache.ws.security.WSSecurityException：签名或解密无效。

如果 WS-Security 运行时未能验证签名或者将接收到的 SOAP 消息中已加密的消息部件解密，就会发生此错误。为了避免发生此错误，请验证您在 server.xml 文件的 <wsSecurityClient> 和 <wsSecurityProvider> 元素中配置 WS-Security 时使用了正确的密钥。如果 Web Service 客户机使用 Bob 的公用密钥对消息部件进行加密，那么 Web Service 提供程序必须对 Bob 的专用密钥具有访问权才能解密该消息。同样，如果 Web Service 客户机使用 Alice 的专用密钥为消息部件签名，那么该 Web Service 提供程序必须使用 Alice 的公用密钥来验证该签名。

org.apache.cxf.ws.policy.PolicyException：无法满足这些替代策略。

如果 WS-Security 运行时无法使用所配置的用来处理此请求的 WS-Security 策略来成功处理入局 SOAP 消息，就会发生此错误。为了避免发生此错误，请查看紧跟着此异常的消息，以确定导致此异常的特定 WS-Security 策略断言。例如，日志文件中可能包含下列消息：

```
Caused by: org.apache.cxf.ws.policy.PolicyException: These policy alternatives can not be satisfied:
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}AsymmetricBinding: The encryption algorithm doe
s not match the requirement
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}InitiatorEncryptionToken
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}RecipientEncryptionToken
at org.apache.cxf.ws.policy.AssertionInfoMap.checkEffectivePolicy(AssertionInfoMap.java:167)
at org.apache.cxf.ws.policy.PolicyVerificationInInterceptor.handle(PolicyVerificationInInterceptor.java:10
1)
```

在这种情况下，因为 Web Service 客户机所使用的加密算法与 AlgorithmSuite 断言所指定的加密算法不匹配，所以发生了错误。在 Web Service 客户机和 Web Service 提供程序的 WS-Security 策略中指定的算法套件必须指定同一加密算法。

javax.xml.ws.soap.SOAPFaultException：消息已到期（**WSSecurityEngine**：时间戳记无效。消息的安全性语义已到期）。

如果消息时间戳记已到期，或者使用将来的时间戳记创建了消息，就会发生此错误。

诊断性能

本部分描述了一些常见性能问题以及您可以选择的解决方案。

应用程序监视器产生高 **CPU** 使用率。

如果应用程序监视器在 apps/ 目录下具有很多文件，且过度频繁地进行轮询，那么会发生此错误。

要避免此问题，可以更改一些设置。

1. 增大 `pollingRate` 属性的值。
2. 更新 server.xml 以将 `applicationMonitor` 元素包含在不是 `polled` 的 `updateTrigger` 中。

```
server.xml
```

```
<applicationMonitor updateTrigger="mbean" />
```

3. 减少 apps/ 目录下的文件数。

有关 applicationMonitor 元素的更多信息，请参阅控制动态更新（见第 1135 页）。

对 SAML 进行故障诊断

本节描述 SAML 的一些常见问题及您必须应用的解决方案。

java.lang.ArrayIndexOutOfBoundsException: 数组下标超出范围：**0**

在未移除身份提供者令牌 (IdP) 的情况下，尝试通过来路不明的服务提供者 (SP) 发起的请求进行多次登录时，可能发生此异常。

为避免此情况，请在相关的来路不明的 server.xml 文件中添加 <httpSession invalidateOnUnauthorizedSessionRequestException="true" />。

java.lang.IllegalStateException: CWWKS0010E: 尝试获取调用者主体

如果 SAML 用户先前直接登录本地用户注册表，那么可能发生此异常。为避免此问题，SAML 用户不得直接登录本地用户注册表。

2.11.1 启动服务器

在 xigemaAS 安装成功后，若启动服务器失败，您可以使用以下建议来排除操作中的常见故障。

1. 系统提示错误信息：

   CWWKE0001I:**已启动服务器** defaultServer。

   CWWKE0005E:**无法启动运行时环境**。

   LICVC0001E:**产品未安装** License **文件**。

   出现此则错误信息，说明 License.dat 缺失，请注册许可证。
2. 系统提示错误信息：

   CWWKE0001I:**已启动服务器** defaultServer。

   CWWKE0005E:**无法启动运行时环境**。

   LICVC0002E: License **文件无效**。

   出现此则错误信息，说明 License.dat 无效，请联系 Vsettan 支持中心或您的销售代表。
3. 系统提示错误信息：

   CWWKE0001I:**已启动服务器** defaultServer。

   [err] The trial period for the copy of xigemaAS has expired.

   CWWKE0005E:**无法启动运行时环境**。

   LICVC0003E: License **已过期**

   出现此则错误信息，说明试用版已过期，请安装正式版 xigemaAS。

2.11.2 日志记录和跟踪

本产品包括统一的日志记录组件，用于处理产品写入的消息以及提供首次故障数据捕获 (FFDC) 服务。

此外，日志记录组件将捕获写入到 System.out、System.err、java.util.logging 和 OSGi 记录的消息。日志记录组件将统一处理这些消息以及该产品写入的其他消息。日志记录组件无法捕获直接由 JVM 进程写入的消息，例如 `-verbose:gc` 输出。

服务器有三个主要日志文件：

1. console.log - 包含来自 JVM 进程的重定向标准输出和标准错误。此控制台输出适用于直接人工使用。如果您使用缺省 consoleLogLevel 配置，那么控制台输出包含主要事件和错误。如果您使用缺省 copySystemStreams 配置，那么控制台输出还包含写入到 System.out 和 System.err 流的任何消息。控制台输出始终包含直接由 JVM 进程写入的消息，例如，`-verbose:gc` 输出。仅当使用 `server start` 命令时，才会创建此文件，此位置只能使用 `LOG_DIR` 环境变量改变。有关更多信息，请参阅从命令行管理 *xigemaAS*（见第 1116 页）。
2. messages.log - 包含除了由日志记录组件写入或者捕获的跟踪消息之外的所有消息。写入到此文件的所有消息包含其他信息，例如，消息时间戳记以及写入该消息的线程的标识。此文件不包含由 JVM 进程直接写入的消息。
3. trace.log - 包含由该产品写入或捕获的所有消息。仅当您启用其他跟踪时，才会创建此文件。此文件不包含由 JVM 进程直接写入的消息。

日志记录配置

可以通过服务器配置来控制日志记录组件。日志记录配置的主要位置是位于 server.xml 文件中。有时，您可能需要配置跟踪以诊断在处理 server.xml 文件之前发生的问题。在这种情况下，可以在 bootstrap.properties 文件中指定等价的配置属性。如果在 bootstrap.properties 文件和 server.xml 文件中都指定配置属性，那么会使用 bootstrap.properties 文件中的值，直到处理了 server.xml 文件为止。然后，使用 server.xml 文件中的值。应避免同时在 bootstrap.properties 和 server.xml 文件中为相同配置属性指定不同值。

表 53: xigemaAS 概要文件的日志记录属性

第 1 列包含可以在 server.xml 文件中设置的属性。第 2 列包含可以在 bootstrap.properties 文件中使用的等价属性。第 3 列提供每个日志记录属性的描述。

属性	等价属性	描述
logDirectory	`com.ibm.ws.logging.log.directory`	此属性对所有日志文件（排除 console.log 文件，但包括 FFDC）设置目录。缺省情况下，logDirectory 设置为 `LOG_DIR` 环境变量。缺省 `LOG_DIR` 环境变量路径为 *WLP_OUTPUT_DIR*/serverName/logs。 注：使用 LOG_DIR 环境变量或 com.ibm.ws.logging.log.directory 属性而不是 logDirectory 属性来配置您要将所有消息写至其中的目录。否则，缺省情况下一开始会将一些消息写至 logs 目录，然后将余下消息写至基于您的配置的指定目录。服务器运行时，可

属性	等价属性	描述
		使用 logDirectory 属性将日志动态更新至指定目录。
maxFileSize	com.ibm.ws.logging.max.file.size	滚动之前日志文件可以达到的最大大小（以 MB 计）。xigemaAS 概要文件运行时仅执行基于大小的日志滚动。要禁用此属性，请将值设置为 0。最大文件大小是适当的。缺省情况下，此值为 20。 👉 注：maxFileSize 不适用于 console.log 文件。
maxFiles	com.ibm.ws.logging.max.files	如果强制最大文件大小存在，那么此设置用于确定每个日志文件保留的字节数。此设置也适用于对任何特定日期发生的异常进行汇总的异常日志数。因此，如果此数目是 10，那么 ffdc/ 目录中可具有 10 个消息日志、10 个跟踪日志以及 10 项异常摘要。缺省情况下，值为 2。 👉 注：maxFiles 不适用于 console.log 文件。
consoleLogLevel	com.ibm.ws.logging.console.log.level	此过滤器控制进入 console.log 文件的消息的详细程度。有效值为 INFO、AUDIT、WARNING、ERROR 和 OFF。缺省情况下，级别为 AUDIT。
copySystemStreams	com.ibm.ws.logging.copy.system.streams	如果为 true，那么会将写入 System.out 和 System.err 流的消息复制到 console.log。如果为 false，那么这些消息会写入到配置的日志，例如，messages.log 或 trace.log，但不复制到 console.log。缺省值为 true。
messageFileName	com.ibm.ws.logging.message.file.name	消息日志的缺省名称为 messages.log。此文件总是存在，而且包含 INFO 和其他 (AUDIT、WARNING、ERROR 和 FAILURE）消息以及 System.out 和 System.err。此日志还包含时间戳记和发放线程标识。如果日志文件滚动，那么之前日志文件的名称的格式为 messages_*timestamp*.log
suppressSensitiveTrace		服务器跟踪可在跟踪非类型化数据（例如，通过网络连接接收到的字节数）时公开敏感数据。此属性如果设置为 true，那么可防止

属性	等价属性	描述
		潜在的敏感信息暴露在日志和跟踪文件中。缺省值为 false。
traceFileName	com.ibm.ws.logging.trace.file.name	只有在启用附加或详细跟踪的情况下，才创建 trace.log 文件。stdout 识别为特殊值，并促使将跟踪定向到原始标准输出流。
traceSpecification	com.ibm.ws.logging.trace.specification	跟踪字符串用于选择性地启用跟踪。缺省值为 *=info。
traceFormat	com.ibm.ws.logging.trace.format	此属性控制跟踪日志的格式。xigemaAS 概要文件的缺省格式为 ENHANCED。
hideMessage	com.ibm.ws.logging.hideMessage	可使用 hideMessage 属性以配置您要在 console.log 和 message.log 文件中隐藏的消息。如果消息配置为隐藏，那么它们会重定向至 trace.log 文件。

要在服务器配置文件中设置日志记录属性，可以在开发者工具的**服务器配置**视图中选择**记录和跟踪**，也可以在服务器配置文件中添加日志记录元素，如下所示：

```
<logging traceSpecification="*=audit:com.myco.mypackage.*=finest"/>
```

日志详细信息级别规范的格式为：

```
<component> = <level>
```

其中 <*component*> 是要为它设置日志详细信息级别的组件，而 <*level*> 是某个有效的记录器级别（关闭(off)、致命(fatal)、严重(severe)、警告(warning)、审计(audit)、信息(info)、配置(config)、详细信息(detail)、精细(fine)、更精细(finer)、最精细(finest)、全部(all)）。用冒号 (:) 分隔多个日志详细信息级别规范。

注意：对于给定记录器，此级别由适用于该记录器的最具体跟踪规范确定。

组件对应 Java™ 包和类，或对应一组 Java™ 包。使用星号 (*) 作为通配符以指示某些组件，这些组件具有所指定组件包含的所有包中所有的类。例如：

*****

指定在应用程序服务器中运行的所有可跟踪代码，包括产品系统代码和客户代码。

com.ibm.ws.*

指定所有包名以 com.ibm.ws 开头的类。

com.ibm.ws.classloader.JarClassLoader

仅指定 JarClassLoader 类。

表 54: 有效日志记录级别

下表列示了 xigemaAS V9 应用程序服务器的有效级别。

V9 及更高版本日志记录级别	内容/重要性
关闭	日志记录已关闭。
致命	任务无法继续，并且组件、应用程序和服务器无法工作。
严重	任务无法继续，但是组件、应用程序和服务器仍可工作。此级别也可以表明即将发生的不可恢复的错误。
警告	可能发生错误或即将发生错误。此级别也可以表明正在向故障发展（例如，潜在的资源泄漏）。
审计	影响服务器状态或资源的重大事件
信息	概述总体任务进度的一般信息
配置	配置更改或状态
详细信息	详细说明子任务进度的一般信息
精细	跟踪信息 - 常规跟踪 + 方法入口、出口和返回值
更精细	跟踪信息 - 详细跟踪
最精细	跟踪信息 - 更详细的跟踪信息，包含调试问题所需的所有详细信息
全部	记录所有事件。如果创建了定制级别，那么全部将包括那些级别，并且能提供比“最精细”更详细的跟踪。

console.log 文件的管理级别与其他日志文件不同。可更改的唯一属性是 consoleLogLevel。如果您担心 console.log 文件的大小不断增长，那么可禁用 console.log 文件并改用消息日志文件。系统将以另一格式编写的相同数据写至消息日志文件，您可使用 maxFileSize 和 maxFiles 属性来控制消息日志文件的大小和数目。例如，以下 bootstrap.properties 文件会导致产生空的 console.log 文件，以及最多 3 个不断增大的 1 MB loggingMessages.log 文件。但是，来自底层 JVM 的消息仍可写入 console.log。由于 bootstrap.properties 文件中的设置在创建消息日志文件之前生效，因此消息日志文件最初创建为 loggingMessages.log 而不是缺省的 messages.log。

```
com.ibm.ws.logging.max.file.size=1
com.ibm.ws.logging.max.files=3
com.ibm.ws.logging.console.log.level=OFF
com.ibm.ws.logging.message.file.name=loggingMessages.log
```

console.log 文件会在服务器重新启动时重置。

注：在所有平台上，日志都以缺省系统编码编写。

- 在 Windows™ 系统上，有两种编码：OEM 代码页（用于控制台输出）和 ANSI 代码页（用于读写文件）。console.log 文件使用 OEM 代码页，而所有其他日志都使用 ANSI 代码页。
- 在所有其他平台上，所有日志文件都使用缺省编码。

2.11.3 定时操作和 JDBC 调用

当应用程序服务器中的 JDBC 调用在操作速度上慢于或快于预期时，定时操作会生成已记录的警告。

概述

如果定时操作功能部件已启用，那么会跟踪 JDBC 操作在应用程序服务器中的运行持续时间。在操作所花费的执行时间超过或少于预期时，定时操作功能部件会记录警告。定时操作功能部件将定期地在应用程序服务器日志中创建报告，详细说明执行时间最长的操作。如果运行 `server dump` 命令，那么定时操作功能部件将生成一个报告，该报告包含此功能部件所跟踪的所有操作的相关信息。您可以使用这些报告中所列出的信息来决定是否有任何项的运行速度慢于或快于预期。

系统会定期向日志中生成一个报告，此报告中包含 10 个最长的 JDBC 定时操作。可在 server.xml 文件中配置此报告的生成频率和是否启用，缺省情况为每天（24 小时）生成一次。

要启用定时操作，请将 `timedOperations-1.0` 功能部件添加至 server.xml 文件。

可使用 `timedOperation` 元素禁止将报告生成到日志或更改报告频率（例如，更改为每 12 小时 1 次），如以下示例中所示：

```
<timedOperation enableReport="false" reportFrequency="12"/>
```

还可以使用 maxNumberTimedOperations 属性来记录警告（用于指示定时操作总数达到此属性指定的值）。系统会监视定时操作的数目，知道此数目很有用，因为系统会从堆中为每个定时操作分配内存，如果您发现定时操作数目过大，那么可禁用定时操作功能。可使用以下示例来配置 maxNumberTimedOperations 属性：

```
<timedOperation enableReport="false" reportFrequency="12" maxNumberTimedOperations="10000"/>
```

在此示例中，当定时操作数超过 10000 时，日志中会出现如下警告消息：

```
[4/18/13 23:01:37:316 EDT] 0000002c com.ibm.wsspi.timedoperations.TimedOperationService W
 TRAS0094I:
The total number of timed operations is 10000, which exceeds the configured maximum number of
 10000.
You can also find the number of timed operations in the report that is periodically generated to
 the logs.
If you find that the number of timed operations is excessive, you can disable the timed
 operations feature.
```

如果您将环境变量 *com.ibm.timedOperations.autoCleanup* 的值设置为 `true`，那么服务器会将所跟踪的定时操作数自动限制为在 <maxNumberTimedOperations> 属性中所指定的值。当定时操作总数达到所指定的最大值时，会记录警告。要限制所跟踪的定时操作数，当需要跟踪新的定时操作时，会删除最近最少使用的定时操作记录。当所跟踪的定时操作数达到所指定的最大值时，那么会显示警告消息，如下所示：

```
TRAS0095I: The total number of timed operations has reached the configured
 maximum of 10000. As new timed operations are created the least recently
 used timed operations will be removed to maintain the total number of tracked
 timed operations at this level.
```

还可使用 `server dump` 命令以在 messages.log 文件中获取所有定时操作的完整报告，这些操作按类型分组，每组内按实际的平均持续时间进行排序。

以下示例显示了一条已记录的样本消息：

```
 [3/14/13 14:01:25:960 CDT] 00000025 TimedOperatio W   TRAS0080W: Operation xigemaas.datasource.execute:
jdbc/exampleDS:insert into cities values ('myHomeCity', 106769, 'myHomeCountry') took 1.541 ms to complete,
 which was longer than the expected duration of 0.213 ms based on past observations.
```

以下示例显示了日志中自动生成的样本报告：

```
[12/13/12 7:42:29:509 CST] 0000001d com.ibm.wsspi.timedoperations.TimedOperationService I TRAS0092I:
The following operations took the longest time to run since the last report has been generated:
Operation xigemaas.datasource.execute:jdbc/exampleDS:insert into cities values ('myHomeCity',
106769, 'myHomeCounty') took 194ms to complete
Operation xigemaas.datasource.execute:jdbc/exampleDS:select county from cities where name=
'myHomeCity' took 187ms to complete
Operation xigemaas.datasource.execute:jdbc/exampleDS:drop table cities took 182ms to
complete\Operation xigemaas.datasource.execute:jdbc/exampleDS:insert into cities values
('myHomeCity', 106769, 'myHomeCounty') took 151ms to complete
```

2.11.4 事件日志记录

作为监视和诊断功能的一部分，xigemaAS 在 Java Platform Enterprise Edition 的各个组件中生成事件以跟踪请求。应用程序请求运行时，`eventLogging-1.0` 功能部件会记录这类事件。通过使用此功能部件，用户可跟踪正在 xigemaAS 中运行的请求。每个请求与唯一相关因子（称为请求标识）及上下文信息相关联，上下文信息帮助用户了解特定于该请求的数据。

Event Logging 功能部件是通过服务器配置控制的。此功能部件是在 server.xml 文件中配置的。

以下样本日志显示对应 *AAY6TalVDTO_AAAAAAAAAAK* 请求标识和 *TradeWeb* 上下文的端到端事件日志：

```
[12/15/14 18:24:29:528 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeScenarioServlet
[12/15/14 18:24:29:531 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeAppServlet
[12/15/14 18:24:29:532 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /quote.jsp
[12/15/14 18:24:29:533 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp
[12/15/14 18:24:29:534 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=?
[12/15/14 18:24:29:547 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=? # duration=12.537ms
[12/15/14 18:24:29:556 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp #
 duration=22.171ms
[12/15/14 18:24:29:671 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp
[12/15/14 18:24:29:672 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=?
[12/15/14 18:24:29:677 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=? # duration=4.968ms
[12/15/14 18:24:29:684 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp #
 duration=12.569ms
[12/15/14 18:24:29:685 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /quote.jsp # duration=152.752ms
[12/15/14 18:24:29:686 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeAppServlet #
 duration=154.616ms
[12/15/14 18:24:29:687 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeScenarioServlet #
 duration=158.283ms
```

此请求以 `BEGIN websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet"` 事件（引用样本代码中的第一行）开头，以 `END websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet"`（引用样本代码中的最后一行）结尾。此请求耗用总时间也会显示在结尾（在样本代码中为 158.283 ms）。

可通过查看主请求的 `BEGIN` 和 `END` 来查看子请求。还可找到每个子请求耗用的时间。

为获得最佳性能，可在启用事件日志记录时使用二进制日志记录。事件日志条目中的 eventType、contextInfo 和 requestID 属性存储为日志记录扩展。可使用这些日志记录扩展以借助 binaryLog 命令来过滤日志。

解析 messages.log 文件中的事件日志条目

事件日志使用以下格式捕获事件信息：

```
[Log mode] [Request Identifier] # [Event Type] # [Context Information] #
 [Duration] (optional)
```

其中

- Log mode 指示日志是在进入还是退出事件时记录的。BEGIN 指进入事件，END 指退出事件。
- Request identifier 是分配给每个请求的唯一字符串。它可用于过滤属于特定请求的事件。例如：**requestId=AAY6TalVDTO_AAAAAAAAAAK**
- Event type 提供有关事件源的信息，可以是下表中给定的任何受支持事件类型。事件类型可用于过滤特定类型的事件。示例：**eventType=websphere.servlet.service**
- Context information 是事件的上下文信息，用于指定与事件类型相关的详细信息。此信息根据事件类型不同而变化。上下文信息可包含多个部分，各部分之间以 |（| 的两边有空格）分隔。下表中给定各种事件类型的上下文信息的示例。
- Duration 指示事件耗用的时间。duration 仅显示在退出事件条目中。示例：**duration=158.283ms**

除 log mode（以空格分隔）外，所有其他日志属性以 #（# 两边有空格）分隔。例如，

```
[12/15/14 18:24:29:687 IST] 0000002e EventLogging  I END
 requestID=AAY6TalVDTO_AAAAAAAAAAK # eventType=websphere.servlet.service #
 contextInfo=TradeWeb | TradeScenarioServlet # duration=158.283ms
```

下表列示事件日志记录支持的事件类型：

表 55: 受支持事件类型及相关上下文信息

组件	事件类型	上下文信息	示例
Servlet	websphere.servlet.destroy websphere.servlet.service	[Application Name] | [Servlet Name] | [Path Information] | [Query String]	contextInfo=**TradeWeb | /displayQuote.jsp**
Session	websphere.session.dbSessionDestroyedByTimeout, websphere.session.dbSessionDestroyed websphere.session.sessionAccessed websphere.session.sessionCreated websphere.session.sessionDestroyedByTimeout websphere.session.sessionDestroyed	[Session Id] [Session Id] | [Session Attribute Name]	contextInfo=**EuitabHZUOD7J2u01HDdAG0** contextInfo=**EuitabHZUOD7J2u01HDdAG0 | userID**

组件	事件类型	上下文信息	示例
	websphere.session.sessionLiveCountDec websphere.session.sessionLiveCountInc websphere.session.sessionReleased websphere.session.getAttribute websphere.session.setAttribute		
JDBC	websphere.datasource.execute websphere.datasource.executeQuery websphere.datasource.executeUpdate websphere.datasource.psExecute websphere.datasource.psExecuteQuery websphere.datasource.psExecuteUpdate websphere.datasource.rsCancelRowUpdates websphere.datasource.rsDeleteRow websphere.datasource.rsInsertRow websphere.datasource.rsUpdateRow	[Jndi Name Of Data Source] | [SQL Query]	`contextInfo=jdbc/TradeDataSource | select * from quoteejb q where q.symbol=?`

2.11.5 检测运行缓慢和挂起的请求

任何请求的持续时间超过所配置阈值时，`requestTiming-1.0` 功能部件提供诊断信息。

Request Timing 功能部件可跟踪进入系统的每个请求的持续时间。可配置该功能部件以查找运行缓慢和挂起的请求。

- 检测运行缓慢的请求（见第 1656 页）
- 检测挂起请求（见第 1658 页）

检测运行缓慢的请求

如果请求的运行时间超过所配置的时间，那么系统会在消息日志文件中写入警告消息。系统会捕获有关该请求及构成该请求的事件的详细信息。

以下样本显示对应一个请求的日志消息，该请求运行时间超过运行缓慢的请求的阈值（缺省值为 10 秒）：

```
[12/1/14 11:58:09:629 IST] 0000001d com.ibm.ws.request.timing.SlowRequestTimer
  W TRAS0112W: Request AABjnS+lIn0_AAAAAAAAAb has been running on thread 00000021 for at least
 10003.571ms. The following stack trace shows what this thread is currently running.

  at java.util.HashMap.getEntry(HashMap.java:516)
  at java.util.HashMap.get(HashMap.java:504)
  at org.apache.derby.iapi.store.access.BackingStoreHashtable.get(Unknown Source)
  at org.apache.derby.impl.sql.execute.HashScanResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.NestedLoopJoinResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.ProjectRestrictResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.DMLWriteResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.DeleteResultSet.setup(Unknown Source)
  at org.apache.derby.impl.sql.execute.DeleteResultSet.open(Unknown Source)
  at org.apache.derby.impl.sql.GenericPreparedStatement.executeStmt(Unknown Source)
  at org.apache.derby.impl.sql.GenericPreparedStatement.execute(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedStatement.executeStatement(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeStatement(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeUpdate(Unknown Source)
  at
 com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.executeUpdate(WSJdbcPreparedStatement.java:626)
  at com.ibm.websphere.samples.trade.direct.TradeDirect.resetTrade(TradeDirect.java:1832)
  at
 com.ibm.websphere.samples.trade.web.TradeConfigServlet.doResetTrade(TradeConfigServlet.java:65)
  at com.ibm.websphere.samples.trade.web.TradeConfigServlet.service(TradeConfigServlet.java:348)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:668)
  at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1275)
  ....
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1121)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:614)
  at java.lang.Thread.run(Thread.java:769)

Duration    Operation
10007.571ms + websphere.servlet.service | TradeWeb | TradeConfigServlet?action=resetTrade
    3.923ms       websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 holdingejb where holdingejb.account_accountid is null
    0.853ms       websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 accountprofileejb where userid like 'ru:%'
 5271.341ms +     websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 orderejb where account_accountid in (select accountid from accountejb a where a.profile_useri
 like 'ru:%')
```

系统会继续监视该请求，如果该请求的运行时间再次超过 10 秒，那么系统会再记录一条警告。日志消息为以下格式：

```
TRAS0112W: Request < (Request ID)> has been running on thread <THREADID> for at least
 <DURATION>. The following stack trace shows what this thread is currently running.
<STACK TRACE>
<DURATION AND OPERATIONS Table>
```

REQUEST ID

此相同标识可用于搜索对应该请求的日志和跟踪消息。尤其是您使用二进制日志记录时，您可搜索使用二进制日志命令来搜索具有相同 `requestID` 扩展的日志和跟踪条目。

STACK TRACE

指示正在运行的方法。在上一样本中，可在 `TRAS0112W` 行后查看当前请求的堆栈跟踪。

DURATION AND OPERATIONS 表

进行堆栈跟踪后，可查找该请求的表格格式，该表格显示持续时间和操作（又称为事件）。**持续时间**列指示该请求的对应操作的耗用时间。加号 (+) 指示该请求内的事件仍在运行。下一行显示不带 + 的持续时间，它指示已在指定持续时间内完成的对应操作。对于该操作，“操作”显示 `EVENT TYPE` 和 `CONTEXT INFO`（这是可选项）。有关事件类型和上下文信息的更多信息，请参阅事件日志记录（见第 1654 页）。

通过分析消息，您可确定请求运行缓慢的原因。但是，可能难以确定请求是在该点停住还是仍在缓慢运行。因此，可按指定 <*slowRequestThreshold*> 的时间间隔来查看针对任何运行缓慢的请求记录的三条消息。通过使用三个不同的堆栈跟踪和请求数据，您可更深入地了解问题。在第三条警告后，系统不会再记录有关该请求的任何警告，除非该请求的持续时间超过挂起请求检测阈值。

检测挂起请求

如果请求超过缺省 `hungRequestThreshold` 或所配置阈值，那么系统会在消息日志文件中写入警告消息及有关该请求的详细信息。系统会捕获有关该请求及构成该请求的事件的详细信息。检测挂起请求时，系统会执行一系列（三个）线程转储（java 核心），各个转储之间存在 1 分钟延迟。以下日志消息样本显示超过挂起请求检测阈值的请求的日志消息。缺省持续时间值为 10 分钟。以下示例中配置的值为 4 分钟。

[WARNING] TRAS0114W: 请求 AAA7WlpP7l7_AAAAAAAAAAA 已在线程 00000021 上运行了至少 240001.015ms。下表显示此请求执行期间运行的事件。

```
Duration       Operation
240001.754ms + websphere.servlet.service | TestWebApp | TestServlet?sleepTime=480000
     0.095ms       websphere.session.setAttribute | mCzBMyzMvAEnjMJJx9zQYIw | userID
     0.007ms       websphere.session.setAttribute | mCzBMyzMvAEnjMJJx9zQYIw | visitCount
```

如果请求稍后完成（请求最初被检测为挂起），那么系统会记录类似以下示例的消息：TRAS0115W: 先前被检测为挂起的请求 AAA7WlpP7l7_AAAAAAAAAAA 在 479999.681 秒后完成。

👉 注：请求被检测为挂起时，系统会启动一系列（三个）线程转储。完成三个线程转储后，仅当新请求被检测为挂起时，系统才会创建其他线程转储。

2.11.6 二进制日志记录

二进制日志记录是一个高性能日志和跟踪工具。

概述

日志和跟踪存储

二进制日志记录提供日志数据存储库和跟踪数据存储库。请参阅下图以了解应用程序和应用程序服务器如何存储日志和跟踪信息。

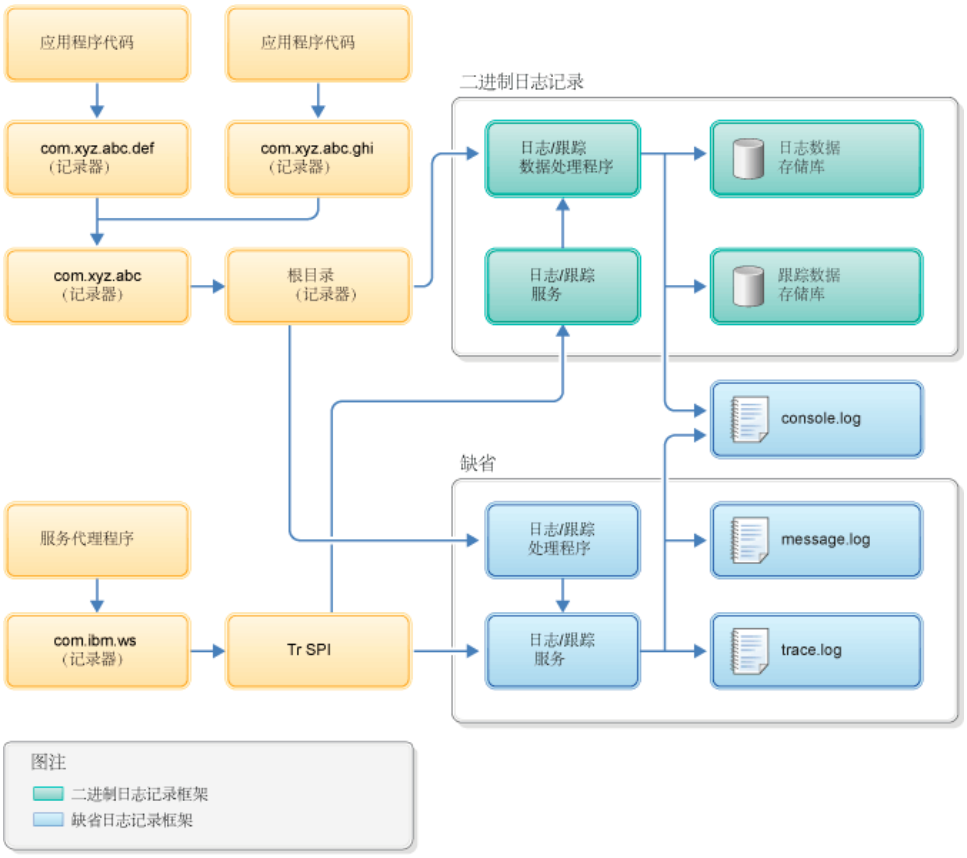


图 38: 用于HPEL日志记录和基本日志记录的日志和跟踪存储器

日志数据存储库

日志数据存储库是用于存储日志记录的存储工具。通常由管理员复查日志数据。这包括应用程序或服务器写入 System.out、System.err、OSGi 记录服务（LOG_INFO 级别或更高级别，包括 LOG_INFO、LOG_WARNING 和 LOG_ERROR）或 java.util.logging（“详细”级别或更高级别，包括“详细”、“配置”、“参考”、“审计”、“警告”、“严重”、“致命”和任何定制级别）的任何信息。

跟踪数据存储库

跟踪数据存储库是用于存储跟踪记录的存储工具。跟踪数据通常供应用程序员或者 xigemaAS 支持团队使用。这包括将应用程序或服务器写入 OSGi 记录服务（LOG_DEBUG 级别）或 `java.util.logging`（“详细”以下的级别，包括精细、较精细、最精细和任何定制级别）的任何信息。

每个日志和跟踪事件都只存储在一个位置

日志事件 System.out 和 System.err 存储在日志数据存储库中。跟踪事件存储在跟踪数据存储库中。仅将每种类型的事件存储在一个位置可确保不会因重复进行数据存储而影响性能。

👉 注：在日志记录性能较重要的情况下，应该禁用控制台日志。写入控制台日志中的任何内容都已经存储在日志数据存储库中。

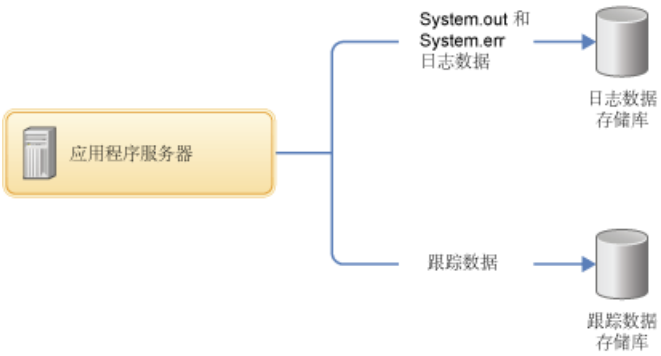


图 39: 连接至日志数据库和跟踪数据存储库的应用程序服务器

除非数据需要格式化，否则不会将数据格式化

格式化数据以便用户阅读会占用处理器时间。日志和跟踪数据会更快速地以专用二进制表示来存储，而不是在运行时格式化日志事件和跟踪事件数据。这将提高日志和跟踪工具的性能。通过延迟日志和跟踪格式化直到 `binaryLog` 命令处于运行状态，从未查看的日志或跟踪部分就决不会格式化。

将日志和跟踪数据写入磁盘之前对其进行缓存

将大块数据写入磁盘比将相同数量的数据写入小块的效率更高。二进制日志记录工具使您能够在将日志和跟踪数据写入磁盘之前对数据进行缓存。缺省情况下，在将日志和跟踪数据写入磁盘之前，会将数据存储在 8 KB 缓冲区中。如果缓冲区在 10 秒之内填满，那么会将缓冲区写入磁盘。如果缓冲区未在 10 秒之内装满，那么会自动地将缓冲区写入磁盘以确保日志具有 最新信息。

日志和跟踪的管理

二进制日志记录的目的是为了使之容易配置和理解。例如，管理员很容易配置供日志和跟踪专用的磁盘空间量、日志和跟踪记录的保留时间以及由服务器来管理日志和跟踪内容。又比如，可以使用一个容易使用的命令 (`binaryLog`) 来访问所有日志、跟踪、System.out 和 System.err 内容，从而避免对于要在哪个文件中访问特定内容可能发生任何混淆。

从日志数据存储库和跟踪数据存储库中读取

日志数据存储库和跟踪数据存储库以 xigemaAS 专用格式进行存储，并且无法使用“记事本”或 VI 之类的文本文件编辑器来阅读。可以使用 `binaryLog` 命令将日志数据存储库和跟踪数据存储库复制到纯文本格式。

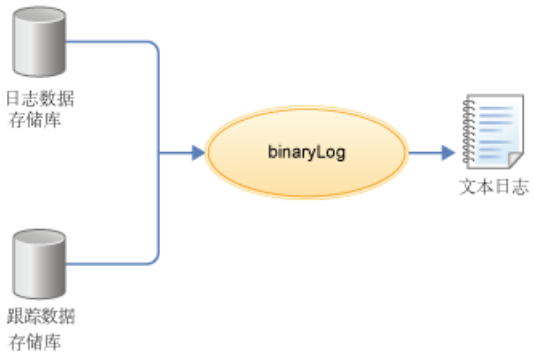


图 40: 拉取数据到文本日志中

binaryLog 命令

binaryLog 是为用户提供的一个容易使用的命令行工具，用来处理日志数据存储库和跟踪数据存储库。binaryLog 提供过滤和格式化选项，使在日志数据存储库和跟踪数据存储库中查找重要内容更容易。例如，用户可以过滤任何错误或警告，然后过滤在 10 秒钟之内发生的所有日志和跟踪条目，以找出同一线程上的关键错误消息。

使用日志和跟踪记录扩展内容进行过滤

二进制日志记录工具可让开发者使用日志记录上下文 API (com.ibm.websphere.logging.hpel.LogRecordContext) 将定制扩展添加到日志和跟踪记录。可以使用 binaryLog 命令行工具根据日志和跟踪记录扩展的内容来过滤记录。

开发资源

已经对二进制日志记录进行设计，以便比缺省日志记录工具更灵活高效地处理日志和跟踪内容。很容易对日志和跟踪内容进行过滤以仅显示感兴趣的记录。您可以使用命令行（请参阅 binaryLog 命令的描述），或者开发者可以使用 HPEL API 来创建功能强大的日志处理程序。

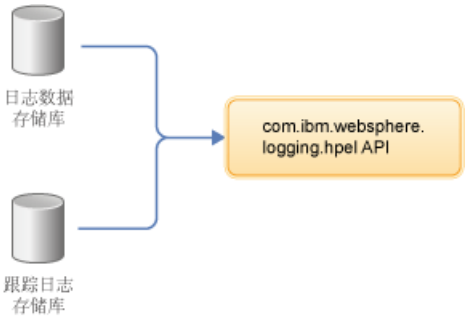


图 41: 连接至 HPEL API 的日志数据存储库和跟踪数据存储库

读取日志数据和跟踪数据

已经提供了一个 API，以使开发者容易开发一些工具来使用二进制日志和跟踪存储库中的内容。例如，开发者可编写 Java™ 程序以搜索日志和跟踪内容来查找其消息标识与已知重要消息标识列表匹配的任何消息。此 API 位于 com.ibm.websphere.logging.hpel 包中。请参阅 API 文档以了解有关 HPEL 日志读取 API 的详细信息。

日志和跟踪记录可扩展性

开发者可以通过日志记录上下文 API (`com.ibm.websphere.logging.hpel.LogRecordContext`) 向日志和跟踪记录添加定制扩展。当二进制日志记录存储日志和跟踪记录时，它会将存在于日志记录上下文中的任何扩展包括在同一线程上。例如，开发者可以编写 Servlet 过滤器，以向日志记录上下文添加重要的 HTTP 请求参数。当该 Servlet 运行时，HPEL API 会向在同一线程上创建的任何日志和跟踪记录添加这些扩展。

与其他日志和跟踪记录字段一样，开发者可以使用 HPEL API 来访问记录扩展。这在编写工具以从日志和跟踪存储库中读取时很有用。在运行时，开发者还可以利用日志记录上下文 API 来访问定制日志处理程序、过滤器和格式化程序中的扩展。

binaryLog 命令选项

使用 `binaryLog` 命令来查看或复制二进制日志记录存储库的内容，或者列示该存储库中可用的服务器进程实例。

二进制日志和跟踪工具以二进制格式写入至存储库。可以使用 binaryLog 命令来查看、查询和过滤存储库。binaryLog 命令提供的选项可快速地将存储库内容转换为各种格式的文本文件，例如基本格式和高级格式。该命令还提供可更方便地从记录中获取所需数据的选项；例如，允许您按级别、记录器名称或日期和时间过滤所需的日志记录。

语法

命令语法如下所示：

```
binaryLog action {serverName | repositoryPath} [options]
```

options 的值随 action 的值不同而不同。

参数

下列操作可用于 binaryLog 命令：

view

读取存储库，可以选择对其进行过滤，并创建用户可以读取的版本。

命令语法如下所示：

```
binaryLog view {serverName | repositoryPath} [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是同时包含 logdata 和 tracedata 目录的目录。

👉 注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

  根据最早记录创建日期进行过滤。必须按日期（例如，`--minDate="2/20/13"`）或者日期和时间（例如，`--minDate="2/20/13 16:47:21:445 EST"`）指定值。
- `--maxDate=`*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，`--maxDate="2/20/13"`）或者日期和时间（例如，`--maxDate="2/20/13 16:47:21:445 EST"`）指定值。
- --minLevel=*value*

  根据最低级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- --maxLevel=*value*

  根据最高级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- --includeLogger=*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- --includeMessage=*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- --includeThread=*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，`--includeThread=2a`）。
- --includeExtension=*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- --includeInstance=*value*

  包括所指定服务器实例中的记录。值必须为 `"latest"` 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

监视选项：

--monitor

连续监视存储库并输出所生成的新内容。

输出选项：

- --format={`basic`|`advanced`|`CBE-1.0.1`}

  指定要使用的输出格式。`"basic"` 为缺省格式。
- --encoding=*value*

  指定要用于输出的字符编码。

copy

读取存储库，可以选择对其进行过滤，并将内容写入新的存储库。

命令语法如下所示：

```
binaryLog copy {serverName | repositoryPath} targetPath [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

targetPath

指定要在其中创建新的存储库的路径。必须指定 *targetPath*。

👉 注：必须指定 *serverName* 或 *repositoryPath*，以及 *targetPath*。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

  根据最早记录创建日期进行过滤。必须按日期（例如，`--minDate="2/20/13"`）或者日期和时间（例如，`--minDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--maxDate=`*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，`--maxDate="2/20/13"`）或者日期和时间（例如，`--maxDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--minLevel=`*value*

  根据最低级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--maxLevel=`*value*

  根据最高级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--includeLogger=`*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--excludeLogger=`*value*[,*value*]*

  排除具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--includeMessage=`*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- `--includeThread=`*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，`--includeThread=2a`）。
- `--includeExtension=`*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- `--includeInstance=`*value*

  包括所指定服务器实例中的记录。值必须为 `"latest"` 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

listInstances

列示存储库中的服务器实例的标识。服务器实例是从服务器启动直到停止期间写入的所有日志记录/跟踪记录的集合。可将服务器实例标识与 binaryLog 查看操作的 --includeInstance 选项配合使用。

命令语法如下所示：

```
binaryLog listInstances {serverName | repositoryPath}
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

请了解 binaryLog 过滤优化。binaryLog 工具在与下列过滤选项配合使用时，能够最高效地过滤日志和跟踪数据：

- --minDate
- --maxDate
- --includeThread
- --minLevel
- --maxLevel

用法示例

请参阅 binaryLog 命令的下列示例。

- 显示 defaultServer 存储库中在 2013 年 7 月 19 日到 2013 年 8 月 2 日期间发生的所有事件。

```
binaryLog view --minDate=07/19/13
--maxDate=08/02/13
```

- 显示 myServer 服务器中发生的、其指定级别为 WARNING 或更高级别的新事件，并在服务器将其写入日志存储库时使用高级格式。

```
binaryLog view myServer --monitor --minLevel=WARNING
--format=advanced
```

- 写入来自存储库（位于 /apps/server1/logs）中的日志消息；仅包括已写入特定存储库的错误流的那些消息。

```
binaryLog
view /apps/server1/logs --includeLogger=SystemErr
```

- 查看 defaultServer 存储库中在 2012 年 9 月 14 日美国东部夏令时下午 4:28 之前发生的事件。

```
binaryLog view
--maxDate="09/14/12 16:28:00:000 EDT"
```

- 写入 defaultServer 存储库中包含“thread”扩展名并且具有以下值的事件：'Default Executor -thread-4'

```
binaryLog view --includeExtension=thread="Default Executor-thread-4" --format=advanced
```

- 查看 defaultServer 存储库中的服务器实例的列表：

```
binaryLog listInstances

使用 D:\wlp\usr\servers\defaultServer\logs 作为存储库目录。

实例标识                        开始日期
1358809441761                   1/21/13 18:04:01:761 EST
1358864476191                   1/22/13 9:21:16:191 EST
1358869523192                   1/22/13 10:45:23:192 EST
1358871281166                   1/22/13 11:14:41:166 EST
1358879829000                   1/22/13 13:37:09:000 EST
1358892222067                   1/22/13 17:03:42:067 EST
```

- 查看 defaultServer 中的使用前一示例中的其中一个实例标识的事件：

```
binaryLog view --includeInstance=1358871281166
```

- 将 defaultServer 中的指定级别为 WARNING 或更高级别的事件从最新服务器实例复制到 d:\toSupport 目录中的新存储库。

```
binaryLog copy defaultServer d:\toSupport --minLevel=warning --includeInstance=latest
```

在 xigemaAS 中配置二进制日志记录

使用此信息作为在 xigemaAS 中配置二进制日志记录的指南。

与缺省 xigemaAS 日志和跟踪框架相比，二进制日志记录提供处理速度更快的日志和跟踪功能以及更灵活的方式来使用日志和跟踪内容。

服务器配置由 bootstrap.properties 文件、server.xml 文件以及这些文件随附的任何可选文件组成。bootstrap.properties 文件指定在处理主要配置之前必须提供的属性，这些属性会保持最少。server.xml 文件是服务器的主要配置文件。

server.xml 文件及其关联文件使用适合于大多数文本编辑器的简单 XML 格式。

bootstrap.properties 文件指定服务器是否应该将二进制日志记录用作日志和跟踪框架，或者用作缺省日志和跟踪框架。

可以通过服务器配置或者 bootstrap.properties 文件来配置二进制日志记录。

- 服务器配置：要从您自己的代码获取日志记录（进行服务器配置处理之后载入），请使用服务器配置来配置二进制日志记录。
- bootstrap.properties 文件：可能需要设置日志记录属性以使其在服务器配置文件得到处理之前生效。例如，如果需要分析服务器启动或配置处理早期发生的问题。在这种情况下，您可以在 bootstrap.properties 文件中配置二进制日志记录。

可以在 bootstrap.properties 或者 server.xml 文件中设置日志记录属性。使用 server.xml 文件中的属性，或者使用 bootstrap.properties 文件中的等价属性。从服务器读取 bootstrap.properties 文件开始，使用 bootstrap.properties 文件中的任何设置，直到完成处理 server.xml 文件为止。如果bootstrap.properties 文件中的日志记录属性未在 server.xml 文件中进行替换或重置，那么将继续使用 bootstrap.properties 文件中的属性值。

如果启用了二进制日志记录，那么会忽略 maxFileSize、maxFiles、messageFileName、traceFileName 和 traceFormat 日志记录元素属性（因为二进制日志记录是在没有 trace.log 和 messages.log 文件的情况下运行）。traceSpecification、consoleLogLevel 和 logDirectory 属性继续用来设置跟踪规范、控制台日志的级别以及日志和跟踪文件的布置。

如果您在 server.xml 文件中设置日志记录或二进制日志记录属性，那么可以通过在 bootstrap.properties 文件中将相应的属性设置为同一值来避免在启动时与运行时之间更改配置。请注意，如果在 bootstrap.properties 文件中未设置任何日志记录属性或者二进制日志记录属性，那么服务器将使用缺省日志记录设置。

- 通过更新 bootstrap.properties 文件对服务器启用二进制日志记录。

  在 bootstrap.properties 文件中，单独添加下列文本行：

  ```
  websphere.log.provider=binaryLogging-1.0
  ```

- 使用下列参数来配置二进制日志记录。

  列出的所有子元素都是 server.xml 文件中日志记录元素的子元素。

  下表列出了可在 server.xml 文件中配置的属性，以及可在 bootstrap.properties 文件中设置的等价属性：

表 56: 可在 server.xml 中配置的二进制日志记录属性以及可在 bootstrap.properties 中设置的等价属性

日志记录子元素	属性	等价的 **bootstrap.properties** 属性
binaryLog	purgeMaxSize	com.ibm.hpel.log.purgeMaxSize
	purgeMinTime	com.ibm.hpel.log.purgeMinTime
	fileSwitchTime	com.ibm.hpel.log.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.log.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.log.outOfSpaceAction
binaryTrace	purgeMaxSize	com.ibm.hpel.trace.purgeMaxSize
	purgeMinTime	com.ibm.hpel.trace.purgeMinTime
	fileSwitchTime	com.ibm.hpel.trace.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.trace.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.trace.outOfSpaceAction

以下示例显示配置为启用二进制日志记录的 bootstrap.properties 文件：

```
websphere.log.provider=binaryLogging-1.0
```

以下示例显示了具有二进制日志记录子元素的 server.xml 文件。日志内容设为在 96 小时后过期，跟踪内容设为最大保留 1024MB：

```
<server description="new server">

    <logging>
        <binaryLog purgeMinTime="96"/>
        <binaryTrace purgeMaxSize="1024"/>
    </logging>
```

```
</server>
```

在重新启动服务器之后，将启用并配置二进制日志记录。

2.11.7 运行时环境已知问题和限制

使用 xigemaAS 运行时环境时，有一些已知问题和限制。

已知问题和限制的列表：

- 一般限制：
  - 支持的最低 *Java* 级别（见第 1669 页）
  - 安装目录名称和路径不能包含非 *ASCII* 字符（见第 1669 页）
  - 运行时更改 *JDBC* 数据源可能导致 *JPA* 失败（见第 1669 页）
  - 依赖于 *getRealPath* 所返回结果的应用程序必须部署为展开的应用程序 *(* 而不是 *WAR* 文件 *)*（见第 1670 页）
  - 文件集限制（见第 1670 页）
  - 覆盖 *Java SDK* 中的类（见第 1670 页）
  - 取消发布共享库时, 只有停止服务器才能删除共享库（见第 1670 页）
  - *java:global* 查询限制（见第 1670 页）
  - 未在嵌入的 *xigemaAS* 服务器中启动应用程序（见第 1671 页）
  - 与 *xigemaMQ* 资源适配器和通用 *JCA* 支持相关的限制（见第 1671 页）
  - 不能对*“dropins”*目录中的应用程序进行版本控制（见第 1671 页）
  - 共享会话应用程序必须将会话对象存储在共享库中（见第 1671 页）
- 特定于 xigemaAS 功能部件的限制：
  - 管理中心功能部件限制（见第 1671 页）
  - *appSecurity-2.0* 功能部件限制（见第 1670 页）
  - *Bean* 验证功能部件限制（见第 1671 页）
  - 动态高速缓存功能部件限制（见第 1672 页）
  - *ejbLite-3.1* 功能部件限制（见第 1672 页）
  - *j2eeManagement-1.1* 功能部件限制（见第 1672 页）
  - *jaxb-2.2* 功能部件限制（见第 1672 页）
  - *jaxws-2.2* 功能部件限制（见第 1673 页）
  - *jpa-2.1* 功能部件限制（见第 1673 页）
  - *jsp-2.2* 功能部件限制（见第 1673 页）
  - *monitor-1.0* 功能部件限制（见第 1673 页）
  - *requestTiming-1.0* 功能部件限制（见第 1674 页）
  - *restConnector-1.0* 功能部件限制（见第 1674 页）
  - *wmqJmsClient-1.1* 功能部件限制（见第 1674 页）
  - *wmqJmsClient-2.0* 功能部件限制（见第 1674 页）
  - *concurrent-1.0* 功能部件限制（见第 1675 页）
  - *jacc-1.5* 功能部件限制（见第 1675 页）

支持的最低 Java™ 级别

此 xigemaAS 概要文件在任何符合 Java™ SE 6 或 Java™ SE 7 运行时环境中受支持，并且受限于针对下列特定实现显示的最低受支持级别。

Java™ SE 6 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 6.0 (J9 2.6) SR 1。对于来自 Oracle 的 JDK，最低受支持级别为 Java™ 6 Update 26。

Java™ SE 7 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 xigemaAS 运行时环境 Java™ Technology Edition 7.0 .4.1。对于 Windows™ 和 Linux™ 上来自 Oracle 的 JDK，最低受支持级别为 Java™ SDK/JRE/JDK 7.0.17。对于 Mac OS X 上来自 Oracle 的 JDK，最低受支持级别为 Java™ SDK/JRE/JDK 7.0 Update 15。

Java™ SE 8 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 xigemaAS SDK Java™ Technology Edition V8。对于来自 Oracle 的 JDK，最低受支持级别为 Java™ 8 update 25。

在分布式平台上，支持 32 位或 64 位 Java™。

对于 Windows™ 和 Linux™ 系统，您可以使用 Oracle JDK 或 xigemaAS JDK。对于 HP 系统和 Mac OS，请使用 Oracle JDK。

安装目录名称和路径不能包含非 ASCII 字符

最近的 JVM 并不完全支持在 `-jar` 和 `-javaagent` 命令中使用非 ASCII 字符。在安装目录名称和路径中只应使用 ASCII 字符。

运行时更改 JDBC 数据源可能导致 JPA 失败

如果未通过属性来指定数据库字典类型，那么在创建第一个实体管理器和建立数据库连接时，OpenJPA 会检测并计算数据库字典类型。此数据库字典类型可用于后续创建的所有实体管理器。如果在应用程序处于运行状态时更改了 JDBC 数据源，那么实体管理器工厂就检测不到此更改，因此对新数据源执行的操作都将继续使用旧字典。如果将数据库更改为其他供应商，那么这可能会导致失败。

将数据库更改为其他供应商之后，请重新启动应用程序。

运行时修改 dataSource、jdbcDriver、connectionManager 和 JDBC 供应商属性可能会导致 JPA 失败

如果在服务器处于运行状态时更新 `dataSource`、`jdbcDriver`、`connectionManager` 或者任何 JDBC 供应商属性列表（例如，`properties.db2.jcc` 或 `properties.oracle`）的配置，那么您可能会看到 `J2CA8040E` 失败。这些失败会说明不能将多个 `dataSource` 元素与单个 `connectionManager` 相关联。即使您的配置只将一个 `connectionManager` 与 `dataSource` 元素相关联，也会生成这些失败。

这些 JDBC 资源的配置只要发生更新，就应该重新启动服务器。

依赖于 getRealPath 所返回结果的应用程序必须部署为展开的应用程序 (而不是 WAR 文件)

Java™ EE 规范说明如果是从 Web 归档 (WAR) 文件提供内容，那么 getRealPath() 方法会返回 null 值。将 WAR 文件部署到 xigemaAS 时，概要文件不会自动将归档文件解压到目录结构。因此，应用程序可能无法启动。如果应用程序依赖于 getRealPath() 所返回的结果，那么必须将应用程序部署为展开的 Web 应用程序（而不是 WAR 文件）。例如，您可以手动解压 WAR 文件并将展开的应用程序复制到 dropins 目录。

文件集限制

下列限制适用于文件集：

- 文件集不会递归地浏览基本目录的子目录。例如，不支持下列指令：

```
<fileset id="testFileset" dir="\temp" includes="**\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="*\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\b\a.jar"/>
```

覆盖 Java™ SDK 中的类

xigemaAS 概要文件支持的某些 Java™ EE 6 技术需要 Java™ SE 6 提供的 API 更新版本。JAX-WS、JAXB 和 javax.annotation.Resource 注解是 Java™ 6 SDK 包含级别低于 Java™ EE 6 所需级别的类的所有示例。当您使用 Java™ SDK V6 针对这些 API 编译应用程序代码时，需要使用 xigemaAS 概要文件提供的类，而不是 Java™ SDK 提供的类。必须执行下列其中一项操作：

- 如果您正在使用 javac 从命令行进行构建，那么使用 javac -endorseddirs 选项以及 ${wlp.install.dir}/dev/specs 目录中的 JAR 文件来编译您的代码。
- 如果您正在使用 Apache Ant 来进行构建，那么使用 javac 任务的 <compilerarg> 子元素以及 ${wlp.install.dir}/dev/specs 目录中的 JAR 文件来编译您的代码。在构建脚本中，作为单独的 <compilerarg> 元素来指定 -endorseddirs 选项和 ${wlp.install.dir}/dev/specs 目录。例如：

```
<javac srcdir="src" destdir="classes"/>
    <compilerarg value="-endorseddirs"/>
    <compilerarg value="${wlp.install.dir}/dev/specs"/>
</javac>
```

取消发布共享库时，只有停止服务器才能删除共享库

从服务器取消发布共享库时，不会立即将库 JAR 文件从服务器释放。因此，操作系统并不知道文件已不再使用，并且不让您删除该文件。下一次停止服务器时，将释放库 JAR 文件，并且您可以将其删除。

java:global 查询限制

应用程序中使用 java:global 查询定义的资源只能用来访问由部署在当前服务器中的应用程序声明的名称。

appSecurity-2.0 功能部件限制

对于 appSecurity-2.0 功能部件，存在下列限制：

- 对于 EJB 应用程序，run-as-mode SYSTEM_IDENTITY 在 ibm-ejb-jar-ext.xml 文件的扩展设置中不受支持。
- 单独会话 Bean 不支持 getCallerIdentity API。

- 角色名称可供 `HttpServletRequest.isUserInRole` 和 `EJBContext.isCallerInRole` API 或者部署描述符中的元素引用，而不先使用 `@DeclareRoles` 注解或者部署描述符中的 `<security-role/>` 元素来声明角色名称。但是，必须先声明角色，然后才能使用这些角色。

未在嵌入的 xigemaAS 服务器中启动应用程序

请确保使用指向 *xigemaASInstallDir*/bin/tools/ws-javaagent.jar 的 JVM 参数 `-javaagent` 启动了用于启动嵌入式 xigemaAS 服务器的 Java™ 进程。如果未使用 `-javaagent` JVM 参数，那么服务器运行时会启动，但应用程序无法启动，同时没有明显异常。

与 xigemaMQ 资源适配器和通用 JCA 支持相关的限制

通过使用 `wmqJmsClient-1.1` 或 `wmqJmsClient-2.0` 功能部件，或通过使用通用 JCA 支持，可在 xigemaAS 内使用 xigemaMQ 资源适配器。

如果您想要使用基于 JMS 2.0 资源适配器的 xigemaMQ 资源适配器 V8.0.0.3 及更高版本，那么必须确保您正在使用与 JMS 2.0 资源适配器兼容的最新 xigemaAS 版本。

👉 注：

- 对于 xigemaAS V9.0.0.1，必须将 `wmqJmsClient-2.0` 功能部件与 xigemaMQ 资源适配器 V8.0.0.3 或更高版本配合使用。

如果正使用类属 JCA 支持，那么以下限制适用：

- 跟踪和日志记录未集成到使用通用 JCA 的 xigemaAS 跟踪系统中。跟踪将写至单独文件，并且必须通过设置系统属性来启用。启用跟踪的过程与为 Java™ 标准环境的 JMS 跟踪工具配置 xigemaMQ 类的过程相同。

不能对“dropins”目录中的应用程序进行版本控制

对于“dropins”目录中的应用程序，应用程序监视器使用文件名和文件扩展名来确定应用程序类型并生成应用程序标识和应用程序名称。因此，不能使用文件名或文件扩展名对应用程序指定版本号。建议不要在生产环境中使用“dropins”目录。

共享会话应用程序必须将会话对象存储在共享库中

使用共享会话上下文应用程序扩展或在 ibm-application-ext.xml 中使用 `<shared-session-context value="true"/>` 时，存储在会话中的所有对象必须在与应用程序相关联的共享库中可用，以便可使这些对象失效。

管理中心功能部件限制

对于 `adminCenter-1.0` 功能部件，将适用以下限制：

- 管理中心浏览工具中仅显示对应服务器、集群和应用程序资源的标记。不会显示对应运行时资源的标记。
- 管理中心的“监视器”视图的“CPU 使用率”图表对未提供进程 CPU 统计信息的 JVM 显示 **`0%`** 或 **`null%`** CPU 使用率。有关该图表的更多信息，请参阅监视管理中心中的指标。

Bean 验证功能部件限制

对于 `beanvalidation-1.0` 功能部件，存在下列限制：

- 不支持在 OSGi 应用程序内部进行 Bean 验证。

对于 beanValidation-1.1 功能部件，存在下列限制：

- 不支持在 OSGi 应用程序内部进行 Bean 验证。
- 在 validation.xml 中为 beanValidation-1.0 功能部件提供定制 ConstraintValidatorFactory 实现的应用程序不会针对 Bean 验证 1.1 API 进行编译。
- 如果 validation.xml 文件不在其关联模块中，那么可能只有一个 validation.xml 文件并且 com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp 属性必须在下列任一文件中设置为 false：
  - jvm.options

    ```
    -Dcom.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false
    ```

  - bootstrap.properties

    ```
    com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false
    ```

动态高速缓存功能部件限制

下列动态高速缓存功能部件不可用或者可用性有限：

- 不支持高速缓存复制。
- 在使用随机逐出和基于大小的逐出技术时，仅支持高性能磁盘高速缓存方式。
- 在 cachespec.xml 文件中不支持 Web Service 客户端和服务器端高速缓存以及 Portlet 高速缓存。
- 不支持对 SingleThreadModel Servlet 进行 Servlet 高速缓存。
- 只包含 Enterprise JavaBeans™ (EJB) 的 JAR 文件不支持使用属性文件来定义高速缓存配置。
- 只能对 32 位 Java™ 虚拟机 (JVM) 限制堆高速缓存大小。

ejbLite-3.1 功能部件限制

对于 ejbLite-3.1 功能部件，存在下列限制：

- 不支持低于 V3.0 的 EJB 模块。此限制也意味着使用 .xmi 文件格式（而不是 .xml 文件格式）的绑定和扩展不受支持。
- 会话 Bean 未绑定到 ejblocal 命名空间，这表示 JNDI 查询和 ejb-ref 绑定名称必须使用 java:global、java:app 或 java:module 名称。忽略 ibm-ejb-jar-bnd.xml 中的 simple-binding-name 和接口 binding-name 元素。
- 有状态 Bean 钝化目录不可配置。文件会钝化到服务器工作区。

j2eeManagement-1.1 功能部件限制

对于 j2eeManagement-1.1 功能部件，以下限制适用：

- 不支持管理 EJB getListenerRegistry() 方法。不能在管理 EJB 组件中注册事件通知侦听器。

jaxb-2.2 功能部件限制

对于 jaxb-2.2 功能部件，适用下列限制：

- 如果应用程序需要 JAXB API 类且已经启动，且要启用 jaxb-2.2 服务器功能部件，那么必须使用 --clean 选项来重新启动服务器，这样应用程序就可以调用由 jaxb-2.2 功能部件提供的 JAXB 2.2 API 和实现类。否则，应用程序仍可能绑定至 Java™ SDK 中提供的 JAXB API 和实现类。
- 如果已启用 jaxb-2.2 服务器功能部件，而且您想要在应用程序中使用您自己的 JAXB API 和实现类，那么必须将您自己的 JAXB API 和实现 JAR 文件放入应用程序的 /WEB-INF/lib 目录中，并且将应用程序的类加载器配置为使用 *parentLast* 授权行为。否则，由 jaxb-2.2 功能部件提供的 JAXB API 和实现类总是会生效。有关在 xigemaAS 上配置应用程序的类加载器行为的更多信息，请参阅使用替代版本来覆盖随附的 *API*（见第 1141 页）。

jaxws-2.2 功能部件限制

对于 jaxws-2.2 功能部件，存在下列限制：

- 如果应用程序提供它自己的 CXF JAR 文件副本来作为应用程序库（例如，在 Web 应用程序的 WEB-INF/lib 目录中），那么您无法在 server.xml 文件中启用 jaxws-2.2 功能部件。
- 因为 jaxws-2.2 功能部件依赖于 jaxb-2.2 功能部件，所以 jaxb-2.2 功能部件限制也适用于 jaxws-2.2 功能部件。
- 如果应用程序需要 JAX-WS API 类且已经启动，但 jaxws-2.2 服务器功能部件未启用，那么必须使用 --clean 选项来重新启动服务器，这样应用程序就可以调用由 jaxws-2.2 功能部件提供的 JAX-WS 2.2 API 和实现类。否则，应用程序仍可能绑定至 Java™ SDK 中提供的 JAX-WS API 和实现类。
- xigemaAS 的 Web Service 绑定文件是 ibm-ws-bnd.xml 文件。
- Apache Axis2 配置或类不受支持。
- 不支持用来实现 javax.xml.ws.Provider<OMElement> 或 javax.xml.ws.Provider<String> 接口的 Web Service 提供程序。
- 对于 xigemaAS ，必须使用尖括号将 MIME 附件的 content-id 属性括起来。例如，<*testID*>。
- xigemaAS 提供的 wsgen 工具不支持 -inlineSchemas 选项。
- 如果您想要使用 JAX-WS Web Service 来传输大型二进制数据以避免发生“内存不足”(OOM) 错误，请启用 MTOM。
- 对于 Web Service 应用程序，如果服务客户机与服务提供程序不在同一应用程序中，并且服务提供程序应用程序中的 WSDL 文件已更改，那么您需要手动重新启动 Web Service 客户机应用程序，以避免发生 WSDL 定义高速缓存问题。

jpa-2.1 功能部件限制

对于 jpa-2.1 功能部件，通过 CORBA/RMI-IIOP 进行的 JPA 实体交换要求参与通信的双方必须启用完全相同的 JPA 功能部件级别。

jsp-2.2 功能部件限制

对于 jsp-2.2 功能部件，存在下列限制：

- 不支持将转换的 JSP 文件仅存储在内存中的 useInMemory 配置选项。

monitor-1.0 功能部件限制

对于 monitor-1.0 功能部件，存在下列限制：

- 从 server.xml 文件中移除该功能部件之后，必须重新启动服务器以使 JAX-WS 应用程序正常运行。

requestTiming-1.0 功能部件限制

对于 requestTiming-1.0 功能部件，以下限制适用：

- 使用 DayTrader 应用程序进行度量时，显示已激活的 requestTiming-1.0 功能部件对可能的最大应用程序吞吐量的影响百分比为 4%。虽然对您的应用程序的影响可能高于或低于此比例，但您还是应注意某些性能下降可能是显而易见的。

restConnector-1.0 功能部件限制

对于 restConnector-1.0 功能部件，以下限制适用：

- 对于 restConnector-1.0 功能部件或包含 restConnector-1.0 的任何功能部件（例如，collectiveMember-1.0 和 collectiveController-1.0）的用户，如果他们想要运行包含定制 JAXRS 2.0 运行时的应用程序，那么他们必须将 jaxrs-2.0 功能部件添加至该服务器。

scim-1.0 功能部件限制

以下限制适用于 scim-1.0 功能部件：

- 搜索 groups 时，不会检索 members 属性。
- 搜索 users 时，不会检索 users 的 groups 属性。
- 不能对 users 的 groups 属性设置规范类型 direct/indirect。
- 只能定义规范类型 work 的用户的一个 email 属性。
- 只能定义规范类型 work 的用户的一个 ims 属性。
- 不能设置或返回 SCIM 的扩展模式属性（例如，entitlements、roles 和 x509Certificates）。
- userName 属性不能与过滤器中的一些其他属性配合使用。
- 对于基本注册表和 SAF 注册表中的用户，只能设置 userName、displayName、id、schema、meta.location 和 groups。userName 和 displayName 将具有相同值。
- 基本注册表和 SAF 注册表的列示/查询的工作方式与 ldapRegistry 注册表不同。
- pr、gt、ge、lt、le、and、or 和 () 之类的运算符对基本注册表及 SAF 注册表不起作用。而且，对于基本注册表和 SAF 注册表，只能在过滤器中使用一个运算符。
- 基本注册表和 SAF 注册表是只读的。
- 创建 user 时，不能设置 groups 属性。

wmqJmsClient-1.1 功能部件限制

对于 wmqJmsClient-1.1 功能部件，存在下列限制：

- 在 Windows™ 环境变量中，必须将 PATH 变量手动设置为指向 xigemaMQ 安装 bin 目录。当应用程序使用“绑定”连接方式时，必须设置此 PATH 变量。
- wmqJmsClient-1.1 功能部件中不包括 xigemaMQ Java™ 类（通称为“基本 Java™”）。这包括在其他应用程序服务器的资源适配器中，但是建议不要将其用于 Java™ Enterprise Edition 环境中的基本 Java™ API。
- wmqJmsClient-1.1 功能部件不支持 xigemaMQ 资源适配器的 BINDINGS_THEN_CLIENT 传输类型。
- wmqJmsClient-1.1 功能部件不包括高级消息传递安全性 (AMS) 功能部件。

wmqJmsClient-2.0 功能部件限制

对于 wmqJmsClient-2.0 功能部件，存在下列限制：

- 在 Windows™ 环境变量中，必须将 PATH 变量手动设置为指向 xigemaMQ 安装 bin 目录。当应用程序使用“绑定”连接方式时，必须设置此 PATH 变量。
- `wmqJmsClient-2.0` 功能部件中未包括 xigemaMQ Java™ 类（通称为“基本 Java™”）。这包括在其他应用程序服务器的资源适配器中，但是建议不要将其用于 Java™ Enterprise Edition 环境中的基本 Java™ API。
- `wmqJmsClient-2.0` 功能部件不支持 xigemaMQ 资源适配器的 BINDINGS_THEN_CLIENT 传输类型。

concurrent-1.0 功能部件限制

对于 concurrent-1.0 功能部件，以下限制适用：

对于类型为 `securityContext` 的线程上下文，不会传播未使用 JAAS 登录模块添加的主体集中的任何定制信息。例如，如果提交者的主体集包含 TAI 添加的定制主体，那么所传播主体集不会包含此定制主体。

jacc-1.5 功能部件限制

对于 jacc-1.5 功能部件，以下配置被忽略：

- 应用程序 ear 文件的 ibm-application-bnd.xml 文件或 ibm-application-bnd.xmi 文件中的授权信息（authorizations 属性的 users 和 groups 属性）。
- server.xml 文件的授权信息（`application-bnd` 元素中的 security-role 属性的 user、group 和 special-subject 属性）。

2.11.8 消息

使用 xigemaAS 时，可能会遇到系统消息。每则消息都具有一个唯一消息标识，并且包含问题的说明以及可用来解决该问题的任何操作的详细信息。

xigemaAS 系统消息是从各种来源（包括应用程序服务器组件和应用程序）记录的。对于 xigemaAS，消息标识的长度是 10 个字符并且具有以下格式：

```
CWXXX9999X
```

其中：

CWXXX

一个五字符字母消息前缀，用于标识 xigemaAS 组件。

9999

一个四字符数字标识，用于标识该组件的特定消息。

X

这是一个可选的字母指示符，用于标识消息类型：I = 参考信息，W = 警告，E = 错误。

表 57: xigemaAS 消息的字母消息前缀

每个字母前缀与特定 xigemaAS 组件相关联。有些前缀会进一步划分，以便字母前缀内的数字范围与特定组件相关联。

xigemaAS 消息前缀	范围	**xigemaAS** 组件
CWIMK	0001-0500	LDAP 注册表联合（配置消息）
CWIML	0001-5000	LDAP 注册表联合（运行时消息）
CWLIB	0001-0100	事务
	CWWKC 0000-0250	注释扫描
CWWKE		核心内核及内核服务
	0001-0099	引导/启动程序
	0100-0199	服务实用程序
	0200-0299	位置服务
	0300-0399	文件安装服务
	0400-0499	FileMonitor 服务
	0500-0599	可扩展类扫描程序
CWWKF		功能部件管理器
CWWKG		配置管理器
CWWKJ		低干涉管理
CWWKL	0001-0100	类装入服务
CWWKM	0001-0050	工件 API 消息（容器工厂）
	0051-0100	重叠消息（所有实现）
	0101-0150	工件 API Zip 实现
	0151-0200	工件 API 文件实现
	0401-0450	自适应 API 实现消息（自适应模块工厂/适配器服务）
	1001-1100	松散归档 API（供基于 Eclipse 的工具选项使用，该选项直接从 Eclipse 工作空间运行应用程序[6]）。
CWWKN	0001-0100	JNDI 缺省名称空间
CWWKO		*CFW* 组件
	0000-0199	CFW
	0200-0399	TCP

[6] Eclipse 平台支持“虚拟”应用程序归档，其中，似乎在同一个归档文件中的文件集实际上会在整个 Eclipse 工作空间中散开。xigemaAS 称此为“松散归档”，并且使用该松散归档 API 的工具选项称为**直接从工作空间运行此应用程序**

xigemaAS 消息前缀	范围		**xigemaAS** 组件
	0400-0599		UDP
	0600-0799		bytebuffer
	0800-0899		SSL 通道 (SSL channel)
	0900-0999		SSL
CWWKS			安全性
	0000 序列		安全性*:* 常规消息
		0000-0099	安全性
		0900-0999	安全性快速入门
	1000 序列		安全性*:* 认证服务
		1000-1099	认证
		1100-1199	JAAS 认证
		1200-1299	TAI 认证
	2000 序列		安全性*:* 授权服务
		2000-2099	权限
		2100-2199	内置授权
		2900-2999	SAF 授权
	3000 序列		安全性*:* 注册服务
		3000-3099	注册表
		3100-3199	基本注册表
		3200-3299	LDAP 注册表
		3300-3399	基于文件的 (VMM) 注册表
		3800-3899	定制注册表
		3900-3999	SAF 注册表
	4000 序列		安全性*:* 令牌服务
		4000-4099	令牌服务
		4100-4199	LTPA
		4200-4299	Kerberos
		4300-4399	SPNEGO
	9000 序列		安全性*:* 协调程序
		9100-9199	Web 协调程序（公共代码）

xigemaAS 消息前缀	范围		**xigemaAS** 组件
		9200-9299	Web 应用程序协调程序
		9300-9399	Web 管理协调程序
CWWKT			HTTP 传输/分派器
CWWKW	0000-0099		JAX-WS 公共
	0100-0199		用于 webContainer 的 JAX-WS
	0200-0299		JAX-WS 安全性
	0300-0399		用于 Java™ EE 公共的 JAX-WS
CWWKX	*0000* 序列		*JMX*
		0001-0100	JMX 安全性
		0101-0200	JMX REST 连接器
		0201-0300	JMX REST 客户机
	1000 序列		管理中心
		1000-1899	管理中心
CWWKZ			应用程序
	0001-0100		应用程序管理器
	0101-0200		WAR
	0201-0300		WAB
	0301-0400		EBA

2.11.9 在 xigemaAS 上对 SIP 容器会话存储库进行故障诊断

对 SIP 容器会话存储库进行故障诊断时，您可能需要 SIP 会话详细信息以转储至指定跟踪文件。

可使用 SIP 会话内存转储实用程序来帮助调试有关 SIP 容器会话的问题。SIP 容器提供 SipContainerMBean 方法以对 SIP 容器执行一些可维护性类型操作，包括通过命令行启动服务器停顿。此任务描述如何使用 SipContainerMBean 方法转储 SIP 容器的内存中会话存储库中包含的 SIP 应用程序会话和 SIP 会话信息。您可以配置 SIPContainerMBean 方法以使用各种跟踪方法，从而指定要转储到指定跟踪文件的 SIP 会话详细信息。

启动会话转储方法后，缺省情况下有关会话的请求信息将显示在 console.log 文件中。还可将此信息发送至 setDumpMethod 方法上指定的预定义源。

可通过以下两种方式运行转储实用程序：简明和详细。如果使用简明会话转储方法，那么每次执行转储方法时仅显示会话标识。如果要使用详细会话转储方法，那么将执行以下操作：

- 每次执行转储方法时，将显示事务用户详细信息和 SIP 会话详细信息（如果存在）。
- 转储至跟踪文件的仅有属性是 JSR 289 规范允许公开的属性。

- 详细方法在跟踪文件中显示以下信息：appName、callID、对话状态、创建时间和属性名称。

对于每个 SIP 应用程序，都会显示跟踪输出；因此，所有 SIP 会话数据结构的排序在显示之前进行。SIPContainerMBean 转储工具在低优先级线程中运行，因此跟踪不影响生产服务器的整体系统的调用处理等待时间。

具有 SIP 会话的事务用户与没有 SipSession 对象的事务用户之间的转储有所区别。不再存在、不再有效或创建跟踪快照时存在的 SIP 会话也以界定方式包含在转储中。

在 xigemaAS 上，可通过两种方式调用 SIPContainerMBean 转储方法：

- 通过运行 `server dump` 命令
- 通过实现 Java™ 管理扩展 (JMX) 客户机，此客户机建立与 JMX 连接器的连接以调用这些方法

以下简明 SipContainerMBean 方法用于转储 SIP 会话标识。

表 58: 用于转储 SIP 会话信息的简明 SipContainerMBean 方法

一个包含两列的表，此表列示方法名称和描述

方法	描述
dumpAllSASIds()	显示所有 SIP 应用程序会话数和 SIP 应用程序会话标识。
dumpAllTUSipSessionIds()	显示事务用户 (TU) 内的事务用户数和 SIP 会话标识（如果存在）。

以下详细 SipContainerMBean 方法用于转储 SIP 会话详细信息。

表 59: 用于转储 SIP 会话信息的详细 SipContainerMBean 方法

一个包含两列的表，此表列示方法名称和描述

方法	描述
dumpAllSASDetails()	显示所有 SIP 应用程序会话数和 SIP 应用程序会话标识详细信息。
dumpAllTUSipSessionDetails()	显示事务用户 (TU) 内的事务用户数和 SIP 会话标识（如果存在）的详细信息。
dumpSASDetails(String sasId)	显示 `sasId` 参数指定的 SIP 应用程序会话的详细信息。
dumpSipSessionDetails(String sessionId)	显示 `sessionId` 参数指定的 SIP 会话的详细信息。

注：使用以下信息来帮助分析打印输出：

- 对于所有打印输出，第一行提供应用程序名称和一些记录。
- 输出之间的定界符是**制表符**。
- 会话属性之间的定界符为 ;（分号）。

- 使用 `server dump` 命令调用 SIPContainerMBean 方法。
  有关 `server dump` 命令的更多信息，请参阅从命令行生成 *xigemaAS* 服务器转储（见第 1124 页）。

1. 在 server.xml 文件中的 `sipIntrospect` 元素上指定转储实用程序方式。

- 对于简明方式，请指定以下代码：

```
<sipContainer>
     <sipIntrospect method="SUCCINCT"/>
</sipContainer>
```

- 对于详细方式，请指定以下代码：

```
<sipContainer>
     <sipIntrospect method="VERBOSE"/>
</sipContainer>
```

2. 打开命令行并切换至 wlp/bin 目录。
3. 要生成 SIP 会话和 SIP 应用程序会话转储，请运行以下命令，其中 *server_name* 是 xigemaAS 服务器：

```
server dump server_name
```

如果未指定服务器名称，那么会使用 defaultServer。

可选择在 archive 参数上指定服务器转储程序包文件的名称：

```
server dump server_name --archive=package_file_name.dump.zip
```

运行该命令后，将生成包含以下文件的服务器转储程序包文件：

- SipContainerIntrospector.txt，它包含有关 SIP 应用程序会话和 SIP 会话的跟踪信息的请求级别
- 其他 *artifact*Introspector.txt 文件，它们提供服务器状态信息

SipContainerIntrospector.txt 包含以下信息，其中 dump.ids.test.app1 是应用程序名称，2 是会话数，local.##### 行为 SAS_id：

```
The description of this introspector:
SIP state details

Succinct dumping

dump.ids.test.app1    2
local.1347524282775_8
local.1347524282775_7

--- End of Dump ---
```

- 通过 JMX 连接器调用特定转储实用程序方法。

有关更多信息，请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）和为 *xigemaAS* 开发 *JMX Java* 客户机（见第 1181 页）。

以下 Java™ 代码示例调用 dumpAllSASIds 方法：

```
    System.setProperty("javax.net.ssl.trustStore", "..\\wlp\\usr\\servers\
\SIPServer\\resources\\security\\key.jks");
    System.setProperty("javax.net.ssl.trustStorePassword", "xigemaAS");

    //If the type of the trustStore is not jks, which is default,
    //set the type by using the following line.
    System.setProperty("javax.net.ssl.trustStoreType", "jks");

   try {
```

```
      HashMap<String, Object> environment = new HashMap<String, Object>();
      environment.put("jmx.remote.protocol.provider.pkgs",
 "com.ibm.ws.jmx.connector.client");

 environment.put("com.ibm.ws.jmx.connector.client.disableURLHostnameVerification",
 Boolean.TRUE);
      environment.put(JMXConnector.CREDENTIALS, new String[] { "theUser",
 "thePassword" });
      environment.put(ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION,
 true);

      JMXServiceURL url = new JMXServiceURL("service:jmx:rest://
localhost:9443/xigemaJMXConnectorREST");
      JMXConnector connector = JMXConnectorFactory.newJMXConnector(url,
 environment);
      connector.connect();
      MBeanServerConnection connection =
 connector.getMBeanServerConnection();

      // test dumping utility
      connection.invoke(new
 ObjectName("WebSphere:name=com.ibm.ws.sip.container.SipContainerMBean"),
 "dumpAllSASIds", null, null);
```

简明 SipContainerMBean 方法具有以下打印格式。

表 60: 简明 SipContainerMBean 方法打印格式

一个包含两列的表，此表列示方法名称及其打印格式的示例

方法	打印格式
dumpAllSASIds()	每行以 `[SAS_ID]` 格式显示 SIP 应用程序会话标识。 例如： `local.1347524282775_8`
dumpAllTUSipSessionIds()	每行以 `[TU_ID] [hasSIPSession] [SipSessionId]` 格式显示事务标识（如果事务具有 SIP 会话）和会话标识（如果存在） 例如： `local.1349965420866_1_0 true local.1349965420866_1_0_1`

详细 SipContainerMBean 方法具有以下打印格式。

表 61: 详细 SipContainerMBean 方法打印格式

一个包含两列的表，此表列示方法名称及其打印格式的示例

方法	描述
dumpAllSASDetails()	每行以 `[SAS_ID] [CreationTime] [attributes]` 格式显示 SIP 应用程序会话标识、创建时间、SIP 会话属性

方法	描述
	例如： `local.1348147884986_2 Sep 20,2012 16:31 DumpSasDetailsAttr;`
dumpAllTUSipSessionDetails()	每行以 [TU_ID] [hasSIPSession] [SipSessionId] [Call-Id] [DialogState] [hasOutgoingTransaction] [initialMethod] [SAS_ID] [CreationTime] [attributes] 格式显示事务标识（如果事务具有 SIP 会话）和会话详细信息（如果存在） 例如： `local.1349965420866_1_0 true local.1349965420866_1_0_1 8-8548@9.148.57.128 2 false INVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAttr2;`

dumpSASDetails(String sasId)	只会显示格式为 [TU_ID] [hasSIPSession] [SipSessionId] 的一行
	例如：
	`local.1349965420866_1_0 true local.1349965420866_1_0_1`
	如果所请求会话不存在，那么将显示以下错误消息：
	`ERROR: Requested session <local.1349965420866_1_0_1> does not exist.`

dumpSipSessionDetails(String sessionId)	只会显示格式为 [SipSessionId] [Call-Id] [DialogState] [hasOutgoingTransaction] [initialMethod] [SAS_ID] [CreationTime] [attributes] 的一行
	例如：
	`local.1349965420866_1_0_1 8-8548@9.148.57.128 2 false INVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAttr2;`
	如果所请求会话不存在，那么将显示以下错误消息：
	`ERROR: Requested session <local.1349965420866_1_0_1> does not exist.`

2.11.10 在 xigemaAS 上跟踪会话启动协议 (SIP) 容器

您可以立即开始或在下次服务器启动之后跟踪会话启动协议 (SIP) 容器。此跟踪将 SIP 事件的记录写入日志文件。

1. 在 server.xml 文件中，添加 `logging` 元素并通过设置 `traceSpecification="*=info:com.ibm.ws.sip=all"` 来对 SIP 容器指定跟踪。

```
<logging logDirectory="${server.config.dir}/logs"
 traceFileName="trace.log" traceSpecification="*=info:com.ibm.ws.sip=all"/
>
```

SIP 级别跟踪消息将显示在 *serverName*/logs/trace.log 中，其中 serverName 是运行要跟踪的 SIP 容器的应用程序服务器的特定实例的名称。这些消息包括应用程序装入事件以及执行语法分析和 SIP Servlet 调用时的 SIP 请求和响应。

2.11.11 xigemaAS 上的会话启动协议 (SIP) 二进制日志和跟踪扩展

二进制日志记录为开发者提供一种方法以向日志和跟踪记录添加扩展字段，并为您提供对应方法以按扩展值过滤日志和跟踪记录。

日志和跟踪记录包含的字段提供该记录的创建时间和所记录消息的内容之类的信息。这些字段是核心字段，每个日志和跟踪记录中都存在。相比之下，扩展字段是应用程序开发者可以添加到日志和跟踪记录的字段，在搜索特定日志和跟踪内容时，您可以将扩展字段用作过滤条件。将文本输出格式配置为使用高级格式时，这些日志和跟踪扩展在二进制日志中可视，您以高级格式使用 binaryLog 命令时，这些扩展也可视。

管理员

应用程序服务器将自动创建一些扩展，您可以使用这些扩展来过滤日志和跟踪记录。还可以使用由应用程序开发者添加的任何扩展来过滤日志和跟踪记录。可以使用 binaryLog 命令行工具根据日志和跟踪记录扩展的内容来过滤记录。有关更多信息，请参阅 *binaryLog 命令选项*（见第 1076 页）。

例如，要查看 SIP 容器处理的所有 SIP 应用程序会话，可使用以下 binaryLog 命令：

```
binaryLog view binaryFile --includeExtension=SIPASId=* --format=advanced
```

开发者

开发者可以使用二进制日志记录以通过日志记录上下文 API（即 com.ibm.websphere.logging.hpel.LogRecordContext）向日志和跟踪记录添加定制扩展。二进制日志记录存储日志和跟踪记录时，它会将存在于日志记录上下文中的任何扩展包括在同一线程上。例如，您可以编写 servlet 过滤器以将重要 HTTP 请求参数添加到日志记录上下文。该 Servlet 运行时，HPEL API 会向在同一线程上创建的任何日志和跟踪记录添加这些扩展。

与其他日志和跟踪记录字段一样，开发者可以使用 HPEL API 来访问记录扩展。编写工具以从日志和跟踪存储库读取时，此 API 很有用。开发者还可以在运行时使用日志记录上下文 API 来访问定制日志处理程序、过滤器和格式化程序中的扩展。

下表描述了日志和跟踪扩展，包括您可用于过滤跟踪的各个方面的标识。

表 62: 日志和跟踪扩展

一个包含两列的表，此表列示二进制日志记录的日志和跟踪扩展。

扩展	描述
appName	指定与日志或跟踪记录相关的 Java™ Platform Enterprise Edition (Java™ EE) 应用程序（如果存在）的名称。
requestID	指定与每个日志或跟踪记录相关的请求（如果存在）的唯一标识。要使应用程序服务器能够将 requestID 扩展添加到日志和跟踪记录，必须启用跨组件跟踪 (XCT)，在管理控制台中也称为“日志和跟踪关联”。仅为某些类型的请求添加请求标识，如 HTTP 或 JMS 请求。
SIPCallId	指定 SIP 代理服务器或 SIP 容器正在处理的 SIP 调用标识。此信息在各个 SIP 代理服务器和 SIP 容器中是相同的。可以使用此扩展在各种组件之间跟踪 SIP 调用流。启用 HPEL 记录后，SIP 代理服务器和 SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPASId	指定 SIP 容器要处理的 SIP 应用程序会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPSessionId	指定 SIP 容器要处理的 SIP 会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPCallId2	指定与同一个 SIP 应用程序会话相关联并且 SIP 容器正在处理的第二个 SIP 调用标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。 如果有两个以上的 SIP 调用标识与单个 SIP 应用程序会话相关联，那么仅记录前两个标识。不会记录其他标识。
SIPSessionId2	指定与同一个 SIP 应用程序会话相关联并且 SIP 容器正在处理的第二个 SIP 会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。 如果有两个以上的 SIP 会话标识与单个 SIP 应用程序会话相关联，那么仅记录前两个标识。不会记录其他标识。
thread	指定每个日志或跟踪记录的相关请求的线程名称。

2.12 xigemaAS 的参考信息

参考信息的建立是为了帮助您快速定位有关 xigemaAS 概要文件的特定事实。本节从整个 xigemaAS 概要文件文档收集参考信息。

设置

设置是可使用管理控制台、配置文件或通过其他方法来配置的属性。请按包含要查看的设置的管理控制台页面的名称或要配置的设置的常规类型来查找设置。

定制属性

定制属性是特殊的“写入”设置。它们是可在特定管理控制台页面上输入的“名称/值”对。

管理员脚本编制接口

查找脚本编制对象或命令类，以找到有关其命令语法的详细信息。

故障诊断提示

查找组件，以找到针对该组件的故障诊断的建议。

2.12.1 JVM 监视

您可使用 xigemaAS 中用于 JVM 监视的 `JvmStats` MXBean。

每个 xigemaAS 实例均具有一个 `JvmStats` MXBean。

用于确定 JVM MXBean 的 `ObjectName` 是：

```
WebSphere:type=JvmStats
```

可用实例数 = 1

此 MXBean 负责 JVM 的报告性能。下列属性适用于 JVM。

堆信息

- 可用堆大小（以字节计）
- 堆中 JVM 使用的内存总大小（以字节计）
- 堆大小（以字节计）

CPU 信息

- 此 JVM 耗用的 CPU 百分比

垃圾回收 **(GC)** 信息

- JVM 启动后进行的 GC 次数
- GC 活动所花的总时间

常规信息

- JVM 启动后经过的时间（以毫秒计）。

计数器定义（**MXBean** 的属性）

- `Heap`：用于当前 JVM 的堆大小。
- `FreeMemory`：适用于当前 JVM 的可用堆。
- `UsedMemory`：当前 JVM 所使用的堆。
- `ProcessCPU`：JVM 进程使用的 CPU 百分比。
- `GcCount`：JVM 启动后进行的 GC 次数。
- `GcTime`：GC 时间的总累计值。
- `UpTime`：JVM 启动后的时间（以毫秒计）。

管理接口

JVM 监视的管理接口是 `com.ibm.websphere.monitor.jmx.JvmMXBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.12.2 ThreadPool 监视

您可使用 xigemaAS 中用于线程池监视的 `ThreadPool` MXBean。

所有 Web 请求都在线程池（名称为 **Default Executor** 的线程池）中执行。可以使用 `ThreadPoolMXBean` 来监视 **Default Executor** 线程池的使用情况。

用来确定线程池的 MXBean 的 `ObjectName` 为：

```
WebSphere:type=ThreadPoolStats,name=Default Executor
```

适用于 `ThreadPool` 的关键性能数据是：

- 池中的线程数，用以表示池大小。
- 服务请求的活动线程数。

`ThreadPool` 的属性

- `ActiveThreads`
- `PoolSize`
- `PoolName`（仅支持 Default Executor 线程池）

管理接口

`ThreadPool` 监视的管理接口是 `com.ibm.websphere.monitor.jmx.ThreadPoolMXBean`。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.12.3 Web 应用程序监视

您可使用 xigemaAS 的用于 Web 应用程序监视的 servlet MXBean。

Web 应用程序中的每个 Servlet 都具有性能数据。每个 servlet 都具有它自己的 MXBean。

用于确定每个 servlet MXBean 的 ObjectName 是：

```
WebSphere:type=ServletStats,name=<AppName>.<ServletName>
```

例如：

```
WebSphere:type=ServletStats,name=snoop.Alpine Snoop Servlet
WebSphere:type=ServletStats,name=MyApp.MyServlet
```

此 MXBean 负责报告每个 servlet 的 ServletStats。ServletStats MXBean 具有下列键数据：

- 请求计数
- 响应时间
- Servlet 名称
- 应用程序名称

计数器定义（**MXBean** 的属性）

- AppName：应用程序的名称。
- ServletName：servlet 的名称。
- RequestCount：此 servlet 的命中次数。
- ResponseTime：平均响应时间（纳秒）。
- Description：计数器的描述。
- RequestCountDetails：RequestCount 详细信息，包括最近的时间戳记。
- ResponseTimeDetails：ResponseTime 详细信息，包括取得的快照数、最小值和最大值。

管理接口

Web 应用程序监视的管理接口是 com.ibm.websphere.webcontainer.ServletStatsMXBean。您可以使用管理接口来获取代理对象。请参阅访问 *MBean* 属性和操作的示例（见第 1186 页）。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.12.4 xigemaAS 功能部件清单文件

xigemaAS 功能部件包含一个功能部件清单文件及一个或多个 OSGi 捆绑软件的集合，这类捆绑软件提供与 xigemaAS 概要文件运行时环境中的特定功能相对应的类和服务。您可以找到有关功能部件清单格式以及清单文件中每个头的含义的简介。

xigemaAS 概要文件中的功能部件清单文件使用 OSGi 企业 R5 规范中的子系统服务元数据格式。功能部件由存储在 lib/features 目录中的功能部件清单文件（.mf 文件）定义，并且必须使用定制类型的子系统：osgi.subsystem.feature。有关 OSGi 清单语法的更多信息，请参阅 OSGi 核心规范第 1.3.2 节。

注：在功能部件清单文件中，属性采用格式 *name=value*，但伪指令采用格式 *name:=value*。

定义了下列头：

表 63: 功能部件清单文件的头

此表显示 xigemaAS中的功能部件清单文件的头。第一列显示了头列表，第二列显示了每个头的描述，而第三列陈述是否需要头。

头	描述	是否必需
Subsystem-ManifestVersion	功能部件清单文件的版本格式。必须设置为 1。	否
Subsystem-SymbolicName	功能部件的符号名称，及任何属性或伪指令。	是
Subsystem-Version	功能部件的版本。有关如何定义此头的详细信息，请参阅 OSGi 核心规范部分 3.2.5 。	否
Subsystem-Type	功能部件的子系统类型。所有功能部件当前都是同一种子系统类型：osgi.subsystem.feature。	是
Subsystem-Content	功能部件的子系统内容。这是运行此功能部件所需的捆绑软件和子系统的逗号分隔列表。如果要允许在 server.xml 文件中配置自动功能部件，那么必须具有包含所需功能部件的功能头，并且也必须在子系统内容中定义所需功能部件。	是
Subsystem-Localization	功能部件的本地化文件的位置。	否
Subsystem-Name	功能部件的人类可读短名称。此值可以本地化。	否
Subsystem-Description	功能部件的描述。此值可以本地化。	否
IBM-Feature-Version	此子系统类型的版本。必须设置为 2。	是
IBM-Provision-Capability	功能头，用来描述是否可以自动地供应功能部件。	否
IBM-API-Package	由此功能部件、其他产品扩展中的功能部件和 xigemaAS 内核向应用程序提供的 API 包。	否
IBM-API-Service	由此功能部件提供给 OSGi 应用程序的 OSGi 服务。	否
IBM-SPI-Package	由此功能部件向其他产品扩展中的功能部件和 xigemaAS 内核提供的 SPI 包。	否
IBM-ShortName	功能部件的短名称。	否
IBM-AppliesTo	此功能部件所适用于的 xigemaAS 版本。	否
Subsystem-License	此功能部件的许可类型。	否
IBM-License-Agreement	许可协议文件的位置的前缀。	否
IBM-License-Information	许可信息文件的位置的前缀。	否

头	描述	是否必需
IBM-App-ForceRestart	指定对运行中服务器安装或卸载功能部件后将重新启动应用程序。	否

Subsystem-SymbolicName

此头的语法与捆绑软件的 Bundle-SymbolicName 语法匹配。它具有遵循包名称样式语法的符号名称，并且可以选择性地接受一组属性和伪指令。

支持下列属性：

- superseded。此属性指示此功能部件是否由功能的一个或多个功能部件或项目取代。它接受下列其中一个值：
  - true - 取代此功能部件。
  - false - 不取代此功能部件。

  此属性是可选的；缺省值为 false。

  有关更多信息，请参阅取代的功能部件。
- superseded-by。此属性指定取代此功能部件的逗号分隔功能部件列表（如果有），可选。

支持以下伪指令：

- visibility。此伪指令采用下列其中一个值：
  - public - 将功能部件认为是 API。此功能部件受开发者工具支持，用于 server.xml 文件中，并且在消息中输出。
  - protected - 将功能部件认为是 SPI。此功能部件不受开发者工具支持，不用于 server.xml 文件中，也不在消息中输出。提供此功能部件是为了扩展程序可以利用它来构建更高级别的功能部件。
  - private -（缺省值）此功能部件是产品的内部功能部件。不支持将此功能部件用于 server.xml 文件中，也不供扩展程序功能部件引用。随时都可以更改此功能部件，其中包括在两个修订包之间更改此功能部件。

例如：

```
Subsystem-SymbolicName: com.ibm.example.feature-1.0;
    visibility:=public; superseded=true; superseded-by="com.ibm.example.feature-2.0"
```

如果 superseded-by 列表中的功能部件名称用方括号 [] 括起来，那么此功能部件已经与取代功能部件分隔开。在以下示例中，功能部件 appSecurity-1.0（此功能部件包含功能部件 servlet-3.0 和 ldapRegistry-3.0）被功能部件 appSecurity-2.0（此功能部件未包含功能部件 servlet-3.0 和 ldapRegistry-3.0）取代：

```
IBM-ShortName: appSecurity-1.0
Subsystem-SymbolicName: com.ibm.websphere.appserver.appSecurity-1.0; visibility:=public;
superseded=true; superseded-by="appSecurity-2.0, [servlet-3.0], [ldapRegistry-3.0]"
```

有关更多信息，请参阅独立的功能部件。

- singleton. 此伪指令采用下列其中一个值：
  - *True*。此功能部件是单例功能部件。
  - *False*。此功能部件不是单例功能部件。

singleton 伪指令是可选的。缺省值为 *False*。

此伪指令用于声明特定功能部件是单例。单例意味着一次只能将给定功能部件的一个版本装入至运行时。缺省情况下，功能部件不是单例。如果该功能部件是单例，并且服务器配置需要给定功能部件的多个版本，那么运行时将尝试查找所有必需功能部件容许的通用版本。有关版本容许的更多信息，请参阅 Subsystem-Content 下的 ibm.tolerates 伪指令。

如果功能部件是单例，那么符号名称值为“<singleton feature name >-<singleton version>”形式，其中 name 和 version 用连字符分隔。单例功能部件名称可包含连字符，但跟在最后一个连字符之后的字符将解释为单例版本。如果跟在最后一个连字符后的字符是无效版本，那么将使用单例版本 0.0.0 并且完整符号名称将用作单例名称。处理 Subsystem-Content 下的 ibm.tolerates 伪指令时将使用单例版本；例如：

```
Subsystem-SymbolicName: com.ibm.example.feature-1.0;
    visibility:=public; singleton:=true
```

Subsystem-Content

此头针对运行时和安装定义功能部件的内容。它遵循的头语法和子系统规范相同，语法如下所示：

```
Subsystem-Content ::= content ( ',' content )*
        content ::= unique-name ( ';' parameter )*
        unique-name ::= unique-name  (see OSGi core spec section 1.3.2)
```

`unique-name` 使用 `Bundle-SymbolicName` 或 `Subsystem-SymbolicName` 头格式。支持下列属性：

- version - 查找捆绑软件时要匹配的版本范围。仅选择此范围中的捆绑软件。版本范围的典型示例是 `[1,1.0.100)`。
- type - 要供应的内容类型。可以指定任何值来指示内容类型；某些类型将促使在使用该功能部件的服务器的 OSGi 框架中安装并启动捆绑软件，而所有类型都会促使将内容包括在含有该功能部件的安装软件包中。预定义了下列值：
  - `osgi.bundle` - 这是缺省值，指示一个 OSGi 捆绑软件（应该同时供应给服务器的 OSGi 框架以及安装软件包）。
  - `osgi.subsystem.feature` - 此值指示应该将功能部件同时供应给服务器的 OSGi 框架以及安装软件包。这些功能部件需要使用 `Subsystem-SymbolicName` 头中指定的名称。
  - `jar` - 此值指示安装软件包中应包括 JAR 文件，并且此 JAR 文件是通过使用版本范围和/或位置值的组合选择的。
  - `file` - 此值指示 location 属性中所标识的文件应该包括在安装软件包中。

以下伪指令受支持：

- location - 捆绑软件的位置。对于捆绑软件或 JAR 类型，此值可以是目录的逗号分隔列表（用于标识 dev 目录中的规范和 API 捆绑软件），也可以是直接指向 JAR 文件的单个条目。如果 type 为 `file`，那么仅允许此值为单个条目，并且它必须直接指向该文件。

  例如：

```
Subsystem-Content: com.ibm.websphere.appserver.api.basics; version="[1,1.0.100)"; type=jar;
 location:="dev/api/xigemaas/,lib/",
                  com.ibm.websphere.appserver.spi.application;
          location:="dev/spi/xigemaas/com.ibm.websphere.appserver.spi.application_1.0.0.jar"; type="jar",
                  com.ibm.websphere.appserver.spi.application_1.0.0-javadoc.zip;
          location:="dev/spi/xigemaas/javadoc/com.ibm.websphere.appserver.spi.application_1.0.0-
javadoc.zip"; type="file"
```

- start-phase - 在系统启动期间捆绑软件应该启动的开始阶段。start-phase 伪指令可采用下列其中一个值：

- SERVICE - 此值指示最早阶段。缺省情况下，它映射到开始阶段 9。
- CONTAINER - 如果未提供任何 start-phase，那么这是缺省值。启动应用程序容器时，它指示容器阶段。缺省情况下，它映射到开始阶段 12。
- APPLICATION - 启动应用程序时，此值指示最晚的阶段。

捆绑软件也可以定义为刚好在这些阶段之前或之后启动，方法是添加晚于关键阶段的 _LATE 或者添加早于关键阶段的 _EARLY。因此，如果要在容器阶段之后立即运行，请使用 CONTAINER_LATE；如果要在 APPLICATION 阶段之前运行，那么使用 APPLICATION_EARLY。

- ibm.tolerates - 指定存在版本冲突时要在系统中提供的单例功能部件 *type=osgi.subsystem.feature* 的备用单例版本。

unique-name 指定单例功能部件的首选版本的符号名称。如果已知包含功能部件使用给定单例功能部件的其他单例版本，那么可使用 ibm.tolerates 伪指令指定这些单例版本。其他功能部件定义的给定单例功能部件的必需版本值存在冲突时，这会给予定义功能部件更高兼容性。

ibm.tolerates 伪指令中列示的单例版本仅在存在版本冲突时使用。ibm.tolerates 伪指令中列示的版本顺序不重要 - 可选择 ibm.tolerates 伪指令中列示的任何版本来满足依赖关系要求。

给定单例功能部件的容许版本必须显式列示在 ibm.tolerates 伪指令中。使用逗号来分隔容许版本列表。不支持指定版本范围。

例如：

```
Subsystem-Content: com.ibm.websphere.appserver.example.featureA-1.1; ibm.tolerates:="1.2";
 type="osgi.subsystem.feature",
                    com.ibm.websphere.appserver.example.featureB-1.1; ibm.tolerates:="1.2, 1.4, 1.6";
 type="osgi.subsystem.feature"
```


注：

容许版本不可转移。这可以避免您的功能部件所依赖的功能部件被自动选为支持更高级别的功能部件而不经测试。

例如：用户功能部件 featureC-1.1 在其清单文件的 Subsystem-Content 头中包含 sipServlet-1.1。sipServlet-1.1 包含 servlet-3.0 并容许 servlet 3.1。如果 featureC-1.1 是在 servlet-3.1 存在之前编写的，然后它使用的功能部件 (sipServlet-1.1) 添加并容许 servlet-3.1，那么 featureC-1.1 应说明它是否也容许 servlet-3.1。

如果将 server.xml 文件配置为具有以下两个功能部件：

```
<feature>usr:featureC-1.1</feature> // includes: sipServlet-1.1
<feature>websocket-1.0</feature> // this feature requires servlet-3.1
```

那么您将见到系统显示类似如下的错误消息：

```
CWWKF0033E: 不能同时装入单例功能部件 servlet-3.0 和 servlet-3.1。
所配置功能部件 usr:featureC-1.1 和 websocket-1.0 包含的一个或多个功能部件导致该冲突。
您的配置不受支持；请更新 server.xml 以移除不兼容的功能部件。
```

报告此错误是因为 featureC-1.1 未被选为容许 servlet-3.1，所以必须具有 servlet-3.0，并且 websocket-1.0 不支持 servlet-3.0，所以必须具有 servlet-3.1。

此解决方案适用于同样直接依赖于 servlet-3.0 并容许 servlet-3.1 的 featureC-1.1。

Subsystem-Name

使用此头，为功能部件提供人类可读的短名称。可以指定文字串或属性名。如果指定属性名，那么可以对该值进行本地化。

例如

```
Subsystem-Name: %name
```

其中，name 的值在 Subsystem-Localization 头（在先前示例中为 loc.properties）指定的位置处的属性文件中按以下格式定义：

```
name=feature_name
```

Subsystem-Description

使用此头，为功能部件提供描述。可以指定文字串或属性名。如果指定属性名，那么可以对该值进行本地化。

例如

```
Subsystem-Description: %desc
```

其中，desc 的值在 Subsystem-Localization 头（在先前示例中为 loc.properties）指定的位置处的属性文件中按以下格式定义：

```
desc=feature_description
```

IBM-Provision-Capability

清单中具有 IBM-Provision-Capability 头的功能部件是自动供应的功能部件。此头描述自动供应此功能部件前必须供应的其他功能部件。列示其他功能部件时，使用该功能部件的 Subsystem-SymbolicName 头。如果所有功能部件都是在 server.xml 文件中配置的，那么 runtime 会检查所有自动供应的功能部件的功能是否得以满足，如果已满足，那么会自动供应这些功能部件。

IBM-Provision-Capability 头的格式使用标准 OSGi LDAP 过滤器。

IBM-API-Package

使用此头来指示对于应用程序可见的 API 包。它与 Export-Package 头语法匹配。这意味着它是一个以逗号分隔的 API 包列表，但是每个 API 包都可以具有一些属性。

支持下列属性：

- type - API 包的类型。type 属性接受下列其中一个值：
  - spec - 指示由标准主体（例如 javax.servlet 或 org.osgi.framework）提供的 API。
  - ibm-api - 指示由 xigemaAS 提供的附加 API。
  - api - 指示用户定义的 API。这是缺省值。
  - third-party - 指示一个可见但并非由 xigemaAS 控制的 API。通常，这些是开放式源代码包。
  - internal - 指示非 API 包，必须向应用程序提供这些非 API 包，才能使应用程序正常运行。如果 Java™ 代码经过字节码增强或交织以在运行时添加对内部代码的引用，那么可使用此选项。

例如：

```
IBM-API-Package: javax.servlet; type="spec",
                 com.ibm.websphere.servlet.session; type="ibm-api",
                 com.ibm.wsspi.webcontainer.annotation; type="internal"
```

IBM-API-Service

使用此头来指示功能部件中对于 OSGi 应用程序可见的服务。此功能部件还必须在 OSGi 服务注册表中注册服务。

它具有下列语法

```
IBM-API-Service ::= service ( ',' service )*
                    service ::= service-name ( ';' attribute )*
                    service-name ::= unique-name
```

`service-name` 是服务的 Java™ 类或接口名称。这些属性解释为服务的服务特性。

例如：

```
IBM-API-Service: com.ibm.example.service.FeatureServiceOne;
                 myServiceAttribute=myAttributeValue,
                 com.ibm.example.service.FeatureServiceTwo
```

如果 OSGi 应用程序想要使用 `IBM-API-Service` 头提供的服务，那么应用程序必须包括对该服务的 Blueprint 引用，才能在该应用程序中供应该服务。

例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <reference id="FeatureServiceOneRef"
        interface="com.ibm.example.service.FeatureServiceOne" />
</blueprint>
```

为使服务在 OSGi 应用程序中可供捆绑软件使用，接口包必须对该捆绑软件可用，这意味着所使用捆绑软件的清单文件中的 `Import-Package` 头必须指定该接口包。功能部件捆绑软件中的 `Export-Package` 头和功能部件清单文件的 `IBM-API-Package` 头也必须指定该接口包。必须使用 OSGi `BundleContext` 接口或任何其他机制（例如，声明式服务或 Blueprint）在 OSGi 服务注册表中注册功能部件提供的服务。有关更多信息，请参阅开发使用简单激活的 *OSGi* 捆绑软件（见第 1243 页）和使用 *OSGi* 声明式服务来编写高级功能部件（见第 1247 页）。

IBM-SPI-Package

创建您自己的 xigemaAS 功能部件时，将它安装到用户产品扩展。您的功能部件中的所有包都可以供安装到用户产品扩展中的任何其他功能部件访问。但是，如果您想要让安装到另一个产品扩展中的功能部件访问您的功能部件中的包，那么必须在 IBM-SPI-Package 头中列出该包的名称。

IBM-SPI-Package 头中列出的任何包必须由 xigemaAS 功能部件中的捆绑软件导出，方法是将包列在捆绑软件清单文件的 Export-Package 头中。

IBM-ShortName

此头是功能部件的短名称，用于在 server.xml 文件中指定功能部件。如果清单文件中没有 `IBM-ShortName` 头，那么缺省情况下会使用 `Subsystem-SymbolicName`。`IBM-ShortName` 头仅对公共功能部件有效。

IBM-AppliesTo

此头指定此功能部件适用于的 xigemaAS 版本。提供以逗号分隔的项目列表，每个项目的格式如下所示：

```
product_id; productVersion=product_version; productInstallType=product_install_type;
 productEdition=product_editions
```

如果您提供多个项目，那么每个项目的 *product_id* 的值必须是唯一的。

productVersion 的值可以是确切版本（例如，9.0.0.1），也可以是最低版本（由以加号 (+) 结尾的版本指示，例如，9.0.0.1+）。

productEdition 的值可以是单个版本，也可以是版本的逗号分隔列表（括在引号中）。

例如：

```
IBM-AppliesTo: com.ibm.websphere.appserver; productVersion=8.5.5.6; productInstallType=Archive;
 productEdition="BASE,DEVELOPERS,EXPRESS,ND"
```

Subsystem-License

此头定义此功能部件的许可类型。如果您为 Subsystem-License 头提供值，但没有为 IBM-License-Agreement 头和 IBM-License-Information 头提供值，那么安装期间会显示 Subsystem-License 头值，供用户验收。

如果已安装具有相同 Subsystem-License 头值的功能部件，那么在安装期间不会显示许可，而且不要求许可批准。如果 `Subsystem-Content` 头中的依赖性意味着正在安装的两个或多个功能部件具有相同的 Subsystem-License 头值，那么安装期间用户只需接受许可一次。

例如：

```
Subsystem-License: L-JTHS-93TMHH
```

```
Subsystem-License: http://www.apache.org/licenses/LICENSE-2.0.html
```

IBM-License-Agreement

此头指定许可协议文件的位置的前缀。提供子系统归档中 LA_*language* 文件的文件路径，文件路径由“_”字符之前的内容表示（语言代码是由安装工具附加）。如果尚未接受此许可，那么安装功能部件时用户必须接受许可。这些许可证文件会复制到 xigemaAS 安装目录。

例如：

```
IBM-License-Agreement: lafiles/LA
```

IBM-License-Information

此头指定许可信息文件的位置的前缀。提供子系统归档中 LI_*language* 文件的文件路径，文件路径由“_”字符之前的内容表示（语言代码是由安装工具附加）。如果尚未接受此许可，那么安装功能部件时用户必须接受许可。这些许可证文件会复制到 xigemaAS 安装目录。

例如：

```
IBM-License-Information: lafiles/LI
```

IBM-App-ForceRestart

对运行中服务器安装或移除功能部件后，此头会导致应用程序重新启动。此头可以采用下列其中一个值：

- `install` - 安装功能部件后重新启动应用程序。
- `uninstall` - 卸载功能部件后重新启动应用程序。
- `install,uninstall` - 安装或卸载功能部件后重新启动应用程序。

示例功能部件清单文件

以下示例显示了 `example-1.0` 功能部件的定义。公共可见性属性允许直接在服务器配置 (server.xml) 文件中指定此功能部件；此功能部件还将包括在开发者工具的**服务器配置**视图中所显示的功能部件下拉列表中，并且将可包括在其他产品扩展中的功能部件中。进行运行时安装时，如果已将此功能部件安装到 usr 产品扩展中，那么可以通过在 server.xml 文件中包括以下代码来将此功能部件配置到服务器中：

```
<featureManager>
 <feature>usr:example-1.0</feature>
</featureManager>
```

在服务器中配置此功能部件将导致在服务器运行时环境的 OSGi 框架中安装并启动所指定捆绑软件 com.ibm.example.bundle1。单个 API 包 (com.ibm.example.publicapi) 将对该服务器中的所有应用程序可见，配置为对 `api` 包类型不可见的 Java™ EE 应用程序除外。如果 OSGi 应用程序希望使用该包，那么它们必须显式导入该包。两个 SPI 包（com.ibm.example.spi.utils 和 com.acme.spi.spiservices）将对服务器中的所有功能部件代码可见，就像 API 包一样。

```
IBM-Feature-Version: 2
Subsystem-ManifestVersion: 1.0
Subsystem-SymbolicName: com.ibm.example-1.0; visibility:=public
Subsystem-Version: 1.0.0.qualifier
Subsystem-Type: osgi.subsystem.feature
Subsystem-Content: com.ibm.example.bundle1; version="1.0.0"
Subsystem-Localization: OSGI-INF/l10n/loc
Manifest-Version: 1.0
Subsystem-Name: %name
Subsystem-Description: %desc
IBM-API-Package: com.ibm.example.publicapi; type="api"
IBM-SPI-Package: com.ibm.example.spi.utils, com.ibm.example.spi.spiservices
IBM-ShortName: example-1.0
```

2.12.5 xigemaAS 功能部件

功能部件是您用于控制载入到特定服务器的运行时环境部件的功能单元。

下表列示了 xigemaAS 中支持的功能部件。

- **Java™ EE 7 Web Profile**
  - *beanValidation-1.1*
  - *cdi-1.2*
  - *ejbLite-3.2*
  - *el-3.0*
  - *jaxrs-2.0*
  - *jaxrsClient-2.0*
  - *jdbc-4.1*
  - *jndi-1.0*
  - *jpa-2.1*

  - *jsf-2.2*
  - *jsonp-1.0*
  - *jsp-2.3*
  - *managedBeans-1.0*
  - *servlet-3.1*
  - *webProfile-7.0*
  - *websocket-1.0*
  - *websocket-1.1*
- **Java EE 7 Full Platform**
  - *appClientSupport-1.0*
  - *appSecurityClient-1.0*
  - *batch-1.0*
  - *concurrent-1.0*
  - *ejb-3.2*
  - *ejbHome-3.2*
  - *ejbPersistentTimer-3.2*
  - *ejbRemote-3.2*
  - *j2eeManagement-1.1*
  - *jacc-1.5*
  - *jaspic-1.1*
  - *javaee-7.0*
  - *javaeeClient-7.0*
  - *javaMail-1.5*
  - *jaxb-2.2*
  - *jaxws-2.2*
  - *jca-1.7*
  - *jcaInboundSecurity-1.0*
  - *jms-2.0*
  - *jmsMdb-3.2*
  - *mdb-3.2*
  - *wasJmsClient-2.0*
  - *wasJmsSecurity-1.0*
  - *wasJmsServer-1.0*
- **Extended Programming Models**
  - *cloudant-1.0*
  - *couchdb-1.0*
  - *distributedMap-1.0*
  - *json-1.0*
  - *microProfile-1.0*
  - *mongodb-2.0*
  - *redisdb-1.0*
  - *rtcomm-1.0*
  - *rtcommGateway-1.0*
- **Enterprise OSGi**
  - *blueprint-1.0*
  - *osgiAppIntegration-1.0*

- *osgi.jpa-1.0*
- *wab-1.0*

- **Operations**
  - *appSecurity-1.0*
  - *appSecurity-2.0*
  - *batchManagement-1.0*
  - *bells-1.0*
  - *eventLogging-1.0*
  - *ldapRegistry-3.0*
  - *localConnector-1.0*
  - *monitor-1.0*
  - *oauth-2.0*
  - *openid-2.0*
  - *openidConnectClient-1.0*
  - *openidConnectServer-1.0*
  - *osgiConsole-1.0*
  - *passwordUtilities-1.0*
  - *requestTiming-1.0*
  - *samlWeb-2.0*
  - *restConnector-1.0*
  - *restConnector-2.0*
  - *serverStatus-1.0*
  - *sessionDatabase-1.0*
  - *redisSession-1.0*
  - *spnego-1.0*
  - *ssl-1.0*
  - *timedOperations-1.0*
  - *webCache-1.0*
  - *wmqJmsClient-2.0*
  - *wsSecurity-1.1*
  - *WS-AT Service*（见第 1023 页）
- **Systems Management**
  - *adminCenter-1.0*
- **Java™ EE 6 Web Profile**
  - *beanValidation-1.0*
  - *cdi-1.0*
  - *ejbLite-3.1*
  - *jdbc-4.0*
  - *jndi-1.0*
  - *jpa-2.0*
  - *jsf-2.0*
  - *jsp-2.2*
  - *servlet-3.0*
  - *webProfile-6.0*
- **Java™ EE 6 Technologies**
  - *jaxb-2.2*

- *jaxrs-1.1*
- *jaxws-2.2*
- *jca-1.6*
- *jcaInboundSecurity-1.0*
- *jms-1.1*
- *jmsMdb-3.1*
- *mdb-3.1*
- *wasJmsClient-1.1*
- *wasJmsSecurity-1.0*
- *wasJmsServer-1.0*
- *wmqJmsClient-1.1*

功能部件描述

以下列表包含可以添加至服务器配置的功能部件的相关信息。在配置中包含功能部件可促使自动装入一个或多个其他功能部件。例如，如果包含 `wab-1.0` 功能部件，那么会自动装入 `servlet-3.0` 和 `blueprint-1.0` 功能部件。每个功能部件都包含一段简短描述，并举例说明如何在 server.xml 文件内部的 `<featureManager>` 元素中声明该功能部件。例如：

```
<server>
  <featureManager>
    <feature>servlet-3.0</feature>
    <feature>localConnector-1.0</feature>
  </featureManager>
</server>
```

Java EE 7 Web 概要文件

Bean 验证

`<feature>beanValidation-1.1</feature>`

`beanvalidation-1.1` 功能部件在应用程序的每一层提供对 JavaBeans™ 的验证。可以通过使用注解或 validation.xml 部署描述符，将验证应用到应用程序中 JavaBeans™ 的所有层。

请参阅 *Bean 验证功能部件限制*（见第 1671 页）。

有关 `beanValidation-1.1` 功能部件配置信息，请参阅 *Bean Validation 1.1*（见第 939 页）。

CDI

`<feature>cdi-1.2</feature>`

`cdi-1.2` 功能部件在 xigemaAS 上启用对上下文和依赖性注入 1.2 规范的支持。

请参阅 *在 xigemaAS 上管理上下文和依赖性注入应用程序*（见第 1222 页）。

有关 `cdi-1.2` 功能部件配置信息，请参阅 *Contexts and Dependency Injection 1.2*（见第 943 页）。

Enterprise JavaBeans (EJB) Lite

`<feature>ejbLite-3.2</feature>`

`ejbLite-3.2` 功能部件为依照 EJB 3.2 规范的 EJB Lite 子集编写的 EJB 应用程序提供支持。

请注意，EJB 3.2 Lite API 组未包含可嵌入 EJB 容器，并且产品未提供 EJB 3.2 可嵌入容器。

而且，以下功能部件与 `ejbLite-3.2` 功能部件不兼容：

- cdi-1.0
- jmsMdb-3.1
- mdb-3.1

请参阅在 *xigemaAS* 上开发 *EJB* 应用程序（见第 1478 页）。

有关 ejbLite-3.2 功能部件配置信息，请参阅*Enterprise JavaBeans Lite 3.2*（见第 949 页）。

表达式语言 3.0

```
<feature>el-3.0</feature>
```

此功能部件启用对表达式语言 (EL) 3.0 的支持。

请参阅配置 *xigemaAS* 以使用 *Expression Language 3.0*（见第 1219 页）。

有关 el-3.0 功能部件配置信息，请参阅*Expression Language 3.0*（见第 952 页）。

Java™ API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-2.0</feature>
```

jaxrs-2.0 功能部件在 xigemaAS 上为 Java™ API for RESTful Web Services 提供支持。

请参阅 配置 *JAX-RS 2.0* 客户机（见第 1513 页）和 将 *JAX-RS 2.0* 与 *EJB* 和 *CDI* 集成（见第 1522 页）。

有关 jaxrs-2.0 功能部件配置信息，请参阅 *Java RESTful Services 2.0*（见第 987 页）。

Java™ EE Client API for JAX-RS 2.0

```
<feature>jaxrsClient-2.0</feature>
```

jaxrsClient-2.0 功能部件提供对 Java Client API for JAX-RS 2.0 的支持

请参阅配置 *JAX-RS 2.0* 客户机（见第 1513 页）和将 *JAX-RS 2.0* 与 *EJB* 和 *CDI* 集成（见第 1522 页）。

有关 jaxrsClient-2.0 功能部件配置信息，请参阅*Java RESTful Services Client 2.0*（见第 988 页）。

Java™ 数据库连接 (JDBC)

```
<feature>jdbc-4.1</feature>
```

您可以采用某一使用 Java™ 数据库连接 (JDBC) 和数据源的现有应用程序，然后将该应用程序部署到服务器。jdbc-4.1 功能部件提供对访问数据库的应用程序的支持。

请参阅在 *xigemaAS* 中配置数据库连接（见第 1196 页）。

有关 jdbc-4.1 功能部件配置信息，请参阅*Java Database Connectivity 4.1*（见第 969 页）。

Java™ 命名和目录接口 (JNDI)

```
<feature>jndi-1.0</feature>
```

jndi-1.0 功能部件为 xigemaAS 的服务器配置中的单一 JNDI 条目定义提供支持。

请参阅在 *xigemaAS* 功能部件中使用 *JNDI* 缺省命名空间开发（见第 1268 页）。

有关 jndi-1.0 功能部件配置信息，请参阅*Java Naming and Directory Interface*（见第 976 页）。

Java 持久性 API 2.1

```
<feature>jpa-2.1</feature>
```

jpa-2.1 功能部件提供对一些应用程序（这些应用程序使用依照 JPA 2.1 规范编写的应用程序管理及容器管理的 JPA）的支持，EclipseLink 支持此功能部件。

请参阅*Java Persistence API (JPA) 功能部件概述*（见第 1069 页）。

有关 jpa-2.1 功能部件配置信息，请参阅*Java Persistence API 2.1*（见第 979 页）。

JavaServer Faces (JSF)

```
<feature>jsf-2.2</feature>
```

此功能部件启用对使用 Java Server Faces (JSF) 2.2 框架的 Web 应用程序的支持。此框架简化了用户界面的构造。

请参阅配置 *xigemaAS 以使用 JavaServer Faces 2.2*（见第 1220 页）。

有关 jsf-2.2 功能部件配置信息，请参阅*JavaServer Faces 2.2*（见第 999 页）。

JavaScript™ 对象表示法处理

```
<feature>jsonp-1.0</feature>
```

Java™ API for JSON Processing (JSON-P) 功能部件提供一种标准化方法以构造和处理要以 JavaScript™ 对象表示法 (JSON) 呈现的数据。

有关 jsonp-1.0 功能部件配置信息，请参阅*JavaScript Object Notation Processing*（见第 996 页）。

JavaServer Pages (JSP)

```
<feature>jsp-2.3</feature>
```

此功能部件启用对写至 JSP 2.3 规范的 Java Server Pages (JSP) 的支持。此框架简化了用户界面的构造。启用此功能部件还会启用 Expression Language (EL) V3.0 功能部件。el-3.0。

请参阅配置 *xigemaAS 以使用 JavaServer Pages 2.3*（见第 1221 页）。

有关 jsp-2.3 功能部件配置信息，请参阅*JavaServer Pages 2.3*（见第 1001 页）。

受管 Bean

```
<feature>managedBeans-1.0</feature>
```

managedBeans-1.0 功能部件提供对受管 Bean 1.0 规范 (JSR-316) 的支持。此功能部件允许使用 javax.annotation.ManagedBean 注释。

有关 managedBeans-1.0 功能部件配置信息，请参阅*Java EE Managed Bean 1.0*（见第 971 页）。

Servlet 3.1

```
<feature>servlet-3.1</feature>
```

servlet-3.1 功能部件启用对依照 Java™ Servlet 3.1 规范编写的 HTTP Servlet 的支持。

请参阅配置 *xigemaAS 以使用 Servlet 3.1*（见第 1211 页）和*Servlet 3.1 行为更改*（见第 1212 页）。

有关 servlet-3.1 功能部件配置信息，请参阅*Java Servlets 3.1*（见第 990 页）。

Web 概要文件 7.0

```
<feature>webProfile-7.0</feature>
```

此功能部件提供支持 Java™ EE 7 Web 概要文件所需 xigemaAS 功能部件的便利组合。

请参阅*xigemaAS 中的 Java EE 7*（见第 24 页）。

有关 `webProfile-7.0` 功能部件配置信息，请参阅*Java EE Web Profile 7.0*（见第 973 页）。

WebSocket

```
<feature>websocket-1.0</feature>
```

```
<feature>websocket-1.1</feature>
```

WebSocket 是一种标准协议，它允许 Web 浏览器或客户机应用程序使用一种全双工连接与 Web 服务器应用程序通信。

请参阅 *xigemaAS: WebSocket* 和*在 xigemaAS 中开发 WebSocket 应用程序*（见第 1493 页）。

有关 `websocket-1.0` 功能部件配置信息，请参阅*Java WebSocket 1.0*（见第 993 页）。

有关 `websocket-1.1` 功能部件配置信息，请参阅*Java WebSocket 1.1*（见第 994 页）。

Java EE 7 完整平台

应用程序客户机支持

```
<feature>appClientSupport-1.0</feature>
```

`appClientSupport-1.0` 功能部件允许服务器处理应用程序的客户机模块内的 Java EE 元数据，例如，读取部署描述符 XML 文件和/或注释并在必要时将它们提供给应用程序中的其他模块。它还允许远程应用程序客户机进程与服务器通信以执行 JNDI 查找。

`appClientSupport-1.0` 功能部件仅在 `server.xml` 文件中启用。

有关 `appClientSupport-1.0` 功能部件配置信息，请参阅*Application Client Support for Server*（见第 931 页）。

应用程序客户机容器安全性

```
<feature>appSecurityClient-1.0</feature>
```

要在客户机容器上启用安全性，请将 `appSecurityClient-1.0` 功能部件添加至 client.xml 文件。

`appSecurityClient-1.0` 功能部件在客户机上启用 SSL、CSIv2 和 JAAS。必须配置 SSL 以确保客户机与服务器之间的通信是安全的和加密的。

请参阅 *xigemaAS 应用程序客户机容器上的安全性*（见第 1057 页） 和 *为 xigemaAS 应用程序客户机容器及其应用程序配置安全性*（见第 1395 页）。

有关 `appSecurityClient-1.0` 功能部件配置信息，请参阅 *Application Security for Client 1.0*（见第 934 页）。

批处理

```
<feature>batch-1.0</feature>
```

`batch-1.0` 功能部件允许使用 JSR-352 编程模型。

请参阅 *Java 批处理和受管批处理概述*（见第 1559 页）。

有关 batch-1.0 功能部件配置信息，请参阅 *Batch API 1.0*（见第 936 页）。

受管执行程序和线程工厂

<feature>concurrent-1.0</feature>

concurrent-1.0 功能部件支持创建受管执行程序服务，这些服务可让应用程序使用应用程序服务器所管理的线程上下文提交要同时运行的任务。此功能部件还允许创建受管线程工厂，以创建与查找该受管线程工厂的组件的线程上下文一起运行的线程。

请参阅配置受管调度执行程序（见第 1174 页）。

有关 concurrent-1.0 功能部件配置信息，请参阅 *Concurrency Utilities for Java EE 1.0*（见第 941 页）。

Enterprise JavaBeans (EJB)

<feature>ejb-3.2</feature>

ejb-3.2 功能部件为依照 EJB 3.2 规范编写的 EJB 应用程序提供支持。

此功能部件包含以下功能部件：

- <feature>ejbLite-3.2</feature>

  此功能部件提供对依照 EJB 3.2 规范的 EJB Lite 子集编写的 EJB 应用程序的支持。有关 ejbLite-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Lite 3.2*（见第 949 页）。
- <feature>ejbHome-3.2</feature>

  此功能部件提供对 EJB 2.x API 的支持。有关 ejbHome-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Home Interfaces 3.2*（见第 947 页）。
- <feature>ejbPersistentTimer-3.2</feature>

  此功能部件提供对持久 EJB 计时器的支持。有关 ejbPersistentTimer-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Persistent Timers 3.2*（见第 950 页）。
- <feature>ejbRemote-3.2</feature>

  此功能部件提供对远程 EJB 接口的支持。有关 ejbRemote-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans Remote 3.2*（见第 951 页）。
- <feature>mdb-3.2</feature>

  此功能部件提供对消息驱动的 Bean 的支持。有关 mdb-3.2 功能部件配置信息，请参阅 *Message-Driven Beans 3.2*（见第 1006 页）。

  mdb-3.2 功能部件取代了 jmsMdb-3.2 功能部件。有关 jmsMdb-3.2 功能部件的信息，请参阅部署消息驱动 *Bean* 以连接至 *xigemaMQ*（见第 1551 页）。

如果不需要完整 EJB 3.2 支持，那么可使用这些功能部件的各种组合来提供所需支持。

请参阅在 *xigemaAS* 上开发 *EJB* 应用程序（见第 1478 页）。

有关 ejb-3.2 功能部件配置信息，请参阅 *Enterprise JavaBeans 3.2*（见第 946 页）。

J2EE 管理 1.1

<feature>j2eeManagement-1.1</feature>

j2eeManagement-1.1 功能部件提供标准接口以用于管理 Java EE 7，并允许应用程序使用 JSR 77 规范中定义的接口。

要调用管理 EJB API，服务器配置必须在功能部件管理器中同时具有 `j2eeManagement-1.1` 和 `ejbRemote-3.2` 功能部件。如果这两个功能部件都已包含在服务器配置中，那么您可通过 JNDI 名称查找来调用管理 EJB API。管理 EJB 绑定名称（JNDI 查找名称）为 ejb/mejb/MEJB。

请参阅*j2eeManagement-1.1 功能部件限制*（见第 1672 页）。

有关 `j2eeManagement-1.1` 功能部件配置信息，请参阅*J2EE Management 1.1*（见第 953 页）。

Java 容器授权合同 1.5

```
<feature>jacc-1.5</feature>
```

`jacc-1.5` 功能部件启用对 Java 容器授权合同 (JACC) V1.5 的支持。为将 `jacc-1.5` 功能部件添加至服务器，您需要添加 xigemaAS 中未包含的第三方 JACC 提供程序。

请参阅开发 *Java Authorization Contract for Containers (JACC)* 授权提供程序（见第 1418 页）。

有关 `jacc-1.5` 功能部件配置信息，请参阅*Java Authorization Contract for Containers 1.5*（见第 964 页）。

Java™ 容器认证 SPI 1.1

```
<feature>jaspic-1.1</feature>
```

`jaspic-1.1` 功能部件支持使用 Java™ 容器认证 SPI (JASPIC) 提供者保护服务器运行时环境和应用程序（如 JSR-196 中所定义）。

请参阅配置 *Java Authentication SPI for Containers (JASPIC)* 用户功能部件（见第 1309 页）。

有关 `jaspic-1.1` 功能部件配置信息，请参阅*Java Authentication SPI for Containers 1.1*（见第 963 页）。

Java EE

```
<feature>javaee-7.0</feature>
```

此功能部件提供支持 Java EE 7.0 完整平台时所需的 xigemaAS 功能部件的便利组合。

请参阅*xigemaAS 中的 Java EE 7*（见第 24 页）。

有关 `javaee-7.0` 功能部件配置信息，请参阅*Java EE Full Platform 7.0*（见第 970 页）。

Java EE 应用程序客户机 7.0

```
<feature>javaeeClient-7.0</feature>
```

此功能部件提供对 Java EE 应用程序客户机 7.0 的支持。

请参阅手动创建 *xigemaAS* 应用程序客户机（见第 1110 页）。

有关 `javaeeClient-7.0` 功能部件配置信息，请参阅*Java EE V7 Application Client*（见第 971 页）。

JavaMail™ API

```
<feature>javaMail-1.5</feature>
```

JavaMail™ API 支持外部邮件服务器与 xigemaAS 应用程序之间的通信。请参阅在 *xigemaAS* 上管理 *JavaMail*（见第 1228 页）。

有关 `javaMail-1.5` 功能部件配置信息，请参阅*JavaMail 1.5*（见第 995 页）。

Java™ XML 绑定体系结构 (JAXB)

```
<feature>jaxb-2.2</feature>
```

jaxb-2.2 功能部件在 xigemaAS 上为“Java™ XML 绑定体系结构”(JAXB) 提供支持。

请参阅 *jaxb-2.2 功能部件限制*（见第 1672 页）。

有关 jaxb-2.2 功能部件配置信息，请参阅*Java XML Bindings 2.2*（见第 994 页）。

Java™ API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

jaxws-2.2 功能部件在 xigemaAS 上为 Java™ API for XML-Based Web Services 提供支持。

- 对于支持 JAX-WS 编程模型的 Web 应用程序，必须在 server.xml 文件中启用 servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于支持 JAX-WS 编程模型的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1、servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

请参阅将 *JAX-WS 应用程序部署到 xigemaAS*（见第 1531 页）和*jaxws-2.2 功能部件限制*（见第 1673 页）。

有关 jaxws-2.2 功能部件配置信息，请参阅*Java Web Services 2.2*（见第 992 页）。

Java EE 连接器体系结构 1.7

```
<feature>jca-1.7</feature>
```

jca-1.7 功能部件提供了配置元素来定义连接工厂、受管对象和激活规范的实例，并将这些实例与安装的资源适配器关联。

有关 jca-1.7 功能部件配置信息，请参阅*Java Connector Architecture 1.7*（见第 967 页）。

Java™ EE 连接器体系结构入站安全性

```
<feature>jcaInboundSecurity-1.0</feature>
```

jcaInboundSecurity-1.0 功能部件针对资源适配器启用安全性流程。

有关 jcaInboundSecurity-1.0 功能部件配置信息，请参阅*Java Connector Architecture 1.0 Security Inflow*（见第 965 页）。

Java™ 消息服务 2.0

```
<feature>jms-2.0</feature>
```

jms-2.0 功能部件支持将资源适配器配置为使用 Java 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.6 规范的任何 JMS 资源适配器。

请参阅对*xigemaAS 启用 JMS 消息传递*和*将消息传递应用程序部署到 xigemaAS*（见第 1547 页）。

有关 jms-2.0 功能部件配置信息，请参阅 *Java 消息传递服务 2.0*。

嵌入式 **xigemaAS 消息传递功能部件**

```
<feature>wasJmsClient-2.0</feature>
```

wasJmsClient-2.0 功能部件取代了 wasJmsClient-1.1 功能部件。wasJmsClient-2.0 功能部件符合 JMS 2.0 规范并且仅在 IBM JDK 7 或更高版本上受支持。

请参阅对*xigemaAS* 启用 *JMS* 消息传递和将消息传递应用程序部署到 *xigemaAS*（见第 1547 页）。

有关 wasJmsClient-2.0 功能部件配置信息，请参阅*JMS 2.0 Client for Message Server*（见第 957 页）。

<feature>wasJmsSecurity-1.0</feature>

wasJmsSecurity-1.0 功能部件支持与消息传递引擎建立安全连接。如果已启用 wasJmsSecurity-1.0 功能部件，那么它会开始对尝试连接至消息传递引擎的用户进行认证和授权。针对在 server.xml 文件中定义的注册表来认证用户。如果用户想要访问主题或队列等目标，那么必须向用户授予必需的许可权。在 server.xml 文件中的 <messagingSecurity> 元素（这是 messagingEngine 元素的子元素）中定义了对于该目标的访问权。如果添加了 wasJmsSecurity-1.0 功能部件，但是 server.xml 文件中未定义 <messagingSecurity> 元素，那么用户将无法连接至消息传递引擎，也无法执行任何消息传递操作（例如，将消息发送至目标或者从目标接收消息）。

有关 wasJmsSecurity-1.0 功能部件配置信息，请参阅*Message Server Security 1.0*（见第 1004 页）。

👉 ：

- 配置用户注册表是 wasJmsSecurity-1.0 功能部件的先决条件。请确保在启用 wasJmsSecurity-1.0 功能部件之前已配置用户注册表。
- 启用 wasJmsSecurity-1.0 功能部件时，您还必须在 server.xml 文件中配置 <messagingSecurity> 元素（这是 <messagingEngine> 元素的子元素）。此配置使得授权用户能够访问消息传递目标。

<feature>wasJmsServer-1.0</feature>

wasJmsServer-1.0 功能部件使 JMS 消息传递引擎运行时能够进行初始化。消息传递运行时负责提供应用程序连接，管理目标（例如，主题或队列）的状态，以及处理服务质量、安全性和事务。此功能部件还支持来自远程消息传递应用程序的入站连接。远程消息传递应用程序可以通过基于 SSL 或者非 SSL 的 TCP/IP 来连接至 JMS 消息传递引擎。

要使用 SSL 进行连接，您必须启用 SSL 功能部件。

请参阅 对 *xigemaAS* 启用安全 *JMS* 消息传递。

有关 wasJmsServer-1.0 功能部件配置信息，请参阅*Message Server 1.0*（见第 1003 页）。

扩展编程模型

Cloudant 集成

<feature>cloudant-1.0</feature>

此功能部件通过提供服务器配置中配置的连接器实例来启用与 Cloudant 的连接。连接器实例可以通过 JNDI 注入或访问。应用程序通过 Cloudant 客户机库来使用连接器实例。请参阅 在 *xigemaAS* 中通过 *Cloudant Java* 客户机库配置 *CouchDB* 连接。

对于 cloudant-1.0 功能部件配置信息，请参阅 *Cloudant Integration 1.0*（见第 940 页）。

CouchDB

<feature>couchdb-1.0</feature>

couchdb-1.0 功能部件支持 CouchDB 实例以及相关联的数据库连接。可以通过 JNDI 查找或资源注入来提供对 CouchDB 连接的访问权。

请参阅在 *xigemaAS* 中使用 *Ektorp* 客户机库配置 *CouchDB* 连接（见第 1209 页）。

有关 couchdb-1.0 功能部件配置信息，请参阅 *CouchDB Integration 1.0*（见第 944 页）。

高速缓存服务

```
<feature>distributedMap-1.0</feature>
```

此功能部件提供可使用 DistributedMap API 访问的本地高速缓存服务。缺省高速缓存绑定在 JNDI 内的 services/cache/distributedmap 中。可通过添加网络高速缓存提供者来分配高速缓存。

有关 distributedMap-1.0 功能部件配置信息，请参阅*Distributed Map interface for Dynamic Caching*（见第 945 页）。

JavaScript™ 对象表示法 (JSON4J) 库

```
<feature>json-1.0</feature>
```

json-1.0 功能部件提供对 JSON4J 库的访问权，该库为 Java™ 环境提供一组 JSON 处理类。JSON4J 库提供简单的 Java™ 模型来构造和操控要作为 JSON 数据来呈现的数据。

有关 json-1.0 功能部件配置信息，请参阅*JavaScript Object Notation for Java*（见第 997 页）。

Micro Profile

```
<feature>microProfile-1.0</feature>
```

microProfile-1.0 功能部件对支持 Micro Profile for enterprise Java 的 xigemaAS 功能部件进行组合。

对于 microProfile-1.0 功能部件配置信息，请参阅 *Micro Profile 1.0*（见第 1007 页）

MongoDB

```
<feature>mongodb-2.0</feature>
```

mongodb-2.0 功能部件支持 MongoDB 实例以及相关联的数据库连接。可以通过 JNDI 查找或资源注入来提供对 MongoDB 连接的访问权。本机 com.mongodb API 可执行数据库处理。

请参阅在 *xigemaAS* 中配置 *MongoDB* 连接（见第 1205 页）。

有关 mongodb-2.0 功能部件配置信息，请参阅 *MongoDB Integration 2.0*（见第 1007 页）。

RedisDB

```
<feature>redisdb-1.0</feature>
```

redisdb-1.0 功能部件允许应用程序通过 Jedis API 与 redis DB 实例交互，提供对 redis 单节点和集群的数据源支持。应用程序可以通过 JNDI 查找或资源注入两种方式来访问 redis DB。此功能部件支持多数据源的配置方式。

请参阅在 *xigemaAS* 中配置 *Redis DB* 连接（见第 1207 页）。

有关 redisdb-1.0 功能部件的配置信息，请参阅 *Redis DB Integration 1.0*（见第 1016 页）。

实时通信

```
<feature>rtcomm-1.0</feature>
```

xigemaAS 实时通信功能部件将启用可高度伸缩的呼叫信号引擎，该引擎可用于将 WebRTC 客户机连接至实时音频/视频/数据呼叫。此功能部件支持客户机注册及在两个端点间创建 WebRTC 对等连接时所需的信号交换。

有关 功能部件配置信息，请参阅*Real-Time Communications*（见第 1018 页）。

RTCommGateway

```
<feature>rtcommGateway-1.0</feature>
```

rtcommGateway-1.0 功能部件添加将会话启动协议 (SIP) 与 RTComm WebRTC 端点连接的功能以交换音频和视频流。

有关 rtcommGateway-1.0 功能部件配置信息，请参阅*WebRTC Rtcomm Gateway*（见第 1026 页）。

SIP Servlet

```
<feature>sipServlet-1.1</feature>
```

sipServlet-1.1 功能部件提供对 SIP Servlet 规范 1.1（又称为 JSR 289）的支持。会话启动协议 (SIP) 是用于许多交互式服务（包括音频、视频和对等通信）的控制协议。

请参阅*xigemaAS: 会话启动协议 (SIP)*（见第 1082 页）和*在 xigemaAS 上管理会话启动协议 (SIP)*（见第 1225 页）。

有关 sipServlet-1.1 功能部件配置信息，请参阅 *SIP Servlet* （见第 1020 页）。

企业 **OSGi**

Blueprint

```
<feature>blueprint-1.0</feature>
```

blueprint-1.0 功能部件为部署使用 OSGi Blueprint Container 规范的 OSGi 应用程序启用支持。借助 xigemaAS中的 OSGi 应用程序支持，您可以开发并部署使用 Java™ EE 和 OSGi 技术的模块化应用程序。

请参阅*查找 OSGi 应用程序*（见第 1267 页）。

有关 blueprint-1.0 功能部件配置信息，请参阅 *OSGi Blueprint*（见第 1009 页）。

OSGi 应用程序集成

```
<feature>osgiAppIntegration-1.0</feature>
```

使用 osgiAppIntegration-1.0 功能部件以允许同一 Java 虚拟机内可用的 OSGi 应用程序相互共享它们的服务。

有关 osgiAppIntegration-1.0 功能部件配置信息，请参阅 *OSGi Application Integration*（见第 1009 页）。

OSGi JPA

```
<feature>osgi.jpa-1.0</feature>
```

osgi.jpa-1.0 功能部件在 xigemaAS 上为 OSGi 应用程序提供 JPA 支持。

请参阅*将 OSGi 应用程序部署到 xigemaAS* （见第 1500 页）。

有关 osgi.jpa-1.0 功能部件配置信息，请参阅 *OSGi Java Persistence API*（见第 1012 页）。

Web 应用程序捆绑软件 (WAB)

```
<feature>wab-1.0</feature>
```

此功能部件为下列过程启用支持：应用程序服务器中某些操作的运行速度慢于预期时对警告进行记录。wab-1.0 功能部件对企业捆绑软件内的 WAB 提供支持。

wab-1.0 功能部件对企业捆绑软件内的 WAB 提供支持。

此功能部件支持打包在 WAB 中的下列资源：

- 静态 Web 内容和 JSP。
- 按照 Servlet 3.0 规范编写的 HTTP Servlet。
- Blueprint 应用程序。

如果包含 wab-1.0 功能部件，那么也包含 servlet-3.0 和 blueprint-1.0 功能部件。

请参阅将 *OSGi 应用程序部署到 xigemaAS*（见第 1500 页）。

有关 wab-1.0 功能部件配置信息，请参阅*OSGi Web Application Bundles*（见第 1013 页）。

操作

API 发现

```
<feature>apiDiscovery-1.0</feature>
```

apiDiscovery-1.0 功能部件允许您发现 REST API 文档。使用此功能部件查找 xigemaAS 服务器上的可用 REST API，然后使用 Swagger 用户接口调用所发现的 REST 端点。请参阅 *在 xigemaAS 服务器上发现 REST API 文档*（见第 1609 页）。

安全性

```
<feature>appSecurity-2.0</feature>
```

此版本的 appSecurity 功能部件显式地根据其他功能部件的存在情况，仅提供某些方面的安全性。此外，它不会自动地包含 servlet-3.0 或 ldapRegistry-3.0 功能部件，从而减少了服务器占用量。要保护 Web 应用程序，必须包含 servlet-3.0 功能部件。要启用 EJB 安全性，必须包含 ejbLite-3.1 功能部件。要支持 LDAP 用户注册表，必须包含 ldapRegistry-3.0 功能部件。

注：

- appSecurity-2.0 功能部件将取代 appSecurity-1.0。除了 appSecurity-2.0 不会自动包括 servlet-3.0 或 ldapRegistry-3.0 之外，这两个功能部件在其他方面都相同。可以选择在服务器配置中改用 appSecurity-2.0 版本。请参阅被取代的功能部件（见第 906 页）。
  - 要启用 Web 安全性，必须在 server.xml 文件中指定 servlet-3.0 功能部件。
  - 要启用对于 LDAP 的支持，必须在 server.xml 文件中指定 ldapRegistry-3.0 功能部件。

appSecurity-1.0 和 appSecurity-2.0 功能部件支持保护服务器运行时环境和应用程序。支持下列方面：

- 基本用户注册表

- 轻量级目录访问协议 (LDAP) 用户注册表
- 基本授权
- Web 应用程序安全性
  - 基本认证登录
  - 表单登录 表单注销
  - 程序化 API：`getRemoteUser`、`getUserPrincipal`、`isUserInRole`、`authenticate`、`logout` 和 `login`。
- EJB 应用程序安全性
  - 可以在 ejb-jar.xml 文件中指定的所有安全性注解及所有安全性元素。
  - 程序化 API：`getCallerPrincipal`、`isCallerInRole` 和 `getCallerIdentity`。单独会话 Bean 不支持 `getCallerIdentity` API。
  - ibm-ejb-jar-ext.xml 文件中用于 run-as-mode CALLER_IDENTITY 和 SPECIFIED_IDENTITY（不支持 SYSTEM_IDENTITY）的 EJB 扩展设置。

当您将 `appSecurity-1.0` 或 `appSecurity-2.0` 功能部件添加到服务器时，还必须配置用户注册表（例如，基本用户注册表或者 LDAP 用户注册表）。

请参阅保护 *xigemaAS* 及应用程序（见第 1274 页） 和 *appSecurity-2.0* 功能部件限制（见第 1670 页）。

有关 `appSecurity-1.0` 功能部件配置信息，请参阅 *Application Security 1.0*（见第 932 页）。

有关 `appSecurity-2.0` 功能部件配置信息，请参阅 *Application Security 2.0*（见第 933 页）。

受管批处理

```
<feature>batchManagement-1.0</feature>
```

`batchManagement-1.0` 功能部件提供 REST 接口以用于远程作业提交和 `batchManager` 命令行客户机实用程序。

请参阅 *Java* 批处理和受管批处理概述（见第 1559 页）

有关 `batchManagement-1.0` 功能部件配置信息，请参阅 *Batch Management*（见第 937 页）。

Bluemix 实用程序

```
<feature>bluemixUtility-1.0</feature>
```

此功能部件可以更方便地配置对 IBM Bluemix 管理服务的访问。

有关 `bluemixUtility-1.0` 功能部件配置信息，请参阅 *bluemixUtility-1.0*。

事件日志记录

```
<feature>eventLogging-1.0</feature>
```

`eventLogging-1.0` 功能部件对包含多个事件（例如，JDBC 请求和 servlet 请求及其持续时间）的记录进行日志记录。

请参阅事件日志记录（见第 1654 页）。

有关 `eventLogging-1.0` 功能部件配置信息，请参阅事件日志记录。

ldapRegistry-3.0

```
<feature>ldapRegistry-3.0</feature>
```

ldapRegistry-3.0 功能部件支持 LDAP 用户注册表。ldapRegistry-3.0 V3.0 功能部件符合 LDAP V3 规范。ldapRegistry-3.0 功能部件未由 appSecurity-2.0 功能部件自动启用。通过使用此功能部件，您可以联合多个 LDAP 存储库。可以在 server.xml 文件中配置两个或两个以上的 LDAP 存储库，并且您可以从多个存储库中获取所有 LDAP 操作的合并结果。

有关 ldapRegistry-3.0 功能部件配置信息，请参阅 *LDAP User Registry*（见第 1002 页）。

本地 JMX 连接器

```
<feature>localConnector-1.0</feature>
```

localConnector-1.0 功能部件提供构建到 JVM 中的本地 JMX 连接器。该 JMX 连接器只能在同一主机上供以同一用户标识和同一 JDK 运行的用户使用。它允许 JMX 客户机（例如，jConsole 或其他使用 Attach API 的 JMX 客户机）进行本地访问。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）。

有关 localConnector-1.0 功能部件配置信息，请参阅*JMX Local Connector*（见第 960 页）。

监视

```
<feature>monitor-1.0</feature>
```

monitor-1.0 功能部件在 xigemaAS 上提供性能监控基础结构 (PMI) 支持。

请参阅监视 *xigemaAS* 服务器运行时环境（见第 1612 页）。

有关 monitor-1.0 功能部件配置信息，请参阅 *Performance Monitoring*（见第 1015 页）。

OAuth

```
<feature>oauth-2.0</feature>
```

oauth-2.0 功能部件支持使用 OAuth 2.0 协议来保护对资源的访问。

有关 oauth-2.0 功能部件配置信息，请参阅 *OAuth*（见第 1008 页）。

OpenID

```
<feature>openid-2.0</feature>
```

此功能部件允许用户对多个实体认证自身而不需要管理多个帐户或多组凭证。xigemaAS 支持 OpenID 2.0 并在 Web 单点登录中充当依赖方的角色。访问 Web 站点之类的各种实体通常需要与每个实体相关联的唯一帐户。OpenID 启用由 OpenID 提供者处理的一组凭证，以授予对支持 OpenID 的任意数目实体的访问权。请参阅*OpenID*（见第 1311 页）。

有关 openid-2.0 功能部件配置信息，请参阅*OpenID*（见第 1013 页）。

OpenID Connect 客户机

```
<feature>openidConnectClient-1.0</feature>
```

此功能部件允许 Web 应用程序集成 OpenID Connect Client 1.0 以认证用户而不是（或以及）所配置用户注册表。请参阅 *OpenID Connect*（见第 1312 页）。

有关 openidConnectClient-1.0 功能部件配置信息，请参阅 *OpenID Connect Client*（见第 1014 页）。

OpenID Connect 提供者

```
<feature>openidConnectServer-1.0</feature>
```

此功能部件允许 Web 应用程序集成 OpenID Connect Server 1.0 以认证用户而不是（或以及）所配置用户注册表。请参阅 *OpenID Connect*（见第 1312 页）。

有关 `openidConnectServer-1.0` 功能部件配置信息，请参阅 *OpenId Connect Provider*（见第 1015 页）。

OSGi 控制台

```
<feature>osgiConsole-1.0</feature>
```

此功能部件可让 OSGi 控制台帮助调试运行时环境。它可用来显示有关捆绑软件、软件包和服务的信息。为产品扩展开发您自己的功能部件时，此信息可能很有用。

请参阅使用 *OSGi* 控制台（见第 1129 页）。

有关 `osgiConsole-1.0` 功能部件配置信息，请参阅*OSGi Debug Console*（见第 1011 页）。

密码实用程序

```
<feature>passwordUtilities-1.0</feature>
```

此功能部件支持使用安全性插接点从应用程序获取 AuthData。

有关 `passwordUtilities-1.0` 功能部件配置信息，请参阅 *passwordUtilities-1.0*。

请求计时

```
<feature>requestTiming-1.0</feature>
```

`requestTiming-1.0` 提供有关运行缓慢或挂起的请求的警告和诊断信息。

请参阅检测运行缓慢和挂起的请求（见第 1656 页）。

有关 `requestTiming-1.0` 功能部件配置信息，请参阅请求计时。

REST 连接器 1.0

```
<feature>restConnector-1.0</feature>
```

`restConnector-1.0` 功能部件提供可以通过任何 JDK 以本地或远程方式使用的安全 JMX 连接器。它支持 JMX 客户机通过基于 REST 的连接器进行远程访问，而且需要 SSL 和基本用户安全性配置。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页），有关 REST 连接器的详细信息，请参阅配置与 *xigemaAS* 的安全 *JMX* 连接（见第 1179 页）。

有关 `restConnector-1.0` 功能部件配置信息，请参阅*JMX REST Connector*（见第 961 页）。

REST 连接器 2.0

```
<feature>restConnector-2.0</feature>
```

`restConnector-2.0` 功能部件提供了安全 JMX 连接器，可以通过任何 JDK 以本地或远程方式进行使用。它支持 JMX 客户机通过基于 REST 的连接器进行远程访问，而且需要 SSL 和基本用户安全性配置。此功能部件取代了 `restConnector-1.0` 功能部件，并且未包含 `jaxrs-1.1` 功能部件。

请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页），有关 REST 连接器的详细信息，请参阅配置与 *xigemaAS* 的安全 *JMX* 连接（见第 1179 页）。

对于 restConnector-2.0 功能部件配置信息，请参阅 *JMX REST Connector 2.0*（见第 962 页）。

SAML Web 浏览器 SSO

<feature>samlWeb-2.0</feature>

samlWeb-2.0 功能部件允许 Web 应用程序将用户认证委派给 SAML 身份提供者（而不是已配置的用户注册表）。

有关 samlWeb-2.0 功能部件配置信息，请参阅*SAML Web Single Sign-on V2.0*（见第 1019 页）。

服务器状态

<feature>serverStatus-1.0</feature>

serverStatus-1.0 功能部件使 xigemaAS 服务器能够将其状态自动发布到作业管理器（它将该服务器视为作业配置中的资源）。已知状态为**已启动**和**已停止**。

有关 serverStatus-1.0 功能部件配置信息，请参阅*Job Manager Integration*（见第 1002 页）。

会话持久性

<feature>sessionDatabase-1.0</feature>

sessionDatabase-1.0 功能部件在xigemaAS 上提供会话亲缘关系和故障转移支持。

请参阅为 *xigemaAS* 配置会话持久性（见第 1145 页）。

有关 sessionDatabase-1.0 功能部件配置信息，请参阅 *Database Session Persistence*（见第 945 页）。

基于 **Redis DB** 的会话持久性

<feature>redisSession-1.0</feature>

redisSession-1.0 功能部件主要用于 HTTP Session 集中式存储，以保证在服务器意外宕机后可以进行数据恢复。开启 redisSession-1.0 功能部件并正确配置 redis 数据源，部署在 redis 单机或集群环境中的所有 Web 应用程序都会将其会话数据存储到对应的 redis 数据库中，并且不同应用程序之间的会话数据互不影响。

请参阅为 *NoSQL* 数据库 *Redis* 配置会话持久性。

有关 redisSession-1.0 功能部件配置信息，请参阅*Redis Database Session Persistence*（见第 1017 页）。

SPNEGO

<feature>spnego-1.0</feature>

此功能部件允许用户登录 Microsoft™ 域控制器一次就可访问 xigemaAS 服务器上的受保护应用程序，而不用提示用户再次进行登录。

有关在 xigemaAS 服务器上配置 SPNEGO 的更多信息，请参阅在 *xigemaAS* 中配置 *SPNEGO* 认证。

有关 spnego-1.0 功能部件配置信息，请参阅*Simple and Protected GSSAPI Negotiation Mechanism*（见第 1022 页）。

安全套接字层 **(SSL)**

<feature>ssl-1.0</feature>

ssl-1.0 功能部件为安全套接字层 (SSL) 连接提供支持。要使用安全 HTTPS 侦听器，必须启用此功能部件。xigemaAS 提供哑元密钥库和哑元信任库。不启动安全 HTTPS 侦听器，除非启用 ssl-1.0 功能部件。如果功能部件不可用，那么会停止 HTTPS 侦听器。要指定 SSL 证书，请在 server.xml 文件中添加一个指针；请参阅保护与 *xigemaAS* 的通信（见第 1278 页）。要更改 HTTPS 端口，请在 server.xml 文件中设置 <httpEndpoint> 元素的 <httpsPort> 属性；请参阅设置应用服务器的引导属性（见第 1102 页）。

有关 ssl-1.0 功能部件配置信息，请参阅*Secure Socket Layer*（见第 1021 页）。

定时操作

```
<feature>timedOperations-1.0</feature>
```

此功能部件为下列过程启用支持：应用程序服务器中某些操作的运行速度慢于预期时对警告进行记录。

请参阅定时操作和 *JDBC* 调用（见第 1653 页）。

有关 timedOperations-1.0 功能部件配置信息，请参阅*Timed Operations*（见第 1022 页）。

动态高速缓存服务

```
<feature>webCache-1.0</feature>
```

此功能部件会对 Web 响应启用本地高速缓存。它包含高速缓存服务 (distributedMap) 功能部件，并对 Web 应用程序响应执行自动高速缓存以缩短响应时间并提高吞吐量。为定制响应高速缓存，可在应用程序中包括 cache-spec.xml 文件。可通过添加网络高速缓存提供者来分配高速缓存。

有关 webCache-1.0 功能部件配置信息，请参阅*Web Response Cache*（见第 1024 页）。

xigemaMQ 消息传递功能部件

```
<feature>wmqJmsClient-2.0</feature>
```

wmqJmsClient-2.0 功能部件允许应用程序通过 JMS 2.0 API 访问 xigemaMQ 上的消息队列。

请参阅 将 *JMS* 应用程序部署到 *xigemaAS* 以使用 *xigemaAS* 消息传递提供程序（见第 1548 页）。

有关 wmqJmsClient-2.0 功能部件配置信息，请参阅 *JMS 2.0 Client for xigemaMQ*（见第 956 页）。

Web Service 安全性

```
<feature>wsSecurity-1.1</feature>
```

wsSecurity-1.1 功能部件支持在消息级别保护 Web Service。要保护 Web Service 消息，必须启用此功能部件以及 appSecurity-2.0 和 jaxws-2.2 功能部件。会忽略 WSDL 文件中所定义的 Web Service 安全策略；除非启用了 wsSecurity-1.1 功能部件，否则不会强制实施此策略。

请参阅 *Web Service* 安全性（见第 1427 页）。

有关 wsSecurity-1.1 功能部件配置信息，请参阅 *Web Service Security*（见第 1025 页）。

Web Service 原子事务

```
<feature>wsAtomicTransaction-1.2</feature>
```

wsAtomicTransaction 是一个可互操作的事务协议。它使您可通过 Web Service 消息传送分布式事务，并可在异构事务基础结构之间以可互操作方式协调。

有关 xigemaAS 中的 `wsAtomicTransaction-1.2` 配置信息，请参阅 *xigemaAS 中的 Web Service 原子事务*（见第 1542 页）。

系统管理

管理中心

```
<feature>adminCenter-1.0</feature>
```

`adminCenter-1.0` 功能部件是一个基于 Web 的图形界面，用于在手机、平板电脑或计算机上从 Web 浏览器管理 xigemaAS 概要文件服务器和应用程序以及其他资源。

请参阅使用管理中心管理 *xigemaAS*和管理中心功能部件限制（见第 1671 页）。

有关 `adminCenter-1.0` 功能部件配置信息，请参阅 *Admin Center 1.0*（见第 930 页）。

Java™ EE 6 Web Profile

Bean 验证

```
<feature>beanValidation-1.0</feature>
```

`beanvalidation-1.0` 功能部件在应用程序的每一层提供对 JavaBeans™ 的验证。可以通过使用注解或 validation.xml 部署描述符，将验证应用到应用程序中 JavaBeans™ 的所有层。

请参阅 *Bean 验证功能部件限制*（见第 1671 页）。

有关 `beanValidation-1.0` 功能部件配置信息，请参阅 *Bean Validation 1.0*（见第 938 页）。

CDI

```
<feature>cdi-1.0</feature>
```

`cdi-1.0` 功能部件在 xigemaAS 上启用对上下文和依赖性注入 1.0 规范的支持。

请参阅在 *xigemaAS* 上管理上下文和依赖性注入应用程序（见第 1222 页）。

有关 `cdi-1.0` 功能部件配置信息，请参阅*Contexts and Dependency Injection 1.0*（见第 942 页）。

Enterprise JavaBeans™ (EJB) Lite 子集

```
<feature>ejbLite-3.1</feature>
```

`ejbLite-3.1` 功能部件为依照 EJB 3.1 规范的 EJB Lite 子集编写的 EJB 应用程序提供支持。

支持下列功能：

- 打包在 EAR 文件中的 EJB 模块。
- 打包在 WAR 文件中的 EJB。
- @Stateful 注解、@Stateless 注解、@Singleton 注解和 @EJB 注解。
- `javax.annotation.security` 注解。
- 将 JPA EntityManager、EntityManagerFactory 和 JDBC DataSource 注入所有类型的会话 Bean。
- ejb-jar.xml。
- EJB 拦截器。
- 非接口视图。
- Bean 管理的事务 (`UserTransaction`)。

请参阅 *ejbLite-3.1 功能部件限制*（见第 1672 页）。

有关 ejbLite-3.1 功能部件配置信息，请参阅 *Enterprise JavaBeans Lite 3.1*（见第 948 页）。

Java™ 数据库连接 (JDBC)

```
<feature>jdbc-4.0</feature>
```

您可以采用某一使用 Java™ 数据库连接 (JDBC) 和数据源的现有应用程序，然后将该应用程序部署到服务器。 jdbc-4.0 功能部件为访问数据库的应用程序提供支持。

请参阅*将现有 JDBC 应用程序部署到 xigemaAS*（见第 1501 页）。

有关 jdbc-4.0 功能部件配置信息，请参阅 *Java Database Connectivity 4.0*（见第 968 页）。

Java™ 命名和目录接口 (JNDI)

```
<feature>jndi-1.0</feature>
```

jndi-1.0 功能部件为 xigemaAS 的服务器配置中的单一 JNDI 条目定义提供支持。

有关 jndi-1.0 功能部件配置信息，请参阅 *Java Naming and Directory Interface*（见第 976 页）。

Java™ 持久性 API (JPA)

```
<feature>jpa-2.0</feature>
```

jpa-2.0 功能部件为应用程序（使用依照 JPA 2.0 规范编写的应用程序管理及容器管理的 JPA）提供支持。此支持基于 Apache OpenJPA，提供扩展来支持容器管理的编程模型。

扩展持久性上下文现在适用于有状态会话 Bean。

请参阅*将 JPA 应用程序部署到 xigemaAS*（见第 1509 页）。

有关 jpa-2.0 功能部件配置信息，请参阅 *Java Persistence API 2.0*（见第 977 页）。

JavaServer Faces (JSF)

```
<feature>jsf-2.0</feature>
```

jsf-2.0 功能部件为使用 JSF 框架的 Web 应用程序提供支持。此框架简化了用户界面的构造。如果包含 jsf-2.0 功能部件，那么也包含 jsp-2.2 功能部件，因为 JSF 框架是 JSP 框架的扩展。

有关 jsf-2.0 功能部件配置信息，请参阅 *JavaServer Faces 2.0*（见第 998 页）。

JavaServer Pages (JSP)

```
<feature>jsp-2.2</feature>
```

如果包含 jsf-2.0 功能部件，那么也包含 jsp-2.2 功能部件，因为 JSF 框架是 JSP 框架的扩展。如果包含 jsp-2.2 功能部件，那么也包含 servlet-3.0 功能部件。

请参阅 *jsp-2.2 功能部件限制*（见第 1673 页）。

有关 jsp-2.2 功能部件配置信息，请参阅 *JavaServer Pages 2.2*（见第 1000 页）。

Servlet 3.0

```
<feature>servlet-3.0</feature>
```

servlet-3.0 功能部件为依照 Java™ Servlet 3.0 规范编写的 HTTP Servlet 提供支持。

请参阅*保护 xigemaAS 及应用程序*（见第 1274 页）。

有关 servlet-3.0 功能部件配置信息，请参阅 *Java Servlets 3.0*（见第 988 页）。

Web 概要文件

```
<feature>webProfile-6.0</feature>
```

此功能部件提供支持 Java™ EE 6.0 Web 概要文件所需 xigemaAS 功能部件的便利组合。

有关 webProfile-6.0 功能部件配置信息，请参阅 *Java EE Web Profile 6.0*（见第 972 页）。

Java EE 6 技术

Java™ XML 绑定体系结构 (JAXB)

```
<feature>jaxb-2.2</feature>
```

jaxb-2.2 功能部件在 xigemaAS 上为“Java™ XML 绑定体系结构”(JAXB) 提供支持。

请参阅 和 *jaxb-2.2 功能部件限制*（见第 1672 页）。

有关 jaxb-2.2 功能部件配置信息，请参阅 *Java XML Bindings 2.2*（见第 994 页）。

Java™ API for RESTful Web Services (JAX-RS)

```
<feature>jaxrs-1.1</feature>
```

jaxrs-1.1 功能部件在 xigemaAS 上为 Java™ API for RESTful Web Services 提供支持。

- 对于使用 jaxrs-1.1 服务器功能部件的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1 功能部件。
- 对于使用 CDI 的 JAX-RS 应用程序，必须在 server.xml 文件中启用 cdi-1.0 功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

有关 jaxrs-1.1 功能部件配置信息，请参阅 *Java RESTful Services 1.1*（见第 984 页）。

Java™ API for XML-Based Web Services (JAX-WS)

```
<feature>jaxws-2.2</feature>
```

jaxws-2.2 功能部件在 xigemaAS 上为 Java™ API for XML-Based Web Services 提供支持。

- 对于支持 JAX-WS 编程模型的 Web 应用程序，必须在 server.xml 文件中启用 servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于支持 JAX-WS 编程模型的 EJB 应用程序，必须在 server.xml 文件中启用 ejbLite-3.1、servlet-3.0 和 jaxws-2.2 服务器功能部件。
- 对于使用全局处理程序服务的应用程序，必须在 server.xml 文件中启用 jaxrs-1.1 或 jaxws-2.2 功能部件。

请参阅 *jaxws-2.2 功能部件限制*（见第 1673 页）。

有关 jaxws-2.2 功能部件配置信息，请参阅 *Java Web Services 2.2*（见第 992 页）。

Java™ EE 连接器体系结构

```
<feature>jca-1.6</feature>
```

jca-1.6 功能部件提供了配置元素来定义连接工厂、受管对象和激活规范的实例，并将这些实例与安装的资源适配器关联。

有关 jca-1.6 功能部件配置信息，请参阅 *Java Connector Architecture 1.6*（见第 966 页）。

Java™ EE 连接器体系结构入站安全性

```
<feature>jcaInboundSecurity-1.0</feature>
```

jcaInboundSecurity-1.0 功能部件针对资源适配器启用安全性流程。

有关 jcaInboundSecurity-1.0 功能部件配置信息，请参阅*Java Connector Architecture 1.0 Security Inflow*（见第 965 页）。

Java™ 消息服务 1.1

```
<feature>jms-1.1</feature>
```

jms-1.1 功能部件支持将资源适配器配置为使用 Java™ 消息服务 API 访问消息传递系统。这还包括配置 JMS 连接工厂、队列、主题和激活规范。可使用符合 JCA 1.6 规范的任何 JMS 资源适配器。

有关 jms-1.1 功能部件配置信息，请参阅 *Java Message Service 1.1*（见第 974 页）。

消息驱动的 Bean

```
<feature>jmsMdb-3.1</feature>
```

jmsMdb-3.1 功能部件支持部署和配置 JMS 资源，在 xigemaAS 中运行 消息驱动 Bean (MDB) 时需要这些 JMS 资源。此功能部件使 MDB 能够与嵌入式 xigemaAS 消息传递或 xigemaMQ 进行交互。

有关 jmsMdb-3.1 功能部件配置信息，请参阅 *JMS Message-Driven Beans 3.1*（见第 959 页）。

消息驱动的 Bean 3.1

```
<feature>mdb-3.1</feature>
```

mdb-3.1 功能部件支持使用消息驱动的 Enterprise JavaBeans™。MDB 允许异步处理 Java™ EE 组件中的消息。

有关 mdb-3.1 功能部件配置信息，请参阅 *Message-Driven Beans 3.1*（见第 1005 页）。

嵌入式 xigemaAS消息传递功能部件

```
<feature>wasJmsClient-1.1</feature>
```

wasJmsClient-1.1 功能部件支持 JMS 资源配置（例如，连接工厂、激活规范以及队列和主题资源），它还提供了客户机库，消息传递应用程序需要这些客户机库才能连接至 xigemaAS 上的 JMS 服务器。

请参阅对*xigemaAS 启用 JMS 消息传递*和*将消息传递应用程序部署到 xigemaAS*（见第 1547 页）。

有关 wasJmsClient-1.1 功能部件配置信息，请参阅 *JMS 1.1 Client for Message Server*（见第 955 页）。

```
<feature>wasJmsSecurity-1.0</feature>
```

wasJmsSecurity-1.0 功能部件支持与消息传递引擎建立安全连接。如果已启用 wasJmsSecurity-1.0 功能部件，那么它会开始对尝试连接至消息传递引擎的用户进行认证和授权。针对在 server.xml 文件中定义的注册表来认证用户。如果用户想要访问主题或队列等目标，那么必须向用户授予必需的许可权。在 server.xml 文件中的 <messagingSecurity> 元素（这是 messagingEngine 元素的子元素）中定义了对于该目标的访问权。如果添加了 wasJmsSecurity-1.0 功能部件，但是 server.xm

l 文件中未定义 <messagingSecurity> 元素，那么用户将无法连接至消息传递引擎，也无法执行任何消息传递操作（例如，将消息发送至目标或者从目标接收消息）。

有关 wasJmsSecurity-1.0 功能部件配置信息，请参阅 *Message Server Security 1.0*（见第 1004 页）。

注：

- 配置用户注册表是 wasJmsSecurity-1.0 功能部件的先决条件。请确保在启用 wasJmsSecurity-1.0 功能部件之前已配置用户注册表。
- 启用 wasJmsSecurity-1.0 功能部件时，您还必须在 server.xml 文件中配置 <messagingSecurity> 元素（这是 <messagingEngine> 元素的子元素）。此配置使得授权用户能够访问消息传递目标。

<feature>wasJmsServer-1.0</feature>

wasJmsServer-1.0 功能部件使 JMS 消息传递引擎运行时能够进行初始化。消息传递运行时负责提供应用程序连接，管理目标（例如，主题或队列）的状态，以及处理服务质量、安全性和事务。此功能部件还支持来自远程消息传递应用程序的入站连接。远程消息传递应用程序可以通过基于 SSL 或者非 SSL 的 TCP/IP 来连接至 JMS 消息传递引擎。

要使用 SSL 进行连接，您必须启用 SSL 功能部件。

请参阅对 *xigemaAS* 启用安全 *JMS* 消息传递。

有关 wasJmsServer-1.0 功能部件配置信息，请参阅 *Message Server 1.0*（见第 1003 页）。

xigemaMQ 消息传递功能部件

<feature>wmqJmsClient-1.1</feature>

wmqJmsClient-1.1 功能部件使应用程序能够使用连接至 xigemaMQ 服务器的 JMS 消息传递。

请参阅将 *JMS* 应用程序部署到 *xigemaAS* 以使用 *xigemaAS* 消息传递提供程序（见第 1548 页）。

有关 wmqJmsClient-1.1 功能部件配置信息，请参阅 *JMS 1.1 Client for xigemaMQ*（见第 954 页）。

2.12.6 应用程序定义的数据源

可以通过注解或借助部署描述符，在应用程序中定义数据源，如 Java™ EE 规范所定义。

注：对于应用程序定义的数据源，建议提供 commonLibraryRef 类加载器属性。privateLibraryRef 属性无法用于 java:global 命名空间，并且不鼓励用于其他作用域。如果多个应用程序声明了同一 java:global 命名空间以指定数据源，那么这些应用程序的 server.xml 文件必须都对同一共享库指定 commonLibraryRef 属性。

在应用程序中定义数据源时，必须使 JDBC 驱动程序可供应用程序使用。这通过在 server.xml 中为应用程序配置共享库来完成。

例如：

```
<application id="myApp" name="myApp" location="myApp.war" type="war">
     <classloader commonLibraryRef="DB2Lib"/>
</application>
<library id="DB2Lib">
     <fileset dir="C:/DB2/java" includes="db2jcc4.jar db2jcc_license_cisuz.jar"/>
```

```
  </library>
```

然后，可以通过注解或借助部署描述符在应用程序中定义数据源。

- 如以下示例中所示使用注解：

```
@DataSourceDefinition(
        name          = "java:comp/env/jdbc/db2",
        className     = "com.ibm.db2.jcc.DB2DataSource",
        databaseName = "SAMPLEDB",
        serverName    = "localhost",
        portNumber    = 50000,
        properties    = { "driverType=4" },
        user          = "user1",
        password      = "pwd1"
        )

public class MyServlet extends HttpServlet {

    @Resource(lookup="java:comp/env/jdbc/db2")
    DataSource ds;
```

- 例如，如以下示例中所示在 web.xml 文件中使用部署描述符：

```
<data-source>
    <name>java:comp/env/jdbc/db2</name>
    <class-name>com.ibm.db2.jcc.DB2DataSource</class-name>
    <server-name>localhost</server-name>
    <port-number>50000</port-number>
    <database-name>SAMPLEDB</database-name>
    <user>user1</user>
    <password>pwd1</password>
    <property><name>driverType</name><value>4</value></property>
  </data-source>
```

通常，在 server.xml 文件中的 dataSource 或 connectionManager 上定义的属性也可以在应用程序定义的数据源上加以指定。此情况的两个例外是 connectionManagerRef 和 jdbcDriverRef，您不能指定它们，因为应用程序定义的数据源隐式定义了连接管理器和 JDBC 驱动程序。将应用程序定义的数据源用于两阶段落实时，可以指定 recoveryAuthDataRef 属性以选择用于事务恢复的认证数据。但是，一定要知道事务恢复只能在应用程序处于运行状态时才能进行。可以在应用程序定义的数据源中使用变量、经过编码的密码以及持续时间语法。

👉 注：持续时间语法不适用于注解中显式定义的属性，例如 loginTimeout 或 maxIdleTime。

下面举例说明了两个使用连接管理器属性的数据源、变量、经过编码的密码以及持续时间语法的数据源。

```
@DataSourceDefinitions(value = {
    @DataSourceDefinition(
        name          = "java:comp/env/jdbc/derby",
        className     = "org.apache.derby.jdbc.EmbeddedDataSource40",
        databaseName = "${shared.resource.dir}/data/SAMPLEDB",
        minPoolSize   = 1,
        maxPoolSize   = 10,
        maxIdleTime = 180,
        properties = { "agedTimeout=10m", "connectionTimeout=30s", "createDatabase=create" }
        ),
    @DataSourceDefinition(
        name          = "java:comp/env/jdbc/oracle",
        className     = "oracle.jdbc.pool.OracleDataSource",
        url           = "jdbc:oracle:thin:@//localhost:1521/SAMPLEDB",
        user          = "user1",
        password      = "{xor}Oz0vKDtt"
        )
})
```

2.12.7 目录位置和属性

在 xigemaAS 中，许多目录具有关联属性。配置服务器时，可以使用这些属性来指定文件位置。

创建后的服务器及配置信息位于 wlp/usr/servers 目录下。关于目录 usr 及其 servers 等子目录的位置、属性和描述信息，请参见目录位置和属性。

配置服务器时，可以使用与每个目录相关联的属性（如果有）来指定文件位置。要获取示例，请参阅在 *xigemaAS* 中部署应用程序（见第 1494 页）。

提示：要确保配置的可移植性，请使用合适的最特定属性，而且不要依赖于资源之间的关系。例如，在某些配置中，安装位置 `${wlp.install.dir}` 可能不是定制实例 `${wlp.user.dir}` 的父代。

位置属性的程序化访问

通过在 server.xml 文件中使用 jndiEntry 配置元素，可将位置属性绑定至所选名称下的 JNDI 命名空间，例如：

```
<jndiEntry jndiName="serverName" value="${wlp.server.name}"/>
```

通过 JNDI 查找，在服务器（应用程序、共享库或功能部件）中运行的任何代码都可访问这类条目：

```
Object serverName = new InitialContext().lookup("serverName");
```

有关如何在配置中使用 JNDI 条目的更多信息，请参阅从服务器配置文件将 *JNDI* 绑定用于常量（见第 1498 页）。

功能部件代码还可使用内核提供的系统编程接口 (SPI) 来解析这些属性的值，例如：

```
ServiceReference <WsLocationAdmin>locationAdminRef =
 bundleContext.getServiceReference(WsLocationAdmin.class);
WsLocationAdmin locationAdmin = bundleContext.getService(locationAdminRef);
String serverName = locationAdmin.resolveString("${wlp.server.name}");
```

2.12.8 对安全性的动态更改

本主题介绍了有关动态更改配置将如何影响 xigemaAS 概要文件的安全性的一些具体信息。

动态更改用户注册表

更改用户注册表会影响服务器配置及使用该服务器的客户机。在动态更改用户注册表之前（不重新启动服务器），请考虑下列事项：

- 如果您更改用户注册表类型或域名，那么所有 Web 客户机都必须清除其 Web 单点登录令牌。
- 如果更改用户注册表类型或域名，那么必须更新在授权绑定中给 accessId 指定的任何值。accessId 接受格式 `user:realmName/uniqueId` 或 `group:realmName/uniqueId`。accessId 中的 realmName 必须与配置用户注册表的 realmName 匹配。

2.12.9 xigemaAS 外部支持

xigemaAS 的外部函数和资源可以直接使用，而且在下一个发行版中可以继续沿用。当您应用服务或升级到未来发行版时，概要文件的内部或附带特征可能会改变。

我可以直接使用 xigemaAS 中的哪些内容并在下一个发行版中依赖这些内容?
下列资源可以直接使用，而且在下一个发行版中仍可以使用：

- ${wlp.install.dir}/dev 目录中的 JAR 文件内容定义的应用程序编程接口 (API) 和系统编程接口 (SPI)。
  - 应用程序类装入器可以看见由服务器配置中的功能部件所提供的 API。产品扩展功能部件可以看见服务器配置中的功能部件所提供的所有 API 和 SPI。
  - 针对 ${wlp.install.dir}/dev 目录中的 JAR 文件编译代码。在 ${wlp.install.dir}/dev 目录中提供 JAR 文件只是为了编译应用程序和功能部件，系统不支持将它们用于运行时。不要在应用程序、库或测试中使用这些 JAR 文件。
- 服务器配置，包括具有 *public* 或 *protected* 可视性的功能部件。可在 server.xml 文件和所包含文件中指定公共功能部件和配置元素；受保护功能部件可包括在您自己的功能部件中。
- ${wlp.install.dir}/bin 目录和子目录中的命令、脚本和归档。
- ${wlp.install.dir}/clients 目录和子目录中的客户机实用程序。

我应该避免依赖哪些内容?

不要构建对产品附带方面的依赖性，否则在应用服务或升级到未来发行版时会受到影响。您应该避免依赖的产品内部内容示例包括但不限于下列情况：

- 产品二进制 JAR 文件的名称，例如，${wlp.install.dir}/dev 目录中的那些 JAR 文件的名称。使用工具或 `javac -extdirs` 选项针对这些 JAR 文件来编译代码。

  如果您要使用 Apache Ant 编译代码，请使用通配符以避免与特定 JAR 版本产生依赖关系；例如：

  ```
  <fileset dir="${wlp.install.dir}/dev/api/spec" includes="com.ibm.ws.javaee.servlet.3.0_*.jar"/>
  ```

  请参阅 覆盖 *Java SDK* 中的类（见第 1670 页）。
- 直接使用 ${wlp.install.dir}/lib 目录中的产品二进制文件。可以直接调用的 JAR 文件仅位于 ${wlp.install.dir}/bin/tools 目录中。
- 由服务器在运行时输出的消息。消息的文本及插入会随服务和版本的升级而改变。只要实践上可能，产品将在特定操作点输出的消息标识方面保持一致，但这无法保证，因为底层实现可能会改变。
- 产品安装的布局，而不是 ${wlp.install.dir}/bin 和 ${wlp.install.dir}/dev 目录。
- ${wlp.install.dir}/templates 目录中的示例和模板文件。将服务应用到安装时可修改这些文件。
- 未显式提供为 API 的专用或第三方 Java™ 包。这些对于运行时的应用程序类加载器不可见。
- 不要对服务器输出的自动处理使用 console.log 文件。请改用 messages.log 文件访问和处理消息，这允许以易于处理的格式提供更多详细信息。

应用服务或升级时可修改哪些内容?

应用服务或升级时可修改下列目录及其子目录的内容。不要自行修改下列位置中的文件，否则它们可能会为产品维护或升级所覆盖：

- ${wlp.install.dir}/bin
- ${wlp.install.dir}/clients

- ${wlp.install.dir}/dev
- ${wlp.install.dir}/java
- ${wlp.install.dir}/lib
- ${wlp.install.dir}/templates

下列目录的内容不会加以修改。这些是您的文件，应用服务或升级将不会修改这些文件：

- ${wlp.install.dir}/etc（您可能在此目录中添加了 server.env 或 jvm.options 文件）。
- ${wlp.install.dir}/usr（用户配置和应用程序的缺省位置）。
- 通过 `WLP_USER_DIR` 环境变量指定的任何非缺省目录。

第三方 API 可能随时间的推移而更改，但未考虑向后兼容性问题。这些是 Java™ 包，视为开放式源代码社区所开发功能部件实现的一部分，且作为 xigemaAS 的一部分来提供。第三方 API 在缺省情况下对于应用程序不可见；具有显式允许第三方访问的类加载器配置的 Java™ EE 应用程序将可以访问应用程序类加载器上的那些包，并且 OSGi 应用程序必须显式导入那些包。决定使用第三方 API 之前，请考虑不兼容更改的影响。

2.12.10 如何应用连接池配置更新

如果在服务器处于运行状态时更改 connectionManager 元素的属性，那么会在不同时间，以不同的方式应用不同属性的更新。

要配置连接池，请在 server.xml 配置文件中指定 connectionManager 元素的属性。如果更改运行中服务器的这些属性，那么会在不同时间，以不同的方式应用更新，取决于更改的属性。下表针对 connectionManager 元素的每个属性，描述了如何在运行时应用配置更改。

表 64: 如何在运行时应用连接管理器配置更新

表的第 1 列列出了 connectionManager 元素的属性。第 2 列针对每个属性，描述了如何在运行时应用配置更新。

属性名称	如何应用配置更新
agedTimeout	更新会立即生效。
connectionTimeout	更新会立即生效。
maxIdleTime	更新会立即生效。
maxNumberOfMCsAllowableInThread	更新会立即生效。
maxPoolSize	更新会立即生效。
minPoolSize	更新会立即生效。
numConnectionsPerThreadLocal	更新会立即生效。
reapTime	更新会立即生效。
purgePolicy	更新会立即生效。

注：属性 agedTimeout 和 maxIdleTime 会立即加以更新。但是，它们不会全部被使用，除非 reapTime 属性的值大于零。

因为连接管理器的更新会立即生效，所以如果更改活动连接，那么可能会发生错误；包括可能提前结束连接的潜在风险。

2.12.11 如何恢复数据库事务

xigemaAS 概要文件事务管理器恢复不确定数据库事务时，它会使用唯一标识或 JNDI 名称来定位当前 dataSource 元素，然后确定要用于恢复的用户标识和密码。

通过在 server.xml 配置文件中指定 dataSource 元素的属性来配置数据源。可对数据源指定唯一标识或 jndiName 属性，如下所示：

```
<dataSource id="ds1" jndiName="jdbc/ds1"... />
```

对数据源参与的事务暂挂恢复时，不得更改 id 或 jndiName 属性的值。如果更改 dataSource 元素的任何其他属性，那么恢复时会保留这些更改。因此，（例如）可添加 recoveryAuthDataRef 属性以指定要用于恢复的数据库用户标识和密码。

用于恢复的数据库用户标识和密码是根据以下优先顺序确定的：

1. 如果 dataSource 元素定义了 recoveryAuthDataRef 属性，那么会使用来自 authData 元素的用户标识和密码。例如：

   ```
   <authData id="recoveryAuth" user="dbuser1" password="{xor}Oz0vKDtu"/>
   <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2"
               recoveryAuthDataRef="recoveryAuth" .../>
   ```

2. 如果使用容器管理的认证，那么会使用来自容器管理的认证别名的用户标识和密码。例如：

   - 在 Vsettan-web-bnd.xml 文件中，您具有下列代码：

     ```
     <resource-ref name="jdbc/ds1ref" binding-name="jdbc/ds1">
         <authentication-alias name="user1Auth"/>
      </resource-ref>
     ```

   - 在 server.xml 文件中，必须定义以下代码：

     ```
     <authData id="user1Auth" user="dbuser1" password="{xor}Oz0vKDtu"/>
     <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" .../>
     ```

3. 使用来自 dataSource 元素的用户标识和密码。例如：

   ```
   <dataSource id="ds1" jndiName="jdbc/ds1" jdbcDriverRef="DB2" ...>
        <properties.db2.jcc databaseName="testdb" user="dbuser1" password="{xor}Oz0vKDtu"/>
   </dataSource>
   ```

4. 如果上述条件都不满足，并且尝试在没有任何用户标识和密码的情况下进行恢复，那么其行为由 JDBC 驱动程序和数据库确定。

👉 注：如果事务恢复由应用程序定义的数据源（例如，@DataSourceDefinition 注解或部署描述符中的 <data-source> 元素）执行，那么必须确保进行恢复时关联应用程序正在运行。不能使用 server.xml 文件中的配置设置来恢复应用程序定义的数据源。

2.12.12 如何应用数据源配置更新

如果在服务器处于运行状态时更改 dataSource 元素的属性，那么会在不同时间，以不同的方式应用不同属性的更新。

要配置数据源，请在 server.xml 配置文件中指定 dataSource 元素的属性。如果更改运行中服务器的这些属性，那么会在不同时间，以不同的方式应用更新，取决于更改的属性。下表针对 dataSource 元素的每个属性，描述了如何在运行时应用配置更改。

表 65: 如何在运行时应用数据源配置更新

表的第 1 列列出了 dataSource 元素的属性。第 2 列针对每个属性，描述了如何在运行时应用配置更新。

属性名称	如何应用配置更新
beginTranForResultSetScrollingAPIs	更新会立即生效。
beginTranForVendorAPIs	更新会立即生效。
commitOrRollbackOnCleanup	更新会立即生效。
connectionManagerRef	将破坏所有连接及连接池。数据源随后由新连接管理器进行管理。
connectionSharing	更新分别随事务中的第一个连接句柄应用。
isolationLevel	更新随新连接请求应用；当前连接会保留其隔离级别。
jdbcDriverRef	将破坏所有连接及连接池。随后会使用新的 JDBC 驱动程序。
jndiName	将破坏所有连接及连接池。随后会使用新的 JNDI 名称。
propertiesRef	如果数据源是 Derby Embedded，那么在新属性生效之前，将破坏所有连接及连接池。对于其他 JDBC 驱动程序，新属性会随新连接请求生效。
queryTimeout	更新会立即生效。
recoveryAuthDataRef	用于事务恢复的认证数据。将破坏所有连接及连接池。随后会使用新的恢复认证数据。
statementCacheSize	下次使用时会调整语句高速缓存的大小。
supplementalJDBCTrace	将破坏所有连接及连接池。随后会使用新设置。
syncQueryTimeoutWithTransactionTimeout	更新会立即生效。
transactional	更新会应用到连接池中的新连接及未使用的现有连接。
type	将破坏所有连接及连接池。随后会使用新设置。

2.12.13 JAX-RS

您可使用 xigemaAS 概要文件中的 Java™ API for RESTful Web Services (JAX-RS)。

实现 **JAX-RS Web** 应用程序

可以使用 JAX-RS 来开发遵循表象化状态转变 (REST) 原理的服务。通过使用 JAX-RS，可以简化 RESTful 服务的开发。

要对 xigemaAS 进行配置更改，您必须将功能部件添加到 server.xml 文件，如下所示：

```
<feature>jaxrs-1.1</feature>
<feature>json-1.0</feature>
```

此外：

在 **JAX-RS** 应用程序请求和响应中使用 **XML** 内容

XML 是 RESTful 服务使用和产生的常用媒体格式。要对 XML 进行反序列化和序列化，您可以使用 Java™ XML 绑定体系结构 (JAXB) 注解对象来表示请求和响应。

在 **JAX-RS** 应用程序请求和响应中使用 **Atom** 内容

可以使用 Atom 联合格式 (Atom) 来格式化 Web 订阅源，该订阅源将在 Web 站点上传达专题节目信息的新闻和更新。通过在 JAX-RS 应用程序中使用 Atom 内容，可以利用 Web 内容联合功能，此功能提供了 RSS 所支持的用于添加新元数据和内容的分散化动态机制，但以有助于保护实现之间的核心互操作性的方式完成此任务。

使用定制实体格式

即使 JAX-RS 运行时环境随附若干个实体提供程序来处理从 Java™ 类型序列化或反序列化为这些类型，但并不支持所有可能的媒体类型。可以开发定制实体提供程序来处理将 Java™ 类型绑定至消息体的操作。

使用内容协商功能在 **JAX-RS** 应用程序中处理多种内容类型

RESTful 应用程序的其中一个优点是，能够返回资源的不同表示。借助表象化状态转变 (REST)，客户机和服务器可以交换同一介质类型的资源，也可以使用不同的介质类型。内容协商使客户机和服务器能够对用来交换数据的内容格式进行协商。

使用 **JAX-RS** 上下文对象来获取有关请求的更多信息

JAX-RS 向资源类和提供程序提供不同类型的上下文。可以使用上下文对象来访问请求信息，例如发现作为该请求的一部分发送的 HTTP 头。上下文对象还提供了一些便于使用的方法，用于对请求进行求值以及构建适当的响应。

使用处理程序来增强请求和响应处理

可以在 JAX-RS 应用程序的服务器端实现处理程序来增强请求和响应处理。

在 **JAX-RS** 应用程序请求和响应中使用多重部件内容

通过使用多重部件消息，服务器和客户机可以通过单一消息来传输多则消息。当客户机和服务器都需要发送多个请求，但希望节省为每个部件发送和接收整个 HTTP 请求和响应的成本时，多重部件消息非常有用。

使用 **WADL** 来生成服务文档

Web 应用程序描述语言 (WADL) 是基于 HTTP 的应用程序的描述语言。它当前是万维网联盟 (W3C) 成员提交项目。程序可以使用 WADL 以机器可处理的方法来提供有关服务的信息。例如，可以使用 XSLT 文档，通过使用一个定制 XSLT 和一个 XSLT 处理器来转换 WADL 文档。

在服务器应用程序内使用 **Apache Wink REST** 客户机以发出请求

可以将 Apache Wink REST 客户机用作可运行以将请求发送到 JAX-RS 应用程序的客户机。

👉 注：如果使用 `jaxrs-1.1` 功能部件，那么必须分别地指定 `servlet-3.0` 或 `jsp-2.2` 功能部件。

2.12.14 LDAP 证书映射方式

证书映射方式用于指定在 xigemaAS 是依照 EXACT_DN 还是 CERTIFICATE_FILTER 将 X.509 证书映射到 LDAP 目录。

EXACT_DN 意味着证书中的专有名称 (DN) 必须与 LDAP 服务器中的用户条目完全匹配（包括大小写和空格都完全匹配）。要对映射使用指定的证书过滤器，您可以使用 CERTIFICATE_FILTER。

证书过滤器

为 LDAP 过滤器指定过滤器证书映射属性。使用过滤器将客户机证书中的属性映射至 LDAP 注册表中的条目。

如果在运行时有多个 LDAP 条目与过滤器规范匹配，认证就会失败，这是因为这会导致模糊匹配。此过滤器的语法是：

```
LDAP attribute=${Client certificate attribute}
```

简单证书过滤器的示例为：`uid=${SubjectCN}`。

您还可以指定多个属性和值作为证书过滤器的一部分。过滤器规范的 LDAP 属性取决于将 LDAP 服务器配置为要使用的模式。客户机证书属性是客户机证书中的一个公共属性。客户机证书属性必须以美元符号 $ 和左花括号 { 开头，以右花括号 } 结尾。这些属性区分大小写。

支持下列 LDAP 属性：

- `uid`
- `initials`
- `sAMAccountName`
- `displayName`
- `distinguishedName`
- `displayName`
- `description`

支持下列客户机证书属性：

- `${SubjectCN}`
- `${SubjectDN}`
- `${IssuerCN}`
- `${IssuerDN}`
- `${SerialNumber}`

示例为启用了证书过滤方式的 LDAP 配置：

```
<ldapRegistry id="LDAP" realm="SampleLdapIDSRealm"
      host="myldap.vsettan.com" port="389" ignoreCase="true"
      baseDN="o=vsettan,c=cn"
      certificateMapMode="CERTIFICATE_FILTER"
      certificateFilter="uid=${SubjectCN}"
      userFilter="(&amp;(uid=%v)(objectclass=ePerson))"
      groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)
          (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
      userIdMap="*:uid"
      groupIdMap="*:cn"
      groupMemberIdMap="vsettan-allGroups:member;vsettan-
allGroups:uniqueMember;
```

```
            groupOfNames:member;groupOfUniqueNames:uniqueMember"
        ldapType="IBM Tivoli Directory Server" searchTimeout="8m" />
```

2.12.15 所提供 MBean 的列表

xigemaAS 提供了您可以用来处理和监视服务器的 MBean 以及相应管理接口的列表。

对于该列表中的每个 MBean 或 MXBean：

- 名称是用来唯一标识 MBean 或 MXBean 的 `javax.management.ObjectName` 值。如果存在 MBean 或 MXBean 的多个实例，那么 `ObjectName` 值可以包含通配符 (*)，在本主题的“注释”条目中对此进行了描述。
- 管理接口条目指定可以用来构造 MBean 或 MXBean 的代理对象（如访问 *MBean* 属性和操作的示例（见第 1186 页）中所述）的 Java™ 接口的名称。有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

`WebSphere:feature=channelfw,type=endpoint,name=*`

- 管理接口：`com.ibm.websphere.endpoint.EndPointInfoMBean`
- 注释：为系统中的每个端点都提供了一个实例，其中 * 是唯一的端点名称。

`WebSphere:feature=restConnector,type=FileService,name=FileService`

- 管理接口：`com.ibm.websphere.filetransfer.FileServiceMXBean`
- 注释：此 MXBean 可让您在 xigemaAS 所在的主机上执行各种与文件相关的操作。

  您可以在下列位置查找此 MXBean 的类和 API 文档：

  ```
  xigemaas_home/dev/api/vsettan/com.vsettan.xigema.appserver.api.restConnector_version.jar
  xigemaas_home/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.restConnector_version-
  javadoc.zip
  ```

  公开的操作包括能够查询所给定文件或目录的特定元数据（最近一次进行修改的日期和大小等等），还能够查询所给定目录的所有子文件（和相应的元数据）。还支持创建和扩展归档，这对于压缩 xigemaAS 日志文件或者在部署应用程序之前将应用程序解压很有用。

  此 MXBean 包含两个属性：读取列表和写入列表。它们表示在使用 xigemaAS 所提供的 `FileService` 或 FileTransfer 功能时用户可以读/写的位置的列表。通过 MXBean，只能读取这些属性，但是可以通过 server.xml 文件中的下列元素来配置或定制这些属性：

  ```
  <remoteFileAccess>
            <readDir>${server.output.dir}/logs</readDir>
     <readDir>${server.output.dir}/apps</readDir>
     <writeDir>${server.output.dir}/dropins</writeDir>
  </remoteFileAccess>
  ```

  如果未指定 `readDir` 元素，那么缺省值为下列各项的组合：`${wlp.install.dir}`、`${wlp.user.dir}` 和 `${server.output.dir}`。如果未指定 `writeDir` 元素，那么缺省值为空集合。

  server.xml 文件中必须包括 `restConnector-1.0` 功能部件，以便载入此 MXBean 以及支持其配置元素

  允许将 xigemaAS 定义的变量与所有采用用于表示文件路径的字符串的服务器端参数配合使用。在 *xigemaas_home*/README.TXT 文件中定义了这类变量。

WebSphere:feature=restConnector,type=FileTransfer,name=FileTransfer

- 管理接口：`com.ibm.websphere.filetransfer.FileTransferMBean`
- 注释：此 MBean 允许您在 xigemaAS 所在的主机上执行各种文件传输操作。

  您可以在下列位置查找此 MXBean 的类和 API 文档：

  ```
  xigemaas_home/dev/api/vsettan/com.vsettan.xigema.appserver.api.restConnector_version.jar
  xigemaas_home/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.restConnector_version-javadoc.zip
  ```

  此 MBean 在正运行其相应 xigemaAS 进程的同一 JVM 中的 `PlatformMBeanServer` 上已注册，但是只能使用 xigemaAS JMX REST 连接器来访问此 MBean。连接可以是本地连接或远程连接，但是必须使用 REST 连接器。

  已公开的操作包括能够下载、上载和删除文件。服务器上的每个读取和写入请求都绑定至可配置的读取和写入列表，可通过 `FileServiceMXBean` 来访问这些列表。如果通过 xigemaAS JMX REST 连接器连接了 JConsole，那么还可以从内置 Java™ JConsole 来完全访问和操作 `FileTransferMBean`。

  允许将 xigemaAS 定义的变量与所有采用用于表示文件路径的字符串的服务器端参数配合使用。在 *xigemaas_home*/README.TXT 文件中定义了这类变量。

WebSphere:name=com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean

- 管理接口：`com.ibm.websphere.config.mbeans.ServerXMLConfigurationMBean`
- 注释：ServerXMLConfigurationMBean 提供接口以检索服务器已知的所有服务器配置文件的文件路径。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean

- 管理接口：`com.ibm.websphere.runtime.update.RuntimeUpdateNotificationMBean`
- 注释：RuntimeUpdateNotificationMBean 为服务器运行时更新提供通知。附加至通知的用户数据对象为 java.util.Map。此 MBean 发出的运行时更新通知的通知类型为 com.ibm.websphere.runtime.update.notification。

WebSphere:name=com.ibm.ws.config.mbeans.FeatureListMBean

- 管理接口：`com.ibm.websphere.config.mbeans.FeatureListMBean`
- 注释：FeatureListMBean 展示单个方法以对运行时安装的所有功能部件生成 XML 报告。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.ws.config.serverSchemaGenerator

- 管理接口：`com.ibm.websphere.config.mbeans.ServerSchemaGenerator`

- 注释：ServerSchemaGenerator MBean 展示用于从已安装映像生成模式（最常用方法）或从当前运行时生成模式的方法。此 MBean 是内核中提供的，所以您不需要启用特殊功能部件。您可以在下列位置查找此 MBean 类和 API 文档：
  - *${wlp.install.dir}*/dev/api/vsettan/com.vsettan.xigema.appserver.api.config_*version*.jar
  - *${wlp.install.dir}*/dev/api/vsettan/javadoc/com.vsettan.xigema.appserver.api.config_*version*-javadoc.zip

WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig

- 管理接口：com.ibm.websphere.webcontainer.GeneratePluginConfigMBean
- 注释：请参阅为 配置 *Web* 服务器插件。

xigemaAS:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean

- 管理接口：com.ibm.websphere.filemonitor.FileNotificationMBean

xigemaAS:service=com.ibm.websphere.application.ApplicationMBean,name=*

- 管理接口：com.ibm.websphere.application.ApplicationMBean
- 注释：系统中的每个应用程序可以使用一个实例，其中 * 是唯一的应用程序名称。

xigemaAS:service=com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean,jndiName=*

- 管理接口：com.ibm.ws.jca.cm.mbean.ConnectionManagerMBean
- 注释：为系统中的每个连接管理器提供了一个实例，其中 * 是连接管理器的 jndiName。如果连接管理器没有 jndiName，并且连接管理器嵌套在 server.xml 中的 DataSource 元素下，那么系统可使用 DataSource 的 jndiName。

xigemaAS:type=JvmStats

- 管理接口：com.ibm.websphere.monitor.jmx.JvmMXBean
- 注释：启用 monitor-1.0 功能部件后可用。请参阅 *JVM* 监视（见第 1613 页）。

xigemaAS:type=ServletStats,name=*

- 管理接口：com.ibm.websphere.webcontainer.ServletStatsMXBean
- 注释：启用 monitor-1.0 功能部件后，所服务的每个 servlet 都可以使用一个实例，其中 * 的格式为 <AppName>.<ServletName>。请参阅 *Web* 应用程序监视（见第 1614 页）。

xigemaAS:type=ThreadPoolStats,name=Default Executor

- 管理接口：com.ibm.websphere.monitor.jmx.ThreadPoolMXBean
- 注释：启用 monitor-1.0 功能部件后可用。请参阅 *ThreadPool* 监视（见第 1615 页）。

WebSphere:feature=jaxws,type=WebServiceStats,service=*,port=*

- 管理接口：org.apache.cxf.management.counters.ResponseTimeCounterMBean
- 注释：启用 monitor-1.0 功能部件后可用。WebServiceStats 可为 Performance.Counter.Server 或者 Performance.Counter.client，其中 service=* 是服务端点的限定名，port=* 是服务端点的端口名。请参阅 *JAX-WS* 监视（见第 1615 页）。

注：这是一个动态模型 MBean。

WebSphere:feature=wasJmsServer, type=MessagingEngine,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.MessagingEngineMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件时可用。每个 xigemaAS 可有一个可用的消息传递引擎实例。name=* 是 MBean 的名称，其中 * 是消息传递引擎 MBean 的唯一名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Queue,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.QueueMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是队列 MBean 的名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Topic,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.TopicMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是主题 MBean 的名称。请参阅 *JMS* 消息传递（见第 1064 页）。

WebSphere:feature=wasJmsServer, type=Subscriber,name=*

- 管理接口：com.ibm.websphere.messaging.mbean.SubscriberMBean
- 注释：当启用了 wasJmsServer-1.0 功能部件并且消息传递引擎的 MBean 可用时，该 MBean 可用。name=* 是 MBean 的名称，其中 * 是订户 MBean 的名称。

  注：SubscriberMBean 是现有 TopicMBean 的订户。请参阅 *JMS* 消息传递（见第 1064 页）。

2.12.16 安全性快速概述

为了解 xigemaAS 中安全性的基本工作流程，详细介绍了一些常见安全性术语以及示例。

安全性关键术语

认证

认证确认用户的身份。最常用的认证形式是用户名和密码，例如通过基本认证或者用于 Web 应用程序的表单登录。认证用户时，请求源在运行时表示为 Subject 对象。

权限

授权决定用户是否能够访问系统中所给定的角色。Java™ EE 模型使用主体、角色和角色映射来确定是否允许访问。

角色

在 Java™ EE 应用程序中定义了角色。系统预定义了一些角色（例如，管理员角色）；另一些角色由应用程序开发者定义。在 Java™ EE 中，通常根据主体 在应用程序中承担的角色来授权或拒绝主体 访问某一角色。

主体

主体（Subject）是常规术语，也是 Java™ 对象：`javax.security.auth.Subject`。通常，术语主体表示系统中的活动实体，例如系统上的用户，甚至是系统进程自身。

安全性工作流示例

以下示例说明了在用户请求访问资源时如何应用安全性。例如，用户 `Bob` 想要访问 servlet `myWebApp`。请参阅 *xigemaAS 安全性入门*（见第 1275 页）中的代码样本。

下列条件必须成立，才能访问 servlet `myWebApp`：

1. `Bob` 必须能登录系统，因为 servlet 是受保护的。
2. `Bob` 必须在 `testing` 角色中，因为 servlet 已使用部署描述符中的 `auth-constraint` 元素进行限制。

如果 `Bob` 无法登录系统，或者 `Bob` 不在 `testing` 角色中，那么将拒绝对 servlet `myWebApp` 进行访问。

另一个用户 `Alice` 可登录系统，因为 `Alice` 是有效用户。但 `Alice` 不在 `testing` 角色中。`Alice` 登录时，会显示 HTTP 403 错误（拒绝/禁止访问）。

2.12.17 资源位置符号

通过使用表示符号位置的变量，可以增强 xigemaAS 用户配置的可移植性。使用这些变量有助于防止编写绝对路径的代码，否则，用户配置会很脆弱且可移植性不强。接收配置属性的功能部件代码可能必须处理包含此类变量的值。

xigemaAS概要文件的位置服务可用于将符号位置解析为物理资源。例如，符号位置 `${wlp.install.dir}/myFile` 可以映射到xigemaAS 概要文件的安装目录中的本地文件 myFile。大多数方法会返回打包了物理资源的 `WsResource` 对象，但您也可以使用 `resolveString` 方法将符号位置变换为可用于获取 `File` 对象的 `String`。

位置服务的名称是 `com.ibm.wsspi.kernel.service.location.WsLocationAdmin`，并且它是由xigemaAS内核提供，因此您不必在 server.xml 文件中指定功能部件以使该功能部件变为可用。

符号

`com.ibm.wsspi.kernel.service.location.WsLocationConstants` 类定义对目录位置进行引用的符号：

- `/`
- `server.config.dir`
- `server.output.dir`
- `server.workarea.dir`
- `shared.app.dir`
- `shared.config.dir`
- `shared.resource.dir`
- `wlp.install.dir`
- `wlp.server.name`
- `wlp.user.dir`
- `usr.extension.dir`

有关每个符号的意义，请参阅目录位置和属性（见第 1101 页）。

2.12.18 Java EE 7 编程模型支持

xigemaAS 遵循 Java™ Platform Enterprise Edition (Java EE) 7。Java EE 7 表和链接显示 xigemaAS 服务器对 Java EE 7 编程模型的支持程度。

Java EE 7 技术

表 66: Java EE 7 （概要文件类型支持）

Java™ EE 技术列表，针对以下各项分为若干部分：Web Service、Web 应用程序、企业应用程序、管理和安全性以及 Java™ SE 中与 Java™ EE 相关的规范。对于每种技术都有规范参考、任何相关 xigemaAS 功能部件以及xigemaAS 是否支持该技术的指示。

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java™ Platform Enterprise Edition 7 (Java EE 7)	*JSR 342*	*javaee-7.0* *javaeeClient-7.0*	✓
Java™ Platform Enterprise Edition 7 Web 概要文件	*JSR 342*	*webProfile-7.0*	✓
Web Service 技术			
Java™ API for RESTful Web Services (JAX-RS) 2.0	*JSR 339*	*jaxrs-2.0*	✓
实现企业 Web Service 1.4	*JSR 109*		✓
Java™ API for XML-Based Web Services (JAX-WS) 2.2	*JSR 224*	*jaxws-2.2*	✓
Web Service 互操作性组织 (WS-I) 基本概要文件	*WS-I 基本概要文件 1.2* *WS-I 基本概要文件 2.0*	*jaxws-2.2*	✓
Java™ XML 绑定体系结构 (JAXB) 2.2	*JSR 222*	*jaxb-2.2*	✓
用于 Java™ 平台的 Web Service 元数据	*JSR 181*		✓
Java™ API for XML-based RPC (JAX-RPC) 1.1（可选）	*JSR 101*		
Java™ API for WSDL (JWSDL)	*JSR 110*		✓
针对 XML 消息传递的 Java™ API 1.3（可选）	*JSR 67*		
带附件的 SOAP Java API (SAAJ) 1.3	*JSR 67*		✓
针对 XML 注册表的 Java™ API (JAXR) 1.0（可选）	*JSR 93*		
Web 应用程序技术			

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
针对 JSON 处理的 Java API (JSON-P) 1.0	*JSR 353*	*jsonp-1.0*	✓
Java™ Servlet 3.1	*JSR 340*	*servlet-3.1*	✓
JavaServer Faces (JSF) 2.2	*JSR 344*	*jsf-2.2*	✓
JavaServer Pages 2.3	*JSR 245*	*jsp-2.3*	✓
Expression Language (JSP/EL) 3.0	*JSR 341*	*el-3.0*	✓
JavaServer Pages 标准标记库 (JSTL) 1.2	*JSR 52*		✓
对其他语言的调试支持 1.0	*JSR 45*		✓
WebSocket 1.1	*JSR 356*	*websocket-1.1*	✓
WebSocket 1.0	*JSR 356*	*websocket-1.0*	✓
企业应用程序技术			
EE 并行实用程序 1.0	*JSR 236*	*concurrent-1.0*	✓
Java™ 上下文和依赖性注入 (Web Beans) 1.2	*JSR 346*	*cdi-1.2*	✓
Java™ 上下文和依赖性注入 (Web Beans) 1.1	*JSR 346*	*cdi-1.2*[7]	✓
Java™ 依赖性注入 1.0	*JSR 330*		✓
Bean 验证 1.1	*JSR 349*	*beanValidation-1.1*	✓
Enterprise JavaBeans™ (EJB) 3.2 完整版	*JSR 345*	*ejb-3.2*[8]	✓
Enterprise JavaBeans (EJB) 3.2 Lite	*JSR 345*	*ejbLite-3.2*	✓
Interceptors 1.2	*JSR 318*		✓
Java™ EE 连接器体系结构 (JCA) 1.7	*JSR 322*	*jca-1.7*	✓
Java™ 持久性 2.1	*JSR 338*	*jpa-2.1*	✓
Java™ 平台的常用注解 1.2	*JSR 250*		✓
Java™ 消息服务 (JMS) API 2.0	*JSR 343*	*jms-2.0*	✓

7 Java EE 7 定义 CDI 1.1。CDI 维护版 CDI 1.2。`cdi-1.2` 功能部件同时支持 CDI 1.1 和 CDI 1.2。

8 *ejb-3.2* 功能部件包含以下 EJB 子功能部件：*ejbLite-3.2*、*ejbHome-3.2*、*ejbPersistentTimer-3.2*、*ejbRemote-3.2* 和 *mdb-3.2*。

9 常用注解 1.2 添加了 `javax.annotation.Priority` 单一注解类型，上下文和依赖性注入 1.2 使用此类型。有关 CDI 1.2 的信息，请参阅 *Contexts and Dependency Injection 1.2*（见第 943 页）。

技术	规范参考	**xigemaAS** 功能部件	**xigemaAS**
Java™ 事务 API (JTA) 1.2	*JSR 907*		✓
JavaMail™ 1.5	*JSR 919*	*javaMail-1.5*	✓
Java 平台的批处理应用程序 1.0	*JSR 352*	*batch-1.0*	✓
管理和安全性技术			
Java™ 容器认证服务提供者接口 (JASPIC) 1.1	*JSR 196*	*jaspic-1.1*	✓
Java™ 容器授权合同 (JACC) 1.5	*JSR 115*	*jacc-1.5*	✓
Java™ EE 应用程序部署 1.2（可选）	*JSR 88*		
J2EE Management 1.1[10]	*JSR 77*	*j2eeManagement-1.1*	✓
Java SE 中与 **Java EE** 相关的规范			
针对 XML 处理的 Java™ API (JAXP) 1.4	*JSR 206*		✓
Java™ 数据库连接 (JDBC) 4.1	*JSR 221*	*jdbc-4.1*	✓
Java™ 管理扩展 (JMX) 2.0	*JSR 255*		✓
JavaBeans™ 激活框架 (JAF) 1.1	*JSR 925*		✓
针对 XML 的流式 API (StAX) 1.0	*JSR 173*		✓

2.12.19 安全性公共 API

xigemaAS 中的安全性公用 API 提供了一种扩展安全性基础结构的方法。

xigemaAS 包含可用来实现安全性功能的公用 API。主要类是 `WSSecurityHelper`、`WSSubject` 和 `RegistryHelper`。此外，还有一个新类 `WebSecurityHelper`。

下列部分描述了这些主要类。还有其他类，例如 `UserRegistry`、`WSCredential` 和其他异常类。

xigemaAS 支持的所有安全性公用 API 都在 Java™ API 文档中。每个 xigemaAS API 的 Java™ API 文档在 ${wlp.install.dir}/dev 目录的某个 javadoc 子目录下的单独 .zip 文件中提供。

`WSSecurityHelper`

此类仅包含方法 `isServerSecurityEnabled()` 和 `isGlobalSecurityEnabled()`。如果此外还启用了 `appSecurity-2.0`，那么这些调用返回 true。否则，这两个方法都将返回 false。

👉 注：

10 要调用管理 EJB API，服务器配置必须在功能部件管理器中同时具有 `j2eeManagement-1.1` 和 `ejbRemote-3.2` 功能部件。如果这两个功能部件都已包含在服务器配置中，那么您可通过 JNDI 名称查找来调用管理 EJB API。管理 EJB 绑定名称（JNDI 查找名称）为 `ejb/mejb/MEJB`。

- xigemaAS 中没有单元，所以 xigemaAS 在全局安全性和服务器安全性之间没有差别。因此，两个方法返回相同值。
- xigemaAS 中不支持方法 `revokeSSOCookies(javax.servlet.http.HttpServletRequest req,javax.servlet.http.HttpServletResponse res)`。
- `getLTPACookieFromSSOToken()` 方法已重命名为新的公用 API 类：`WebSecurityHelper`。

`WSSubject`

此类提供实用程序方法来查询和设置安全性线程上下文。

注：Java™ 2 安全性在 xigemaAS 概要文件中受支持，但缺省情况下未启用。所以，在缺省情况下，不会在 `WSSubject` 中执行 Java™ 2 安全性检查。

`RegistryHelper`

此类提供对 `UserRegistry` 对象及可信域信息的访问权。

`WebSecurityHelper`

此类包含重命名的 `getLTPACookieFromSSOToken()` 方法，该方法已从 `WSSecurityHelper` 中移出：

```
public static Cookie getLTPACookieFromSSOToken() throws Exception
```

安全性公用 API 代码示例

以下示例说明如何在 xigemaAS 中使用安全性公用 API 来执行程序化登录并操作 Subject 。

- 示例 *1:* 创建主体并将其用于授权
- 示例 *2:* 创建主体并使其成为线程上的当前 *Subject*
- 示例 *3:* 获取线程上当前主体的信息

示例 **1:** 创建主体并将其用于授权

此示例说明如何使用 `WSSecurityHelper`、`WSSubject` 和 `UserRegistry` 来执行程序化登录以创建 Java™ 主体，然后执行操作并将该主体用于任何需要的授权。

注：下列代码在执行进一步的安全处理之前使用 `WSSecurityHelper` 来检查是否启用了安全性。此检查由于 xigemaAS 的模块化性质而得到广泛使用：如果未启用安全性，那么不会载入安全性运行时。`WSSecurityHelper` 总是会载入，即使安全性并未启用也是如此。

```
import java.rmi.RemoteException;
import java.security.PrivilegedAction;

import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.CustomRegistryException;
import com.ibm.websphere.security.UserRegistry;
import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
import com.ibm.wsspi.security.registry.RegistryHelper;
public class myServlet {

 ...
 if (WSSecurityHelper.isServerSecurityEnabled()) {
```

```
  UserRegistry ur = null;
  try {
   ur = RegistryHelper.getUserRegistry(null);
   } catch (WSSecurityException e1) {
      // record some diagnostic info
   return;
  }
   String userid = "user1";
   String password = "user1password";
  try {
   if (ur.isValidUser(userid)) {
    // create a Subject, authenticating with
    // a userid and password
    CallbackHandler wscbh = new WSCallbackHandlerImpl(userid, password);
    LoginContext ctx;
    ctx = new LoginContext("WSLogin", wscbh);
    ctx.login();
    Subject subject = ctx.getSubject();
    // Perform an action using the Subject for
    // any required authorization
    WSSubject.doAs(subject, action);
   }
  } catch (CustomRegistryException e) {
   // record some diagnostic info
   return;
  } catch (RemoteException e) {
   // record some diagnostic info
   return;
   } catch (LoginException e) {
   // record some diagnostic info
   return;
  }
 }
 ...
  private final PrivilegedAction action = new PrivilegedAction() {
  @Override
   public Object run() {
         // do something useful here
   return null;
  }
 };

}
```

示例 **2:** 创建主体并使其成为线程上的当前 **Subject**

以下示例说明如何使用 `WSSecurityHelper` 和 `WSSubject` 来执行程序化登录以创建 Java™ 主体，使该主体成为线程上的当前主体，最后复原原始安全性线程上下文。

👉 注**:** 下列代码在执行进一步的安全处理之前使用 `WSSecurityHelper` 来检查是否启用了安全性。

```
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl;
...
if (WSSecurityHelper.isServerSecurityEnabled()) {
 CallbackHandler wscbh = new WSCallbackHandlerImpl("user1", "user1password");
  LoginContext ctx;
         try {
          // create a Subject, authenticating with
          // a userid and password
   ctx = new LoginContext("WSLogin", wscbh);
   ctx.login();
    Subject mySubject = ctx.getSubject();
    Subject oldSubject = null;
```

```
          try {
    // Save a ref to the current Subject on the thread
    oldSubject = WSSubject.getRunAsSubject();
    // Make mySubject the current Subject on the thread
    WSSubject.setRunAsSubject(mySubject);
    // Do something useful here. Any authorization
          // required will be performed using mySubject
    } catch (WSSecurityException e) {
     // record some diagnostic info
     return;
  } finally {
    // Put the original Subject back on the thread context
    if (oldSubject != null) {
            try {
     WSSubject.setRunAsSubject(oldSubject);
      } catch (WSSecurityException e) {
      // record some diagnostic info
                  }
            }
  }
  } catch (LoginException e) {
  // record some diagnostic info
  return;
 }
}
```

示例 **3:** 获取线程上当前主体的信息

以下示例说明如何使用 `WSSecurityHelper`、`WSSubject` 和 `WSCredential` 来获取有关线程上当前主体的信息。

👉 注**:** 下列代码在执行进一步的安全处理之前使用 `WSSecurityHelper` 来检查是否启用了安全性。

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Set;

import javax.security.auth.Subject;
import javax.security.auth.login.CredentialExpiredException;

import com.ibm.websphere.security.WSSecurityException;
import com.ibm.websphere.security.WSSecurityHelper;
import com.ibm.websphere.security.auth.CredentialDestroyedException;
import com.ibm.websphere.security.auth.WSSubject;
import com.ibm.websphere.security.cred.WSCredential;
...
 if (WSSecurityHelper.isServerSecurityEnabled()) {
  // Get the caller's subject
  Subject callerSubject;
 try {
    callerSubject = WSSubject.getCallerSubject();
  } catch (WSSecurityException e) {
     // record some diagnostic info
  return;
 }
  WSCredential wsCred = null;
  Set<WSCredential> wsCredentials =
    callerSubject.getPublicCredentials(WSCredential.class);
  Iterator<WSCredential> wsCredentialsIterator = wsCredentials.iterator();
  if (wsCredentialsIterator.hasNext()) {
   wsCred = wsCredentialsIterator.next();
          try {
    // Print out the groups
    ArrayList<String> groups = wsCred.getGroupIds();
    for (String group : groups) {
    System.out.println("Group name: " + group);
           }
          } catch (CredentialExpiredException e) {
      // record some diagnostic info
      return;
    } catch (CredentialDestroyedException e) {
```

```
        // record some diagnostic info
        return;
             }
           }
         }
}
```

2.12.20 SSL 配置属性

SSL 配置包含用来控制 xigemaAS 上服务器 SSL 传输层的行为的属性。此主题复述适用于 SSL 配置的所有设置。

SSL 功能部件

要在服务器上启用 SSL，必须在 server.xml 文件中包括 SSL 功能部件：

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>
```

SSL 缺省值

可以配置多项 SSL 配置。如果配置了多项 SSL 配置，那么必须在 server.xml 文件中使用 `sslDefault` 服务配置来指定缺省 SSL 配置。

表 67: sslDefault 元素的属性

此表描述 `sslDefault` 元素的属性。

属性	描述	缺省值
sslRef	`sslRef` 属性指定要用作缺省值的 SSL 配置的名称。	缺省 SSL 配置名称是 `defaultSSLConfig`。

在 server.xml 文件中，条目如下所示：

```
<sslDefault sslRef="mySSLSettings" />
```

SSL 配置

使用 SSL 配置属性来定制 SSL 环境以适合需求。可以在 server.xml 文件中的 `ssl` 服务配置元素上设置这些属性。

表 68: SSL 元素的属性

此表描述 `ssl` 元素的属性。

属性	描述	缺省值
id	id 属性对 SSL 配置对象指定唯一名称。	没有缺省值；必须指定唯一名称。

属性	描述	缺省值
keyStoreRef	keyStoreRef 属性指定用来定义 SSL 配置密钥库的密钥库服务对象。密钥库保存建立 SSL 连接所需的密钥。	没有缺省值；必须指定密钥库引用。
trustStoreRef	trustStoreRef 属性指定用来定义 SSL 配置信任库的密钥库服务对象。信任库保存对验证进行签名所需的证书。	`trustStoreRef` 是可选属性。如果缺少引用，那么会使用 `keyStoreRef` 所指定的密钥库。
clientAuthentication	clientAuthentication 属性确定是否需要 SSL 客户机认证。	缺省值为 false。
clientAuthenticationSupported	clientAuthenticationSupported 属性确定是否支持 SSL 客户机认证。客户机不必提供客户机证书。如果 clientAuthentication 属性设置为 true，那么会覆盖 clientAuthenticationSupported 属性的值。	缺省值为 false。
sslProtocol	sslProtocol 属性定义 SSL 握手协议。协议可能依赖于 SDK，因此如果您修改协议，请确保据以运行的 SDK 支持该值。	缺省值为 SSL_TLS。
securityLevel	securityLevel 属性确定 SSL 握手要使用的密码套件组。该属性具有下列其中一个值： • HIGH（128 位密码及更高） • MEDIUM（40 位密码） • WEAK（适用于所有密码，不加密） • CUSTOM（如果定制了密码套件组）。 如果使用特定的密码列表来设置 enabledCiphers 属性，那么系统会忽略此属性。	缺省值为 HIGH。
enabledCiphers	enabledCiphers 属性用于指定唯一的密码套件列表。使用空格来分隔列表中的每个密码套件。如果设置了 enabledCiphers 属性，那么会忽略 securityLevel 属性。	没有缺省值。
serverKeyAlias	serverKeyAlias 属性指定密钥库中要用作 SSL 配置密钥的密钥。只有在密钥库中具有多个密钥条目时，才需要此属性。如果密钥库具有多个密钥条目，且	没有缺省值。

属性	描述	缺省值
	此属性未指定密钥，那么 JSSE 会选取密钥。	
clientKeyAlias	clientKeyAlias 属性指定密钥库中要用作 SSL 配置密钥的密钥（在启用 clientAuthentication 的情况下）。只有在密钥库中包含多个密钥条目时，才需要此属性。	没有缺省值。

注：

- SSL 握手使用密钥管理器来确定要使用的证书别名。密钥管理器不是在 server.xml 文件中进行配置，而是从 SDK 的安全性属性 `ssl.KeyManagerFactory.algorithm` 中检索。
- SSL 握手使用信任管理器来作出信任决策。信任管理器不是在 server.xml 文件中进行配置，而是从 SDK 的安全性属性 `ssl.TrustManagerFactory.algorithm` 中检索。

下面举例说明了如何在 server.xml 文件中配置 `ssl` 元素：

```
<!--  Simple ssl configuration service object. This assumes there is a keystore object named -->
<!--  defaultKeyStore and a truststore object named defaultTrustStore in the server.xml file. --
>
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       trustStoreRef="defaultTrustStore" />


<!--  A ssl configuration service object that enabled clientAuthentication -->
<!--  and specifies the TLS protocol be used. -->
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       trustStoreRef="defaultTrustStore"
       clientAuthentication="true"
       sslProtocol="TLS" />

<!-- An SSL configuration service object that names the serverKeyAlias -->
<!-- to be used by the handshake. This assumes there is a certificate -->
<!-- called "default" in the keystore defined by keyStoreRef. -->
  <ssl id="myDefaultSSLConfig"
       keyStoreRef="defaultKeyStore"
       serverKeyAlias="default" />
```

密钥库配置

`keystore` 配置由装入密钥库时所需的属性组成。可以在 server.xml 文件中的密钥库服务配置上设置这些属性。

表 69: keystore 元素的属性

此表说明了 `keystore` 元素的属性。

属性	描述	缺省值
id	id 属性定义密钥库对象的唯一标识。	没有缺省值，必须指定唯一名称。

属性	描述	缺省值
location	location 属性指定密钥库文件名。值可以包含文件的绝对路径。如果未提供绝对路径，那么代码会在 ${server.config.dir}/resources/security 目录中查找该文件。	在 *SSL 最低配置*中，会假定该文件的位置为 ${server.config.dir}/resources/security/key.jks。
type	type 属性指定密钥库的类型。检查据以运行的 SDK 支持您所指定的密钥库类型。	缺省值为 jks。
password	password 属性指定用于装入密钥库文件的密码。可以采用明文或编码格式存储该密码。有关如何对密码进行编码的信息，请参阅 *security Utility encode* 选项。	必须提供。
provider	provider 属性指定要用于装入密钥库的提供程序。某些密钥库类型需要提供程序，而不是 SDK 缺省值。	缺省情况下，未指定任何提供程序。
fileBased	fileBased 属性指定密钥库是否基于文件。	缺省值为 true。
pollingRate	服务器检查密钥库文件更新的频率。	500ms。
updateTrigger	该方法用于触发服务器以重新装入密钥库文件。指定 polled 以允许服务器检查密钥库文件的更改，指定 mbean 以允许服务器等待 mbean 重新装入密钥库文件，或指定 disabled 以禁用文件监视。	disabled。

如果 updateTrigger 属性设置为 polled 或 mbean，那么服务器可重新装入密钥库文件。如果启用了 polled，那么服务器会根据 pollingRate 属性中设置的速率监视密钥库文件的更改。如果 updateTrigger 属性设置为 mbean，那么服务器从 `xigemaAS:service=com.ibm.ws.kernel.filemonitor.FileNotificationMBean` MBean 接收到通知时将重新装入密钥库文件。缺省情况下，文件监视被禁用。

下面举例说明了如何在 server.xml 文件中配置 `keystore` 元素：

```
<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be monitored every 5 seconds -->
<!-- for updates -->
   <keyStore id="defaultKeyStore"
           location="MyKeyStoreFile.jks"
           type="JKS" password="myPassword"
           pollingRate="5s"
           updateTrigger="polled" />
```

```
<!-- A keystore object called defaultKeyStore provides a location, -->
<!-- type, and password. The MyKeyStoreFile.jks file is assumed -->
<!-- to be located in ${server.config.dir}/resources/security -->
<!-- This keystore is configured to be reloaded when the server -->
<!-- recieves an mbean notification to do so -->
   <keyStore id="defaultKeyStore"
           location="MyKeyStoreFile.jks"
           type="JKS" password="myPassword"
           updateTrigger="mbean" />
```

完整 SSL 配置示例

下面举例说明了 server.xml 文件中的完整 SSL 配置。此示例具有下列 SSL 配置：

- defaultSSLSettings
- mySSLSettings

缺省情况下，SSL 配置会设置为 defaultSSLConfig。

```
<featureManager>
  <feature>ssl-1.0</feature>
</featureManager>


<!-- default SSL configuration is defaultSSLSettings ->
  <sslDefault sslRef="defaultSSLSettings" />
  <ssl id="defaultSSLSettings"
       keyStoreRef="defaultKeyStore"
       trustStoreRef="defaultTrustStore"
       clientAuthenticationSupported="true" />
  <keyStore id="defaultKeyStore"
            location="key.jks"
            type="JKS" password="defaultPWD" />
  <keyStore id="defaultTrustStore"
            location="trust.jks"
            type="JKS" password="defaultPWD" />

  <ssl id="mySSLSettings"
       keyStoreRef="myKeyStore"
       trustStoreRef="myTrustStore"
       clientAuthentication="true" />
  <keyStore id="LDAPKeyStore"
            location="${server.config.dir}/myKey.p12"
            type="PKCS12"
            password="{xor}CDo9Hgw=" />    <keyStore id="LDAPTrustStore"
            location="${server.config.dir}/myTrust.p12"
            type="PKCS12"
            password="{xor}CDo9Hgw=" />
```

2.12.21 访问 MBean 属性和操作的示例

您可使用 xigemaAS 访问 Java™ 管理扩展 (JMX) 管理 Bean (MBean) 的属性以及调用其操作。

在您获得 MBeanServer 实例（对于 xigemaAS 上运行的应用程序）或 MBeanServerConnection 实例（对于外部客户机）之后，可以访问 xigemaAS 所提供 MBean 的属性或者调用这些 MBean 的操作。请参阅在 *xigemaAS* 上使用 *JMX MBean*（见第 1182 页）。

以下代码示例假定变量 mbs 是 MBeanServer 或 MBeanServerConnection 实例。可以使用所提供的方法，以类似于 Java™ 反射的方式来访问属性和调用操作。此外，每个 MBean 的管理接口具有用于属性的 getter 方法及用于操作的方法。可以使用这些接口，通过其中一个 javax.managementJMX.newMBeanProxy 方法或者其中一个 javax.management.JMX.newMXBeanProxy 方法（用于 MXBean）来获取代理对象。管理接口的名称以“MXBean”结尾。对于管理接口的名称，请参阅所提供 *MBean* 的列表（见第 1182 页）。

示例 1：检查应用程序“myApp”的状态

```
import javax.management.ObjectName;
import javax.management.JMX;
import com.ibm.websphere.application.ApplicationMBean;
...

ObjectName myAppMBean = new ObjectName(
"xigemaAS:service=com.ibm.websphere.application.ApplicationMBean,name=myApp");
if (mbs.isRegistered(myAppMBean)) {
  String state = (String) mbs.getAttribute(myAppMBean, "State");
  // alternatively, obtain a proxy object
  ApplicationMBean app = JMX.newMBeanProxy(mbs, myAppMBean, ApplicationMBean.class);
  state = app.getState();
}
```

示例 2：从应用程序“myApp”获取 servlet“Example Servlet”的响应时间统计信息

```
import javax.management.ObjectName;
import javax.management.openmbean.CompositeData;
import javax.management.JMX;
import com.ibm.websphere.webcontainer.ServletStatsMXBean;

...

ObjectName servletMBean = new ObjectName("xigemaAS:type=ServletStats,name=myApp.Example
 Servlet");
if (mbs.isRegistered(servletMBean)) {
  CompositeData responseTimeDetails = (CompositeData) mbs.getAttribute(servletMBean,
 "ResponseTimeDetails");
  CompositeData responseTimeReading = (CompositeData) responseTimeDetails.get("reading");
  Double mean = (Double) responseTimeReading.get("mean");
  Double standardDeviation = (Double) responseTimeReading.get("standardDeviation");
  // alternatively, obtain a proxy object
  ServletStatsMXBean servletStats = JMX.newMXBeanProxy(mbs, servletMBean,
 ServletStatsMXBean.class);
  StatisticsMeter meter = servletStats.getResponseTimeDetails();
  StatisticsReading reading = meter.getReading();
  mean = reading.getMean();
  standardDeviation = reading.getStandardDeviation();
}
```

示例 3：创建 Web 服务器插件配置文件

```
import com.ibm.websphere.webcontainer.GeneratePluginConfigMBean;

...

ObjectName pluginMBean = new
 ObjectName("WebSphere:name=com.ibm.ws.jmx.mbeans.generatePluginConfig");
if (mbs.isRegistered(pluginMBean)) {
  mbs.invoke(pluginMBean, "generatePluginConfig", new Object[] {
    "installRoot", "serverName"}, new String[] {
    String.class.getName(), String.class.getName()
  });
  // alternatively, use a proxy object
  GeneratePluginConfigMBean plugin = JMX.newMBeanProxy(mbs, name,
 GeneratePluginConfigMBean.class);
  plugin.generatePluginConfig("installRoot", "serverName");
}
```

示例 4：查询 Web Service 端点的状态

```
import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;
```

```
...

// Init mbs as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-
Bus,
 type=Bus.Service.Endpoint,service=\"{http://vsettan.com.cn/}TestEndpointService\",
    port=\"TestEndpoint\",instance.id=1816106538");
MBeanInfo beanInfo = mbsc.getMBeanInfo(objName);

// Go through attributes to find the interested one
for (MBeanAttributeInfo attr : beanInfo.getAttributes()) {
  if (attr.getName().equals("State")) {
          String status = String.valueOf(mbs.getAttribute(objName, attr.getName()));
      break; }
}
```

示例 5：关闭 CXF 服务器总线

```
import javax.management.ObjectName;
import javax.management.MBeanServerConnection;
import javax.management.MBeanInfo;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanOperationInfo;

...

// Init mbsc as needed
MBeanServerConnection mbs;

// Get MBeanInfo for specific ObjectName
ObjectName objName = new ObjectName("WebSphere:feature=jaxws,bus.id=testCXFJMXSupport-Server-
Bus,
    type=Bus,instance.id=1618108530");
MBeanInfo beanInfo = mbsc.getMBeanInfo(objName);

// Go through operation to find the interested one and invoke
for (MBeanOperationInfo operation : beanInfo.getOperations()) {
  if (operation.getName().equals("shutdown")) {
    mbs.invoke(objName, operation.getName(), new Object[] { true }, new String[]
 { boolean.class.getName() });
      break; }
}
```

2.12.22 注册 MBean 的示例

应用程序可在 xigemaAS 上注册其自己的 MBean 实例。之后，该 MBean 实例可供其他应用程序或外部管理员使用。

任何应用程序都可以通过使用 `MBeanServer` 实例来注册 MBean。假定应用程序包含称为 `org.example.Example` 的类，这个类实现接口 `org.example.ExampleMBean` 来定义一些属性和操作。如以下示例中所示，应用程序可能只需实例化 `Example` 类，然后使用唯一的 `ObjectName` 来注册这个类。如果所选择的 `ObjectName` 已在使用中，那么会报告 `javax.management.InstanceAlreadyExistsException`。

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.Example;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
```

```
Object mbean = new Example();
ObjectName name = new ObjectName("org.example.MyApplication:name=Example");
mbs.registerMBean(mbean, name);
```

此外，应用程序还可注册 MBean 来扩展 `java.lang.ClassLoader` 并提供对任何数量的 MBean 实现类的访问权。然后，可以使用任何其他 JMX 客户机（本地或远程）来创建并注册应用程序所随附的 MBean。例如，假设应用程序具有可执行下列任务的 MBean 类 `org.example.ApplicationClassLoader`：

- 实现任何空接口 `org.example.ApplicationClassLoaderMBean`
- 扩展 `java.lang.Classloader`，并且
- 提供对 `org.example.Example` MBean 实现类的访问权

应用程序可以注册 `ApplicationClassLoader` 的实例，以使 `Example` MBean 可供其他 JMX 客户机使用，如下所示：

```
import java.lang.management.ManagementFactory;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.example.ApplicationClassLoader;

...

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
Object classLoader = new ApplicationClassLoader();
ObjectName name = new ObjectName("org.example.MyApplication:name=ClassLoader");
mbs.registerMBean(classLoader, name);
```

任何 JMX 客户机都可以创建 `Example` 实例。以下示例假定变量 `mbs` 是 `MBeanServer` 或 `MBeanServerConnection` 实例。请参阅在 *xigemaAS* 上使用 *JMX MBean*（见第 1182 页）。

```
import javax.management.ObjectName;

...

ObjectName loaderName = new ObjectName("org.example.MyApplication:name=ClassLoader");
ObjectName exampleName = new ObjectName("org.example.MyApplication:name=Example");
mbs.createMBean("org.example.Example", exampleName, loaderName);
```

必要时，可以使用其他形式的 `MBeanServer.createMBean` 方法，通过使用非缺省构造函数来创建 MBean。

有关管理接口的更多信息，请参阅 xigemaAS 的 Java™ API 文档。

2.12.23 xigemaAS故障诊断提示

xigemaAS 故障诊断提示。

为帮助您确定并解决问题，本产品包括统一的记录组件。请参阅日志记录和跟踪（见第 1649 页）。

如果您收到异常消息，那么可以在消息（见第 1675 页）中找到有关该消息的信息。

下列两个主题中提供了使用 xigemaAS 时适用的主要已知限制的详细信息：

- 运行时环境已知问题和限制（见第 1668 页）.

下面一组提示可帮助您对常见问题进行故障诊断。

- *JDK* 故障诊断
- 安全性故障诊断（见第 1746 页）

- *LDAP* 故障诊断（见第 1748 页）
- *SSL* 故障诊断（见第 1748 页）
- 对日志记录和跟踪进行故障诊断（见第 1749 页）
- 对 *JAX-WS* 进行故障诊断（见第 1749 页）
- 对 *WS-Security* 进行故障诊断（见第 1752 页）
- 诊断性能（见第 1754 页）
- 对 *SAML* 进行故障诊断

检查 JDK 是否处于支持的级别

如果您遇到尚不具有现成解释的问题，请检查您正在使用的 JDK 是否符合 Java™ V6 或更高版本的标准，以及是否处于当前服务级别。请参阅支持的最低 *Java* 级别（见第 1669 页）。

注：使用基于 Oracle 的 JVM（使用 Java™ V6）时，可能会发生死锁。如果是使用受影响的 JVM 或 JDK，那么下列设置可以帮助您防止发生死锁：

- 启用以下 VM 选项：`-XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass`
- 要设置 Equinox 框架选项以将类名锁定用于类装入，请设置以下 Equinox 配置选项：`-Dosgi.classloader.lock=classname`

启动 xigemaAS 服务器时，可以在 Java™ 属性文件（例如，jvm.options）中进行这些设置。

安全性故障诊断

此部分描述了一些常见安全问题以及您可以选择的解决方案。

SESN0008E: 认证为匿名的用户尝试访问以下用户所拥有的会话：**LdapRegistry/cn=steven,o=myCompany, c=CN**。

如果未认证的用户尝试访问由已认证的用户所创建的会话，那么会发生此错误。缺省情况下启用的会话安全性会阻止对会话进行未授权访问。仅创建会话的用户可以访问该会话。

如果使用 JSP（例如，login.jsp）进行表单登录，而且浏览器发送的 SSO 令牌到期，那么可能会发生此错误。因为 SSO 令牌到期，所以会提示用户使用针对表单登录而配置的 login.jsp 页面重新登录。缺省情况下，jsp 页面会尝试获取会话。此会话最初是由其令牌已到期的用户创建。但是，访问此会话时，令牌到期，用户未获认证以及未建立凭证，都会导致此错误。

要避免此错误，请重新启动那个启动新会话的浏览器，或者将 login.jsp 文件配置成缺省情况下不创建会话。如果选择更新 login.jsp 文件，请设置 `<%@ page session="false" %>`。

CWWKS9104A: 对 **{2}** 调用 **{1}** 时，授权用户 **{0}** 失败。未向用户授予访问任何必需角色的权限：**{3}**。

如果您对应用程序所需的角色没有权限，那么会发生此错误。请确保用户或用户所属的组已至少映射到错误消息中所提及的其中一个角色。ibm-application-bnd.xmi/xml 文件中所定义的用户到角色映射优先于 server.xml 文件中所定义的映射。检查两个资源以确保定义了正确的映射。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。

CWWKZ0013E: 无法启动名为 **{0}** 的两个应用程序，其后是异常安全行为和错误消息，例如 **CWWKS9104A**。

如果使用 `application` 元素在 server.xml 中和 dropins 文件夹中都指定您的应用程序，那么会发生此错误。因此，会尝试安装您的应用程序两次，这样 server.xml 文件中的安全性配置可能会生效，也可能不会生效。要修正此问题，必须将您的应用程序从 dropins 文件夹中移除，然后将其复制到另一个目录。如果

必须将您的应用程序保留在 dropins 文件夹中，那么必须通过在 server.xml 文件中使用下列代码来禁用应用程序监视：

```
<applicationMonitor dropinsEnabled="false" />
```

未认证的用户可以访问我的 **servlet** 或 **JSP**。

如果认证失败或者您的 servlet 或 JSP 未受保护，那么将创建主体为 UNAUTHENTICATED 的用户。如果您未定义任何安全性约束或者您将 EVERYONE 特殊主体集映射到您的应用程序所需的角色，那么未认证的用户可以访问您的 servlet 或 JSP。复审 ibm-application-bnd.xmi/xml 和 server.xml 文件中用户到角色的映射。采用下列其中一个选项：

- 如果您的 servlet 或 JSP 未受保护，请在应用程序的部署描述符中添加安全性约束。请参阅认证（见第 1035 页）。
- 如果不想任何未认证的用户访问您的应用程序，请将 EVERYONE 特殊主体集从该角色的映射中移除。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。
- 如果不能授权某一用户访问您的 servlet 或 JSP，请通过检查 ibm-application-bnd.xmi/xml 和 server.xml 文件来复审映射到该角色的用户。您可以将角色映射到用户、组或特殊主体集。如果角色已映射到 EVERYONE 特殊主体集，那么会向所有用户授予访问权。如果角色已映射到 ALL_AUTHENTICATED 特殊主体集，那么会向所有已认证的用户授予访问权。如果要进一步限制可以访问您的 servlet 或 JSP 的用户，请移除这些特殊主体集。最后，检查用户所属的组。虽然用户可能没有显式访问权，但组可能已映射到该角色。在这种情况下，将用户从获授权的组中移除或者将组从角色映射中移除。请参阅在 *xigemaAS* 上为应用程序配置授权（见第 1365 页）。

单点登录 **(SSO)** 不起作用。

如果 SSO 不起作用，请确保不同的 xigemaAS 概要文件服务器（使用 `webAppSecurity` 元素的相同 LTPA 密钥、密码以及 ssoCookieName 属性）具有相同的世界时间 (UTC) 且共享同一个用户注册表。此外，如果令牌到期或者如果在以不一致的方式更改用户注册表（例如，修改域或移除 cookie 所表示的用户）后从 Web 浏览器发送 cookie，那么 SSO 会失败并且用户会收到重新输入凭证信息的提示。请参阅针对 *xigemaAS* 使用 *LTPA cookie* 来定制 *SSO* 配置（见第 1316 页）。

调试安全性公用 **API**。

装入 `WSSecurityHelper` 和 `RegistryHelper`，即使未启用安全性亦如此，例如，即使未指定安全性功能部件 - `appSecurity-1.0`或`appSecurity-2.0`。如果未启用安全性，那么 `WSSecurityHelper.isServerSecurityEnabled()` 和 `WSSecurityHelper.isGlobalSecurityEnabled()` 方法都返回 false，而 `RegistryHelper.getUserRegistry()` 方法返回空值。

如果未启用安全性，那么可能不会装入其他安全性公用 API 类。如果尝试访问这些类，并调用其中一个类上的方法，那么您可能会收到 `java.lang.NoClassDefFoundError` 异常。

要避免收到 `java.lang.NoClassDefFoundError` 异常，必须先通过调用 `WSSecurityHelper.isServerSecurityEnabled()` 或 `WSSecurityHelper.isGlobalSecurityEnabled()` 类来测试以查看是否启用了安全性，然后只有在启用了安全性的情况下才调用其他安全性公用 API 类。有关此编码方法的示例，请参阅 安全性公共 *API*（见第 1060 页）。

无法认证使用 **Unicode** 字符的用户

如果要对名称中包含 Unicode 字符的用户进行认证，必须通过将以下 jvm 选项添加至服务器启动命令以将 xigemaAS 服务器使用的字符编码类型设置为 UTF-8：

```
-Dclient.encoding.override=UTF-8
```

还必须在登录页面中指定 `charset` 和 `pageEncoding`。以下是在登录 JSP 页面上指定这些参数的示例：

```
<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

java.lang.annotation.AnnotationFormatError: java.lang.IllegalArgumentException:Wrong type at constant pool index at sun.reflect.annotation.AnnotationParser.parseAnnotations(AnnotationParser.java:87)

OpenID Connect 或 OAuth 提供者使用数据库存储器向某些版本的 JDK 7 注册客户机时，可能发生此错误。

为避免此问题，请升级至 JDK V7.1。

LDAP 故障诊断

此部分描述了一些常见 LDAP 问题以及您可以选择的解决方案。

FFDC1015I: 已创建 **FFDC** 事件：**javax.naming.ServiceUnavailableException**：**myldapserver.mycompany.com**：**636**；套接字已关闭 **com.ibm.ws.security.registry.ldap.internal.LdapRegistry 298**

messages.log 中的此消息指示无法访问所配置的 LDAP 服务器。请检查 LDAP 服务器以验证它是否处于运行状态，以及验证它的 IP 地址是否可以从 xigemaAS 服务器进行访问。

javax.naming.CommunicationException：简单绑定失败：**myldapserver.mycompany.com**：**636 [**根异常是 **javax.net.ssl.SSLHandshakeException**：**com.ibm.jsse2.util.g**：**PKIX** 路径构建失败：**java.security.cert.CertPathBuilderException**：找不到所请求目标的有效认证路径**]**

如果在 LDAP 服务器上启用 SSL，而未将 LDAP 服务器的签署者复制到 `LDAPSSLSettings` 元素中所引用的信任库，那么与 LDAP 服务器的连接会失败，且产生 SSL 握手错误。请确保将 LDAP 服务器的签署者复制到信任库。

javax.naming.CommunicationException：**myldapserver.mycompany.com**：**389 [**根异常是 **java.net.BindException**：地址已在使用中：连接**]**

此消息可能出现在 FFDC 日志中，并指示本地服务器上的可用套接字已耗尽，这可能会导致无法连接至 LDAP 服务器。请确保套接字未使用，然后再试一次。

CWWKS1100A: 用户标识 **xxxxx** 认证失败。指定的用户标识或密码无效

即使消息中提及的用户在高负载的 LDAP 服务器上是有效的，服务器上也可能会发生此 FFDC 异常。借助 LDAP 配置，您可以使用开发者工具来添加 `reuseConnection=false` 属性或者将其禁用。要修正问题，请将此属性设置为缺省值 `true`。

SSL 故障诊断

此部分描述了一些常见 SSL 问题以及您可以选择的解决方案。

CWWKS9105E: 无法确定用于自动重定向的 **SSL** 端口。

如果您尝试访问的应用程序会重定向至 SSL 端口，但 SSL 端口不可用，那么会发生此错误。该端口可能会由于缺少 SSL 配置或 SSL 配置定义发生某些错误而不可用。请在 server.xml 文件中检查 SSL 配置，以确保它存在并且是正确的。请参阅保护与 *xigemaAS* 的通信（见第 1278 页）。

FFDC1015I: 在 **ffdc_12.04.18_16.09.42.0.log** 中创建了 **FFDC** 事件：**"java.lang.IllegalArgumentException**：配置未知、不完整：缺少标识字段 **com.ibm.ws.config.internal.cm.ManagedServiceFactoryTracker aSyncReadNupdate**。尝试读取配置并更新 **ManagedServiceFactory** 时抛出异常。异常 **= java.lang.IllegalArgumentException**：配置未知、不完整：缺少标识字段**"**

如果配置中存在不含标识字段的 `keystore` 元素，那么会发生此错误。如果使用最低 SSL 配置，请将标识字段设置为 defaultKeyStore。

如果使用启用了 **SSL** 的 **LDAP** 用户注册表，但未指定 **sslRef** 值，那么可能会收到异常。

例如，配置将 `sslEnabled` 设置为 true 但没有 `sslRef` 值，如以下示例中所示：

```
<ltldapRegistry id="ldap" realm="SampleLdapIDSRealm"
                                          host="ccwin12.austin.vsettan.com.c
n" port="636" ignoreCase="true"
baseDN="o=vsettan,c=cn"
bindDN="cn=root"
bindPassword="rootpwd"
ldapType="IBM Tivoli Directory Server"
idsFilters="ibm_dir_server"
sslEnabled="true"
searchTimeout="8m" />
```

必须输入 `sslRef` 值。如果使用的是类似如下的最低 SSL 配置：

```
<ltkeyStore id="defaultKeyStore" location="key.jks"
password="mypassword" />
```

`sslRef` 字段应该设置为 `defaultSSLConfig`。

如果已设定定制 SSL 配置，那么应该将该配置的名称放入 `sslRef` 字段。

对日志记录和跟踪进行故障诊断

此部分描述了日志记录和跟踪的一些常见问题。

java.util.logging -- Java 日志记录编程接口。

xigemaAS 概要文件不支持使用 logging.properties 文件配置 java.util.logging。使用 java 代码（例如，在部署的应用程序或用户界面中使用），来创建和添加 java.util.logging 处理程序、过滤器或格式化程序。

由于 xigemaAS 概要文件服务器根据日志记录配置元素的 `traceSpecification` 属性管理 java.util.logging 记录器级别，因此应避免使用 Logger.setLevel 方法。

对 JAX-WS 进行故障诊断

本节描述了一些常见 JAX-WS 问题以及您可以选择的解决方案。

在 **xigemaAS** 概要文件上部署 **Web Service** 应用程序时，发生 **org.apache.cxf.bus.extension.ExtensionException**。

如果应用程序中已嵌入 CXF 库和配置，并且您想要将该应用程序部署到 xigemaAS，那么必须确保通过从 server.xml 文件中移除 `jaxws-2.2` 服务器功能部件来禁用此功能部件。

将二进制文件从 **MTOM** 服务取回到 **MTOM** 客户机时返回空文件。

如果 MTOM 客户机已将二进制文件成功发送至 MTOM 服务，但是从 MTOM 服务取回这些二进制文件时返回空文件，就会发生这种情况。

作为一种变通方法，您可以创建新的 `DataHandler`，然后通过填充二进制文件的内容来初始化该 `DataHandler`。例如，使用 `FileDataStore` 对象从 I/O 进行读取，返回新创建的 `DataHandler`，然后将该 `DataHandler` 传递到其他 Web Service。

```
...
File f = new File("received_image");
  if (f.exists()) {
      connection.start();
 }

  FileOutputStream fos = new FileOutputStream(f);
        callerSubject.getPublicCredentials(WSCredential.class);

  FileDataSource fos_out = new FileDataSource(f);
  DataHandler img_out = new DataHandler(fos_out);
  return img_out;
...
```

将采用 **xsd:gMonth** 格式的 **XMLGregorianCalendar** 与 **Oracle JVM** 配合使用时发生了 **javax.xml.ws.soap.SOAPFaultException**：取消编组错误。

如果您使用 `XMLGregorianCalendar` 类将 `gMonth` 格式的常量作为客户端的 SOAP 请求的一部分来存储，并且在具有 Oracle JVM 的 xigemaAS 上启用了 `jaxws-2.2` 功能部件，就会发生此错误。造成此错误的根本原因是，对于 `xsd:gMonth` 类型，Oracle JVM 支持新的标准格式 *--MM*，而最新的 JAXB RI（参考实现）仅支持 *--MM--* 格式。

要解决此问题，可以针对客户端应用程序和服务器端应用程序将 Oracle JVM 更改为 IBM® JVM。

解析由 **wsdlLocation** 属性指定的 **WSDL** 文件时发生了 **java.io.FileNotFoundException**。

如果您像在 `wsdlLocation = "file:/WEB-INF/wsdl/a.wsdl"` 代码中一样指定 WSDL 文件，并且该 WSDL 文件打包在应用程序中，就会发生此错误。如果您想要引用打包在应用程序中的 WSDL 文件，那么必须对 `@WebService`、`@WebServiceProvider`、`@WebServiceClient` 或 `@WebServieRef` 注释中定义的 wsdlLocation 属性使用相对 URI。

`wsdlLocation` 属性的相对 URI 必须为下列其中一个值：

- `wsdlLocation = "/WEB-INF/wsdl/a.wsdl"`
- `wsdlLocation = "WEB-INF/wsdl/a.wsdl"`

messages.log 文件中出现大量**“Creating service”**消息。

当您调用所生成客户机存根类中的 `getPort` 或相关方法时，会发生这种情况。xigemaAS 配置为使用缺省日志记录配置，并且所有参考消息都写入 messages.log 文件。出现了大量消息，可能类似如下：

```
00000037 org.apache.cxf.service.factory.ReflectionServiceFactoryBean I Creating Service {http
://www.echo.org/}EchoService from WSDL:
    wsjar:file:/wlp/usr/servers/default/workarea/org.eclipse.osgi/bundles/100/data/cache/
```

```
    com.ibm.ws.app.manager_gen_15a42559-f979-4ee6-b488-9fa1fb212c96/.cache/Echo.war!/WEB-INF/
wsdl/Echo.wsdl
```

要禁止这些参考消息，可以在 server.xml 文件中更改日志记录配置，如下所示：

```
<logging traceSpecification="org.apache.cxf.*=warning=enabled"/>
```

SOAPFaultException：给定的 **SOAPAction** ***test*** 与客户机应用程序发送 **SOAP Action** 时进行的操作不匹配。

👉 注：*test* 是请求 HTTP 头中 `soapAction` 属性的值。

可以使 SOAP Web Service 中的每个操作与一个 SOAP Action 字符串相关联，例如在 WSDL 绑定中或者通过注释。Web Service 客户机将 SOAP Action 字符串作为头，和请求一起发送，以与 Web Service 提供程序的操作匹配。在以下场景中显示错误消息：客户机发送的 SOAP Action 不匹配操作的 SOAP Action。

要解决此问题，请确保对 Web Service 客户机和 Web Service 提供程序指定完全相同的 SOAP 操作。

使用 **Service.create()** 方法来创建服务时，发生 **javax.xml.ws.WebServiceException**。

如果在未提供 WSDL 文档的情况下使用 `Service.create()` 方法来创建服务，那么会发生此错误。如果您想要使用 `Service.create()` 来创建服务以及使用 `getPort()` 方法来获取端口号，那么必须使用 `addPort()` 方法来提供绑定信息。

下面提供了样本示例代码，说明如何使用 `addPort()` 方法：

```
QName serviceName = new QName("http://test.com/wssvt/acme/InsBusiness/", "MTOM11Service");
QName portName = new QName("http://test.com/wssvt/acme/InsBusiness/", "MTOM11Port");

// Setup the necessary JAX-WS artifacts
Service svc = Service.create(serviceName);
// Add the port in the service instance
svc.addPort(portName, SOAPBinding.SOAP11HTTP_MTOM_BINDING, mtom11URL);

port = svc.getPort(portName, MTOMInterface.class);
```

如果 **@WebResult** 注释中的 **name** 属性与 **WSDL** 文档中的 **name** 元素不同，那么会返回 **NULL** 响应。

如果使用 SEI 类来定义 `@WebResult` 注释的 name 属性，并且 SEI 类已提供如下所示的 WSDL 位置，那么会发生此问题：

```
@WebService(wsdlLocation="WEB-INF/wsdl/WebServiceIfc.wsdl")
public interface WebServiceRuntimeIfc {
        @WebMethod
        @WebResult(name="nononoreturn")
        public String echo (String parm);
}
```

但是所提供 WSDL 文档中的 XML 元素声明如下所示：

```
<xs:element name="echoResponse">
 <xs:complexType>
  <xs:sequence>
          <xs:element name="return" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

调用 `echo()` 方法时，Web Service 客户机将获取 NULL 响应。

要解决此问题，必须确保 `@WebResult` 注释的 name 属性与 WSDL 文档中 `name` 元素的值匹配。

JAX-WS 客户机应用程序无法从服务器端获取 **WSDL** 文档更改。

如果 Web Service 客户机和 Web Service 提供程序位于 xigemaAS 概要文件上的两个不同应用程序中，并且客户机需要动态地从 Web Service 提供程序检索 WSDL 文档，那么会发生此问题。这是因为首次访问 WSDL 文档时 xigemaAS 概要文件会高速缓存 WSDL 定义，此行为与完整概要文件中的行为不同。通过在 xigemaAS 概要文件中高速缓存 WSDL 定义，应用程序可以避免频繁地连接和解析 WSDL 文档。

要解决此问题，必须重新启动客户机应用程序以便能重新装入所更新的 WSDL 定义。

如果 **Web Service** 引用其自身，那么 **xigemaAS** 概要文件中的 **Web** 应用程序会挂起。

如果将注释 `@WebService` 用于 Web Service，使用注释 `@WebServiceRef` 引用其自身，那么服务端点发布会失败。例如，

```
@WebService
public class Test {
      @WebServiceRef
      private Test selfservice;

      public String echo(String msg) {
            return selfservice.invoke();
      }

      public String invoke() {
            return "hello world";
      }
}
```

要避免此问题，请使用服务存根（在以下示例中显示为 `TestService`）来获取服务实例，而不是使用 `@WebServiceRef` 引用自助服务。 例如，

```
@WebService
public class Test {

      private Test selfservice;

      public String echo(String msg) {
            TestService ts=new TestService();
            selfservice=ts.getTestPort();
            return selfservice.invoke();
      }

      public String invoke() {
            return "hello world";
      }
}
```

对 WS-Security 进行故障诊断

本节描述了您可以用来解决 WS-Security 问题的一些常见解决方案。

1. 请检查 server.xml 文件以确保正确配置了必需的 JAX-WS (`jaxws-2.2`) 功能部件和安全性 (`appSecurity-2.0`) 功能部件。

2. 要使用 WS-Security 来保护 Web Service 应用程序，JAX-WS 应用程序必须包含一个具有嵌入式 WS-Security 策略的 WSDL 文件。在 `wsdl:binding` 和/或 `wsdl:operation` 部分，必须存在对于该嵌入式 WS-Security 策略的 PolicyReference。
3. 如果您使用回调处理程序来检索密码以生成 UsernameToken，或者用于从密钥库文件中检索专用密钥的密码，那么必须将密码回调处理程序作为 xigemaAS 中的用户功能部件进行打包和部署。

启用 WS-Security 跟踪

如果根据错误消息中的信息无法确定发生此问题的原因，那么可以使用 xigemaAS 的跟踪和日志记录工具来收集对于 WS-Security 组件的跟踪。请参阅 *xigemaAS:* 跟踪和日志记录。

可以使用以下跟踪字符串来收集跟踪以调试 WS-Security 故障：

```
org.apache.ws.security.*=all=enabled:
org.apache.cxf.ws.security.*=all=enabled:
org.apache.cxf.ws.policy.*=all=enabled
org.apache.xml.security.*=all=enabled:
com.ibm.ws.wssecurity.*=all=enabled
```

本节描述了一些常见 WS-Security 问题以及您可以选择的解决方案。

org.apache.cxf.ws.policy.PolicyException：不能满足任何替代策略。

当 server.xml 文件中未定义 WS-Security 功能部件 `wsSecurity-1.1` 时，通常会发生此错误。为了避免发生此错误，您必须在 server.xml 文件中按如下所示定义 `wsSecurity-1.1` 功能部件：

```
<featureManager>
  <feature>usr:wsseccbh-1.0</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
  <feature>jaxws-2.2</feature>
  <feature>wsSecurity-1.1</feature>
</featureManager>
```

org.apache.cxf.ws.policy.PolicyException：不存在回调处理程序，没有密码可用。

如果 WS-Security 运行时无法检索在生成 UsernameToken 时所需要的密码，或者无法访问密钥库中的专用密钥，就会发生此错误。为了避免发生此错误，请检查以下配置：

1. 请确保密码回调处理程序功能部件 `wsseccbh-1.0` 在 server.xml 文件中定义为用户功能部件：

```
<featureManager>
  <feature>usr:wsseccbh-1.0</feature>
  <feature>servlet-3.0</feature>
  <feature>appSecurity-2.0</feature>
  <feature>jaxws-2.2</feature>
  <feature>wsSecurity-1.1</feature>
</featureManager>
```

2. 请确保在 server.xml 文件的 <wsSecurityClient> 或 <wsSecurityProvider> 元素中定义了回调处理程序属性 `ws-security.callback-handler`：

```
ws-security.callback-handler="<full class name of the callback handler>"

example:
ws-security.callback-handler="com.ibm.ws.wssecurity.example.cbh.CommonPasswordCallback"
```

3. 请确保密码回调处理程序返回了在 server.xml 文件中指定的用户名或密钥别名的正确密码。该密码必须为明文，并且未编码，也未加密。

org.apache.ws.security.WSSecurityException：签名未涵盖必需的元素 **(soap:Body)**。

如果 Web Service 提供程序应用程序期望指定请求消息中的 SOAP 主体，但是接收到的 SOAP 消息未指定 SOAP 主体，就会发生此错误。为了避免发生此错误，请确保为 Web Service 客户机所配置的正确 WS-Security 策略与 Web Service 提供程序的策略相匹配。

org.apache.ws.security.WSSecurityException：签名或解密无效。

如果 WS-Security 运行时未能验证签名或者将接收到的 SOAP 消息中已加密的消息部件解密，就会发生此错误。为了避免发生此错误，请验证您在 server.xml 文件的 <wsSecurityClient> 和 <wsSecurityProvider> 元素中配置 WS-Security 时使用了正确的密钥。如果 Web Service 客户机使用 Bob 的公用密钥对消息部件进行加密，那么 Web Service 提供程序必须对 Bob 的专用密钥具有访问权才能解密该消息。同样，如果 Web Service 客户机使用 Alice 的专用密钥为消息部件签名，那么该 Web Service 提供程序必须使用 Alice 的公用密钥来验证该签名。

org.apache.cxf.ws.policy.PolicyException：无法满足这些替代策略。

如果 WS-Security 运行时无法使用所配置的用来处理此请求的 WS-Security 策略来成功处理入局 SOAP 消息，就会发生此错误。为了避免发生此错误，请查看紧跟着此异常的消息，以确定导致此异常的特定 WS-Security 策略断言。例如，日志文件中可能包含下列消息：

```
Caused by: org.apache.cxf.ws.policy.PolicyException: These policy alternatives can not be satisfied:
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}AsymmetricBinding: The encryption algorithm doe
s not match the requirement
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}InitiatorEncryptionToken
{http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702}RecipientEncryptionToken
at org.apache.cxf.ws.policy.AssertionInfoMap.checkEffectivePolicy(AssertionInfoMap.java:167)
at org.apache.cxf.ws.policy.PolicyVerificationInInterceptor.handle(PolicyVerificationInInterceptor.java:10
1)
```

在这种情况下，因为 Web Service 客户机所使用的加密算法与 AlgorithmSuite 断言所指定的加密算法不匹配，所以发生了错误。在 Web Service 客户机和 Web Service 提供程序的 WS-Security 策略中指定的算法套件必须指定同一加密算法。

javax.xml.ws.soap.SOAPFaultException：消息已到期（**WSSecurityEngine**：时间戳记无效。消息的安全性语义已到期）。

如果消息时间戳记已到期，或者使用将来的时间戳记创建了消息，就会发生此错误。

诊断性能

本部分描述了一些常见性能问题以及您可以选择的解决方案。

应用程序监视器产生高 **CPU** 使用率。

如果应用程序监视器在 apps/ 目录下具有很多文件，且过度频繁地进行轮询，那么会发生此错误。

要避免此问题，可以更改一些设置。

1. 增大 `pollingRate` 属性的值。
2. 更新 server.xml 以将 `applicationMonitor` 元素包含在不是 `polled` 的 `updateTrigger` 中。

```
server.xml
<applicationMonitor updateTrigger="mbean" />
```

3. 减少 apps/ 目录下的文件数。

有关 `applicationMonitor` 元素的更多信息，请参阅控制动态更新（见第 1135 页）。

对 SAML 进行故障诊断

本节描述 SAML 的一些常见问题及您必须应用的解决方案。

java.lang.ArrayIndexOutOfBoundsException: 数组下标超出范围：**0**

在未移除身份提供者令牌 (IdP) 的情况下，尝试通过来路不明的服务提供者 (SP) 发起的请求进行多次登录时，可能发生此异常。

为避免此情况，请在相关的来路不明的 server.xml 文件中添加 `<httpSession invalidateOnUnauthorizedSessionRequestException="true" />`。

java.lang.IllegalStateException: CWWKS0010E: 尝试获取调用者主体

如果 SAML 用户先前直接登录本地用户注册表，那么可能发生此异常。为避免此问题，SAML 用户不得直接登录本地用户注册表。

启动服务器

在 xigemaAS 安装成功后，若启动服务器失败，您可以使用以下建议来排除操作中的常见故障。

1. 系统提示错误信息：

   `CWWKE0001I:已启动服务器 defaultServer。`

   `CWWKE0005E:无法启动运行时环境。`

   `LICVC0001E:产品未安装 License 文件。`

   出现此则错误信息，说明 License.dat 缺失，请注册许可证。
2. 系统提示错误信息：

   `CWWKE0001I:已启动服务器 defaultServer。`

   `CWWKE0005E:无法启动运行时环境。`

   `LICVC0002E: License 文件无效。`

   出现此则错误信息，说明 License.dat 无效，请联系 Vsettan 支持中心或您的销售代表。
3. 系统提示错误信息：

   `CWWKE0001I:已启动服务器 defaultServer。`

   `[err] The trial period for the copy of xigemaAS has expired.`

   `CWWKE0005E:无法启动运行时环境。`

   `LICVC0003E: License 已过期`

   出现此则错误信息，说明试用版已过期，请安装正式版 xigemaAS。

日志记录和跟踪

本产品包括统一的日志记录组件，用于处理产品写入的消息以及提供首次故障数据捕获 (FFDC) 服务。

此外，日志记录组件将捕获写入到 System.out、System.err、java.util.logging 和 OSGi 记录的消息。日志记录组件将统一处理这些消息以及该产品写入的其他消息。日志记录组件无法捕获直接由 JVM 进程写入的消息，例如 `-verbose:gc` 输出。

服务器有三个主要日志文件：

1. console.log - 包含来自 JVM 进程的重定向标准输出和标准错误。此控制台输出适用于直接人工使用。如果您使用缺省 consoleLogLevel 配置，那么控制台输出包含主要事件和错误。如果您使用缺省 copySystemStreams 配置，那么控制台输出还包含写入到 System.out 和 System.err 流的任何消息。控制台输出始终包含直接由 JVM 进程写入的消息，例如，`-verbose:gc` 输出。仅当使用 `server start` 命令时，才会创建此文件，此位置只能使用 `LOG_DIR` 环境变量改变。有关更多信息，请参阅从命令行管理 *xigemaAS*（见第 1116 页）。
2. messages.log - 包含除了由日志记录组件写入或者捕获的跟踪消息之外的所有消息。写入到此文件的所有消息包含其他信息，例如，消息时间戳记以及写入该消息的线程的标识。此文件不包含由 JVM 进程直接写入的消息。
3. trace.log - 包含由该产品写入或捕获的所有消息。仅当您启用其他跟踪时，才会创建此文件。此文件不包含由 JVM 进程直接写入的消息。

日志记录配置

可以通过服务器配置来控制日志记录组件。日志记录配置的主要位置是位于 server.xml 文件中。有时，您可能需要配置跟踪以诊断在处理 server.xml 文件之前发生的问题。在这种情况下，可以在 bootstrap.properties 文件中指定等价的配置属性。如果在 bootstrap.properties 文件和 server.xml 文件中都指定配置属性，那么会使用 bootstrap.properties 文件中的值，直到处理了 server.xml 文件为止。然后，使用 server.xml 文件中的值。应避免同时在 bootstrap.properties 和 server.xml 文件中为相同配置属性指定不同值。

表 70: xigemaAS 概要文件的日志记录属性

第 1 列包含可以在 server.xml 文件中设置的属性。第 2 列包含可以在 bootstrap.properties 文件中使用的等价属性。第 3 列提供每个日志记录属性的描述。

属性	等价属性	描述
logDirectory	`com.ibm.ws.logging.log.directory`	此属性对所有日志文件（排除 console.log 文件，但包括 FFDC）设置目录。缺省情况下，logDirectory 设置为 `LOG_DIR` 环境变量。缺省 `LOG_DIR` 环境变量路径为 *WLP_OUTPUT_DIR*/serverName/logs。 👉 注：使用 LOG_DIR 环境变量或 com.ibm.ws.logging.log.directory 属性而不是 logDirectory 属性来配置您要将所有消息写至其中的目录。否则，缺省情况下一开始会将一些消息写至 logs 目录，然后将余下消息写至基于您的配置的指定目录。服务器运行时，可使用 logDirectory 属性将日志动态更新至指定目录。
maxFileSize	`com.ibm.ws.logging.max.file.size`	滚动之前日志文件可以达到的最大大小（以 MB 计）。xigemaAS 概要文件运行时仅执行基于大小的日志滚动。要禁用此属性，请将值设置为 0。最大文件大小是适当的。缺省情况下，此值为 20。

属性	等价属性	描述
		👉 注：maxFileSize 不适用于 console.log 文件。
maxFiles	`com.ibm.ws.logging.max.files`	如果强制最大文件大小存在，那么此设置用于确定每个日志文件保留的字节数。此设置也适用于对任何特定日期发生的异常进行汇总的异常日志数。因此，如果此数目是 10，那么 ffdc/ 目录中可具有 10 个消息日志、10 个跟踪日志以及 10 项异常摘要。缺省情况下，值为 2。 👉 注：maxFiles 不适用于 console.log 文件。
consoleLogLevel	`com.ibm.ws.logging.console.log.level`	此过滤器控制进入 console.log 文件的消息的详细程度。有效值为 INFO、AUDIT、WARNING、ERROR 和 OFF。缺省情况下，级别为 AUDIT。
copySystemStreams	`com.ibm.ws.logging.copy.system.streams`	如果为 true，那么会将写入 System.out 和 System.err 流的消息复制到 console.log。如果为 false，那么这些消息会写入到配置的日志，例如，messages.log 或 trace.log，但不复制到 console.log。缺省值为 true。
messageFileName	`com.ibm.ws.logging.message.file.name`	消息日志的缺省名称为 messages.log。此文件总是存在，而且包含 INFO 和其他 (AUDIT、WARNING、ERROR 和 FAILURE）消息以及 System.out 和 System.err。此日志还包含时间戳记和发放线程标识。如果日志文件滚动，那么之前日志文件的名称的格式为 messages_*timestamp*.log
suppressSensitiveTrace		服务器跟踪可在跟踪非类型化数据（例如，通过网络连接接收到的字节数）时公开敏感数据。此属性如果设置为 true，那么可防止潜在的敏感信息暴露在日志和跟踪文件中。缺省值为 false。
traceFileName	`com.ibm.ws.logging.trace.file.name`	只有在启用附加或详细跟踪的情况下，才创建 trace.log 文件。stdout 识别为特殊值，并促使将跟踪定向到原始标准输出流。
traceSpecification	`com.ibm.ws.logging`	跟踪字符串用于选择性地启用跟踪。缺省值为 *=info。

属性	等价属性	描述
	.trace.specification	
traceFormat	com.ibm.ws.logging.trace.format	此属性控制跟踪日志的格式。xigemaAS 概要文件的缺省格式为 ENHANCED。
hideMessage	com.ibm.ws.logging.hideMessage	可使用 hideMessage 属性以配置您要在 console.log 和 message.log 文件中隐藏的消息。如果消息配置为隐藏，那么它们会重定向至 trace.log 文件。

要在服务器配置文件中设置日志记录属性，可以在开发者工具的**服务器配置**视图中选择**记录和跟踪**，也可以在服务器配置文件中添加日志记录元素，如下所示：

```
<logging traceSpecification="*=audit:com.myco.mypackage.*=finest"/>
```

日志详细信息级别规范的格式为：

```
<component> = <level>
```

其中 <*component*> 是要为它设置日志详细信息级别的组件，而 <*level*> 是某个有效的记录器级别（关闭(off)、致命(fatal)、严重(severe)、警告(warning)、审计(audit)、信息(info)、配置(config)、详细信息(detail)、精细(fine)、更精细(finer)、最精细(finest)、全部(all)）。用冒号 (:) 分隔多个日志详细信息级别规范。

注意：对于给定记录器，此级别由适用于该记录器的最具体跟踪规范确定。

组件对应 Java™ 包和类，或对应一组 Java™ 包。使用星号 (*) 作为通配符以指示某些组件，这些组件具有所指定组件包含的所有包中所有的类。例如：

*****

指定在应用程序服务器中运行的所有可跟踪代码，包括产品系统代码和客户代码。

com.ibm.ws.*

指定所有包名以 com.ibm.ws 开头的类。

com.ibm.ws.classloader.JarClassLoader

仅指定 JarClassLoader 类。

表 71: 有效日志记录级别

下表列示了 xigemaAS V9 应用程序服务器的有效级别。

V9 及更高版本日志记录级别	内容/重要性
关闭	日志记录已关闭。
致命	任务无法继续，并且组件、应用程序和服务器无法工作。

V9 及更高版本日志记录级别	内容/重要性
严重	任务无法继续，但是组件、应用程序和服务器仍可工作。此级别也可以表明即将发生的不可恢复的错误。
警告	可能发生错误或即将发生错误。此级别也可以表明正在向故障发展（例如，潜在的资源泄漏）。
审计	影响服务器状态或资源的重大事件
信息	概述总体任务进度的一般信息
配置	配置更改或状态
详细信息	详细说明子任务进度的一般信息
精细	跟踪信息 - 常规跟踪 + 方法入口、出口和返回值
更精细	跟踪信息 - 详细跟踪
最精细	跟踪信息 - 更详细的跟踪信息，包含调试问题所需的所有详细信息
全部	记录所有事件。如果创建了定制级别，那么全部将包括那些级别，并且能提供比“最精细”更详细的跟踪。

console.log 文件的管理级别与其他日志文件不同。可更改的唯一属性是 consoleLogLevel。如果您担心 console.log 文件的大小不断增长，那么可禁用 console.log 文件并改用消息日志文件。系统将以另一格式编写的相同数据写至消息日志文件，您可使用 maxFileSize 和 maxFiles 属性来控制消息日志文件的大小和数目。例如，以下 bootstrap.properties 文件会导致产生空的 console.log 文件，以及最多 3 个不断增大的 1 MB loggingMessages.log 文件。但是，来自底层 JVM 的消息仍可写入 console.log。由于 bootstrap.properties 文件中的设置在创建消息日志文件之前生效，因此消息日志文件最初创建为 loggingMessages.log 而不是缺省的 messages.log。

```
com.ibm.ws.logging.max.file.size=1
com.ibm.ws.logging.max.files=3
com.ibm.ws.logging.console.log.level=OFF
com.ibm.ws.logging.message.file.name=loggingMessages.log
```

console.log 文件会在服务器重新启动时重置。

注：在所有平台上，日志都以缺省系统编码编写。

- 在 Windows™ 系统上，有两种编码：OEM 代码页（用于控制台输出）和 ANSI 代码页（用于读写文件）。console.log 文件使用 OEM 代码页，而所有其他日志都使用 ANSI 代码页。
- 在所有其他平台上，所有日志文件都使用缺省编码。

定时操作和 JDBC 调用

当应用程序服务器中的 JDBC 调用在操作速度上慢于或快于预期时，定时操作会生成已记录的警告。

概述

如果定时操作功能部件已启用，那么会跟踪 JDBC 操作在应用程序服务器中的运行持续时间。在操作所花费的执行时间超过或少于预期时，定时操作功能部件会记录警告。定时操作功能部件将定期地在应用程序服务器日志中创建报告，详细说明执行时间最长的操作。如果运行 `server dump` 命令，那么定时操作功能部件将生成一个报告，该报告包含此功能部件所跟踪的所有操作的相关信息。您可以使用这些报告中所列出的信息来决定是否有任何项的运行速度慢于或快于预期。

系统会定期向日志中生成一个报告，此报告中包含 10 个最长的 JDBC 定时操作。可在 server.xml 文件中配置此报告的生成频率和是否启用，缺省情况为每天（24 小时）生成一次。

要启用定时操作，请将 `timedOperations-1.0` 功能部件添加至 server.xml 文件。

可使用 `timedOperation` 元素禁止将报告生成到日志或更改报告频率（例如，更改为每 12 小时 1 次），如以下示例中所示：

```
<timedOperation enableReport="false" reportFrequency="12"/>
```

还可以使用 maxNumberTimedOperations 属性来记录警告（用于指示定时操作总数达到此属性指定的值）。系统会监视定时操作的数目，知道此数目很有用，因为系统会从堆中为每个定时操作分配内存，如果您发现定时操作数目过大，那么可禁用定时操作功能。可使用以下示例来配置 maxNumberTimedOperations 属性：

```
<timedOperation enableReport="false" reportFrequency="12" maxNumberTimedOperations="10000"/>
```

在此示例中，当定时操作数超过 10000 时，日志中会出现如下警告消息：

```
[4/18/13 23:01:37:316 EDT] 0000002c com.ibm.wsspi.timedoperations.TimedOperationService W
 TRAS0094I:
The total number of timed operations is 10000, which exceeds the configured maximum number of
 10000.
You can also find the number of timed operations in the report that is periodically generated to
 the logs.
If you find that the number of timed operations is excessive, you can disable the timed
 operations feature.
```

如果您将环境变量 *com.ibm.timedOperations.autoCleanup* 的值设置为 `true`，那么服务器会将所跟踪的定时操作数自动限制为在 <maxNumberTimedOperations> 属性中所指定的值。当定时操作总数达到所指定的最大值时，会记录警告。要限制所跟踪的定时操作数，当需要跟踪新的定时操作时，会删除最近最少使用的定时操作记录。当所跟踪的定时操作数达到所指定的最大值时，那么会显示警告消息，如下所示：

```
TRAS0095I: The total number of timed operations has reached the configured
 maximum of 10000. As new timed operations are created the least recently
 used timed operations will be removed to maintain the total number of tracked
 timed operations at this level.
```

还可使用 `server dump` 命令以在 messages.log 文件中获取所有定时操作的完整报告，这些操作按类型分组，每组内按实际的平均持续时间进行排序。

以下示例显示了一条已记录的样本消息：

```
 [3/14/13 14:01:25:960 CDT] 00000025 TimedOperatio W   TRAS0080W: Operation xigemaas.datasource.execute:
jdbc/exampleDS:insert into cities values ('myHomeCity', 106769, 'myHomeCountry') took 1.541 ms to complete,
 which was longer than the expected duration of 0.213 ms based on past observations.
```

以下示例显示了日志中自动生成的样本报告：

```
[12/13/12 7:42:29:509 CST] 0000001d com.ibm.wsspi.timedoperations.TimedOperationService I TRAS0092I:
The following operations took the longest time to run since the last report has been generated:
Operation xigemaas.datasource.execute:jdbc/exampleDS:insert into cities values ('myHomeCity',
106769, 'myHomeCounty') took 194ms to complete
Operation xigemaas.datasource.execute:jdbc/exampleDS:select county from cities where name=
'myHomeCity' took 187ms to complete
Operation xigemaas.datasource.execute:jdbc/exampleDS:drop table cities took 182ms to
complete\Operation xigemaas.datasource.execute:jdbc/exampleDS:insert into cities values
('myHomeCity', 106769, 'myHomeCounty') took 151ms to complete
```

事件日志记录

作为监视和诊断功能的一部分，xigemaAS 在 Java Platform Enterprise Edition 的各个组件中生成事件以跟踪请求。应用程序请求运行时，`eventLogging-1.0` 功能部件会记录这类事件。通过使用此功能部件，用户可跟踪正在 xigemaAS 中运行的请求。每个请求与唯一相关因子（称为请求标识）及上下文信息相关联，上下文信息帮助用户了解特定于该请求的数据。

Event Logging 功能部件是通过服务器配置控制的。此功能部件是在 server.xml 文件中配置的。

以下样本日志显示对应 *AAY6TalVDTO_AAAAAAAAAAK* 请求标识和 *TradeWeb* 上下文的端到端事件日志：

```
[12/15/14 18:24:29:528 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeScenarioServlet
[12/15/14 18:24:29:531 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeAppServlet
[12/15/14 18:24:29:532 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /quote.jsp
[12/15/14 18:24:29:533 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp
[12/15/14 18:24:29:534 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=?
[12/15/14 18:24:29:547 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=? # duration=12.537ms
[12/15/14 18:24:29:556 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp #
 duration=22.171ms
[12/15/14 18:24:29:671 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp
[12/15/14 18:24:29:672 IST] 0000002e EventLogging  I BEGIN requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=?
[12/15/14 18:24:29:677 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.datasource.psExecuteQuery # contextInfo=jdbc/TradeDataSource | select *
 from quoteejb q where q.symbol=? # duration=4.968ms
[12/15/14 18:24:29:684 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | /displayQuote.jsp #
 duration=12.569ms
[12/15/14 18:24:29:685 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK #
 eventType=websphere.servlet.service # contextInfo=TradeWeb | /quote.jsp # duration=152.752ms
[12/15/14 18:24:29:686 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeAppServlet #
 duration=154.616ms
[12/15/14 18:24:29:687 IST] 0000002e EventLogging  I END requestID=AAY6TalVDTO_AAAAAAAAAAK
 # eventType=websphere.servlet.service # contextInfo=TradeWeb | TradeScenarioServlet #
 duration=158.283ms
```

此请求以 `BEGIN websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet"` 事件（引用样本代码中的第一行）开头，以 `END websphere.servlet.service "contextInfo=TradeWeb | TradeScenarioServlet"`（引用样本代码中的最后一行）结尾。此请求耗用总时间也会显示在结尾（在样本代码中为 158.283 ms）。

可通过查看主请求的 `BEGIN` 和 `END` 来查看子请求。还可找到每个子请求耗用的时间。

为获得最佳性能，可在启用事件日志记录时使用二进制日志记录。事件日志条目中的 eventType、contextInfo 和 requestID 属性存储为日志记录扩展。可使用这些日志记录扩展以借助 binaryLog 命令来过滤日志。

解析 messages.log 文件中的事件日志条目

事件日志使用以下格式捕获事件信息：

```
[Log mode] [Request Identifier] # [Event Type] # [Context Information] #
 [Duration] (optional)
```

其中

- Log mode 指示日志是在进入还是退出事件时记录的。BEGIN 指进入事件，END 指退出事件。
- Request identifier 是分配给每个请求的唯一字符串。它可用于过滤属于特定请求的事件。例如：**requestId=AAY6TalVDTO_AAAAAAAAAAK**
- Event type 提供有关事件源的信息，可以是下表中给定的任何受支持事件类型。事件类型可用于过滤特定类型的事件。示例：**eventType=websphere.servlet.service**
- Context information 是事件的上下文信息，用于指定与事件类型相关的详细信息。此信息根据事件类型不同而变化。上下文信息可包含多个部分，各部分之间以 |（| 的两边有空格）分隔。下表中给定各种事件类型的上下文信息的示例。
- Duration 指示事件耗用的时间。duration 仅显示在退出事件条目中。示例：**duration=158.283ms**

除 log mode（以空格分隔）外，所有其他日志属性以 #（# 两边有空格）分隔。例如，

```
[12/15/14 18:24:29:687 IST] 0000002e EventLogging  I END
 requestID=AAY6TalVDTO_AAAAAAAAAAK # eventType=websphere.servlet.service #
 contextInfo=TradeWeb | TradeScenarioServlet # duration=158.283ms
```

下表列示事件日志记录支持的事件类型：

表 72: 受支持事件类型及相关上下文信息

组件	事件类型	上下文信息	示例
Servlet	websphere.servlet.destroy websphere.servlet.service	[Application Name] | [Servlet Name] | [Path Information] | [Query String]	contextInfo=**TradeWeb | /displayQuote.jsp**
Session	websphere.session.dbSessionDestroyedByTimeout, websphere.session.dbSessionDestroyed websphere.session.sessionAccessed websphere.session.sessionCreated websphere.session.sessionDestroyedByTimeout websphere.session.sessionDestroyed	[Session Id] [Session Id] | [Session Attribute Name]	contextInfo=**EuitabHZUOD7J2u01HDdAG0** contextInfo=**EuitabHZUOD7J2u01HDdAG0 | userID**

组件	事件类型	上下文信息	示例
	websphere.session.sessionLiveCountDec websphere.session.sessionLiveCountInc websphere.session.sessionReleased websphere.session.getAttribute websphere.session.setAttribute		
JDBC	websphere.datasource.execute websphere.datasource.executeQuery websphere.datasource.executeUpdate websphere.datasource.psExecute websphere.datasource.psExecuteQuery websphere.datasource.psExecuteUpdate websphere.datasource.rsCancelRowUpdates websphere.datasource.rsDeleteRow websphere.datasource.rsInsertRow websphere.datasource.rsUpdateRow	[Jndi Name Of Data Source] | [SQL Query]	`contextInfo=jdbc/TradeDataSource | select * from quoteejb q where q.symbol=?`

检测运行缓慢和挂起的请求

任何请求的持续时间超过所配置阈值时，`requestTiming-1.0` 功能部件提供诊断信息。

Request Timing 功能部件可跟踪进入系统的每个请求的持续时间。可配置该功能部件以查找运行缓慢和挂起的请求。

- 检测运行缓慢的请求（见第 1763 页）
- 检测挂起请求（见第 1765 页）

检测运行缓慢的请求

如果请求的运行时间超过所配置的时间，那么系统会在消息日志文件中写入警告消息。系统会捕获有关该请求及构成该请求的事件的详细信息。

以下样本显示对应一个请求的日志消息，该请求运行时间超过运行缓慢的请求的阈值（缺省值为 10 秒）：

```
[12/1/14 11:58:09:629 IST] 0000001d com.ibm.ws.request.timing.SlowRequestTimer
  W TRAS0112W: Request AABjnS+lIn0_AAAAAAAAAb has been running on thread 00000021 for at least
 10003.571ms. The following stack trace shows what this thread is currently running.
```

```
  at java.util.HashMap.getEntry(HashMap.java:516)
  at java.util.HashMap.get(HashMap.java:504)
  at org.apache.derby.iapi.store.access.BackingStoreHashtable.get(Unknown Source)
  at org.apache.derby.impl.sql.execute.HashScanResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.NestedLoopJoinResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.ProjectRestrictResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.DMLWriteResultSet.getNextRowCore(Unknown Source)
  at org.apache.derby.impl.sql.execute.DeleteResultSet.setup(Unknown Source)
  at org.apache.derby.impl.sql.execute.DeleteResultSet.open(Unknown Source)
  at org.apache.derby.impl.sql.GenericPreparedStatement.executeStmt(Unknown Source)
  at org.apache.derby.impl.sql.GenericPreparedStatement.execute(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedStatement.executeStatement(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeStatement(Unknown Source)
  at org.apache.derby.impl.jdbc.EmbedPreparedStatement.executeUpdate(Unknown Source)
  at
 com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.executeUpdate(WSJdbcPreparedStatement.java:626)
  at com.ibm.websphere.samples.trade.direct.TradeDirect.resetTrade(TradeDirect.java:1832)
  at
 com.ibm.websphere.samples.trade.web.TradeConfigServlet.doResetTrade(TradeConfigServlet.java:65)
  at com.ibm.websphere.samples.trade.web.TradeConfigServlet.service(TradeConfigServlet.java:348)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:668)
  at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1275)
  ....
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1121)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:614)
  at java.lang.Thread.run(Thread.java:769)

Duration    Operation
10007.571ms + websphere.servlet.service | TradeWeb | TradeConfigServlet?action=resetTrade
    3.923ms       websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 holdingejb where holdingejb.account_accountid is null
    0.853ms       websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 accountprofileejb where userid like 'ru:%'
 5271.341ms +     websphere.datasource.psExecuteUpdate | jdbc/TradeDataSource | delete from
 orderejb where account_accountid in (select accountid from accountejb a where a.profile_useri
 like 'ru:%')
```

系统会继续监视该请求，如果该请求的运行时间再次超过 10 秒，那么系统会再记录一条警告。日志消息为以下格式：

```
TRAS0112W: Request < (Request ID)> has been running on thread <THREADID> for at least
 <DURATION>. The following stack trace shows what this thread is currently running.
<STACK TRACE>
<DURATION AND OPERATIONS Table>
```

REQUEST ID

此相同标识可用于搜索对应该请求的日志和跟踪消息。尤其是您使用二进制日志记录时，您可搜索使用二进制日志命令来搜索具有相同 `requestID` 扩展的日志和跟踪条目。

STACK TRACE

指示正在运行的方法。在上一样本中，可在 `TRAS0112W` 行后查看当前请求的堆栈跟踪。

DURATION AND OPERATIONS 表

进行堆栈跟踪后，可查找该请求的表格格式，该表格显示持续时间和操作（又称为事件）。**持续时间**列指示该请求的对应操作的耗用时间。加号 (+) 指示该请求内的事件仍在运行。下一行显示不带 + 的持续时间，它指示已在指定持续时间内完成的对应操作。对于该操作，“操作”显示 `EVENT TYPE` 和 `CONTEXT INFO`（这是可选项）。有关事件类型和上下文信息的更多信息，请参阅事件日志记录（见第 1654 页）。

通过分析消息，您可确定请求运行缓慢的原因。但是，可能难以确定请求是在该点停住还是仍在缓慢运行。因此，可按指定 *<slowRequestThreshold>* 的时间间隔来查看针对任何运行缓慢的请求记录的三条消息。通过使用三个不同的堆栈跟踪和请求数据，您可更深入地了解问题。在第三条警告后，系统不会再记录有关该请求的任何警告，除非该请求的持续时间超过挂起请求检测阈值。

检测挂起请求

如果请求超过缺省 hungRequestThreshold 或所配置阈值，那么系统会在消息日志文件中写入警告消息及有关该请求的详细信息。系统会捕获有关该请求及构成该请求的事件的详细信息。检测挂起请求时，系统会执行一系列（三个）线程转储（java 核心），各个转储之间存在 1 分钟延迟。以下日志消息样本显示超过挂起请求检测阈值的请求的日志消息。缺省持续时间值为 10 分钟。以下示例中配置的值为 4 分钟。

[WARNING] TRAS0114W: 请求 AAA7WlpP7l7_AAAAAAAAAAA 已在线程 00000021 上运行了至少 240001.015ms。下表显示此请求执行期间运行的事件。

```
Duration       Operation
240001.754ms + websphere.servlet.service | TestWebApp | TestServlet?sleepTime=480000
     0.095ms       websphere.session.setAttribute | mCzBMyzMvAEnjMJJx9zQYIw | userID
     0.007ms       websphere.session.setAttribute | mCzBMyzMvAEnjMJJx9zQYIw | visitCount
```

如果请求稍后完成（请求最初被检测为挂起），那么系统会记录类似以下示例的消息：TRAS0115W: 先前被检测为挂起的请求 AAA7WlpP7l7_AAAAAAAAAAA 在 479999.681 秒后完成。

👉 注：请求被检测为挂起时，系统会启动一系列（三个）线程转储。完成三个线程转储后，仅当新请求被检测为挂起时，系统才会创建其他线程转储。

二进制日志记录

二进制日志记录是一个高性能日志和跟踪工具。

概述

日志和跟踪存储

二进制日志记录提供日志数据存储库和跟踪数据存储库。请参阅下图以了解应用程序和应用程序服务器如何存储日志和跟踪信息。

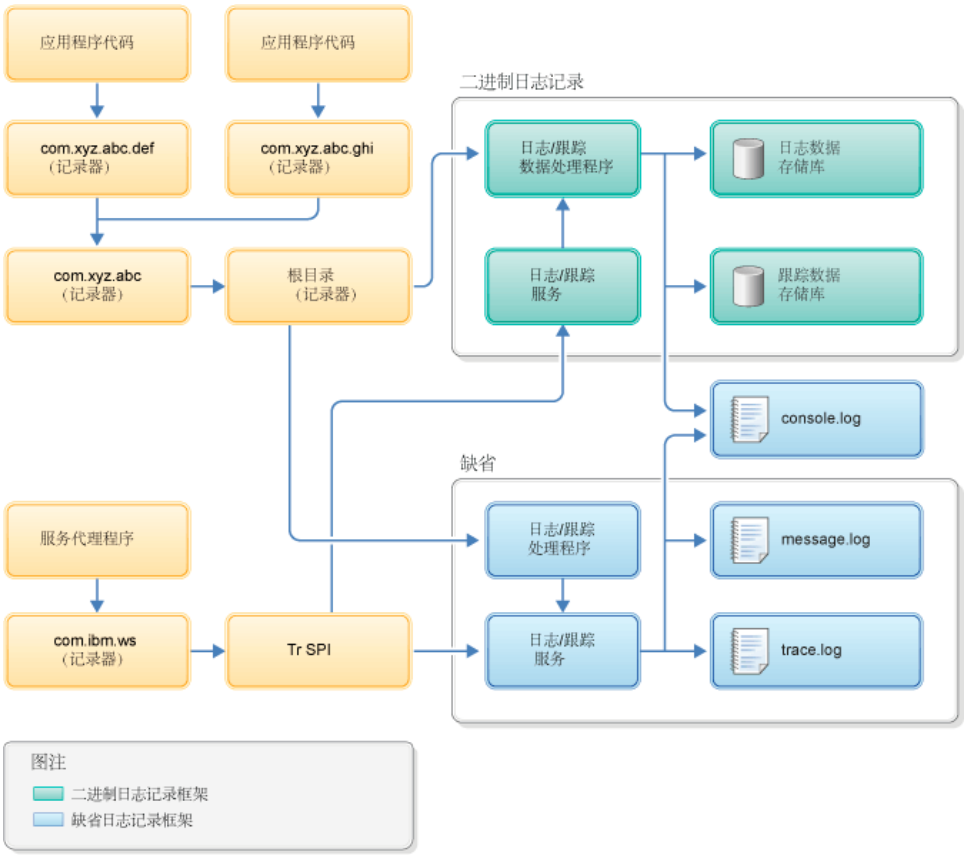


图 42: 用于HPEL日志记录和基本日志记录的日志和跟踪存储器

日志数据存储库

日志数据存储库是用于存储日志记录的存储工具。通常由管理员复查日志数据。这包括应用程序或服务器写入 System.out、System.err、OSGi 记录服务（LOG_INFO 级别或更高级别，包括 LOG_INFO、LOG_WARNING 和 LOG_ERROR）或 java.util.logging（“详细”级别或更高级别，包括“详细”、“配置”、“参考”、“审计”、“警告”、“严重”、“致命”和任何定制级别）的任何信息。

跟踪数据存储库

跟踪数据存储库是用于存储跟踪记录的存储工具。跟踪数据通常供应用程序员或者 xigemaAS 支持团队使用。这包括将应用程序或服务器写入 OSGi 记录服务（LOG_DEBUG 级别）或 `java.util.logging`（“详细”以下的级别，包括精细、较精细、最精细和任何定制级别）的任何信息。

每个日志和跟踪事件都只存储在一个位置

日志事件 System.out 和 System.err 存储在日志数据存储库中。跟踪事件存储在跟踪数据存储库中。仅将每种类型的事件存储在一个位置可确保不会因重复进行数据存储而影响性能。

👉 注：在日志记录性能较重要的情况下，应该禁用控制台日志。写入控制台日志中的任何内容都已经存储在日志数据存储库中。

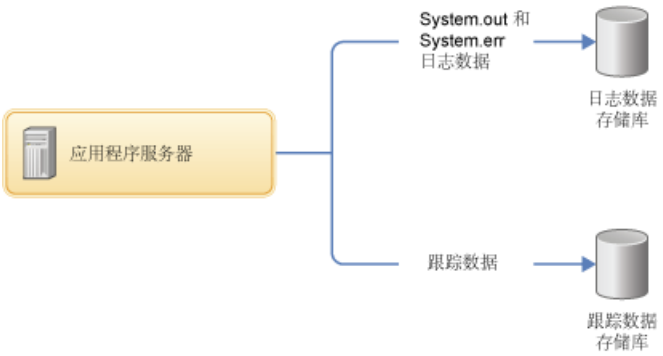


图 43: 连接至日志数据库和跟踪数据存储库的应用程序服务器

除非数据需要格式化，否则不会将数据格式化

格式化数据以便用户阅读会占用处理器时间。日志和跟踪数据会更快速地以专用二进制表示来存储，而不是在运行时格式化日志事件和跟踪事件数据。这将提高日志和跟踪工具的性能。通过延迟日志和跟踪格式化直到 `binaryLog` 命令处于运行状态，从未查看的日志或跟踪部分就决不会格式化。

将日志和跟踪数据写入磁盘之前对其进行缓存

将大块数据写入磁盘比将相同数量的数据写入小块的效率更高。二进制日志记录工具使您能够在将日志和跟踪数据写入磁盘之前对数据进行缓存。缺省情况下，在将日志和跟踪数据写入磁盘之前，会将数据存储在 8 KB 缓冲区中。如果缓冲区在 10 秒之内填满，那么会将缓冲区写入磁盘。如果缓冲区未在 10 秒之内装满，那么会自动地将缓冲区写入磁盘以确保日志具有 最新信息。

日志和跟踪的管理

二进制日志记录的目的是为了使之容易配置和理解。例如，管理员很容易配置供日志和跟踪专用的磁盘空间量、日志和跟踪记录的保留时间以及由服务器来管理日志和跟踪内容。又比如，可以使用一个容易使用的命令 (`binaryLog`) 来访问所有日志、跟踪、System.out 和 System.err 内容，从而避免对于要在哪个文件中访问特定内容可能发生任何混淆。

从日志数据存储库和跟踪数据存储库中读取

日志数据存储库和跟踪数据存储库以 xigemaAS 专用格式进行存储，并且无法使用“记事本”或 VI 之类的文本文件编辑器来阅读。可以使用 `binaryLog` 命令将日志数据存储库和跟踪数据存储库复制到纯文本格式。

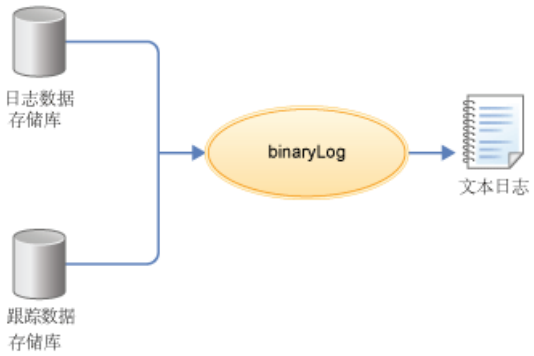


图 44: 拉取数据到文本日志中

binaryLog 命令

binaryLog 是为用户提供的一个容易使用的命令行工具，用来处理日志数据存储库和跟踪数据存储库。binaryLog 提供过滤和格式化选项，使在日志数据存储库和跟踪数据存储库中查找重要内容更容易。例如，用户可以过滤任何错误或警告，然后过滤在 10 秒钟之内发生的所有日志和跟踪条目，以找出同一线程上的关键错误消息。

使用日志和跟踪记录扩展内容进行过滤

二进制日志记录工具可让开发者使用日志记录上下文 API (com.ibm.websphere.logging.hpel.LogRecordContext) 将定制扩展添加到日志和跟踪记录。可以使用 binaryLog 命令行工具根据日志和跟踪记录扩展的内容来过滤记录。

开发资源

已经对二进制日志记录进行设计，以便比缺省日志记录工具更灵活高效地处理日志和跟踪内容。很容易对日志和跟踪内容进行过滤以仅显示感兴趣的记录。您可以使用命令行（请参阅 binaryLog 命令的描述），或者开发者可以使用 HPEL API 来创建功能强大的日志处理程序。

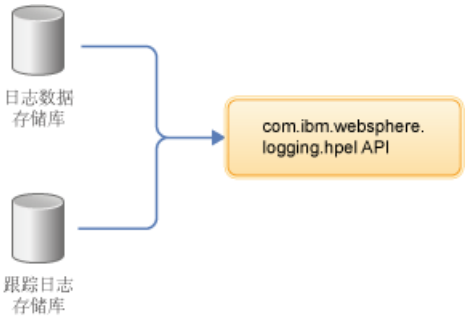


图 45: 连接至 HPEL API 的日志数据存储库和跟踪数据存储库

读取日志数据和跟踪数据

已经提供了一个 API，以使开发者容易开发一些工具来使用二进制日志和跟踪存储库中的内容。例如，开发者可编写 Java™ 程序以搜索日志和跟踪内容来查找其消息标识与已知重要消息标识列表匹配的任何消息。此 API 位于 com.ibm.websphere.logging.hpel 包中。请参阅 API 文档以了解有关 HPEL 日志读取 API 的详细信息。

日志和跟踪记录可扩展性

开发者可以通过日志记录上下文 API (`com.ibm.websphere.logging.hpel.LogRecordContext`) 向日志和跟踪记录添加定制扩展。当二进制日志记录存储日志和跟踪记录时，它会将存在于日志记录上下文中的任何扩展包括在同一线程上。例如，开发者可以编写 Servlet 过滤器，以向日志记录上下文添加重要的 HTTP 请求参数。当该 Servlet 运行时，HPEL API 会向在同一线程上创建的任何日志和跟踪记录添加这些扩展。

与其他日志和跟踪记录字段一样，开发者可以使用 HPEL API 来访问记录扩展。这在编写工具以从日志和跟踪存储库中读取时很有用。在运行时，开发者还可以利用日志记录上下文 API 来访问定制日志处理程序、过滤器和格式化程序中的扩展。

binaryLog 命令选项

使用 `binaryLog` 命令来查看或复制二进制日志记录存储库的内容，或者列示该存储库中可用的服务器进程实例。

二进制日志和跟踪工具以二进制格式写入至存储库。可以使用 binaryLog 命令来查看、查询和过滤存储库。binaryLog 命令提供的选项可快速地将存储库内容转换为各种格式的文本文件，例如基本格式和高级格式。该命令还提供可更方便地从记录中获取所需数据的选项；例如，允许您按级别、记录器名称或日期和时间过滤所需的日志记录。

语法

命令语法如下所示：

```
binaryLog action {serverName | repositoryPath} [options]
```

options 的值随 action 的值不同而不同。

参数

下列操作可用于 binaryLog 命令：

view

读取存储库，可以选择对其进行过滤，并创建用户可以读取的版本。

命令语法如下所示：

```
binaryLog view {serverName | repositoryPath} [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是同时包含 logdata 和 tracedata 目录的目录。

👉 注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

根据最早记录创建日期进行过滤。必须按日期（例如，--minDate="2/20/13"）或者日期和时间（例如，--minDate="2/20/13 16:47:21:445 EST"）指定值。

- --maxDate=*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，--maxDate="2/20/13"）或者日期和时间（例如，--maxDate="2/20/13 16:47:21:445 EST"）指定值。
- --minLevel=*value*

  根据最低级别进行过滤。值必须为下列其中一个值：FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL。
- --maxLevel=*value*

  根据最高级别进行过滤。值必须为下列其中一个值：FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL。
- --includeLogger=*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- --includeMessage=*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- --includeThread=*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，--includeThread=2a）。
- --includeExtension=*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- --includeInstance=*value*

  包括所指定服务器实例中的记录。值必须为 "latest" 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

监视选项：

--monitor

连续监视存储库并输出所生成的新内容。

输出选项：

- --format={basic|advanced|CBE-1.0.1}

  指定要使用的输出格式。"basic" 为缺省格式。
- --encoding=*value*

  指定要用于输出的字符编码。

copy

读取存储库，可以选择对其进行过滤，并将内容写入新的存储库。

命令语法如下所示：

```
binaryLog copy {serverName | repositoryPath} targetPath [options]
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

targetPath

指定要在其中创建新的存储库的路径。必须指定 *targetPath*。

👉 注：必须指定 *serverName* 或 *repositoryPath*，以及 *targetPath*。

过滤器选项：

所有过滤器均为可选。当使用多个过滤器时，它们将在逻辑上相“与”。

- `--minDate=`*value*

  根据最早记录创建日期进行过滤。必须按日期（例如，`--minDate="2/20/13"`）或者日期和时间（例如，`--minDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--maxDate=`*value*

  根据最迟记录创建日期进行过滤。必须按日期（例如，`--maxDate="2/20/13"`）或者日期和时间（例如，`--maxDate="2/20/13 16:52:32:808 EST"`）指定值。
- `--minLevel=`*value*

  根据最低级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--maxLevel=`*value*

  根据最高级别进行过滤。值必须为下列其中一个值：`FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`。
- `--includeLogger=`*value*[,*value*]*

  包括具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--excludeLogger=`*value*[,*value*]*

  排除具有所指定的记录器名称的记录。值可包含 * 或 ? 作为通配符。
- `--includeMessage=`*value*

  根据消息名称进行过滤。值可包含 * 或 ? 作为通配符。
- `--includeThread=`*value*

  包括具有所指定的线程标识的记录。值必须采用十六进制（例如，`--includeThread=2a`）。
- `--includeExtension=`*name*=*value*[,*name*=*value*]*

  包括具有所指定的扩展名和值的记录。值可包含 * 或 ? 作为通配符。要在值中包含逗号，必须使用“\,”
- `--includeInstance=`*value*

  包括所指定服务器实例中的记录。值必须为 `"latest"` 或者有效的实例标识。请使用 list Instances 操作来运行此命令以查看有效实例标识的列表。

listInstances

列示存储库中的服务器实例的标识。服务器实例是从服务器启动直到停止期间写入的所有日志记录/跟踪记录的集合。可将服务器实例标识与 binaryLog 查看操作的 --includeInstance 选项配合使用。

命令语法如下所示：

```
binaryLog listInstances {serverName | repositoryPath}
```

serverName

指定具有要从其中读取的存储库的 xigemaAS 服务器的名称。

repositoryPath

指定要从其中读取的存储库的路径。这通常是包含 logdata 和 tracedata 目录的目录。

👉 注：如果在命令行上既未指定 *serverName*，也未指定 *repositoryPath*，那么将对缺省服务器实例 defaultServer（如果它存在）执行此任务。

请了解 binaryLog 过滤优化。binaryLog 工具在与下列过滤选项配合使用时，能够最高效地过滤日志和跟踪数据：

- --minDate
- --maxDate
- --includeThread
- --minLevel
- --maxLevel

用法示例

请参阅 binaryLog 命令的下列示例。

- 显示 defaultServer 存储库中在 2013 年 7 月 19 日到 2013 年 8 月 2 日期间发生的所有事件。

```
binaryLog view --minDate=07/19/13
--maxDate=08/02/13
```

- 显示 myServer 服务器中发生的、其指定级别为 WARNING 或更高级别的新事件，并在服务器将其写入日志存储库时使用高级格式。

```
binaryLog view myServer --monitor --minLevel=WARNING
--format=advanced
```

- 写入来自存储库（位于 /apps/server1/logs）中的日志消息；仅包括已写入特定存储库的错误流的那些消息。

```
binaryLog
view /apps/server1/logs --includeLogger=SystemErr
```

- 查看 defaultServer 存储库中在 2012 年 9 月 14 日美国东部夏令时下午 4:28 之前发生的事件。

```
binaryLog view
--maxDate="09/14/12 16:28:00:000 EDT"
```

- 写入 defaultServer 存储库中包含“thread”扩展名并且具有以下值的事件：'Default Executor -thread-4'

```
binaryLog view --includeExtension=thread="Default Executor-thread-4" --format=advanced
```

- 查看 defaultServer 存储库中的服务器实例的列表：

```
binaryLog listInstances

使用 D:\wlp\usr\servers\defaultServer\logs 作为存储库目录。
```

```
实例标识                        开始日期
1358809441761                   1/21/13 18:04:01:761 EST
1358864476191                   1/22/13 9:21:16:191 EST
1358869523192                   1/22/13 10:45:23:192 EST
1358871281166                   1/22/13 11:14:41:166 EST
1358879829000                   1/22/13 13:37:09:000 EST
1358892222067                   1/22/13 17:03:42:067 EST
```

- 查看 defaultServer 中的使用前一示例中的其中一个实例标识的事件：

```
binaryLog view --includeInstance=1358871281166
```

- 将 defaultServer 中的指定级别为 WARNING 或更高级别的事件从最新服务器实例复制到 d:\toSupport 目录中的新存储库。

```
binaryLog copy defaultServer d:\toSupport --minLevel=warning --includeInstance=latest
```

在 xigemaAS 中配置二进制日志记录

使用此信息作为在 xigemaAS 中配置二进制日志记录的指南。

与缺省 xigemaAS 日志和跟踪框架相比，二进制日志记录提供处理速度更快的日志和跟踪功能以及更灵活的方式来使用日志和跟踪内容。

服务器配置由 bootstrap.properties 文件、server.xml 文件以及这些文件随附的任何可选文件组成。bootstrap.properties 文件指定在处理主要配置之前必须提供的属性，这些属性会保持最少。server.xml 文件是服务器的主要配置文件。

server.xml 文件及其关联文件使用适合于大多数文本编辑器的简单 XML 格式。

bootstrap.properties 文件指定服务器是否应该将二进制日志记录用作日志和跟踪框架，或者用作缺省日志和跟踪框架。

可以通过服务器配置或者 bootstrap.properties 文件来配置二进制日志记录。

- 服务器配置：要从您自己的代码获取日志记录（进行服务器配置处理之后载入），请使用服务器配置来配置二进制日志记录。
- bootstrap.properties 文件：可能需要设置日志记录属性以使其在服务器配置文件得到处理之前生效。例如，如果需要分析服务器启动或配置处理早期发生的问题。在这种情况下，您可以在 bootstrap.properties 文件中配置二进制日志记录。

可以在 bootstrap.properties 或者 server.xml 文件中设置日志记录属性。使用 server.xml 文件中的属性，或者使用 bootstrap.properties 文件中的等价属性。从服务器读取 bootstrap.properties 文件开始，使用 bootstrap.properties 文件中的任何设置，直到完成处理 server.xml 文件为止。如果bootstrap.properties 文件中的日志记录属性未在 server.xml 文件中进行替换或重置，那么将继续使用 bootstrap.properties 文件中的属性值。

如果启用了二进制日志记录，那么会忽略 maxFileSize、maxFiles、messageFileName、traceFileName 和 traceFormat 日志记录元素属性（因为二进制日志记录是在没有 trace.log 和 messages.log 文件的情况下运行）。traceSpecification、consoleLogLevel 和 logDirectory 属性继续用来设置跟踪规范、控制台日志的级别以及日志和跟踪文件的布置。

如果您在 server.xml 文件中设置日志记录或二进制日志记录属性，那么可以通过在 bootstrap.properties 文件中将相应的属性设置为同一值来避免在启动时与运行时之间更改配置。请注意，如果在 bootstrap.properties 文件中未设置任何日志记录属性或者二进制日志记录属性，那么服务器将使用缺省日志记录设置。

- 通过更新 bootstrap.properties 文件对服务器启用二进制日志记录。

在 bootstrap.properties 文件中，单独添加下列文本行：

```
websphere.log.provider=binaryLogging-1.0
```

- 使用下列参数来配置二进制日志记录。
  列出的所有子元素都是 server.xml 文件中日志记录元素的子元素。
  下表列出了可在 server.xml 文件中配置的属性，以及可在 bootstrap.properties 文件中设置的等价属性：

表 73: 可在 server.xml 中配置的二进制日志记录属性以及可在 bootstrap.properties 中设置的等价属性

日志记录子元素	属性	等价的 **bootstrap.properties** 属性
binaryLog	purgeMaxSize	com.ibm.hpel.log.purgeMaxSize
	purgeMinTime	com.ibm.hpel.log.purgeMinTime
	fileSwitchTime	com.ibm.hpel.log.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.log.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.log.outOfSpaceAction
binaryTrace	purgeMaxSize	com.ibm.hpel.trace.purgeMaxSize
	purgeMinTime	com.ibm.hpel.trace.purgeMinTime
	fileSwitchTime	com.ibm.hpel.trace.fileSwitchTime
	bufferingEnabled	com.ibm.hpel.trace.bufferingEnabled
	outOfSpaceAction	com.ibm.hpel.trace.outOfSpaceAction

以下示例显示配置为启用二进制日志记录的 bootstrap.properties 文件：

```
websphere.log.provider=binaryLogging-1.0
```

以下示例显示了具有二进制日志记录子元素的 server.xml 文件。日志内容设为在 96 小时后过期，跟踪内容设为最大保留 1024MB：

```
<server description="new server">

    <logging>
        <binaryLog purgeMinTime="96"/>
        <binaryTrace purgeMaxSize="1024"/>
    </logging>

</server>
```

在重新启动服务器之后，将启用并配置二进制日志记录。

运行时环境已知问题和限制

使用 xigemaAS 运行时环境时，有一些已知问题和限制。

已知问题和限制的列表：

- 一般限制：
  - 支持的最低 *Java* 级别（见第 1776 页）
  - 安装目录名称和路径不能包含非 *ASCII* 字符（见第 1776 页）
  - 运行时更改 *JDBC* 数据源可能导致 *JPA* 失败（见第 1776 页）
  - 依赖于 *getRealPath* 所返回结果的应用程序必须部署为展开的应用程序 *(* 而不是 *WAR* 文件 *)*（见第 1776 页）
  - 文件集限制（见第 1777 页）
  - 覆盖 *Java SDK* 中的类（见第 1777 页）
  - 取消发布共享库时, 只有停止服务器才能删除共享库（见第 1777 页）
  - *java:global* 查询限制（见第 1777 页）
  - 未在嵌入的 *xigemaAS* 服务器中启动应用程序（见第 1778 页）
  - 与 *xigemaMQ* 资源适配器和通用 *JCA* 支持相关的限制（见第 1778 页）
  - 不能对*"dropins"*目录中的应用程序进行版本控制（见第 1778 页）
  - 共享会话应用程序必须将会话对象存储在共享库中（见第 1778 页）
- 特定于 xigemaAS 功能部件的限制：
  - 管理中心功能部件限制（见第 1778 页）
  - *appSecurity-2.0* 功能部件限制（见第 1777 页）
  - *Bean* 验证功能部件限制（见第 1778 页）
  - 动态高速缓存功能部件限制（见第 1779 页）
  - *ejbLite-3.1* 功能部件限制（见第 1779 页）
  - *j2eeManagement-1.1* 功能部件限制（见第 1779 页）
  - *jaxb-2.2* 功能部件限制（见第 1779 页）
  - *jaxws-2.2* 功能部件限制（见第 1780 页）
  - *jpa-2.1* 功能部件限制（见第 1780 页）
  - *jsp-2.2* 功能部件限制（见第 1780 页）
  - *monitor-1.0* 功能部件限制（见第 1780 页）
  - *requestTiming-1.0* 功能部件限制（见第 1780 页）
  - *restConnector-1.0* 功能部件限制（见第 1781 页）
  - *wmqJmsClient-1.1* 功能部件限制（见第 1781 页）
  - *wmqJmsClient-2.0* 功能部件限制（见第 1781 页）
  - *concurrent-1.0* 功能部件限制（见第 1782 页）
  - *jacc-1.5* 功能部件限制（见第 1782 页）
  - 运行时修改 *dataSource*、*jdbcDriver*、*connectionManager* 和 *JDBC* 供应商属性可能会导致 *JPA* 失败（见第 1776 页）

支持的最低 Java™ 级别

此 xigemaAS 概要文件在任何符合 Java™ SE 6 或 Java™ SE 7 运行时环境中受支持，并且受限于针对下列特定实现显示的最低受支持级别。

Java™ SE 6 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 6.0 (J9 2.6) SR 1。对于来自 Oracle 的 JDK，最低受支持级别为 Java™ 6 Update 26。

Java™ SE 7 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 xigemaAS 运行时环境 Java™ Technology Edition 7.0 .4.1。对于 Windows™ 和 Linux™ 上来自 Oracle 的 JDK，最低受支持级别为 Java™ SDK/JRE/JDK 7.0.17。对于 Mac OS X 上来自 Oracle 的 JDK，最低受支持级别为 Java™ SDK/JRE/JDK 7.0 Update 15。

Java™ SE 8 运行时环境

对于来自 xigemaAS 的 Java™ SDK，最低受支持级别为 xigemaAS SDK Java™ Technology Edition V8。对于来自 Oracle 的 JDK，最低受支持级别为 Java™ 8 update 25。

在分布式平台上，支持 32 位或 64 位 Java™。

对于 Windows™ 和 Linux™ 系统，您可以使用 Oracle JDK 或 xigemaAS JDK。对于 HP 系统和 Mac OS，请使用 Oracle JDK。

安装目录名称和路径不能包含非 ASCII 字符

最近的 JVM 并不完全支持在 `-jar` 和 `-javaagent` 命令中使用非 ASCII 字符。在安装目录名称和路径中只应使用 ASCII 字符。

运行时更改 JDBC 数据源可能导致 JPA 失败

如果未通过属性来指定数据库字典类型，那么在创建第一个实体管理器和建立数据库连接时，OpenJPA 会检测并计算数据库字典类型。此数据库字典类型可用于后续创建的所有实体管理器。如果在应用程序处于运行状态时更改了 JDBC 数据源，那么实体管理器工厂就检测不到此更改，因此对新数据源执行的操作都将继续使用旧字典。如果将数据库更改为其他供应商，那么这可能会导致失败。

将数据库更改为其他供应商之后，请重新启动应用程序。

运行时修改 dataSource、jdbcDriver、connectionManager 和 JDBC 供应商属性可能会导致 JPA 失败

如果在服务器处于运行状态时更新 `dataSource`、`jdbcDriver`、`connectionManager` 或者任何 JDBC 供应商属性列表（例如，`properties.db2.jcc` 或 `properties.oracle`）的配置，那么您可能会看到 `J2CA8040E` 失败。这些失败会说明不能将多个 `dataSource` 元素与单个 `connectionManager` 相关联。即使您的配置只将一个 `connectionManager` 与 `dataSource` 元素相关联，也会生成这些失败。

这些 JDBC 资源的配置只要发生更新，就应该重新启动服务器。

依赖于 `getRealPath` 所返回结果的应用程序必须部署为展开的应用程序 (而不是 WAR 文件)

Java™ EE 规范说明如果是从 Web 归档 (WAR) 文件提供内容，那么 `getRealPath()` 方法会返回 `null` 值。将 WAR 文件部署到 xigemaAS 时，概要文件不会自动将归档文件解压到目录结构。因此，应用程序可能无法启动。如果应用程序依赖于 `getRealPath()` 所返回的结果，那么必须将应用程序部署为展开的 Web 应用程序（而不是 WAR 文件）。例如，您可以手动解压 WAR 文件并将展开的应用程序复制到 dropins 目录。

文件集限制

下列限制适用于文件集：

- 文件集不会递归地浏览基本目录的子目录。例如，不支持下列指令：

```
<fileset id="testFileset" dir="\temp" includes="**\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="*\a.jar"/>
<fileset id="testFileset" dir="\temp" includes="a\b\a.jar"/>
```

覆盖 Java™ SDK 中的类

xigemaAS 概要文件支持的某些 Java™ EE 6 技术需要 Java™ SE 6 提供的 API 更新版本。JAX-WS、JAXB 和 `javax.annotation.Resource` 注解是 Java™ 6 SDK 包含级别低于 Java™ EE 6 所需级别的类的所有示例。当您使用 Java™ SDK V6 针对这些 API 编译应用程序代码时，需要使用 xigemaAS 概要文件提供的类，而不是 Java™ SDK 提供的类。必须执行下列其中一项操作：

- 如果您正在使用 `javac` 从命令行进行构建，那么使用 `javac -endorseddirs` 选项以及 ${wlp.install.dir}/dev/specs 目录中的 JAR 文件来编译您的代码。
- 如果您正在使用 Apache Ant 来进行构建，那么使用 `javac` 任务的 `<compilerarg>` 子元素以及 ${wlp.install.dir}/dev/specs 目录中的 JAR 文件来编译您的代码。在构建脚本中，作为单独的 `<compilerarg>` 元素来指定 `-endorseddirs` 选项和 ${wlp.install.dir}/dev/specs 目录。例如：

```
<javac srcdir="src" destdir="classes"/>
    <compilerarg value="-endorseddirs"/>
    <compilerarg value="${wlp.install.dir}/dev/specs"/>
</javac>
```

取消发布共享库时，只有停止服务器才能删除共享库

从服务器取消发布共享库时，不会立即将库 JAR 文件从服务器释放。因此，操作系统并不知道文件已不再使用，并且不让您删除该文件。下一次停止服务器时，将释放库 JAR 文件，并且您可以将其删除。

java:global 查询限制

应用程序中使用 `java:global` 查询定义的资源只能用来访问由部署在当前服务器中的应用程序声明的名称。

appSecurity-2.0 功能部件限制

对于 `appSecurity-2.0` 功能部件，存在下列限制：

- 对于 EJB 应用程序，run-as-mode SYSTEM_IDENTITY 在 ibm-ejb-jar-ext.xml 文件的扩展设置中不受支持。
- 单独会话 Bean 不支持 `getCallerIdentity` API。
- 角色名称可供 `HttpServletRequest.isUserInRole` 和 `EJBContext.isCallerInRole` API 或者部署描述符中的元素引用，而不先使用 `@DeclareRoles` 注解或者部署描述符中的 `<security-role/>` 元素来声明角色名称。但是，必须先声明角色，然后才能使用这些角色。

未在嵌入的 xigemaAS 服务器中启动应用程序

请确保使用指向 *xigemaASInstallDir*/bin/tools/ws-javaagent.jar 的 JVM 参数 `-javaagent` 启动了用于启动嵌入式 xigemaAS 服务器的 Java™ 进程。如果未使用 `-javaagent` JVM 参数，那么服务器运行时会启动，但应用程序无法启动，同时没有明显异常。

与 xigemaMQ 资源适配器和通用 JCA 支持相关的限制

通过使用 `wmqJmsClient-1.1` 或 `wmqJmsClient-2.0` 功能部件，或通过使用通用 JCA 支持，可在 xigemaAS 内使用 xigemaMQ 资源适配器。

如果您想要使用基于 JMS 2.0 资源适配器的 xigemaMQ 资源适配器 V8.0.0.3 及更高版本，那么必须确保您正在使用与 JMS 2.0 资源适配器兼容的最新 xigemaAS 版本。

👉 注：

- 对于 xigemaAS V9.0.0.1，必须将 `wmqJmsClient-2.0` 功能部件与 xigemaMQ 资源适配器 V8.0.0.3 或更高版本配合使用。

如果正使用类属 JCA 支持，那么以下限制适用：

- 跟踪和日志记录未集成到使用通用 JCA 的 xigemaAS 跟踪系统中。跟踪将写至单独文件，并且必须通过设置系统属性来启用。启用跟踪的过程与为 Java™ 标准环境的 JMS 跟踪工具配置 xigemaMQ 类的过程相同。

不能对“dropins”目录中的应用程序进行版本控制

对于“dropins”目录中的应用程序，应用程序监视器使用文件名和文件扩展名来确定应用程序类型并生成应用程序标识和应用程序名称。因此，不能使用文件名或文件扩展名对应用程序指定版本号。建议不要在生产环境中使用“dropins”目录。

共享会话应用程序必须将会话对象存储在共享库中

使用共享会话上下文应用程序扩展或在 ibm-application-ext.xml 中使用 `<shared-session-context value="true"/>` 时，存储在会话中的所有对象必须在与应用程序相关联的共享库中可用，以便可使这些对象失效。

管理中心功能部件限制

对于 `adminCenter-1.0` 功能部件，将适用以下限制：

- 管理中心浏览工具中仅显示对应服务器、集群和应用程序资源的标记。不会显示对应运行时资源的标记。
- 管理中心的“监视器”视图的“CPU 使用率”图表对未提供进程 CPU 统计信息的 JVM 显示 **`0%`** 或 **`null%`** CPU 使用率。有关该图表的更多信息，请参阅监视管理中心中的指标。

Bean 验证功能部件限制

对于 `beanvalidation-1.0` 功能部件，存在下列限制：

- 不支持在 OSGi 应用程序内部进行 Bean 验证。

对于 `beanValidation-1.1` 功能部件，存在下列限制：

- 不支持在 OSGi 应用程序内部进行 Bean 验证。

- 在 validation.xml 中为 beanValidation-1.0 功能部件提供定制 ConstraintValidatorFactory 实现的应用程序不会针对 Bean 验证 1.1 API 进行编译。
- 如果 validation.xml 文件不在其关联模块中，那么可能只有一个 validation.xml 文件并且 com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp 属性必须在下列任一文件中设置为 false：
  - jvm.options

    ```
    -Dcom.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false
    ```

  - bootstrap.properties

    ```
    com.ibm.ws.beanvalidation.allowMultipleConfigsPerApp=false
    ```

动态高速缓存功能部件限制

下列动态高速缓存功能部件不可用或者可用性有限：

- 不支持高速缓存复制。
- 在使用随机逐出和基于大小的逐出技术时，仅支持高性能磁盘高速缓存方式。
- 在 cachespec.xml 文件中不支持 Web Service 客户端和服务器端高速缓存以及 Portlet 高速缓存。
- 不支持对 SingleThreadModel Servlet 进行 Servlet 高速缓存。
- 只包含 Enterprise JavaBeans™ (EJB) 的 JAR 文件不支持使用属性文件来定义高速缓存配置。
- 只能对 32 位 Java™ 虚拟机 (JVM) 限制堆高速缓存大小。

ejbLite-3.1 功能部件限制

对于 ejbLite-3.1 功能部件，存在下列限制：

- 不支持低于 V3.0 的 EJB 模块。此限制也意味着使用 .xmi 文件格式（而不是 .xml 文件格式）的绑定和扩展不受支持。
- 会话 Bean 未绑定到 ejblocal 命名空间，这表示 JNDI 查询和 ejb-ref 绑定名称必须使用 java:global、java:app 或 java:module 名称。忽略 ibm-ejb-jar-bnd.xml 中的 simple-binding-name 和接口 binding-name 元素。
- 有状态 Bean 钝化目录不可配置。文件会钝化到服务器工作区。

j2eeManagement-1.1 功能部件限制

对于 j2eeManagement-1.1 功能部件，以下限制适用：

- 不支持管理 EJB getListenerRegistry() 方法。不能在管理 EJB 组件中注册事件通知侦听器。

jaxb-2.2 功能部件限制

对于 jaxb-2.2 功能部件，适用下列限制：

- 如果应用程序需要 JAXB API 类且已经启动，且要启用 jaxb-2.2 服务器功能部件，那么必须使用 --clean 选项来重新启动服务器，这样应用程序就可以调用由 jaxb-2.2 功能部件提供的 JAXB 2.2 API 和实现类。否则，应用程序仍可能绑定至 Java™ SDK 中提供的 JAXB API 和实现类。
- 如果已启用 jaxb-2.2 服务器功能部件，而且您想要在应用程序中使用您自己的 JAXB API 和实现类，那么必须将您自己的 JAXB API 和实现 JAR 文件放入应用程序的 /WEB-INF/lib 目录中，并且将应用程序的类加载器配置为使用 *parentLast* 授权行为。否则，由 jaxb-2.2 功能部件提供的 JAXB API 和实现类

总是会生效。有关在 xigemaAS 上配置应用程序的类加载器行为的更多信息，请参阅使用替代版本来覆盖随附的 *API*（见第 1141 页）。

jaxws-2.2 功能部件限制

对于 `jaxws-2.2` 功能部件，存在下列限制：

- 如果应用程序提供它自己的 CXF JAR 文件副本来作为应用程序库（例如，在 Web 应用程序的 WEB-INF/lib 目录中），那么您无法在 server.xml 文件中启用 `jaxws-2.2` 功能部件。
- 因为 `jaxws-2.2` 功能部件依赖于 `jaxb-2.2` 功能部件，所以 `jaxb-2.2` 功能部件限制也适用于 `jaxws-2.2` 功能部件。
- 如果应用程序需要 JAX-WS API 类且已经启动，但 `jaxws-2.2` 服务器功能部件未启用，那么必须使用 --clean 选项来重新启动服务器，这样应用程序就可以调用由 `jaxws-2.2` 功能部件提供的 JAX-WS 2.2 API 和实现类。否则，应用程序仍可能绑定至 Java™ SDK 中提供的 JAX-WS API 和实现类。
- xigemaAS 的 Web Service 绑定文件是 ibm-ws-bnd.xml 文件。
- Apache Axis2 配置或类不受支持。
- 不支持用来实现 `javax.xml.ws.Provider<OMElement>` 或 `javax.xml.ws.Provider<String>` 接口的 Web Service 提供程序。
- 对于 xigemaAS ，必须使用尖括号将 MIME 附件的 content-id 属性括起来。例如，*<testID>*。
- xigemaAS 提供的 `wsgen` 工具不支持 -inlineSchemas 选项。
- 如果您想要使用 JAX-WS Web Service 来传输大型二进制数据以避免发生“内存不足”(OOM) 错误，请启用 MTOM。
- 对于 Web Service 应用程序，如果服务客户机与服务提供程序不在同一应用程序中，并且服务提供程序应用程序中的 WSDL 文件已更改，那么您需要手动重新启动 Web Service 客户机应用程序，以避免发生 WSDL 定义高速缓存问题。

jpa-2.1 功能部件限制

对于 `jpa-2.1` 功能部件，通过 CORBA/RMI-IIOP 进行的 JPA 实体交换要求参与通信的双方必须启用完全相同的 JPA 功能部件级别。

jsp-2.2 功能部件限制

对于 `jsp-2.2` 功能部件，存在下列限制：

- 不支持将转换的 JSP 文件仅存储在内存中的 `useInMemory` 配置选项。

monitor-1.0 功能部件限制

对于 `monitor-1.0` 功能部件，存在下列限制：

- 从 server.xml 文件中移除该功能部件之后，必须重新启动服务器以使 JAX-WS 应用程序正常运行。

requestTiming-1.0 功能部件限制

对于 `requestTiming-1.0` 功能部件，以下限制适用：

- 使用 DayTrader 应用程序进行度量时，显示已激活的 `requestTiming-1.0` 功能部件对可能的最大应用程序吞吐量的影响百分比为 4%。虽然对您的应用程序的影响可能高于或低于此比例，但您还是应注意某些性能下降可能是显而易见的。

restConnector-1.0 功能部件限制

对于 restConnector-1.0 功能部件，以下限制适用：

- 对于 restConnector-1.0 功能部件或包含 restConnector-1.0 的任何功能部件（例如，collectiveMember-1.0 和 collectiveController-1.0）的用户，如果他们想要运行包含定制 JAXRS 2.0 运行时的应用程序，那么他们必须将 jaxrs-2.0 功能部件添加至该服务器。

scim-1.0 功能部件限制

以下限制适用于 scim-1.0 功能部件：

- 搜索 groups 时，不会检索 members 属性。
- 搜索 users 时，不会检索 users 的 groups 属性。
- 不能对 users 的 groups 属性设置规范类型 direct/indirect。
- 只能定义规范类型 work 的用户的一个 email 属性。
- 只能定义规范类型 work 的用户的一个 ims 属性。
- 不能设置或返回 SCIM 的扩展模式属性（例如，entitlements、roles 和 x509Certificates）。
- userName 属性不能与过滤器中的一些其他属性配合使用。
- 对于基本注册表和 SAF 注册表中的用户，只能设置 userName、displayName、id、schema、meta.location 和 groups。userName 和 displayName 将具有相同值。
- 基本注册表和 SAF 注册表的列示/查询的工作方式与 ldapRegistry 注册表不同。
- pr、gt、ge、lt、le、and、or 和 () 之类的运算符对基本注册表及 SAF 注册表不起作用。而且，对于基本注册表和 SAF 注册表，只能在过滤器中使用一个运算符。
- 基本注册表和 SAF 注册表是只读的。
- 创建 user 时，不能设置 groups 属性。

wmqJmsClient-1.1 功能部件限制

对于 wmqJmsClient-1.1 功能部件，存在下列限制：

- 在 Windows™ 环境变量中，必须将 PATH 变量手动设置为指向 xigemaMQ 安装 bin 目录。当应用程序使用“绑定”连接方式时，必须设置此 PATH 变量。
- wmqJmsClient-1.1 功能部件中不包括 xigemaMQ Java™ 类（通称为“基本 Java™”）。这包括在其他应用程序服务器的资源适配器中，但是建议不要将其用于 Java™ Enterprise Edition 环境中的基本 Java™ API。
- wmqJmsClient-1.1 功能部件不支持 xigemaMQ 资源适配器的 BINDINGS_THEN_CLIENT 传输类型。
- wmqJmsClient-1.1 功能部件不包括高级消息传递安全性 (AMS) 功能部件。

wmqJmsClient-2.0 功能部件限制

对于 wmqJmsClient-2.0 功能部件，存在下列限制：

- 在 Windows™ 环境变量中，必须将 PATH 变量手动设置为指向 xigemaMQ 安装 bin 目录。当应用程序使用“绑定”连接方式时，必须设置此 PATH 变量。
- wmqJmsClient-2.0 功能部件中未包括 xigemaMQ Java™ 类（通称为“基本 Java™”）。这包括在其他应用程序服务器的资源适配器中，但是建议不要将其用于 Java™ Enterprise Edition 环境中的基本 Java™ API。
- wmqJmsClient-2.0 功能部件不支持 xigemaMQ 资源适配器的 BINDINGS_THEN_CLIENT 传输类型。

concurrent-1.0 功能部件限制

对于 concurrent-1.0 功能部件，以下限制适用：

对于类型为 `securityContext` 的线程上下文，不会传播未使用 JAAS 登录模块添加的主体集中的任何定制信息。例如，如果提交者的主体集包含 TAI 添加的定制主体，那么所传播主体集不会包含此定制主体。

jacc-1.5 功能部件限制

对于 jacc-1.5 功能部件，以下配置被忽略：

- 应用程序 ear 文件的 ibm-application-bnd.xml 文件或 ibm-application-bnd.xmi 文件中的授权信息（authorizations 属性的 users 和 groups 属性）。
- server.xml 文件的授权信息（`application-bnd` 元素中的 security-role 属性的 user、group 和 special-subject 属性）。

消息

使用 xigemaAS 时，可能会遇到系统消息。每则消息都具有一个唯一消息标识，并且包含问题的说明以及可用来解决该问题的任何操作的详细信息。

xigemaAS 系统消息是从各种来源（包括应用程序服务器组件和应用程序）记录的。对于 xigemaAS，消息标识的长度是 10 个字符并且具有以下格式：

```
CWXXX9999X
```

其中：

CWXXX

一个五字符字母消息前缀，用于标识 xigemaAS 组件。

9999

一个四字符数字标识，用于标识该组件的特定消息。

X

这是一个可选的字母指示符，用于标识消息类型：I = 参考信息，W = 警告，E = 错误。

表 74: xigemaAS 消息的字母消息前缀

每个字母前缀与特定 xigemaAS 组件相关联。有些前缀会进一步划分，以便字母前缀内的数字范围与特定组件相关联。

xigemaAS 消息前缀	范围		**xigemaAS** 组件
CWIMK	0001-0500		LDAP 注册表联合（配置消息）
CWIML	0001-5000		LDAP 注册表联合（运行时消息）
CWLIB	0001-0100		事务
	CWWKC	0000-0250	注释扫描
CWWKE			核心内核及内核服务

xigemaAS 消息前缀	范围	**xigemaAS** 组件
	0001-0099	引导/启动程序
	0100-0199	服务实用程序
	0200-0299	位置服务
	0300-0399	文件安装服务
	0400-0499	FileMonitor 服务
	0500-0599	可扩展类扫描程序
CWWKF		功能部件管理器
CWWKG		配置管理器
CWWKJ		低干涉管理
CWWKL	0001-0100	类装入服务
CWWKM	0001-0050	工件 API 消息（容器工厂）
	0051-0100	重叠消息（所有实现）
	0101-0150	工件 API Zip 实现
	0151-0200	工件 API 文件实现
	0401-0450	自适应 API 实现消息（自适应模块工厂/适配器服务）
	1001-1100	松散归档 API（供基于 Eclipse 的工具选项使用，该选项直接从 Eclipse 工作空间运行应用程序[11]）。
CWWKN	0001-0100	JNDI 缺省名称空间
CWWKO		*CFW* 组件
	0000-0199	CFW
	0200-0399	TCP
	0400-0599	UDP
	0600-0799	bytebuffer
	0800-0899	SSL 通道 (SSL channel)
	0900-0999	SSL
CWWKS		安全性

[11] Eclipse 平台支持“虚拟”应用程序归档，其中，似乎在同一个归档文件中的文件集实际上会在整个 Eclipse 工作空间中散开。xigemaAS 称此为“松散归档”，并且使用该松散归档 API 的工具选项称为**直接从工作空间运行此应用程序**

xigemaAS 消息前缀	范围		**xigemaAS** 组件
	0000 序列		安全性*:* 常规消息
		0000-0099	安全性
		0900-0999	安全性快速入门
	1000 序列		安全性*:* 认证服务
		1000-1099	认证
		1100-1199	JAAS 认证
		1200-1299	TAI 认证
	2000 序列		安全性*:* 授权服务
		2000-2099	权限
		2100-2199	内置授权
		2900-2999	SAF 授权
	3000 序列		安全性*:* 注册服务
		3000-3099	注册表
		3100-3199	基本注册表
		3200-3299	LDAP 注册表
		3300-3399	基于文件的 (VMM) 注册表
		3800-3899	定制注册表
		3900-3999	SAF 注册表
	4000 序列		安全性*:* 令牌服务
		4000-4099	令牌服务
		4100-4199	LTPA
		4200-4299	Kerberos
		4300-4399	SPNEGO
	9000 序列		安全性*:* 协调程序
		9100-9199	Web 协调程序（公共代码）
		9200-9299	Web 应用程序协调程序
		9300-9399	Web 管理协调程序
CWWKT			HTTP 传输/分派器
CWWKW	0000-0099		JAX-WS 公共

xigemaAS 消息前缀	范围		**xigemaAS** 组件
	0100-0199		用于 webContainer 的 JAX-WS
	0200-0299		JAX-WS 安全性
	0300-0399		用于 Java™ EE 公共的 JAX-WS
CWWKX	*0000* 序列		*JMX*
		0001-0100	JMX 安全性
		0101-0200	JMX REST 连接器
		0201-0300	JMX REST 客户机
	1000 序列		管理中心
		1000-1899	管理中心
CWWKZ			应用程序
	0001-0100		应用程序管理器
	0101-0200		WAR
	0201-0300		WAB
	0301-0400		EBA

在 xigemaAS 上对 SIP 容器会话存储库进行故障诊断

对 SIP 容器会话存储库进行故障诊断时，您可能需要 SIP 会话详细信息以转储至指定跟踪文件。

可使用 SIP 会话内存转储实用程序来帮助调试有关 SIP 容器会话的问题。SIP 容器提供 SipContainerMBean 方法以对 SIP 容器执行一些可维护性类型操作，包括通过命令行启动服务器停顿。此任务描述如何使用 SipContainerMBean 方法转储 SIP 容器的内存中会话存储库中包含的 SIP 应用程序会话和 SIP 会话信息。您可以配置 SIPContainerMBean 方法以使用各种跟踪方法，从而指定要转储到指定跟踪文件的 SIP 会话详细信息。

启动会话转储方法后，缺省情况下有关会话的请求信息将显示在 console.log 文件中。还可将此信息发送至 setDumpMethod 方法上指定的预定义源。

可通过以下两种方式运行转储实用程序：简明和详细。如果使用简明会话转储方法，那么每次执行转储方法时仅显示会话标识。如果要使用详细会话转储方法，那么将执行以下操作：

- 每次执行转储方法时，将显示事务用户详细信息和 SIP 会话详细信息（如果存在）。
- 转储至跟踪文件的仅有属性是 JSR 289 规范允许公开的属性。
- 详细方法在跟踪文件中显示以下信息：appName、callID、对话状态、创建时间和属性名称。

对于每个 SIP 应用程序，都会显示跟踪输出；因此，所有 SIP 会话数据结构的排序在显示之前进行。SIPContainerMBean 转储工具在低优先级线程中运行，因此跟踪不影响生产服务器的整体系统的调用处理等待时间。

具有 SIP 会话的事务用户与没有 SipSession 对象的事务用户之间的转储有所区别。不再存在、不再有效或创建跟踪快照时存在的 SIP 会话也以界定方式包含在转储中。

在 xigemaAS 上，可通过两种方式调用 SIPContainerMBean 转储方法：

- 通过运行 `server dump` 命令
- 通过实现 Java™ 管理扩展 (JMX) 客户机，此客户机建立与 JMX 连接器的连接以调用这些方法

以下简明 SipContainerMBean 方法用于转储 SIP 会话标识。

表 75: 用于转储 SIP 会话信息的简明 SipContainerMBean 方法

一个包含两列的表，此表列示方法名称和描述

方法	描述
dumpAllSASIds()	显示所有 SIP 应用程序会话数和 SIP 应用程序会话标识。
dumpAllTUSipSessionIds()	显示事务用户 (TU) 内的事务用户数和 SIP 会话标识（如果存在）。

以下详细 SipContainerMBean 方法用于转储 SIP 会话详细信息。

表 76: 用于转储 SIP 会话信息的详细 SipContainerMBean 方法

一个包含两列的表，此表列示方法名称和描述

方法	描述
dumpAllSASDetails()	显示所有 SIP 应用程序会话数和 SIP 应用程序会话标识详细信息。
dumpAllTUSipSessionDetails()	显示事务用户 (TU) 内的事务用户数和 SIP 会话标识（如果存在）的详细信息。
dumpSASDetails(String sasId)	显示 `sasId` 参数指定的 SIP 应用程序会话的详细信息。
dumpSipSessionDetails(String sessionId)	显示 `sessionId` 参数指定的 SIP 会话的详细信息。

注：使用以下信息来帮助分析打印输出：

- 对于所有打印输出，第一行提供应用程序名称和一些记录。
- 输出之间的定界符是**制表符**。
- 会话属性之间的定界符为 ;（分号）。

- 使用 `server dump` 命令调用 SIPContainerMBean 方法。

  有关 `server dump` 命令的更多信息，请参阅从命令行生成 *xigemaAS* 服务器转储（见第 1124 页）。

  1. 在 server.xml 文件中的 `sipIntrospect` 元素上指定转储实用程序方式。
     - 对于简明方式，请指定以下代码：

```
<sipContainer>
     <sipIntrospect method="SUCCINCT"/>
</sipContainer>
```

     - 对于详细方式，请指定以下代码：

```
<sipContainer>
```

```
      <sipIntrospect method="VERBOSE"/>
</sipContainer>
```

2. 打开命令行并切换至 wlp/bin 目录。
3. 要生成 SIP 会话和 SIP 应用程序会话转储，请运行以下命令，其中 *server_name* 是 xigemaAS 服务器：

```
server dump server_name
```

如果未指定服务器名称，那么会使用 defaultServer。

可选择在 archive 参数上指定服务器转储程序包文件的名称：

```
server dump server_name --archive=package_file_name.dump.zip
```

运行该命令后，将生成包含以下文件的服务器转储程序包文件：

- SipContainerIntrospector.txt，它包含有关 SIP 应用程序会话和 SIP 会话的跟踪信息的请求级别
- 其他 *artifact*Introspector.txt 文件，它们提供服务器状态信息

SipContainerIntrospector.txt 包含以下信息，其中 dump.ids.test.app1 是应用程序名称，2 是会话数，local.##### 行为 SAS_id：

```
The description of this introspector:
SIP state details

Succinct dumping

dump.ids.test.app1   2
local.1347524282775_8
local.1347524282775_7

--- End of Dump ---
```

- 通过 JMX 连接器调用特定转储实用程序方法。

  有关更多信息，请参阅使用 *JMX* 来连接至 *xigemaAS*（见第 1178 页）和为 *xigemaAS* 开发 *JMX Java* 客户机（见第 1181 页）。

  以下 Java™ 代码示例调用 dumpAllSASIds 方法：

```
    System.setProperty("javax.net.ssl.trustStore", "..\\wlp\\usr\\servers\
\SIPServer\\resources\\security\\key.jks");
    System.setProperty("javax.net.ssl.trustStorePassword", "xigemaAS");

    //If the type of the trustStore is not jks, which is default,
    //set the type by using the following line.
    System.setProperty("javax.net.ssl.trustStoreType", "jks");

   try {
      HashMap<String, Object> environment = new HashMap<String, Object>();
      environment.put("jmx.remote.protocol.provider.pkgs",
"com.ibm.ws.jmx.connector.client");

environment.put("com.ibm.ws.jmx.connector.client.disableURLHostnameVerification",
Boolean.TRUE);
      environment.put(JMXConnector.CREDENTIALS, new String[] { "theUser",
"thePassword" });
      environment.put(ConnectorSettings.DISABLE_HOSTNAME_VERIFICATION,
true);
```

```
      JMXServiceURL url = new JMXServiceURL("service:jmx:rest://
localhost:9443/xigemaJMXConnectorREST");
      JMXConnector connector = JMXConnectorFactory.newJMXConnector(url,
 environment);
      connector.connect();
      MBeanServerConnection connection =
 connector.getMBeanServerConnection();

      // test dumping utility
      connection.invoke(new
 ObjectName("WebSphere:name=com.ibm.ws.sip.container.SipContainerMBean"),
 "dumpAllSASIds", null, null);
```

简明 SipContainerMBean 方法具有以下打印格式。

表 77: 简明 SipContainerMBean 方法打印格式

一个包含两列的表，此表列示方法名称及其打印格式的示例

方法	打印格式
dumpAllSASIds()	每行以 [SAS_ID] 格式显示 SIP 应用程序会话标识。 例如： `local.1347524282775_8`
dumpAllTUSipSessionIds()	每行以 [TU_ID] [hasSIPSession] [SipSessionId] 格式显示事务标识（如果事务具有 SIP 会话）和会话标识（如果存在） 例如： `local.1349965420866_1_0 true local.1349965 420866_1_0_1`

详细 SipContainerMBean 方法具有以下打印格式。

表 78: 详细 SipContainerMBean 方法打印格式

一个包含两列的表，此表列示方法名称及其打印格式的示例

方法	描述
dumpAllSASDetails()	每行以 [SAS_ID] [CreationTime] [attributes] 格式显示 SIP 应用程序会话标识、创建时间、SIP 会话属性 例如： `local.1348147884986_2 Sep 20,2012 16:31 Du mpSasDetailsAttr;`
dumpAllTUSipSessionDetails()	每行以 [TU_ID] [hasSIPSession] [SipSessionId] [Call-Id] [DialogState] [hasOutgoingTransact ion] [initialMethod] [SAS_ID] [CreationTime] [attributes] 格式显示事务标识（如果事务具有 SIP 会话）和会话详细信息（如果存在）

方法	描述
	例如： `local.1349965420866_1_0 true local.1349965 420866_1_0_1 8-8548@9.148.57.128 2 false I NVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAttr2;`

dumpSASDetails(String sasId)	只会显示格式为 `[TU_ID] [hasSIPSession] [SipSess ionId]` 的一行
	例如：
	`local.1349965420866_1_0 true local.1349965 420866_1_0_1`
	如果所请求会话不存在，那么将显示以下错误消息：
	`ERROR: Requested session <local.13499654208 66_1_0_1> does not exist.`

dumpSipSessionDetails(String sessionId)	只会显示格式为 `[SipSessionId] [Call-Id] [Dialog State] [hasOutgoingTransaction] [initialMeth od] [SAS_ID] [CreationTime] [attributes]` 的一行
	例如：
	`local.1349965420866_1_0_1 8-8548@9.148.57. 128 2 false INVITE local.1349965420866_1 Jan 24,2013 14:41 TestSSAttr1; TestSSAt tr2;`
	如果所请求会话不存在，那么将显示以下错误消息：
	`ERROR: Requested session <local.13499654208 66_1_0_1> does not exist.`

在 xigemaAS 上跟踪会话启动协议 (SIP) 容器

您可以立即开始或在下次服务器启动之后跟踪会话启动协议 (SIP) 容器。此跟踪将 SIP 事件的记录写入日志文件。

1. 在 server.xml 文件中，添加 `logging` 元素并通过设置 `traceSpecification="*=info:com.ibm.ws.sip=all"` 来对 SIP 容器指定跟踪。

```
<logging logDirectory="${server.config.dir}/logs"
 traceFileName="trace.log" traceSpecification="*=info:com.ibm.ws.sip=all"/
>
```

SIP 级别跟踪消息将显示在 *serverName*/logs/trace.log 中，其中 serverName 是运行要跟踪的 SIP 容器的应用程序服务器的特定实例的名称。这些消息包括应用程序装入事件以及执行语法分析和 SIP Servlet 调用时的 SIP 请求和响应。

xigemaAS 上的会话启动协议 (SIP) 二进制日志和跟踪扩展

二进制日志记录为开发者提供一种方法以向日志和跟踪记录添加扩展字段，并为您提供对应方法以按扩展值过滤日志和跟踪记录。

日志和跟踪记录包含的字段提供该记录的创建时间和所记录消息的内容之类的信息。这些字段是核心字段，每个日志和跟踪记录中都存在。相比之下，扩展字段是应用程序开发者可以添加到日志和跟踪记录的字段，在搜索特定日志和跟踪内容时，您可以将扩展字段用作过滤条件。将文本输出格式配置为使用高级格式时，这些日志和跟踪扩展在二进制日志中可视，您以高级格式使用 binaryLog 命令时，这些扩展也可视。

管理员

应用程序服务器将自动创建一些扩展，您可以使用这些扩展来过滤日志和跟踪记录。还可以使用由应用程序开发者添加的任何扩展来过滤日志和跟踪记录。可以使用 binaryLog 命令行工具根据日志和跟踪记录扩展的内容来过滤记录。有关更多信息，请参阅 *binaryLog 命令选项*（见第 1076 页）。

例如，要查看 SIP 容器处理的所有 SIP 应用程序会话，可使用以下 binaryLog 命令：

```
binaryLog view binaryFile --includeExtension=SIPASId=* --format=advanced
```

开发者

开发者可以使用二进制日志记录以通过日志记录上下文 API（即 com.ibm.websphere.logging.hpel.LogRecordContext）向日志和跟踪记录添加定制扩展。二进制日志记录存储日志和跟踪记录时，它会将存在于日志记录上下文中的任何扩展包括在同一线程上。例如，您可以编写 servlet 过滤器以将重要 HTTP 请求参数添加到日志记录上下文。该 Servlet 运行时，HPEL API 会向在同一线程上创建的任何日志和跟踪记录添加这些扩展。

与其他日志和跟踪记录字段一样，开发者可以使用 HPEL API 来访问记录扩展。编写工具以从日志和跟踪存储库读取时，此 API 很有用。开发者还可以在运行时使用日志记录上下文 API 来访问定制日志处理程序、过滤器和格式化程序中的扩展。

下表描述了日志和跟踪扩展，包括您可用于过滤跟踪的各个方面的标识。

表 79: 日志和跟踪扩展

一个包含两列的表，此表列示二进制日志记录的日志和跟踪扩展。

扩展	描述
appName	指定与日志或跟踪记录相关的 Java™ Platform Enterprise Edition (Java™ EE) 应用程序（如果存在）的名称。
requestID	指定与每个日志或跟踪记录相关的请求（如果存在）的唯一标识。要使应用程序服务器能够将 requestID 扩展添加到日志和跟踪记录，必须启用跨组件跟踪 (XCT)，在管理控制台中也称为“日志和跟踪关联”。仅为某些类型的请求添加请求标识，如 HTTP 或 JMS 请求。
SIPCallId	指定 SIP 代理服务器或 SIP 容器正在处理的 SIP 调用标识。此信息在各个 SIP 代理服务器和 SIP 容器中是相同的。可以使用此扩展在各种组件之间跟

扩展	描述
	踪 SIP 调用流。启用 HPEL 记录后，SIP 代理服务器和 SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPASId	指定 SIP 容器要处理的 SIP 应用程序会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPSessionId	指定 SIP 容器要处理的 SIP 会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。
SIPCallId2	指定与同一个 SIP 应用程序会话相关联并且 SIP 容器正在处理的第二个 SIP 调用标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。 如果有两个以上的 SIP 调用标识与单个 SIP 应用程序会话相关联，那么仅记录前两个标识。不会记录其他标识。
SIPSessionId2	指定与同一个 SIP 应用程序会话相关联并且 SIP 容器正在处理的第二个 SIP 会话标识。此信息在各个 SIP 容器中是相同的。可以使用此扩展跟踪 SIP 调用流。启用 HPEL 记录后，SIP 容器会自动将此标识添加到每个日志和跟踪记录。 如果有两个以上的 SIP 会话标识与单个 SIP 应用程序会话相关联，那么仅记录前两个标识。不会记录其他标识。
thread	指定每个日志或跟踪记录的相关请求的线程名称。

3 欢迎使用xigemaAS 管理中心

3.1 xigemaAS 管理中心概述

xigemaAS 管理中心 (xigemaAS Management Center， xigemaAS MC) 是对 xigemaAS 服务器和 Web 服务器进行集中管理的 B/S 图形化工具，使用户能够通过一台安装了 xigemaAS 管理中心的主机集中管理本地或远端的服务器和集群，以及部署于服务器和集群上的应用，实时监控服务器和应用的状态。提供多种灵活的服务，并简化管理，有助于用户提高工作效率、降低业务风险、保障业务安全。

xigemaAS 管理中心提供对以下功能模块的管理：节点管理、服务器管理（包括应用服务器管理和 Web 服务器管理）、集群管理、应用管理、资源管理（包括共享库管理、连接池管理、数据源管理和 JMS 管理）、系统管理（包括参数配置和数据备份）。

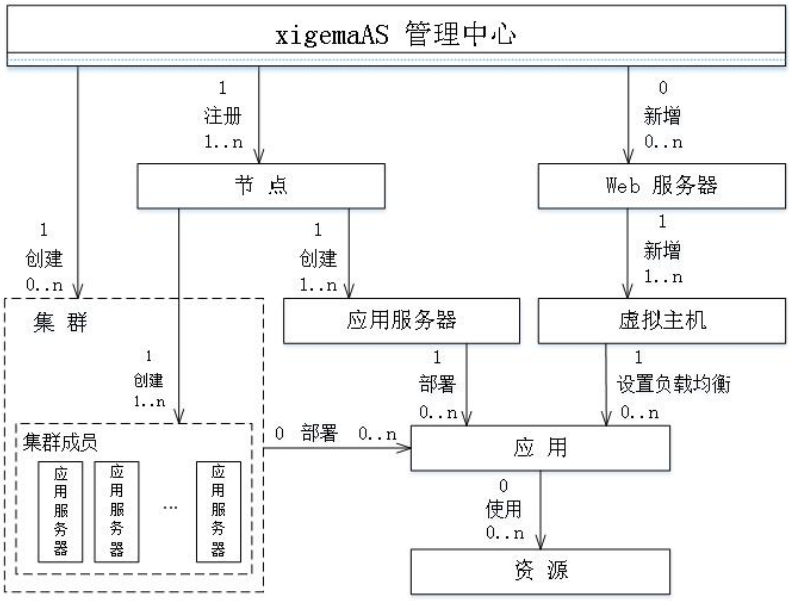
xigemaAS 管理中心的主要优势包括：

- 用户可以集中管理节点、服务器、集群、应用和资源，无需登录多台服务器对其分别进行管理。同时，通过集群管理功能，可以直接创建包含多个相同配置应用服务器的集群，并针对该集群进行应用部署等操作，无需手动定义和配置大量服务器，从而大幅提高用户的工作效率。
- 支持服务器指标监控，并提供监控指标定制，监控视图样式自主选择的个性化监控功能，帮助确保服务器最佳性能，显著降低业务风险。
- 使用标签定义节点、服务器、集群、应用和资源的可识别属性，利用标签对这些对象进行分组管理，便于用户进行检索，减少操作复杂度。
- 提供列表定制功能，用户可以根据需要自定义列表项，直观展示不同用户关注的不同信息，提升用户体验好感度。

关于如何安装 xigemaAS 管理中心，请参见 xigemaAS 手册的安装内容。

3.1.1 概念体系

本节将介绍 xigemaAS 管理中心的关键概念及其相互关系。



xigemaAS 管理中心

集中管理 xigemaAS 服务器和 Web 服务器的 B/S 图形化工具。能够集中式地管理已注册的节点、创建于节点上的服务器和集群、部署到服务器和集群上的应用、应用使用的资源，以及用来配置负载均衡的 Web 服务器和虚拟主机。

主机

物理计算机或虚拟机。主机是各节点的实际载体。

节点

主机上的一份 xigemaAS 物理安装。可以在节点上创建一个或多个 xigemaAS 服务器。

👉 注：节点由主机、用户和 xigemaAS 安装目录确定。

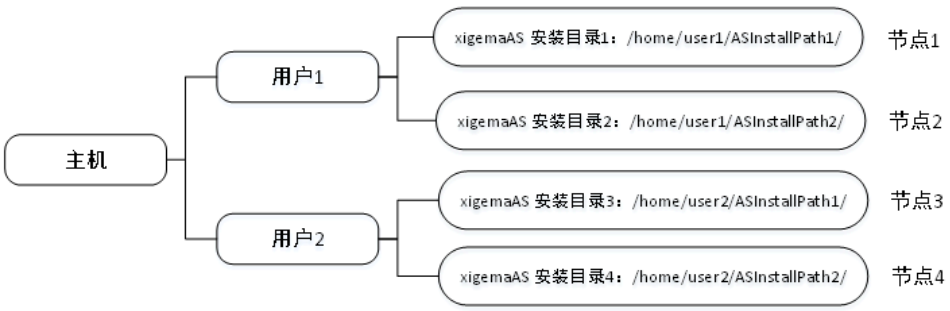


图 46: 主机、用户、xigemaAS 安装目录和节点的概念关系图

一个主机上可以存在多个用户，每个用户可以存在多个 xigemaAS 安装目录。一个主机、一个用户和一个 xigemaAS 安装目录唯一确定一个节点。

本机节点

在 xigemaAS 管理中心所在主机上注册的节点。

远程节点

在远程主机上注册的节点，注册时需远程认证。

应用服务器

创建在节点上的 xigemaAS 服务器实例。可以在应用服务器上部署一个或多个应用。

Web 服务器

已经安装好的 Web 服务器可以添加到 xigemaAS 管理中心进行管理。一个 Web 服务器只能由一个 xigemaAS 管理中心进行管理。

虚拟主机

可以在 Web 服务器上新增多个虚拟主机。一个虚拟主机可以为多个应用设置负载均衡，一个应用也可以使用多个虚拟主机进行负载配置。

集群

为提升业务系统的处理能力，而组成的多个 xigemaAS 服务器实例的集合，xigemaAS 集群中应至少存在一个集群成员。可以在集群上部署一个或多个应用，集群中所有集群成员部署的应用必须相同。

集群成员

从属于集群的 xigemaAS 服务器实例。创建在节点上的服务器实例。一个集群成员只能从属于一个集群，同一个集群中的集群成员，可以创建于不同节点，但部署的应用必须相同。

应用

部署在服务器或集群中的符合规范的应用程序，xigemaAS 管理中心支持 Web 应用、EJB 应用、企业应用、OSGi 应用和连接器应用。

资源

应用程序运行时需要的各类资源，包括共享库、连接池、数据源和 JMS 资源。通过 xigemaAS 管理中心可以对各类资源进行管理和配置。

3.1.2 部署结构

通过 xigemaAS 管理中心能够对本地或远程主机上的对象进行管理，包括对节点、服务器、集群、应用和资源五个部分的管理。

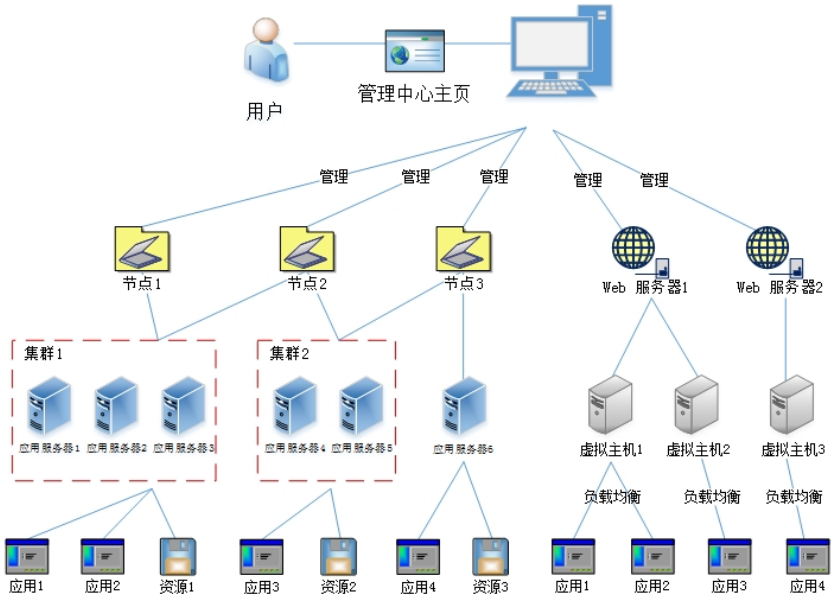


图 47: xigemaAS 管理中心部署结构

xigemaAS 管理中心采用 B/S 模式实现对 xigemaAS 服务器的集中管理。用户登录管理中心主页，对已注册的本地或远程节点上的服务器和应用等进行管理。

节点、服务器和应用的物理载体都是主机。一台主机的一个用户可以注册多个节点。每个节点可以创建多个 xigemaAS 服务器，各应用和资源部署于应用服务器之上，由 Web 服务器上的虚拟主机来为应用设置负载均衡。一个 Web 服务器上可以创建多个虚拟主机，一个虚拟主机可以为多个应用设置负载均衡，一个应用也可以使用多个虚拟主机进行负载配置。集群中的集群成员可以是不同节点上的应用服务器，但同一集群内的集群成员必须部署相同的应用。应用可以使用多个资源，资源也可以被多个应用使用。xigemaAS 管理中心可以对各服务器和部署于其上的应用进行状态监控和动态配置。

3.1.3 实现机制

xigemaAS 基于 SSH 协议、HTTP 协议与节点进行通信。使用 xigemaAS 管理中心创建应用服务器时，xigemaAS 管理中心的环境信息会通过 SSH 协议推送到应用服务器端。应用服务器使用 HTTP 协议，从 xigemaAS 管理中心下载资源。xigemaAS 管理中心采用 Derby 数据库存储节点、服务器、应用的基本信息以及服务器的配置信息，使用资源仓库管理 xigemaAS 安装介质、应用文件以及服务器配置信息等资源。

节点通信

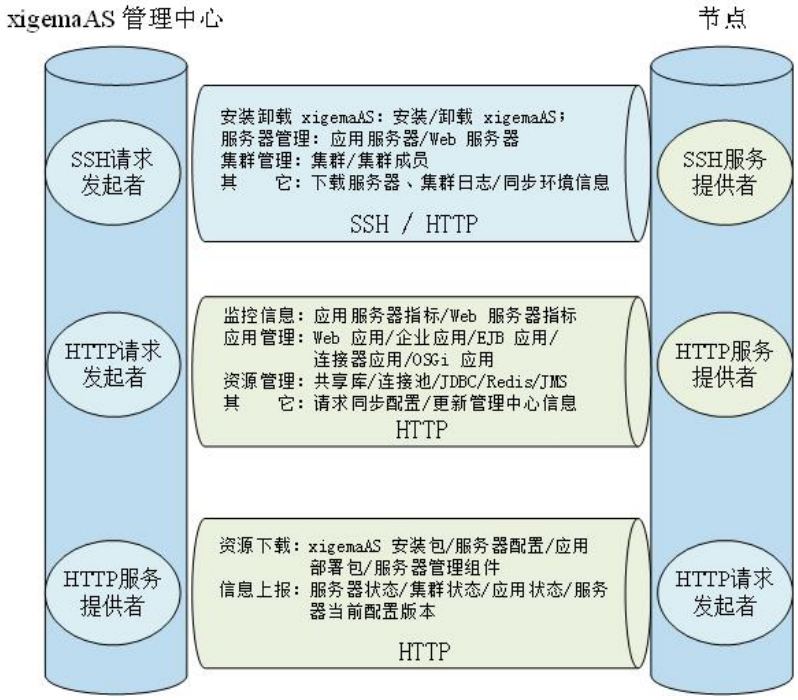


图 48: xigemaAS 管理中心通信机制

xigemaAS 管理中心在被管理的节点上安装节点管理组件 toolkit-*${version}*.jar，以辅助管理节点和服务器。该组件保存在 xigemaAS 管理中心的资源仓库中。在用户注册节点时，xigemaAS 管理中心会通过 SSH 协议登录到节点上。节点通过 `curl` 命令从 xigemaAS 管理中心下载该组件，并保存到节点用户主目录 .xigemaAS/runtime/toolkit/1.0 下。

xigemaAS 管理中心通过 RESTful HTTP 与被管理的应用服务器进行通信。xigemaAS 管理中心与应用服务器之间通过 HTTP 协议实现以下管理功能：

- 获取监控信息：xigemaAS 管理中心通过定期访问应用服务器的 REST 服务，获取内存、CPU、类、JVM 线程、应用会话等监控信息；
- 应用管理：通过访问服务器相应 REST 服务来实现。

由应用服务器作为 HTTP 请求发起者，xigemaAS 管理中心响应，完成如下业务功能：

- 下载 xigemaAS 服务器管理组件 (toolkit-*${version}*.jar) ；
- 从 xigemaAS 管理中心下载并部署应用；
- 定期向 xigemaAS 管理中心汇报应用服务器及部署在其上应用的当前运行状态；
- 定期向 xigemaAS 管理中心汇报当前应用服务器的配置版本。

节点代理

xigemaAS 支持以下几种代理类型：

- 本机：在 xigemaAS 管理中心所在主机上注册节点，管理应用服务器以及 Web 服务器。
- SSH：使用 SSH 代理类型需要在节点所在主机（一般为远程主机）上安装并启用 SSH 服务。通过 xigemaAS 管理中心的节点注册功能，可以将 SSH 认证信息登记到 xigemaAS，然后管理中心通过 SSH 协议登录到远程主机，执行应用服务器和 Web 服务器的管理操作。

- Agent：xigemaAS 提供的代理服务组件，实现了 HTTP 服务。从 xigemaAS 管理中心可以下载该组件，将其安装到远程主机，配置并启动后，可以自动向管理中心注册一条节点信息，然后可以基于该节点执行应用服务器和 Web 服务器的管理操作。

配置应用服务器

应用服务器包含四种配置文件：server.xml、bootstrap.properties、server.env 和 jvm.options。用户使用 xigemaAS 管理中心创建或者修改应用服务器配置信息时，实际上是在修改这四个文件，并将修改信息保存在 xigemaAS 管理中心端；但应用服务器的基本信息（如标签、描述等）数据保存在数据库中，所以在修改基本信息时不会修改上述文件。

1. server.xml：用来配置 xigemaAS 服务器的核心配置文件，可以在此文件中声明需要载入的功能部件。

   注：开启 xigemaAS 管理中心对 xigemaAS 服务器的管理功能，需在被管理 xigemaAS 服务器的 server.xml 文件中配置 `mcMember-1.0` 功能部件，配置片段如下：

   ```
   <featureManager>
   <!--mcMember-1.0不可被删除，否则MC管理中心将无法管理此服务器 -->
         <feature>mcMember-1.0</feature>
   </featureManager>
   ```

   mcMember-1.0 关联了如下功能部件：

   `servlet-3.0、jsp-2.2、ssl-1.0、json-1.0、restConnector-1.0、jaxrs-1.1、jndi-1.0、di`

   若配置了 `mcMember-1.0`，启动 xigemaAS 服务器时将会自动开启上述所有功能部件。
2. bootstrap.properties：以键值的方式保存在 xigemaAS 服务器的 server.xml 中或运行过程中会使用的系统变量。
3. server.env：保存 xigemaAS 服务器的环境变量，如 JAVA_HOME；
4. jvm.options：保存 JVM 的配置信息。

上述文件均位于 *${server.config.dir}* 目录下。

应用服务器信息同步

环境信息同步：xigemaAS 管理中心在创建服务器时，通过 SSH 协议将环境信息推送到应用服务器端。 当 xigemaAS 管理中心的 IP 地址或管理员账户信息被修改时，会重新生成环境信息并推送到应用服务器端。

配置信息同步：对应用服务器任一配置文件的修改，均需重启应用服务器才能使之生效。应用服务器每次启动都会自动下载最新的配置文件到相应目录，并依据配置信息执行启动操作。若由于网络等原因下载失败，应用服务器仍能启动；但启动完毕后，会定期向 xigemaAS 管理中心汇报当前应用服务器的各个配置文件的版本。xigemaAS 管理中心收到配置文件版本后，会与存储的配置文件版本进行比较。若不一致，则会生成**"服务器配置版本与 `xigemaAS` 管理中心版本不一致"**的警告消息。

服务器状态监控

xigemaAS 管理中心通过各服务器定期向其汇报当前状态。服务器启动时，将启动一个定时任务，周期性的向 xigemaAS 管理中心汇报其状态，以及部署在应用服务器上的各应用的当前状态。xigemaAS 管理中心收到服务器汇报的状态后，会及时更新状态信息。服务器停止时，xigemaAS 管理中心收到服务器停止的汇报消息后，更新服务器状态为"停止"，并更新部署在该应用服务器上的应用状态为"停止"。此外，xigemaAS 管理中心会定期扫描处于启动状态的服务器的 HTTP 端口，若端口不可达，则设置相应服务器及部署在其上的应用状态为"未知"。有关服务器状态的更多信息，请参见应用服务器状态（见第 1887 页）。

集群、集群成员配置文件

xigemaAS 管理中心为集群保存四个配置文件：server.xml、bootstrap.properties、server.env 和 jvm.options；为集群成员保存 bootstrap.properties、server.env 和 jvm.options 三个配置文件。集群成员的 server.xml 配置文件与所属集群的相同，本质上是集群 server.xml 配置文件的拷贝，由所属集群统一维护。而 bootstrap.properties、server.env 和 jvm.options 配置文件则由各个集群成员独自维护。创建集群时，若所选模板为已创建的服务器或集群，则集群成员的初始配置文件 bootstrap.properties、server.env 和 jvm.options 与所选模板的配置文件相同。

集群、集群成员信息同步

与服务器相同，集群和集群成员要向 xigemaAS 管理中心汇报其状态信息，同时，对集群任一配置文件的修改，均需重启集群才能使之生效。集群、集群成员环境信息、配置信息的同步原理，与服务器信息同步的原理基本相同。区别在于，将对集群成员汇报的 server.xml 配置文件的版本与所属集群 server.xml 配置文件的版本进行比较；对集群成员汇报的 bootstrap.properties、server.env 和 jvm.options 配置文件的版本与集群成员自身同类型配置文件的版本进行比较，比较后，向 xigemaAS 管理中心汇报或推送相关信息。

配置负载均衡

将已安装好的 Web 服务器添加到 xigemaAS 管理中心后，用户可以通过 xigemaAS 管理中心设置负载均衡。负载均衡的设置，主要通过 xigemaAS 管理中心远程修改配置文件来实现。

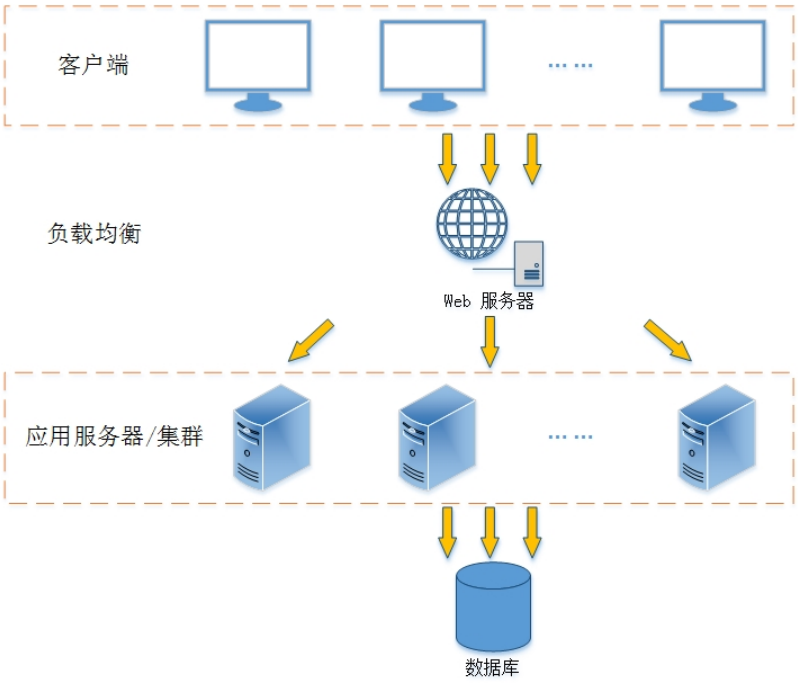


图 49: 负载均衡架构

xigemaAS 管理中心支持的 Web 服务器包括 Apache 和 Nginx。

通过执行以下命令，实现 xigemaAS 管理中心对 Windows 平台上 Apache 的管理。其中，部分常用命令如下：

```
注册 Windows 服务      httpd.exe -k uninstall  [-n windows服务名称]
启动     httpd.exe -k start -n windows服务名称
停止     httpd.exe -k stop/shutdown -n windows服务名称
平滑重启     httpd.exe -k restart -n windows服务名称
检查配置文件是否有效     httpd.exe -t
获取服务器版本     httpd.exe -v
```

通过执行以下命令，实现 xigemaAS 管理中心对 Linux 平台上 Apache 的管理。其中，部分常用命令如下：

```
启动     ./apachectl start
停止     ./apachectl stop
平滑停止     ./apachectl graceful-stop
重启    ./apachectl restart
平滑重启     ./apachectl graceful
检查配置文件是否有效     ./apachectl -t
获取服务器版本     ./apachectl -v
```

通过 xigemaAS 管理中心配置了应用的 Apache 负载均衡，会自动在 httpd.conf 配置文件中生成以下配置片段：

```
#other configurations
Listen 172.16.173.20:8889
<VirtualHost 172.16.173.20:8889>
    #virtual_host_name:outsideVirtualHost-1 (Warning: This comment is used
 by xigemaAS Management Center, please do not modify this line.)
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass /demoAppWebServerContextRoot/balancer://balancer_demoApp/
demoAppContextRoot/
    ProxyPassReverse /demoAppWebServerContextRoot/balancer://
balancer_demoApp/demoAppContextRoot/
</VirtualHost>
<Proxy balancer://balancer_demoApp>
    BalancerMember http://172.16.173.19:9080
    BalancerMember http://172.16.173.20:9080
</Proxy>
#other configurations
```

通过执行以下命令，实现 xigemaAS 管理中心对 Linux 平台上 Nginx 的管理。其中，部分常用命令如下：

```
启动     ./nginx
停止     ./nginx -s stop
重启     ./nginx -s reload
检查配置文件是否有效     ./nginx -t
获取服务器版本和配置信息     ./nginx -V
```

通过 xigemaAS 管理中心配置了应用的 Nginx 负载均衡，会自动在 nginx.conf 配置文件中生成以下配置片段：

```
#other configurations
```

```
http{
#other configurations
   server{
        #server_block_name:defaultVirtualHost  (Warning:this comment is used
 by xigemaAS Management Center, please do not modify this line.)
        listen 8081;
        location /demoApp/ {
           proxy_pass  http://up4demoApp#1/demoApp/;
        }
      }
     upstream up4demoApp#1 {
       server 172.16.173.19:9080;
       server 172.16.173.20:9080; }
#other configurations
}
#other configurations
```

应用部署

用户通过 xigemaAS 管理中心部署的所有应用均存储在 xigemaAS 管理中心的资源仓库中。应用服务器通过 RESTful HTTP 从 xigemaAS 管理中心下载并部署应用。通过 xigemaAS 管理中心在应用服务器上部署、卸载和编辑应用，实质上是修改应用服务器的 server.xml 配置文件，在该文件中添加、卸载或编辑应用的配置片段。其中，location 属性的值为应用在 xigemaAS 管理中心保存下载地址。修改完毕的 `server.xml` 配置文件将保存在 xigemaAS 管理中心。应用服务器下载配置文件后，将自动对配置文件的修改进行检测，然后根据检测结果自动部署、卸载或更新应用。对未发生变更的配置，服务器不会做任何处理，因此不会对其它配置片段造成影响。

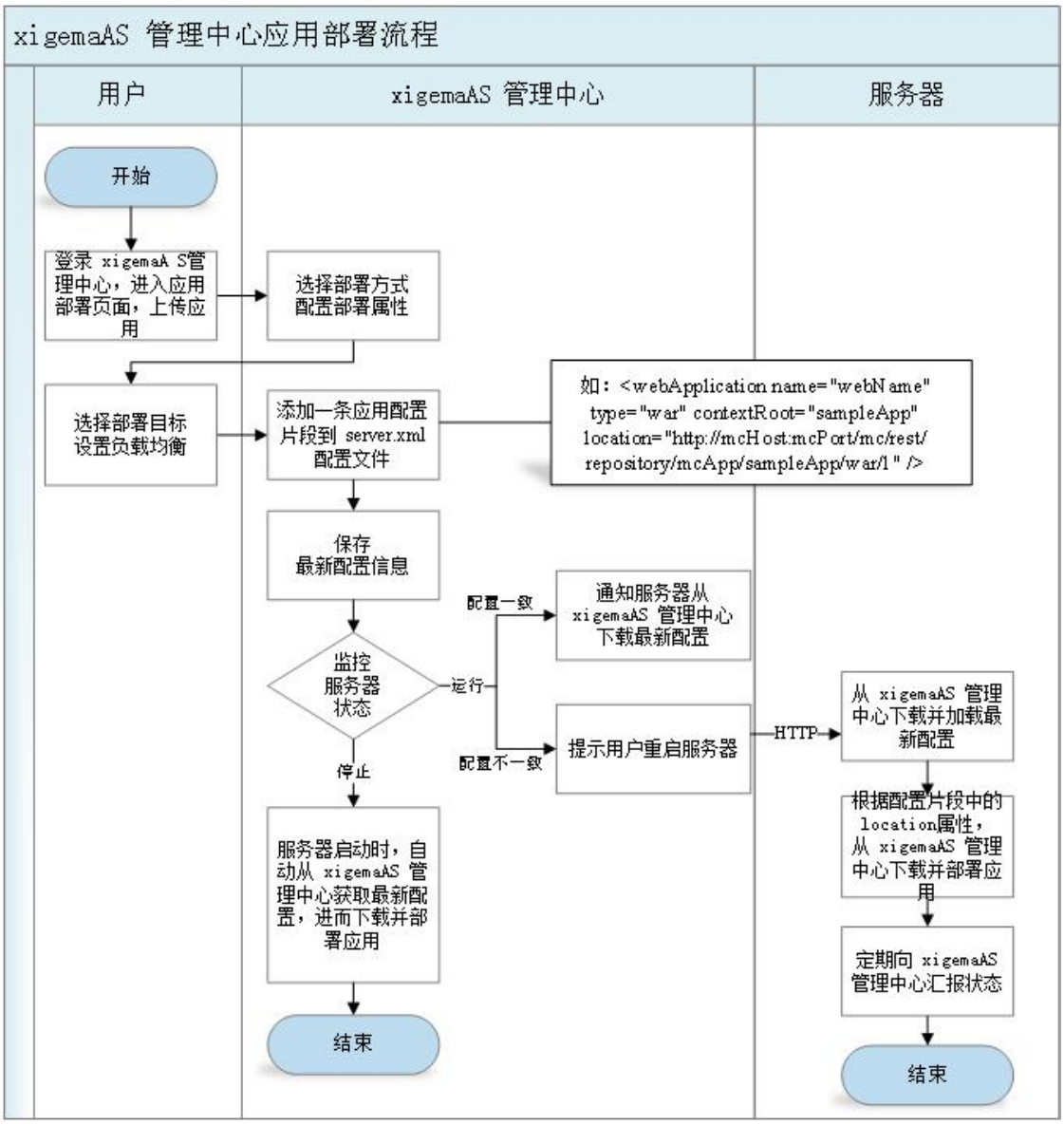


图 50: 应用部署业务流程图

资源管理

资源管理包括对共享库、连接池和数据源的管理，用户通过 xigemaAS 管理中心管理的所有资源均存储在 xigemaAS 管理中心的资源仓库中。通过 xigemaAS 管理中心在应用服务器上新增、删除和编辑资源，实质上是修改应用服务器的 server.xml 配置文件：在该文件中添加、卸载或编辑资源的配置片段。修改完毕的应用服务器 `server.xml` 配置文件将保存在 xigemaAS 管理中心。

3.2 快速入门

登录 xigemaAS 管理中心后，可以对节点、服务器、集群、应用以及资源进行集中管理。

xigemaAS 管理中心的业务流程如下图所示：

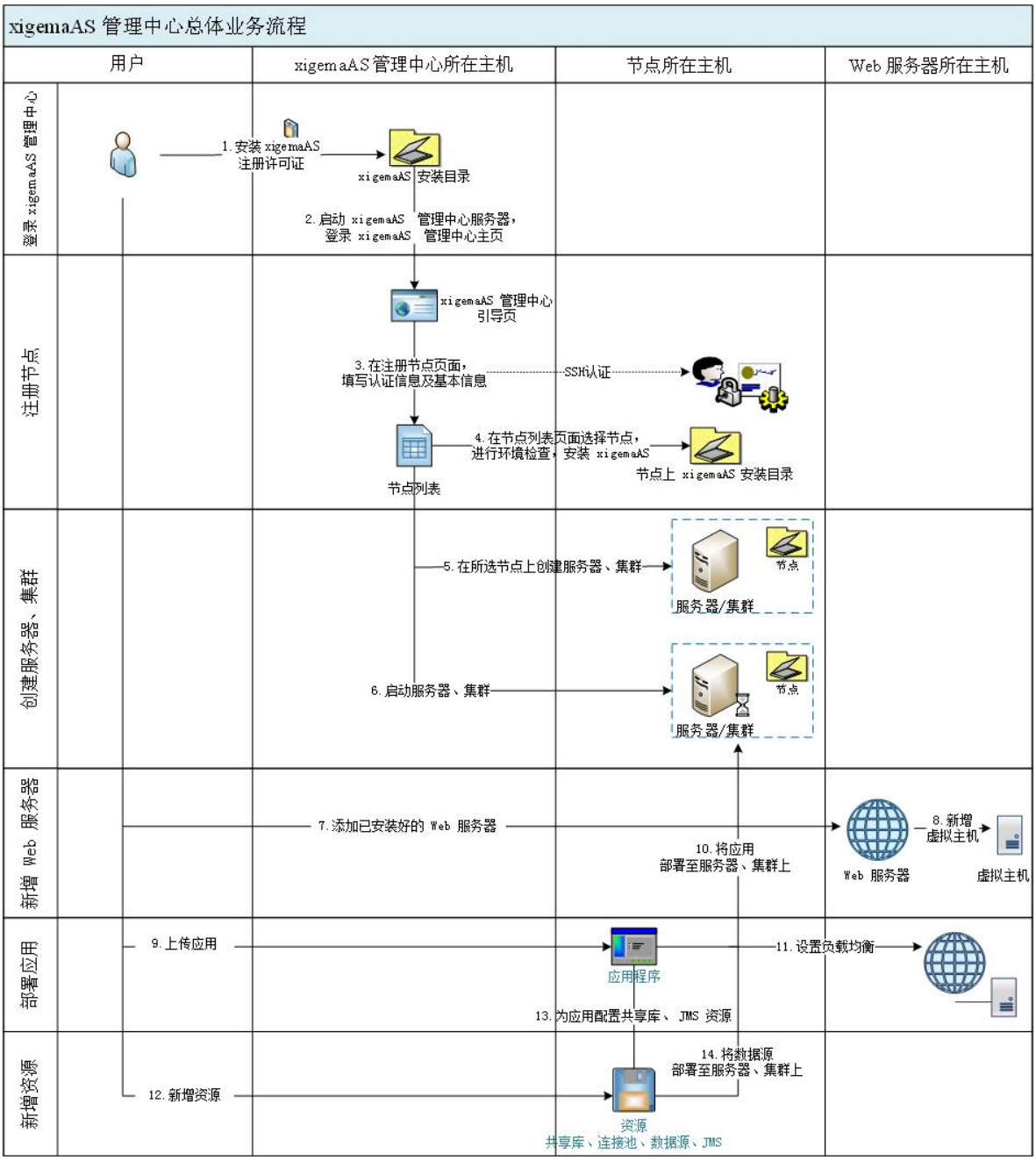


图 51: xigemaAS 管理中心业务流程图

3.2.1 启动 xigemaAS 管理中心

安装成功后，可以从命令行窗口启动 xigemaAS 管理中心。

在启动 xigemaAS 管理中心 之前，必须已成功安装 xigemaAS。

- 进入 xigemaAS 的 wlp/bin 目录。

类 UNIX 系统环境下执行以下命令启动 xigemaAS 管理中心：

```
./server run mc
```

或

```
./server start mc
```

Windows 系统环境下执行以下命令启动 xigemaAS 管理中心：

```
server run mc
```

或

```
server start mc
```

xigemaAS 管理中心启动成功，如下图所示：

若命令窗口显示"**服务器 mc 已准备就绪，可开始运行 xigemaAS**"，则说明xigemaAS 管理中心已启动成功。

3.2.2 登录 xigemaAS 管理中心

启动成功后，可以从浏览器登录 xigemaAS 管理中心。

表 80: xigemaAS 管理中心支持以下浏览器

浏览器			
xigemaAS 管理中心支持的最低版本	Internet Explorer 10	Google Chrome 39.0	Firefox 38.0.5

若使用其它浏览器登录 xigemaAS 管理中心，请确保该浏览器支持 Websocket 协议。

- 访问 xigemaAS 管理中心，在浏览器地址栏输入 `http://hostname:port/mc` 进入 xigemaAS 管理中心登录页面。首次登录需要输入默认的用户名和密码 (admin/admin)。

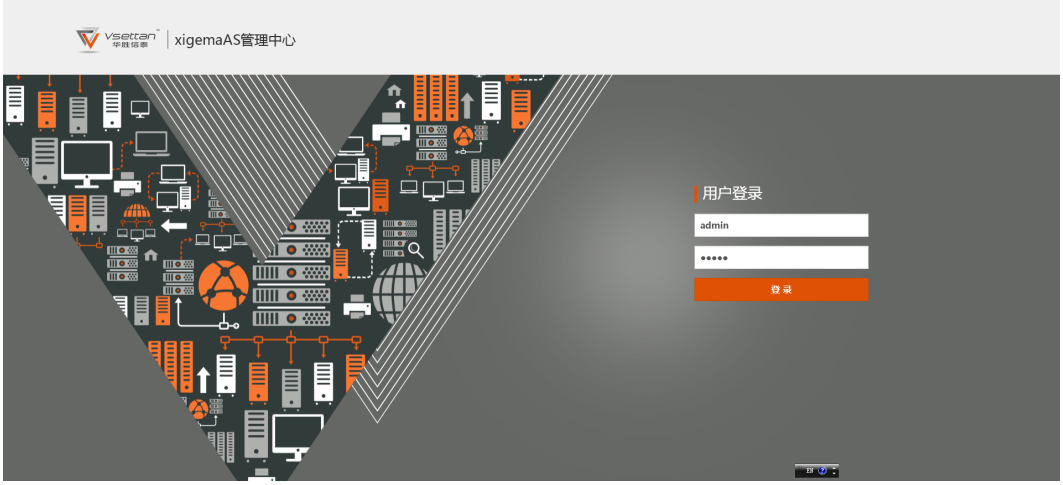


图 52: xigemaAS 管理中心登录页面

注：HTTP 端口号默认值为9060，HTTPS 端口号默认值为9043。

- 首次登录 xigemaAS 管理中心，登录后弹出 xigemaAS 管理中心所在主机 IP 地址配置引导页面。由 xigemaAS 管理中心管理的所有服务器通过在此配置的 IP 地址进行通信。配置完成后，进入 xigemaAS 管理中心。

图 53: xigemaAS 管理中心 IP 配置引导页面

注：进入 xigemaAS 管理中心后，若需要变更 xigemaAS 管理中心 IP 地址，可以单击**系统管理 > 参数配置**，进入参数配置页面变更 IP 地址。详细操作和注意事项请参阅参数配置（见第 1867 页）。

3.2.3 注册节点

注册节点是使用 xigemaAS 管理中心进行节点管理，以及其他模块管理的第一步。

请确认已满足以下前置条件：

- 若使用 xigemaAS 管理中心选择 SSH 代理方式注册节点，则需要满足以下环境要求：
  - 若使用证书认证方式，请确保已为 xigemaAS 管理中心生成了 SSH 公、私钥证书文件，并已将 SSH 公钥证书文件添加到待注册节点的受信文件中；
  - 已安装 curl 工具，并且当前 SSH 用户具备 curl 命令的执行权限；

- xigemaAS 管理中心与当前节点所在主机的网络连通。

- 若使用 xigemaAS 管理中心选择 Agent 代理方式注册节点，详细信息和具体操作请参阅安装 *Agent*（见第 1820 页）。
- 注册本机节点，节点所在主机可以是 Windows 系统或类 UNIX 系统。

当所有前置条件已满足，请在已登录 xigemaAS 管理中心主页状态下执行以下操作注册节点。

1. 单击**节点 > 注册**，进入注册节点页面。
2. 通过 xigemaAS 管理中心可以**“创建新的认证”**或**“使用已有认证”**注册节点。若使用已有认证信息注册节点，可以在**“认证信息”**下拉列表中选择已注册的节点，选择完成后，当前节点将直接使用已选节点的认证信息；若选择创建新的认证信息，请执行以下操作注册节点。

   xigemaAS 管理中心支持本机节点和远程节点的注册。

   a. **远程节点**：从**“代理类型”**栏选中 **SSH** 项，在远程主机上注册节点。xigemaAS 管理中心默认注册远程节点，须填写认证信息和基本信息。

      - 在**“认证信息”**面板填写以下信息：

图 54: 认证信息面板

      - 选择并填写节点所在主机的计算机名，或 IPv4 地址。
      - SSH 用户名，即登录节点所在主机的用户名。
      - SSH 端口号，即节点所在主机用于 SSH 通信的端口号，默认值为22。
      - 选择认证方式。默认选择“用户名/密码”方式认证，需要填写登录节点所在主机使用的密码。若选择“证书”方式认证，需要填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径。
      - 在**“基本信息”**面板填写以下信息：

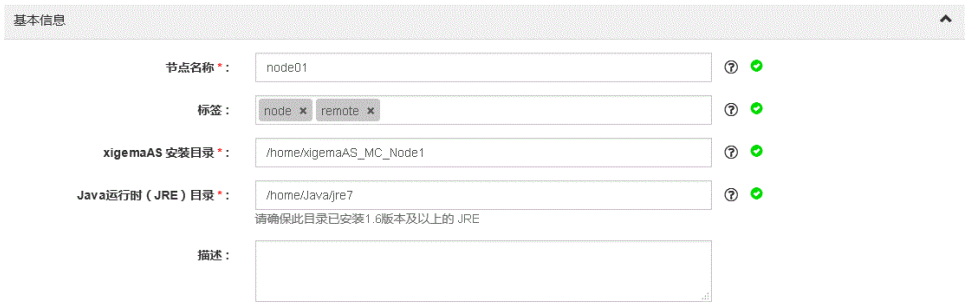


图 55: 基本信息面板

- 节点名称。节点名称是节点在 xigemaAS 管理中心的唯一标识，保存后不可修改。
- xigemaAS 安装目录。
- Java 运行时（JRE）目录。

  注：保存后，在该节点下创建的服务器，将使用该目录作为其环境变量 JAVA_HOME 的值。修改此值不会修改已创建的服务器的 JAVA_HOME，但会作为此后创建服务器的 JAVA_HOME 路径。
- 标签及描述，用于标识和分类管理节点。

b. **本机节点**：从**“代理类型”**栏选择**本机**项，在**“基本信息”**面板填写基本信息。注册 xigemaAS 管理中心所在主机上的节点，无需进行远程认证。

3. 单击**保存**按钮，保存节点注册信息。若提示 **“保存成功”** ，说明该节点注册成功。若存在无效信息，则无法保存。请参阅注册节点（见第 1882 页）中界面元素的含义描述和操作规范，修改相应的节点信息，再进行保存操作以完成节点的注册。

   节点注册成功后，可继续注册新的节点或返回节点列表查看已注册节点。在节点列表中通过节点名称、主机名、xigemaAS 安装状态、标签可以查询已注册的节点。

3.2.4 在节点上安装 xigemaAS

节点注册完成后，需要在节点上安装 xigemaAS，安装成功后方可创建服务器。

1. 单击**节点**，进入节点管理页面。
2. 在节点列表中选择要安装 xigemaAS 的节点，单击**安装 AS** 按钮。提示“xigemaAS **安装成功**”，则该节点 xigemaAS 安装状态更新为“已安装”。

   安装过程中，xigemaAS 管理中心自动进行该节点的环境检查。环境检查是否通过，直接影响该节点的 xigemaAS 能否安装成功。

   也可以在安装 xigemaAS 前，单击**环境检查**按钮以手动进行环境检查。有关环境检查的具体检查内容和操作请参阅节点环境检查（见第 1819 页）。

   xigemaAS 管理中心支持批量安装 xigemaAS，同时在安装过程中提供安装详情提示，以查看当前操作进度及详情。

   注：若所选节点上已安装 xigemaAS 、正在安装 xigemaAS、或正在卸载 xigemaAS，则不能重复执行安装操作。

3.2.5 创建应用服务器

在节点上成功安装 xigemaAS 后，可以通过 xigemaAS 管理中心在该节点上创建一个或多个应用服务器。

目标节点上已安装 xigemaAS，方能创建应用服务器。

1. 单击**服务器 > 应用服务器 > 创建**，进入**“应用服务器创建”**页面。
2. 在**“基本信息”**面板输入要创建的应用服务器的基本信息。

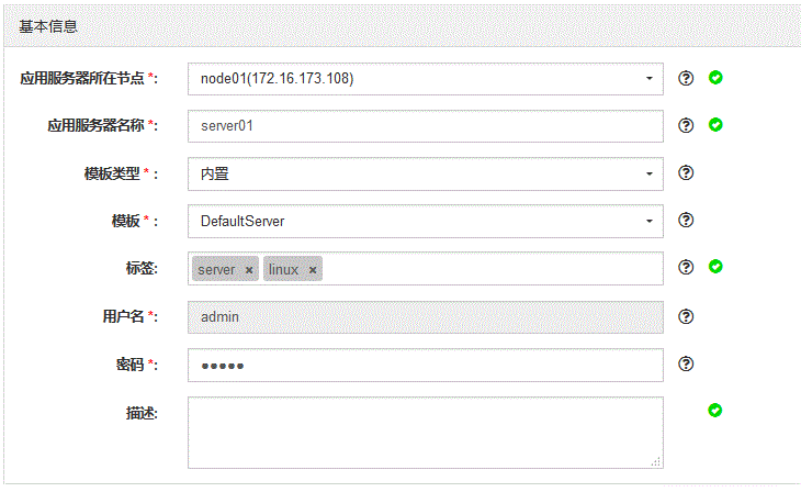


图 56: 应用服务器基本信息面板

a. 在**应用服务器所在节点**栏，可以从下拉列表中选择创建应用服务器的目标节点，也可以直接输入目标节点的名称，从而快速找到目标节点。
b. 在**应用服务器名称**栏输入要创建的应用服务器的名称。
c. 在**模板类型**栏选择应用服务器模板类型，默认模板类型为内置模板。选择完成后在**模板**栏选择相应类型的模板，默认模板为 DefaultServer。
d. 可选： 在**标签**、**描述**栏分别输入该应用服务器的标签和描述信息，用于标识和分类管理应用服务器。

3. 可选： 在**“系统属性”**面板编辑当前应用服务器的系统属性。

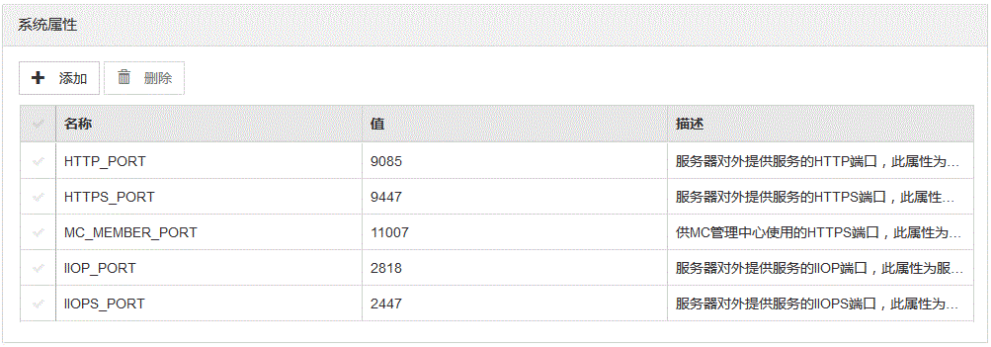


图 57: 应用服务器系统属性面板

a. 单击**添加**或**删除**按钮，添加或删除当前应用服务器的系统属性。其中，默认系统属性包括HTTP_PORT、HTTPS_PORT、MC_MEMBER_PORT、IIOP_PORT、IIOPS_PORT，默认系统属性不可删除。

b. 单击系统属性的名称、值、描述，可对该系统属性进行编辑。默认系统属性仅支持属性值的编辑，名称和描述不可编辑。

4. 单击**保存 > 确定**，若提示 “**应用服务器创建成功**” ，说明已成功创建应用服务器。

   应用服务器创建成功后，可继续创建新的应用服务器，或返回应用服务器列表查看已创建的应用服务器。在应用服务器列表中通过应用服务器名称、应用服务器状态、标签、部署在应用服务器上的应用名称、应用服务器所在节点名称以及所属集群名称可以查询已创建的应用服务器。

有关创建应用服务器时界面元素的含义描述和操作规范，请参阅创建应用服务器（见第 1885 页）。

3.2.6 创建集群

通过 xigemaAS 管理中心可以直接创建包含多个应用服务器的集群，并针对该集群进行应用部署等操作，无需手动定义和配置大量应用服务器。

1. 单击**集群 > 创建**，进入**“集群创建”**页面。在**“基本信息”**面板输入集群的基本信息。

图 58: 集群创建基本信息面板

a. 在**集群名称**栏输入要创建的集群名称。

b. 在**模板类型**栏选择集群模板类型，默认模板类型为内置模板。选择完成后在**模板**栏选择相应类型的模板，默认模板为 DefaultServer。

   xigemaAS 管理中心支持的模板类型包括内置、集群、服务器三个类型。若选择服务器模板类型，可以将模板服务器添加为当前集群的集群成员。

c. 可选： 在**标签**、**描述**栏分别输入当前集群的标签和描述。

2. 在**“集群成员”**面板新增、删除、批量创建集群成员。

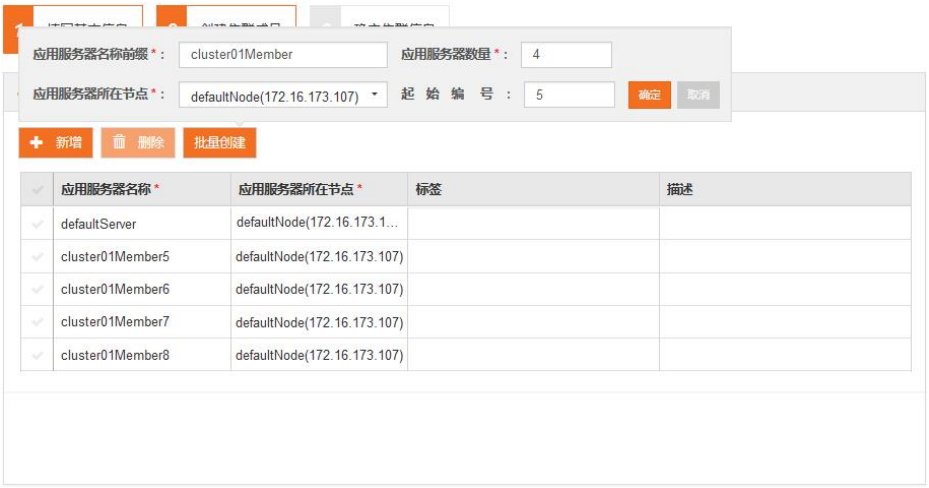


图 59: 集群创建成员信息面板

- 新增集群成员：单击**新增**，在集群成员列表中输入应用服务器名称，标签、描述，从**应用服务器所在节点**下拉列表中选择新增应用服务器的目标节点。
- 删除集群成员：从集群成员列表中勾选需要删除的集群成员，单击**删除**即可。
- 批量创建集群成员：单击**批量创建**，在批量创建窗口输入应用服务器名称前缀、应用服务器数量，并选择应用服务器所在节点。单击**确定**，批量创建的集群成员显示在集群成员列表中，说明已创建成功。用户可以根据需要输入集群成员的起始编号，首次创建集群成员时，编号最小值为1；多次批量创建时，编号必须大于此前批量创建的集群成员的数量。

注：一个集群中必须至少创建一个集群成员。

3. 在**“确认集群信息”**面板确认创建的集群信息是否填写正确。若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若所有集群成员的操作进度均为**“服务器已创建”**，则完成该集群的创建。

   集群创建成功后，可继续创建新的集群或返回集群列表查看已创建的集群。在集群列表中通过集群名称、集群成员名称、集群状态、标签以及部署在该集群上的应用名称可以查询已创建的集群。

有关创建集群时界面元素的含义描述和操作规范，请参阅创建集群（见第 1894 页）。

3.2.7 新增 Web 服务器

通过 xigemaAS 管理中心可以添加已安装好的 Web 服务器。

确保已存在安装好的 Web 服务器及第三方模块。

xigemaAS 管理中心 V2.1 支持以下 Web 服务器：

- Nginx：应用于以下操作系统和平台：Windows、Linux、AIX、FreeBSD、Solaris、HP-UX、MacOS，建议使用 nginx (0.7.53+)。详细信息请以 Nginx 官网信息为准。

  第三方模块具体包括：

  - gcc-4.8.3
  - gcc-c++-4.8.3
  - autoconf-2.69

  - automake-1.13.4
  - pcre-8.32
  - pcre-devel-8.32
  - zlib-1.2.7
  - zlib-devel-1.2.7
  - openssl-1.0.1e
  - openssl-devel-1.0.1e
  - readline-6.3.1

- Apache： 应用于类 Unix 平台和 Windows 平台，建议使用 Apache (2.2+)。详细信息请以 Apache 官网信息为准。

  第三方模块具体包括：

  - apr-1.4.5.tar.gz
  - apr-util-1.3.12.tar.gz
  - pcre-8.10.zip

👉 注：若新增 Web 服务器时选择 SSH 代理方式，则需要满足以下环境要求：

1. 若选择证书认证方式，请确保已为 xigemaAS 管理中心生成了 SSH 公、私钥证书文件，并已将 SSH 公钥证书文件添加到待注册节点的受信文件中；
2. 已安装 curl 工具，并且当前 SSH 用户具备 curl 命令的执行权限；
3. xigemaAS 管理中心与当前节点所在主机、Web 服务器所在主机的网络连通。

1. 单击**服务器 > Web 服务器 > 新增**，进入**“新增 Web 服务器”**页面。
2. 通过 xigemaAS 管理中心可以**“创建新的认证”**或**“使用已有认证”**新增 Web 服务器。若认证信息中存在已认证的 Web 服务器的所在机器，可以直接选择使用已有认证，选择完成后，填写 Web 服务器的基本信息即可；若选择创建新的认证信息，请执行以下操作新增 Web 服务器。
3. xigemaAS 管理中心支持 Nginx 和 Apache 的新增。

   a. 新增 Nginx：

      - 新增远程 Nginx：选择 **SSH** 代理类型，添加远程主机上的 Nginx。xigemaAS 管理中心默认新增远程 Web 服务器，须填写认证信息和基本信息。

        1. 在认证信息面板中，填写以下信息：

           - 选择并填写Web 服务器所在主机的计算机名，或 IPv4 地址。
           - 输入 SSH 用户名以及端口号。
           - 选择认证方式：xigemaAS 管理中心默认选择“用户名/密码”方式认证，需要填写登录 Web 服务器所在主机使用的密码；若选择“证书”方式认证，需要填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径和证书密码。

图 60: 认证信息面板

2. 在基本信息面板中选择 Nginx 作为 Web 服务器类型，并填写以下信息：
   - 输入 Web 服务器名称，该名称是 Web 服务器在 xigemaAS 管理中心的唯一标识，保存后不可修改。
   - 输入当前 Web 服务器的安装目录和 Java 运行时（JRE）目录。其中，Java 运行时（JRE）目录在保存后将作为其环境变量 JAVA_HOME 的值。
   - 输入标签及描述，用于标识和分类管理 Web 服务器。

图 61: Nginx 基本信息面板

- 新增本机 Nginx：在**“认证信息”**面板选择**本机**作为代理类型，添加 xigemaAS 管理中心所在主机上的 Nginx，只需要填写基本信息，不需要进行远程认证。

b. 新增 Apache：

- 新增远程 Apache：选择 **SSH** 代理类型，添加远程主机上的 Apache，须填写认证信息和基本信息。

  1. 填写认证信息，具体步骤请参阅步骤 *1*。
  2. 在基本信息面板中选择 Apache 作为 Web 服务器类型，并填写以下信息：

     - 输入 Web 服务器名称，该名称是 Web 服务器在 xigemaAS 管理中心的唯一标识，保存后不可修改。
     - 选择当前 Web 服务器的 OS 类型。若选择 **Windows** 平台，还需要输入将 Apache 注册为 Windows 服务时的 **Windows 服务名称**。若当前 Apache 未注册 Windows 服务，请使用管理员用户执行以下命令注册 Windows 服务：

       ```
       http.exe -k install [-n Windows服务名称]
       ```

     - 输入当前 Web 服务器的安装目录和 Java 运行时（JRE）目录。其中，Java 运行时（JRE）目录在保存后将作为其环境变量 JAVA_HOME 的值。
     - 输入标签及描述，用于标识和分类管理 Web 服务器。

图 62: Apache 基本信息面板

- 新增本机 Apache：在**“认证信息”**面板选择**本机**作为代理类型，添加 xigemaAS 管理中心所在主机上的 Apache，只需要填写基本信息，不需要进行远程认证。

4. 可选： 信息填写完成后，单击**环境检查**，进行当前 Web 服务器的环境检查。有关环境检查的具体检查内容和注意事项，请参阅 *Web* 服务器环境检查（见第 1833 页）。
5. 单击**保存**按钮，保存当前 Web 服务器信息，若提示 “Web **服务器创建成功**”，说明该 Web 服务器已添加成功；若提示环境检查未通过，请根据环境检查的详情提示修改相应信息；若存在无效信息，则无法保存当前信息，请参阅新增 *Web* 服务器（见第 1888 页）中界面元素的含义描述和操作规范，修改相应信息，再进行保存操作以完成 Web 服务器的添加。

   Web 服务器添加成功后，可继续添加新的 Web 服务器或返回 Web 服务器列表查看已添加的 Web 服务器。在 Web 服务器列表中通过 Web 服务器名称、主机、状态、标签可以查询已添加的 Web 服务器。

   注：一个 Web 服务器只能由一个 xigemaAS 管理中心进行管理。

Web 服务器添加完成后，xigemaAS 管理中心自动生成一个默认的虚拟主机。在 Web 服务器列表，单击该 Web 服务器操作栏中的 **虚拟主机管理**，查看虚拟主机列表。

3.2.8 部署应用

应用服务器或集群创建成功后，可以作为应用的部署目标。一个应用可以选择多个应用服务器或集群作为部署目标。同时，当 Web 服务器添加到 xigemaAS 管理中心后，可以为 Web 应用和企业应用配置负载均衡。

请确认已满足以下前置条件：

- 应用部署所在应用服务器已成功创建，即存在部署目标。
- 所要部署的应用已放置在本地主机目录或者 xigemaAS 管理中心所在主机目录。

1. 在页面左上角，单击**部署**。进入**“选择部署文件”**页面。

图 63: 选择部署文件面板

a. 选择**应用类型**。若部署的应用类型是 Web 应用，可以填写**应用上下文**。

   xigemaAS 管理中心支持的应用类型包括：

   - Web 应用

- 企业应用
- EJB 应用
- 连接器应用
- OSGi 应用

b. 选择应用部署方式。对于 Web 应用和企业应用，xigemaAS 管理中心支持本地上传、远程部署以及目录部署三种方式；针对其他应用，仅支持本地上传和远程部署。

根据所选部署方式，选择应执行的操作：

- 本地上传或远程部署：

  1. 单击**选择文件 > 开始上传**，若提示**上传成功**，则所选文件已上传。

- 目录部署：

  在部署Web 应用和企业应用时，如果选择**目录部署**，

  1. 在下方显示应用目录输入框，输入应用所在目录，如 D:\installAS\wlp\samples\examples。该应用必须是解压后文件。

  2. 填写**应用上下文**和**应用名称**。

  👉 注：对于目录部署：

  - xigemaAS 管理中心运行用户必须对应用所在目录有读权限。
  - 如果多点部署（如集群），要求应用存在于各个部署服务器的同一目录下。

2. 配置部署属性。

图 64: 配置部署属性面板

a. 可选： 选择私有共享库、公有共享库、以及类加载优先顺序。若应用类型为 OSGi 应用，选择共享库和类加载优先顺序。

b. 可选： 在**标签**、**描述**栏分别输入该应用的标签和描述信息，用于标识和分类管理应用。

3. 选择部署目标，设置负载均衡。

a. 在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。

图 65: 选择部署目标面板

b. 可选： 若当前应用类型为 Web 应用或企业应用，可以为应用配置负载均衡。在**“负载均衡设置”**面板输入 Web 服务器端应用上下文，选择配置负载均衡的虚拟主机。

图 66: 设置负载均衡面板

4. 确认部署信息。确认部署应用相关信息是否填写正确。若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若提示 **“应用已部署”** ，则该应用已成功部署。

应用部署成功后，可继续部署新的应用或返回应用列表查看已部署的应用。在应用列表中通过应用名称、应用类型、应用状态、标签可以查询已部署的应用。

有关部署应用时界面元素的含义描述和操作规范，请参阅应用部署（见第 1897 页）。

3.3 xigemaAS 管理中心功能概览

xigemaAS 管理中心支持以下功能模块的管理：节点管理、服务器管理（包括应用服务器管理、Web 服务器管理）、集群管理、应用管理、资源管理（包括共享库管理、连接池管理、数据源管理、数据源管理、JMS 管理）、系统管理（包括参数配置、数据备份）、用户管理和审计日志。

- xigemaAS 管理中心各模块具体功能如下：

	节点	应用服务器	**Web** 服务器	集群	应用	资源	用户管理	审计日志	系统
注册	√								
创建		√		√					
新增			√			√	√		
部署					√				
删除	√	√	√	√		√	√		
强制删除	√	√	√	√					
卸载					√				
查询	√	√	√	√	√	√	√	√	
编辑	√	√	√	√	√	√	√		
详情查看	√	√	√	√	√	√			
环境检查	√		√						
启动、停止、重启		√	√	√	√				
平滑重启			√						
安装/卸载 xigemaAS	√								
监控		√	√		√				
配置		√	√	√	√				
更新					√				
日志下载		√	√	√					
访问链接、子模块详情查看					√				
参数配置、数据备份、日志配置									√

3.3.1 节点管理

集中管理本机节点和远程节点，xigemaAS 管理中心提供已安装 xigemaAS 的默认节点，用户可以直接进行服务器创建、应用部署等操作。

注册节点

注册节点是使用 xigemaAS 管理中心进行节点管理，以及其他模块管理的第一步。

请确认已满足以下前置条件：

- 若使用 xigemaAS 管理中心选择 SSH 代理方式注册节点，则需要满足以下环境要求：
  - 若使用证书认证方式，请确保已为 xigemaAS 管理中心生成了 SSH 公、私钥证书文件，并已将 SSH 公钥证书文件添加到待注册节点的受信文件中；
  - 已安装 curl 工具，并且当前 SSH 用户具备 curl 命令的执行权限；
  - xigemaAS 管理中心与当前节点所在主机的网络连通。
- 若使用 xigemaAS 管理中心选择 Agent 代理方式注册节点，详细信息和具体操作请参阅安装 *Agent*（见第 1820 页）。
- 注册本机节点，节点所在主机可以是 Windows 系统或类 UNIX 系统。

当所有前置条件已满足，请在已登录 xigemaAS 管理中心主页状态下执行以下操作注册节点。

1. 单击**节点 > 注册**，进入注册节点页面。
2. 通过 xigemaAS 管理中心可以**“创建新的认证”**或**“使用已有认证”**注册节点。若使用已有认证信息注册节点，可以在**“认证信息”**下拉列表中选择已注册的节点，选择完成后，当前节点将直接使用已选节点的认证信息；若选择创建新的认证信息，请执行以下操作注册节点。

   xigemaAS 管理中心支持本机节点和远程节点的注册。

   a. **远程节点**：从**“代理类型”**栏选中 **SSH** 项，在远程主机上注册节点。xigemaAS 管理中心默认注册远程节点，须填写认证信息和基本信息。

      - 在**“认证信息”**面板填写以下信息：

图 67: 认证信息面板

      - 选择并填写节点所在主机的计算机名，或 IPv4 地址。
      - SSH 用户名，即登录节点所在主机的用户名。
      - SSH 端口号，即节点所在主机用于 SSH 通信的端口号，默认值为22。
      - 选择认证方式。默认选择“用户名/密码”方式认证，需要填写登录节点所在主机使用的密码。若选择“证书”方式认证，需要填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径。
      - 在**“基本信息”**面板填写以下信息：

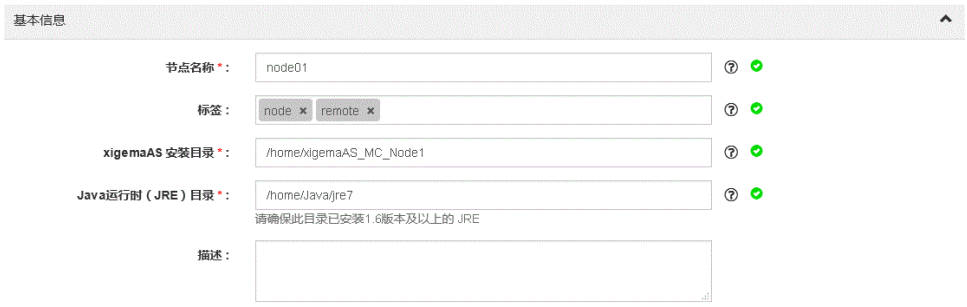


图 68: 基本信息面板

- 节点名称。节点名称是节点在 xigemaAS 管理中心的唯一标识，保存后不可修改。
- xigemaAS 安装目录。
- Java 运行时（JRE）目录。

  注：保存后，在该节点下创建的服务器，将使用该目录作为其环境变量 JAVA_HOME 的值。修改此值不会修改已创建的服务器的 JAVA_HOME，但会作为此后创建服务器的 JAVA_HOME 路径。
- 标签及描述，用于标识和分类管理节点。

b. **本机节点**：从**“代理类型”**栏选择**本机**项，在**“基本信息”**面板填写基本信息。注册 xigemaAS 管理中心所在主机上的节点，无需进行远程认证。

3. 单击**保存**按钮，保存节点注册信息。若提示 **“保存成功”** ，说明该节点注册成功。若存在无效信息，则无法保存。请参阅注册节点（见第 1882 页）中界面元素的含义描述和操作规范，修改相应的节点信息，再进行保存操作以完成节点的注册。

节点注册成功后，可继续注册新的节点或返回节点列表查看已注册节点。在节点列表中通过节点名称、主机名、xigemaAS 安装状态、标签可以查询已注册的节点。

删除节点

通过 xigemaAS 管理中心可以删除不再需要进行管理的节点。

- 请确保待删除的节点上不存在应用服务器。若节点上已创建应用服务器，需要先删除该应用服务器，再进行节点的删除操作。应用服务器删除的具体操作，请参阅删除应用服务器（见第 1823 页）。
- 若要删除代理类型为 Agent 的节点，请确保该节点上不存在应用服务器和 Web 服务器。若所选 Agent 节点上已添加 Web 服务器，需要先删除该 Web 服务器，再进行 Agent 节点的删除操作。Web 服务器删除的具体操作，请参阅删除 *Web* 服务器（见第 1833 页）。

节点上的 xigemaAS 卸载后，方能删除该节点。为了简化用户操作，xigemaAS 管理中心在删除节点时，提供自动帮您卸载 xigemaAS 的功能。

1. 单击**节点**，进入节点管理页面。

2. 在节点列表中选择要删除的节点，单击**删除 > 确定**，若所选节点上存在 xigemaAS，则 xigemaAS 管理中心先卸载该节点上的 xigemaAS，再删除节点。若提示 **“节点已删除”** ，则该节点删除成功。

注：若要完全卸载代理类型为 Agent 的节点，还需要手动删除 Agent 所在磁盘目录。

xigemaAS 管理中心支持批量删除节点，同时在删除过程中提供节点删除详情提示列表，列出所选节点的删除详情。若所选操作目标的操作进度都显示为 **“节点已删除”** ，表明所选节点删除成功。

3. 可选： 若删除出现异常，可以通过强制删除功能解除 xigemaAS 管理中心对所选节点的管理。在节点列表中选择需要强制删除的节点，单击**强制删除 > 确定**。

   强制删除功能可以将节点从 xigemaAS 管理中心节点管理列表中移除，但仍可能存在残留文件需要手动清理。

节点环境检查

注册节点后，可以在该节点上安装 xigemaAS。安装前，须进行环境检查。通过 xigemaAS 管理中心可以手动进行环境检查，也可以在安装 xigemaAS 时，自动执行环境检查。

节点的环境检查具体包括以下检查内容：

- IP 层连通性检测。
- SSH 连通性检测；若所选节点的代理类型为 Agent，则检测 Agent 连通性。
- xigemaAS 安装环境检查：
  - xigemaAS 安装路径是否有效，用户是否具有该目录的读写权限。
  - 目标节点的 JRE 版本是否为 JRE 1.6 或以上，并且能够正常运行。

1. 单击**节点**，进入节点管理页面。
2. 选择要进行环境检查的节点，单击**环境检查**按钮以手动进行环境检查。
3. 若提示 **“环境检查通过”** ，则完成所选节点的环境检查。

   xigemaAS 管理中心支持批量节点的环境检查，并提供环境检查详情提示列表。若所有节点的操作进度均为 **“环境检查通过”** ，则所选节点的环境检查全部通过；若存在环境检查未通过的节点，单击**详情**可以查看该节点环境检查具体成功项和失败项，根据提示信息检查环境配置，并修改相关信息，以完成当前节点的环境检查。

   注：
   - 注册节点时，环境检查可以在保存节点信息前进行，也可以在保存节点信息后进行。环境检查是否通过，不影响该节点的注册。但环境检查不通过，无法在该节点上安装 xigemaAS。
   - 对已安装 xigemaAS 并注册的节点进行编辑时，若修改 **Java 运行时 (JRE) 目录**，须先进行环境检查，再保存本次编辑。

在节点上安装 xigemaAS

节点注册完成后，需要在节点上安装 xigemaAS，安装成功后方可创建服务器。

1. 单击**节点**，进入节点管理页面。
2. 在节点列表中选择要安装 xigemaAS 的节点，单击**安装 AS** 按钮。提示“xigemaAS **安装成功**”，则该节点 xigemaAS 安装状态更新为“已安装”。

   安装过程中，xigemaAS 管理中心自动进行该节点的环境检查。环境检查是否通过，直接影响该节点的 xigemaAS 能否安装成功。

   也可以在安装 xigemaAS 前，单击**环境检查**按钮以手动进行环境检查。有关环境检查的具体检查内容和操作请参阅节点环境检查（见第 1819 页）。

   xigemaAS 管理中心支持批量安装 xigemaAS，同时在安装过程中提供安装详情提示，以查看当前操作进度及详情。

   注：若所选节点上已安装 xigemaAS 、正在安装 xigemaAS、或正在卸载 xigemaAS，则不能重复执行安装操作。

在节点上卸载 xigemaAS

xigemaAS 管理中心提供手动卸载按钮便于用户卸载节点上的 xigemaAS。

在节点上卸载 xigemaAS 需要满足以下前置条件：

节点上不存在服务器。

若节点上已创建服务器，需要先删除该服务器，再卸载节点上的 xigemaAS。服务器删除的具体操作，请参阅删除应用服务器（见第 1823 页）。

1. 单击**节点**，进入节点管理页面。
2. 在节点列表中选择要卸载 xigemaAS 的节点，单击**卸载 AS** 按钮。提示"**卸载成功**"，则该节点 xigemaAS 安装状态更新为"未安装"。

   xigemaAS 管理中心支持批量卸载 xigemaAS，同时在卸载过程中提供卸载详情提示，以查看当前操作进度及详情。

   👉 注：若所选节点已卸载 xigemaAS 、正在卸载 xigemaAS、或正在安装 xigemaAS，则不能重复操作。

安装 Agent

从 xigemaAS 管理中心下载的 Agent 内置了服务器 mc-agent 和应用 agent.war，面向 xigemaAS 管理中心提供 HTTPS 服务。同时，Agent 安装完成后，可以基于 Agent 管理 Web 服务器，并监控被 Agent 管理的应用服务器。

1. 在 xigemaAS 管理中心顶端导航栏，单击**下载 Agent**，将 Agent 下载到已指定的安装目录 。
2. 解压并安装 Agent。

   在命令行窗口中执行命令：

   ```
   ${JAVA_HOME}/bin/java -jar xigemaAS_${version}--Agent.jar
   ```

   安装成功后，将会在已指定的安装目录新增一个名为"wlp"的文件夹，该文件夹即为 Agent 的安装目录(${WLP_INSTALL_DIR})。
3. 配置 Agent。Agent 安装成功后，在 ${wlp_install_dir}/usr/servers/mc-agent 目录下，编辑 bootstrap.properties 文件来配置 Agent。

   a. 配置端口。在 bootstrap.properties 文件中，通过编辑以下片段来配置 Agent 对外提供的 HTTPS 服务端口：

      ```
      agent.https.port=port_number
      ```

      将 *port_number* 改为 Agent 的端口值。

   b. 配置 xigemaAS 管理中心的 IP 地址。在 bootstrap.properties 文件中编辑以下片段：

      ```
      mc.url=https://mc:mcport_number/mc
      ```

      将 *https://mc:mcport_number/mc* 改为 xigemaAS 管理中心的地址。

   c. 配置 Agent 节点的名称。在 bootstrap.properties 文件中的 `node.name` 处输入 Agent 节点的名称。

      👉 注：节点名称是 Agent 节点的唯一标识，请确保 xigemaAS 管理中心不存在同名节点。

d. 配置 Agent 节点所在主机的主机名称或 IP 地址。在 bootstrap.properties 文件中的 `host.name` 处输入 Agent 节点的主机名称或 IP 地址。若输入主机名称且未配置 DNS 服务器，还需要在 xigemaAS 管理中心所在主机的 host 表中将添加 Agent IP 地址和主机名映射，使 xigemaAS 管理中心通过该主机名能够访问 Agent。

e. 可选： 配置 Agent 的监控和故障恢复功能。在 bootstrap.properties 文件中编辑 process.watch 属性：

属性名称	数据类型	值	描述
process.watch	布尔型	true	启动监控和故障恢复功能。通过 Agent 可以监控被 Agent 管理的应用服务器的进程，若进程被意外关闭（正常情况下，应使用 xigemaAS 管理中心或命令 `server stop servername` 执行应用服务器的停止操作。非以上方式发生的应用服务器进程关闭，属于进程意外关闭），Agent 会自动启动该应用服务器。
		false	缺省值。关闭监控和故障恢复功能。

配置完成后，方可启动 Agent。

👉 注：Agent 启动后，若对以上配置项进行了修改，需要重启 Agent 使配置生效。

4. 启动 Agent。进入 Agent 的 wlp/bin 目录，执行以下命令启动 Agent。

```
server start mc-agent
```

Agent 启动成功，如下图所示：

命令窗口显示“**服务器 mc-agent 已准备就绪，可开始运行 xigemaAS**”，则说明 Agent 已启动成功。

👉 注：

- 启动 Agent 时，请确保 xigemaAS 管理中心已启动，并且已配置正确的 xigemaAS 管理中心地址。若 xigemaAS 管理中心已启动，Agent 会自动在 xigemaAS 管理中心注册一个节点，可以从节点管理列表中查看该节点的详细信息。该节点代理类型为 Agent，具备与 SSH 代理类型节点相同的业务功能。
- 若直接复制已安装完成的 Agent 到新的主机，作为该主机的 Agent 注册 xigemaAS 管理中心，需要删除 ${wlp_install_dir}/lib/versions 目录下的 node.info 文件，检查 bootstrap.properties 文件中各配置项是否正确，并重新启动 Agent。

3.3.2 服务器管理

xigemaAS 管理中心的服务器管理包括应用服务器管理和 Web 服务器管理。

应用服务器管理

通过 xigemaAS 管理中心能够集中管理本机节点和远程节点上创建的应用服务器。xigemaAS 管理中心提供默认本地应用服务器，用户可以直接使用并进行应用部署等操作。

创建应用服务器

在节点上成功安装 xigemaAS 后，可以通过 xigemaAS 管理中心在该节点上创建一个或多个应用服务器。

目标节点上已安装 xigemaAS，方能创建应用服务器。

1. 单击**服务器 > 应用服务器 > 创建**，进入**“应用服务器创建”**页面。
2. 在**“基本信息”**面板输入要创建的应用服务器的基本信息。

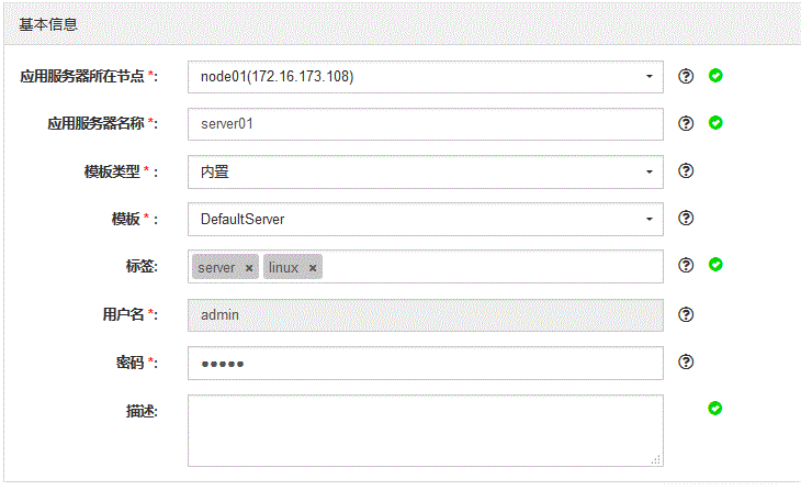


图 69: 应用服务器基本信息面板

   a. 在**应用服务器所在节点**栏，可以从下拉列表中选择创建应用服务器的目标节点，也可以直接输入目标节点的名称，从而快速找到目标节点。
   b. 在**应用服务器名称**栏输入要创建的应用服务器的名称。
   c. 在**模板类型**栏选择应用服务器模板类型，默认模板类型为内置模板。选择完成后在**模板**栏选择相应类型的模板，默认模板为 DefaultServer。
   d. 可选： 在**标签**、**描述**栏分别输入该应用服务器的标签和描述信息，用于标识和分类管理应用服务器。
3. 可选： 在**“系统属性”**面板编辑当前应用服务器的系统属性。

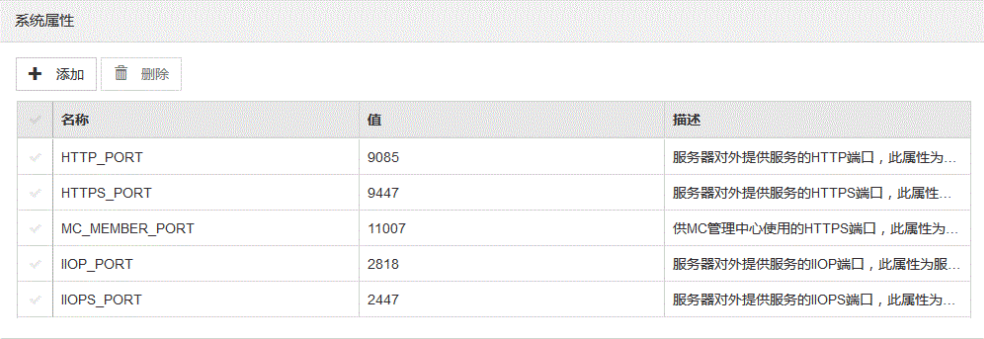


图 70: 应用服务器系统属性面板

a. 单击**添加**或**删除**按钮，添加或删除当前应用服务器的系统属性。其中，默认系统属性包括 HTTP_PORT、HTTPS_PORT、MC_MEMBER_PORT、IIOP_PORT、IIOPS_PORT，默认系统属性不可删除。

b. 单击系统属性的名称、值、描述，可对该系统属性进行编辑。默认系统属性仅支持属性值的编辑，名称和描述不可编辑。

4. 单击**保存 > 确定**，若提示 “**应用服务器创建成功**”，说明已成功创建应用服务器。

   应用服务器创建成功后，可继续创建新的应用服务器，或返回应用服务器列表查看已创建的应用服务器。在应用服务器列表中通过应用服务器名称、应用服务器状态、标签、部署在应用服务器上的应用名称、应用服务器所在节点名称以及所属集群名称可以查询已创建的应用服务器。

有关创建应用服务器时界面元素的含义描述和操作规范，请参阅创建应用服务器（见第 1885 页）。

删除应用服务器

通过 xigemaAS 管理中心可以删除不需要的应用服务器。

1. 单击**服务器 > 应用服务器**，进入应用服务器管理页面。
2. 从应用服务器列表中勾选需要删除的应用服务器，单击**删除 > 确定**，若提示 “**应用服务器已删除**”，则该应用服务器已删除成功。

   xigemaAS 管理中心支持批量删除应用服务器，同时在删除过程中提供应用服务器删除详情提示列表，列出所选应用服务器的删除详情。若所选操作目标的操作进度都显示为 “**应用服务器已删除**”，表明所选应用服务器删除成功。

   注：若所选应用服务器上已部署应用，那么，这个应用服务器即为当前应用的一个部署目标，若该应用只有一个部署目标，则删除应用服务器会一并删除应用服务器上的应用；若应用有多个部署目标，删除一个或多个部署目标后，如果该应用仍存在部署目标，则不会删除这个应用。

3. 可选： 若删除出现异常，可以通过强制删除功能解除 xigemaAS 管理中心对所选应用服务器的管理。在应用服务器列表中选择需要强制删除的应用服务器，单击**强制删除 > 确定**。

   强制删除功能可以将应用服务器从 xigemaAS 管理中心应用服务器管理列表中移除，但仍可能存在残留文件需要手动清理。

配置应用服务器

通过编辑应用服务器基本信息、JVM 参数以及 `server.xml` 配置文件来配置应用服务器。

1. 单击**服务器 > 应用服务器**进入应用服务器管理页面。从**“应用服务器列表”**中选择要配置的应用服务器，单击**应用服务器名称**，进入该应用服务器详情页面。

2. 单击**基本信息 > 编辑**，进入应用服务器基本信息编辑页面。
   a. 在**“基本信息”**面板单击**修改密码**，可以修改当前应用服务器的密码。若当前应用服务器是集群成员，则使用默认的应用服务器用户名和密码（admin/admin），并且不可修改。在**标签、服务器描述**处输入标签和描述信息，以标识和分类应用服务器。
   b. 在**“系统属性”**面板，单击**添加、删除**、以及具体列表项来编辑系统属性。默认属性的名称和描述不可编辑，仅可编辑属性值。
   c. 编辑完成后，单击**保存 > 确定**，完成应用服务器基本信息的编辑。
3. 单击 **JVM 配置 > 编辑**，进入 JVM 配置信息编辑页面。

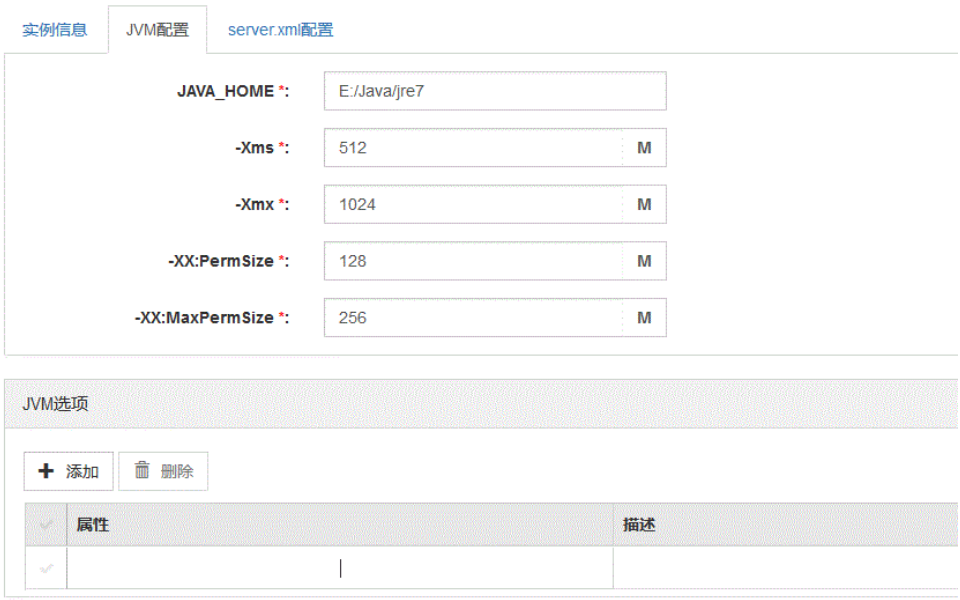


图 71: JVM 信息配置面板

   a. 在 JAVA_HOME 栏输入环境变量 JAVA_HOME 的值。在 **-Xms**、**-Xmx** 栏输入堆的最小值和最大值。在 **-XX:PermSize** 和 **-XX:MaxPermSize** 栏输入永久域的最小值和最大值。
   b. 在 **“JVM 选项”** 面板，单击**添加、删除**以及具体列表项来编辑 JVM 属性和 JVM 属性描述。
   若启用应用服务器调试功能，需要在 **“JVM 选项”**面板配置如下属性：

```
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=${PORT}
```

   其中 *${PORT}* 为调试端口的端口值，须确保该端口为可用端口，未被占用。

   👉 注：JVM 属性配置完成后，需要重启应用服务器使配置生效。

   c. 编辑完成后，单击**保存 > 确定**完成 JVM 配置信息的编辑。
4. 单击 **“server.xml 配置” > 编辑**，进入 `server.xml` 编辑页面。`server.xml` 是用来配置 xigemaAS 服务器的核心配置文件，可以在此文件中声明需要载入的功能部件。

图 72: server.xml 配置面板

xigemaAS 管理中心提供源窗格和设计窗格两种方式编辑 `server.xml` 配置文件。

a. 单击**“源”**，可以在功能部件管理器中查看并编辑 `server.xml` 配置文件。

b. 单击**“设计”**，切换到设计面板。左侧面板列出当前应用服务器已配置元素目录，右侧面板可以添加或移除子代。

c. 单击**设计 > 添加子代 > 添加**，将要添加的配置元素添加到应用服务器配置文件。单击左侧目录中的配置元素，可以对该元素详细配置信息进行查看和编辑。在左侧目录中选择要移除的元素，单击**移除 > 确定**，将该元素从 `server.xml` 文件中移除。

d. 配置完成后，单击**保存**，完成 `server.xml` 配置文件的编辑。

注：server.xml 配置文件修改后，需要重启应用服务器。

有关配置应用服务器时界面元素的含义描述和操作规范，请参阅配置应用服务器（见第 1886 页）。

监控应用服务器

xigemaAS 管理中心提供有关应用服务器性能信息的监控功能，使用户能够实时跟踪服务器已使用堆内存、已装入类、活动 JVM 线程、中央处理器 (CPU) 使用率及其他指标。

请确保满足以下前置条件：

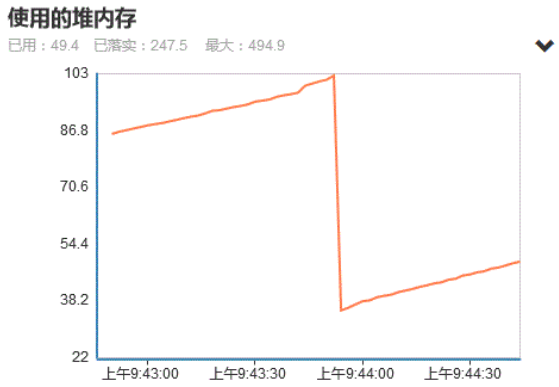
- 应用服务器处于启动状态。
- JRE 版本要求在 JRE 1.7 或以上，方可查看 CPU 使用情况。

1. 单击**服务器 > 应用服务器 > 应用服务器名称 > 监控**进入应用服务器监控页面。
2. 单击 **“编辑图表”**，选择要监控或隐藏的监控指标。
3. 单击**保存**，将保存对监控指标所做的选择，再次登录并打开监控页面时，将显示相同指标监控图表。

首次监控时，默认显示以下监控指标：

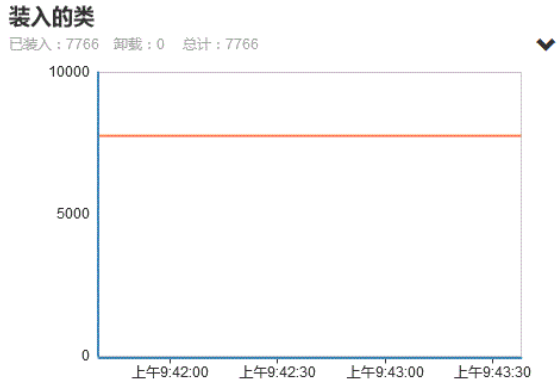
“使用的堆内存”

显示应用服务器使用的堆内存，单位为兆字节（MB），每2秒查询一次。此图表还显示已使用堆内存、已落实堆内存和最大堆内存。



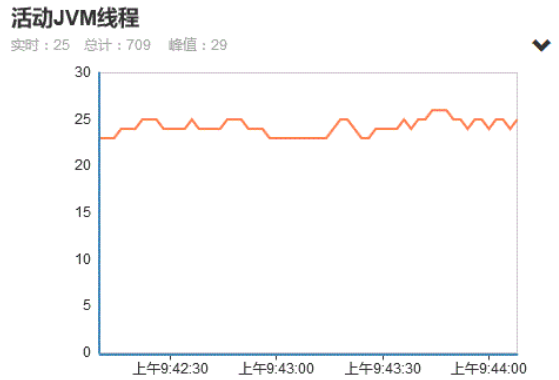
“装入的类”

显示已装入的类的数目，每2秒查询一次。此图表还显示已装入类数、未加载类数和总类数。



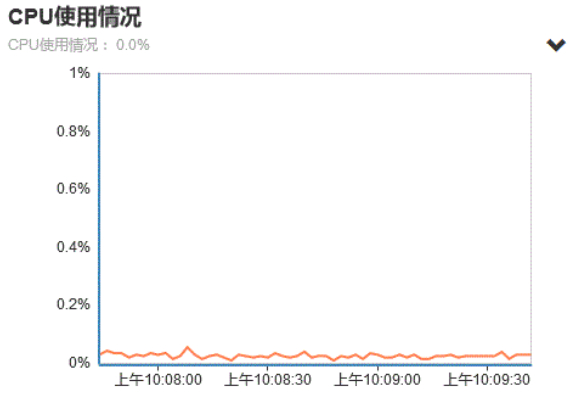
“活动 JVM 线程”

显示 JVM 线程数，每2秒查询一次。此图表还显示活动线程数、总线程数和峰值线程数。



“CPU 使用情况”

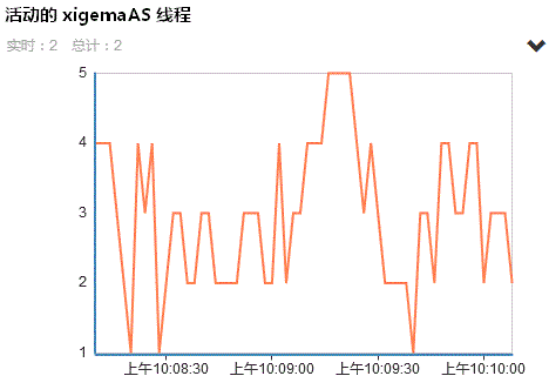
显示 CPU 的已使用百分比，每2秒查询一次。



若应用服务器已启用 monitor-1.0 功能部件，则 xigemaAS 管理中心还支持**“活动 xigemaAS 线程”**和**“活动会话”**的监控。有关启用功能部件的更多信息，请参阅配置应用服务器（见第 1823 页）。

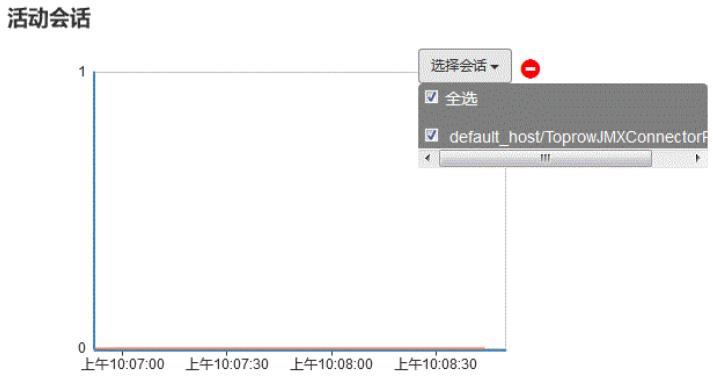
“活动 xigemaAS 线程”

显示 xigemaAS 线程数，每2秒查询一次。

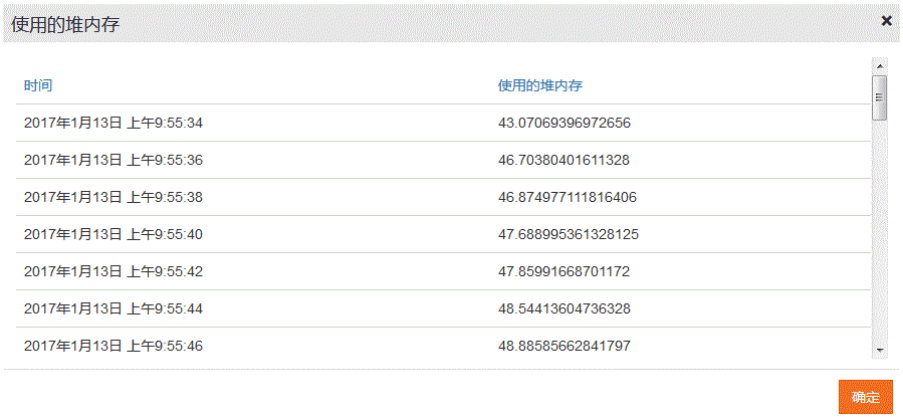


“活动会话”

可选择多个会话同时监控，每2秒查询一次。单击**编辑图表 > 活动会话**，从下拉列表中选择要监控的活动会话。



xigemaAS 管理中心提供图形和表格两种方式查看数据，默认显示图形，单击 可以通过表格形式查看相应数据。



应用服务器的常用操作

应用服务器常用操作包括应用服务器的启动、停止和重启。

- 启动应用服务器：应用服务器处于停止状态或未知状态时，方可启动应用服务器。

  在应用服务器列表中选择要启动的应用服务器，单击**启动**按钮。

  若提示**应用服务器已启动**，该应用服务器状态更新为“启动”。若启动出现异常，该应用服务器状态更新为“未知”。
- 停止应用服务器：应用服务器处于启动状态或未知状态时，方可停止应用服务器。

  在应用服务器列表中选择要停止的应用服务器，单击**停止**按钮。

  若提示**应用服务器已停止**，该应用服务器状态更新为“停止”。若停止出现异常，该应用服务器状态更新为“未知”。
- 重启应用服务器：不受应用服务器状态限制。

  在应用服务器列表中选择要重启的应用服务器，单击**重启**按钮。

  若提示**应用服务器已重启**，该应用服务器状态更新为“启动”。若重启失败，该应用服务器状态更新为“未知”。

应用服务器的状态是影响应用服务器常用操作的重要因素。有关应用服务器状态的更多信息，请参阅应用服务器状态（见第 1887 页）。

JNDI 树

xigemaAS 管理中心支持 JNDI 树的查看功能。

JNDI 树在应用服务器启动状态下方可查看。

1. 单击**服务器 > 应用服务器**，进入应用服务器管理页面。
2. 从应用服务器列表中勾选需要查看 JNDI 的应用服务器，单击**应用服务器名称 > JNDI** 进入 JNDI 树查看页面，查看当前应用服务器上配置的所有可用的 JNDI 信息。

图 73: JNDI 信息面板

应用服务器日志

通过 xigemaAS 管理中心可以下载或删除应用服务器日志。

1. 单击**服务器 > 应用服务器**，进入应用服务器管理页面。
2. 从应用服务器列表中勾选需要查看日志的应用服务器，单击**应用服务器名称 > 基本信息**，进入所选应用服务器的详情页面。
3. 单击**日志下载**，下载并查看当前应用服务器日志；单击**日志删除**，则删除当前应用服务器日志。

   应用服务器日志的数据来源于当前应用服务器下的 logs 目录。

Web 服务器管理

通过 xigemaAS 管理中心能够集中管理本机 Web 服务器和远程 Web 服务器。

新增 Web 服务器

通过 xigemaAS 管理中心可以添加已安装好的 Web 服务器。

确保已存在安装好的 Web 服务器及第三方模块。

xigemaAS 管理中心 V2.1 支持以下 Web 服务器：

- Nginx：应用于以下操作系统和平台：Windows、Linux、AIX、FreeBSD、Solaris、HP-UX、MacOS，建议使用 nginx (0.7.53+)。详细信息请以 Nginx 官网信息为准。

  第三方模块具体包括：

  - gcc-4.8.3
  - gcc-c++-4.8.3
  - autoconf-2.69
  - automake-1.13.4
  - pcre-8.32
  - pcre-devel-8.32
  - zlib-1.2.7
  - zlib-devel-1.2.7
  - openssl-1.0.1e
  - openssl-devel-1.0.1e
  - readline-6.3.1

- Apache： 应用于类 Unix 平台和 Windows 平台，建议使用 Apache (2.2+)。详细信息请以 Apache 官网信息为准。

  第三方模块具体包括：

  - apr-1.4.5.tar.gz
  - apr-util-1.3.12.tar.gz
  - pcre-8.10.zip

👉 注：若新增 Web 服务器时选择 SSH 代理方式，则需要满足以下环境要求：

1. 若选择证书认证方式，请确保已为 xigemaAS 管理中心生成了 SSH 公、私钥证书文件，并已将 SSH 公钥证书文件添加到待注册节点的受信文件中；
2. 已安装 curl 工具，并且当前 SSH 用户具备 curl 命令的执行权限；
3. xigemaAS 管理中心与当前节点所在主机、Web 服务器所在主机的网络连通。

1. 单击**服务器 > Web 服务器 > 新增**，进入**“新增 Web 服务器”**页面。
2. 通过 xigemaAS 管理中心可以**“创建新的认证”**或**“使用已有认证”**新增 Web 服务器。若认证信息中存在已认证的 Web 服务器的所在机器，可以直接选择使用已有认证，选择完成后，填写 Web 服务器的基本信息即可；若选择创建新的认证信息，请执行以下操作新增 Web 服务器。
3. xigemaAS 管理中心支持 Nginx 和 Apache 的新增。

   a. 新增 Nginx：

   - 新增远程 Nginx：选择 **SSH** 代理类型，添加远程主机上的 Nginx。xigemaAS 管理中心默认新增远程 Web 服务器，须填写认证信息和基本信息。

     1. 在认证信息面板中，填写以下信息：

        - 选择并填写Web 服务器所在主机的计算机名，或 IPv4 地址。
        - 输入 SSH 用户名以及端口号。
        - 选择认证方式：xigemaAS 管理中心默认选择“用户名/密码”方式认证，需要填写登录 Web 服务器所在主机使用的密码；若选择“证书”方式认证，需要填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径和证书密码。

图 74: 认证信息面板

2. 在基本信息面板中选择 Nginx 作为 Web 服务器类型，并填写以下信息：
   - 输入 Web 服务器名称，该名称是 Web 服务器在 xigemaAS 管理中心的唯一标识，保存后不可修改。
   - 输入当前 Web 服务器的安装目录和 Java 运行时（JRE）目录。其中，Java 运行时（JRE）目录在保存后将作为其环境变量 JAVA_HOME 的值。
   - 输入标签及描述，用于标识和分类管理 Web 服务器。

图 75: Nginx 基本信息面板

- 新增本机 Nginx：在**“认证信息”**面板选择**本机**作为代理类型，添加 xigemaAS 管理中心所在主机上的 Nginx，只需要填写基本信息，不需要进行远程认证。

b. 新增 Apache：

- 新增远程 Apache：选择 **SSH** 代理类型，添加远程主机上的 Apache，须填写认证信息和基本信息。

  1. 填写认证信息，具体步骤请参阅步骤 *1*。
  2. 在基本信息面板中选择 Apache 作为 Web 服务器类型，并填写以下信息：

     - 输入 Web 服务器名称，该名称是 Web 服务器在 xigemaAS 管理中心的唯一标识，保存后不可修改。
     - 选择当前 Web 服务器的 OS 类型。若选择 **Windows** 平台，还需要输入将 Apache 注册为 Windows 服务时的 **Windows 服务名称**。若当前 Apache 未注册 Windows 服务，请使用管理员用户执行以下命令注册 Windows 服务：

       ```
       http.exe -k install [-n Windows服务名称]
       ```

     - 输入当前 Web 服务器的安装目录和 Java 运行时（JRE）目录。其中，Java 运行时（JRE）目录在保存后将作为其环境变量 JAVA_HOME 的值。
     - 输入标签及描述，用于标识和分类管理 Web 服务器。

图 76: Apache 基本信息面板

- 新增本机 Apache：在**“认证信息”**面板选择**本机**作为代理类型，添加 xigemaAS 管理中心所在主机上的 Apache，只需要填写基本信息，不需要进行远程认证。

4. 可选： 信息填写完成后，单击**环境检查**，进行当前 Web 服务器的环境检查。有关环境检查的具体检查内容和注意事项，请参阅 *Web* 服务器环境检查（见第 1833 页）。
5. 单击**保存**按钮，保存当前 Web 服务器信息，若提示 “Web **服务器创建成功**”，说明该 Web 服务器已添加成功；若提示环境检查未通过，请根据环境检查的详情提示修改相应信息；若存在无效信息，则无法保存当前信息，请参阅新增 *Web* 服务器（见第 1888 页）中界面元素的含义描述和操作规范，修改相应信息，再进行保存操作以完成 Web 服务器的添加。

   Web 服务器添加成功后，可继续添加新的 Web 服务器或返回 Web 服务器列表查看已添加的 Web 服务器。在 Web 服务器列表中通过 Web 服务器名称、主机、状态、标签可以查询已添加的 Web 服务器。

   注：一个 Web 服务器只能由一个 xigemaAS 管理中心进行管理。

Web 服务器添加完成后，xigemaAS 管理中心自动生成一个默认的虚拟主机。在 Web 服务器列表，单击该 Web 服务器操作栏中的 **虚拟主机管理**，查看虚拟主机列表。

删除 Web 服务器

xigemaAS 管理中心可以删除 Web 服务器的配置信息并解除对其的管理，但不会删除 Web 服务器的物理安装。

Web 服务器处于启动状态、停止状态或未知状态下，方可删除。

1. 单击**服务器 > Web 服务器**，进入 Web 服务器管理页面。
2. 从 Web 服务器列表中勾选需要删除的 Web 服务器，单击**删除 > 确定**，若提示 “Web **服务器已删除**”，则该 Web 服务器已删除成功。

   xigemaAS 管理中心支持批量删除 Web 服务器，同时在删除过程中提供 Web 服务器删除详情提示列表，列出所选 Web 服务器的删除详情。若所选操作目标的操作进度都显示为 “Web **服务器已删除**”，表明所选 Web 服务器已成功删除。

   注：删除完成后，除默认虚拟主机外，Web 服务器上的虚拟主机和应用的负载配置也将一并删除。

3. 可选： 若删除出现异常，可以通过强制删除功能解除 xigemaAS 管理中心对所选 Web 服务器的管理。在 Web 服务器列表中选择需要强制删除的 Web 服务器，单击**强制删除 > 确定**。强制删除会将所选的 Web 服务器以及 Web 服务器上的虚拟主机从 xigemaAS 管理中心移除，解除对其的管理。若所选 Web 服务器上配置了应用的负载均衡，则负载配置也一并删除。存在负载配置未删除完整的情况，此时需要用户手动清理该负载配置。

   强制删除会将所选的 Web 服务器以及 Web 服务器上的虚拟主机从 xigemaAS 管理中心移除，解除对其的管理。若所选 Web 服务器上配置了应用的负载均衡，则负载配置也一并删除，但仍可能存在残留文件需要手动清理。

Web 服务器环境检查

Web 服务器的环境检查通过后，方能保存 Web 服务器的新增信息。

Web 服务器的环境检查具体包括以下检查内容：

- IP 层连通性检测。
- SSH 连通性检测； 若所选 Web 服务器 的节点代理类型为 Agent，则检测 Agent 连通性。
- Web 服务器安装环境检查：

- Web 服务器的版本是否为 nginx (0.7.53+)，安装路径是否有效，用户是否具有该目录的读写权限。
- Web 服务器所在主机的 JRE 版本是否为 JRE 1.6 或以上，并且能够正常运行。

1. 单击**服务器 > Web 服务器**，进入 Web 服务器管理页面。
2. 从 Web 服务器管理列表中勾选需要进行环境检查的 Web 服务器，单击**环境检查**按钮以手动进行环境检查。
3. 若提示 “**环境检查通过**”，则完成所选 Web 服务器的环境检查。

   xigemaAS 管理中心支持批量 Web 服务器的环境检查，并提供环境检查详情提示列表。若所有 Web 服务器的操作进度均为 “**环境检查通过**”，则所选 Web 服务器的环境检查全部通过；若存在环境检查未通过的 Web 服务器，单击**详情**可以查看该 Web 服务器环境检查具体成功项和失败项，根据提示信息检查环境配置，并修改相关信息，以完成当前 Web 服务器的环境检查。

   👉 注：Web 服务器的环境检查是否通过，直接影响 Web 服务器的新增和后续操作。

配置 Web 服务器

通过编辑配置信息来配置 Web 服务器。

Web 服务器状态为启动或停止时，方可编辑配置信息。

1. 单击**服务器 > Web 服务器**进入 Web 服务器管理页面。从 **“Web 服务器列表”**中选择要配置的 Web 服务器，单击 **Web 服务器名称**，进入当前 Web 服务器详情页面。
2. 单击**配置信息 > 编辑**，进入配置信息的编辑页面。默认显示配置文件 nginx.conf。
3. 从下拉列表中选择要编辑的配置文件，在文本编辑框中对已选择的文件内容进行编辑。
4. 可选： 编辑完成后，单击**配置文件校验**，检查配置文件的有效性。
5. 可选： 若配置信息编辑有误，可以单击**内容重置**，重置当前配置文件的内容。
6. 单击**保存**按钮，保存当前配置文件的配置信息。若提示**保存成功**，说明配置信息修改成功。若存在无效信息，则无法保存当前信息，请根据提示信息修改相应信息，再进行保存操作以完成配置信息的编辑。
7. 配置信息保存成功后，在左侧导航栏中单击**服务器 > Web 服务器**进入 Web 服务器管理页面。在列表中勾选已完成配置信息更改的 Web 服务器，单击**平滑重启 > 确定**，更新该 Web 服务器的配置信息。

有关配置 Web 服务器的操作规范和更多信息，请参阅配置 *Web* 服务器（见第 1889 页）。

配置 Session 亲和

可以通过编辑 Web 服务器的配置文件来手动配置 Session 亲和，使得来自同一用户的所有请求，均由同一台 xigemaAS 服务器进行处理。

- 若配置 Nginx 的 Session 亲和，请确保已安装模块 nginx-sticky-module。
- 若使用第三方模块 Session sticky 配置 Apache 的 Session 亲和，请确保已启动模块 LoadModule headers_module modules/mod_headers.so。

xigemaAS 管理中心默认 Web 服务器的 Session 配置为 Session 非亲和，若要配置 Web 服务器的 Session 亲和，请参照以下步骤。

- **手动配置 Apache 的 Session 亲和**:

  1. 打开 Apache 的配置文件 httpd.conf。
  2. 在 httpd.conf 配置文件中添加配置片段如下（添加片段以加粗斜体显示）：

```
#other configurations
Listen *:80
```

```
<VirtualHost *:80>
    #virtual_host_name:outsideVirtualHost-1 (Warning: This comment is
 used by xigemaAS Management Center, please do not modify this line.)
    Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
 env=BALANCER_ROUTE_CHANGED
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass /examples/ balancer://balancer_examples_examples_1/
examples/
    ProxyPassReverse /examples/ balancer://balancer_examples_examples_1/
examples/
</VirtualHost>
<Proxy balancer://balancer_examples_examples_1>
    BalancerMember http://172.16.173.107:9081
  route=server1[loadfactor20]
    BalancerMember http://172.16.173.107:9082
  route=server2[loadfactor80]
    ProxySet stickysession=ROUTEID
    [ProxySet lbmethod=byrequests[bytraffic][bybusyness]]
</Proxy>
#other configurations
```

其中，*route* 是负载成员的标识名，用于做 session sticky；*loadfactor* 表示集群成员的权重，取值范围为 [1,100]，默认值为1。

通过 ProxySet 指令设置负载均衡策略：

- byrequests：根据用户请求次数，按照集群成员的权重分发。
- bytraffic：根据各负载的流量，按照集群成员的权重分发。
- bybusyness：根据服务器的繁忙程度进行分发。

- **使用第三方模块 Session sticky 手动配置 Nginx 的 Session 亲和：**

1. 打开 Nginx 的配置文件 nginx.conf。
2. 在 nginx.conf 配置文件中的 server 块中添加配置片段如下（添加片段以加粗斜体显示）：

```
#other configurations
   server{
        #server_block_name:defaultVirtualHost  (Warning:this comment is
 used by xigemaAS Management Center, please do not modify this line.)
        listen 172.16.173.108:90;
        server_name vsettan.com vsettan.cn;
        location examples/{
            proxy_pass http://up4examples#1/examples/;
         }
     }
#other configurations
   upstream up4examples#1{
      sticky;
      server 172.16.173.107:9080
      server 172.16.173.108:9080
    }
#other configurations
```

- **使用 ip_hash 方式手动配置 Nginx 的 Session 亲和：**

1. 打开 Nginx 的配置文件 nginx.conf。

2. 在 nginx.conf 配置文件中的 server 块中添加配置片段如下（添加片段以加粗斜体显示）：

```
#other configurations
   server{
        #server_block_name:defaultVirtualHost  (Warning:this comment is
 used by xigemaAS Management Center, please do not modify this line.)
        listen 172.16.173.108:90;
        server_name vsettan.com vsettan.cn;
        location examples/{
            proxy_pass http://up4examples#1/examples/;
         }
     }
#other configurations
   upstream up4examples#1{
      ip_hash;
      server 172.16.173.107:9080
      server 172.16.173.108:9080
    }
#other configurations
```

👉 注：ip_hash 是 Nginx 自带的 Session 亲和配置。但是当一个客户端发生大量负载并发时，此方式无法使得请求被分配到其他应用服务器上，可能导致负载均衡失效。

监控 Web 服务器

xigemaAS 管理中心提供有关 Nginx 活跃指标的监控功能。

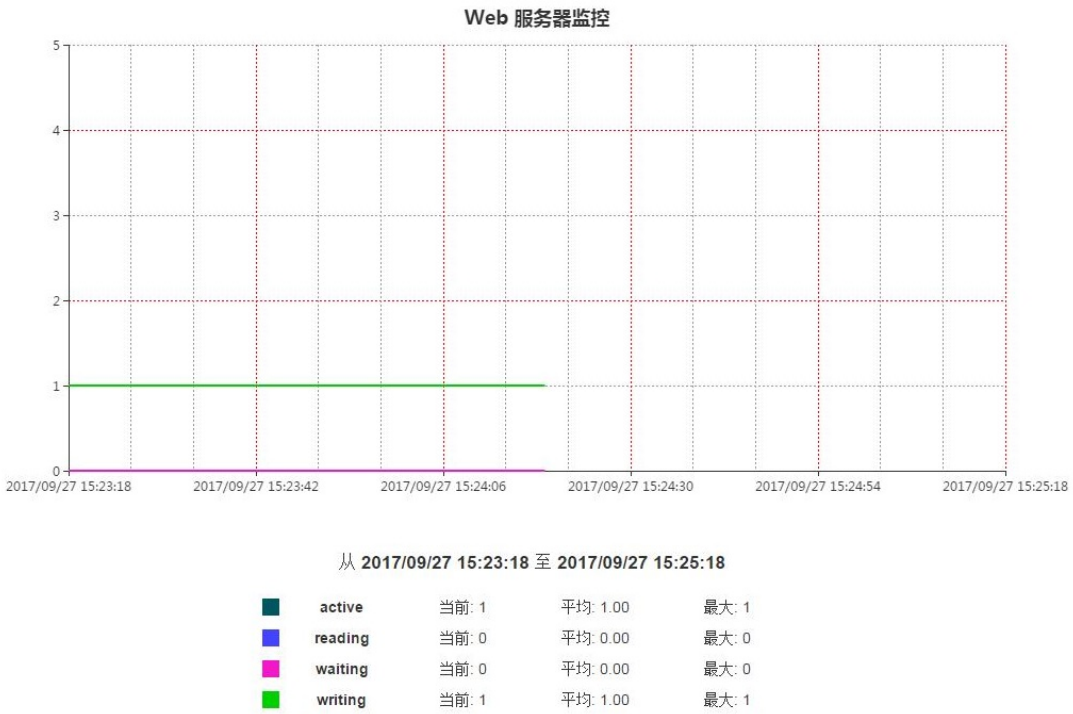
请确保满足以下前置条件：

- Nginx 处于启动状态。
- ngx_http_stub_status_module 模块已编译。

  Nginx 的监控功能由 ngx_http_stub_status_module 模块提供，该模块不是内建模块，因此需要在编译时添加以下配置项来激活：

```
--with-http_stub_status_module
```

1. 单击**服务器 > Web 服务器 > 名称**，进入 Web 服务器管理页面。
2. 在 Web 服务器管理列表中选择需要查看监控指标的 Nginx，单击 **Web 服务器名称 > 监控**，进入当前 Web 服务器监控页面。



活跃指标（Active、Waiting、Reading、Writing）随着请求量而增减。

Active

当前活跃的客户端连接数。

Waiting

活跃的连接处于 Waiting 状态。

Reading

当接收到请求时，连接离开 Waiting 状态，Reading 状态计数增加。

Writing

请求被读取后，Writing 状态计数增加。

Web 服务器的常用操作

Web 服务器常用操作包括 Web 服务器的启动、停止、重启和平滑重启。

- 启动 **Web** 服务器： Web 服务器处于停止状态或未知状态时，方可启动 Web 服务器。

  在 Web 服务器列表中选择要启动的 Web 服务器，单击**启动**按钮。

  若提示 `Web` **服务器已启动**，该 Web 服务器状态更新为“启动”。若启动出现异常，该 Web 服务器状态更新为“未知”。
- 停止 **Web** 服务器： Web 服务器处于启动状态或未知状态时，方可停止 Web 服务器。

  在 Web 服务器列表中选择要停止的 Web 服务器，单击**停止**按钮。

  若提示 `Web` **服务器已停止**，该 Web 服务器状态更新为“停止”。若停止出现异常，该 Web 服务器状态更新为“未知”。
- 重启 **Web** 服务器：Web 服务器处于启动状态、停止状态或未知状态时，方可重启 Web 服务器。

在 Web 服务器列表中选择要重启的 Web 服务器，单击**重启**按钮。

若提示 `Web` **服务器已重启**，该 Web 服务器状态更新为“启动”。若重启失败，该 Web 服务器状态更新为“未知”。

- **平滑重启**：Web 服务器处于启动状态或未知状态时，方可进行平滑重启。

  在 Web 服务器列表中选择要进行平滑重启的 Web 服务器，单击**平滑重启**按钮。

  xigemaAS 管理中心支持批量平滑重启，并提供平滑重启详情提示列表。若 Web 服务器的操作进度均显示为 “**平滑重启成功**”，说明已完成所选 Web 服务器的平滑重启。

Web 服务器的状态是影响 Web 服务器常用操作的重要因素。有关 Web 服务器状态的更多信息，请参阅 *Web 服务器状态*（见第 1892 页）。

Web 服务器日志

通过 xigemaAS 管理中心可以下载或删除 Web 服务器日志。

1. 单击**服务器 > Web 服务器**，进入 Web 服务器管理页面。
2. 从 Web 服务器列表中勾选需要查看日志的 Web 服务器，单击 **Web 服务器名称**，进入所选 Web 服务器的详情页面。
3. 单击**日志下载**，下载并查看当前 Web 服务器日志；单击**日志删除**，则删除当前 Web 服务器日志。

   Web 服务器日志的数据来源于当前 Web 服务器下的 logs 目录。

虚拟主机管理

可以通过 xigemaAS 管理中心新增或删除虚拟主机，并对已新增的虚拟主机进行编辑。

新增虚拟主机

Web 服务器新增完成后，xigemaAS 管理中心自动为其生成一个默认的虚拟主机，便于用户设置负载均衡。

虚拟主机所属 Web 服务器处于启动状态或停止状态、方可新增虚拟主机。

1. 单击**服务器 > Web 服务器 >**  ，进入虚拟主机管理页面。
2. 单击**新增**按钮，进入虚拟主机新增页面，如下图所示。

图 77: 新增虚拟主机

3. 在“**基本信息**”面板输入要新增的虚拟主机的基本信息。其中，**Web 服务器**默认为当前虚拟主机所属Web 服务器的名称和 IP 地址。

   a. 在**虚拟主机名称**栏输入要新增的虚拟主机的名称，保存后不可修改。
   b. 在**域名**栏输入访问虚拟主机的域名，可以输入多个域名。
   c. 可选： 在**标签**、**描述**栏分别输入该虚拟主机的标签和描述信息，用于标识和分类管理虚拟主机。

4. 在“**监听信息**”面板单击**添加**按钮，输入监听地址和监听端口。在“**监听信息列表**”中选择监听信息，单击**删除**按钮，可删除已添加的监听信息。

   注：新增虚拟主机时，需要至少输入一条监听信息。

5. 单击**保存 > 确定**，若提示 “**虚拟主机新增成功**” ，说明已成功新增虚拟主机。

   虚拟主机新增成功后，可继续添加新的虚拟主机，或返回虚拟主机列表查看已新增的虚拟主机。在虚拟主机列表中通过虚拟主机名称、域名、监听地址、端口号以及应用名称可以查询已新增的虚拟主机。

有关新增虚拟主机时界面元素的含义描述和操作规范，请参阅新增虚拟主机（见第 1893 页）。

删除虚拟主机

通过 xigemaAS 管理中心可以删除不需要的虚拟主机。

虚拟主机所在的 **Web 服务器**处于启动状态或停止状态，方可删除虚拟主机。

1. 单击**服务器 > Web 服务器 >** ，进入虚拟主机管理页面。
2. 从虚拟主机列表中勾选需要删除的虚拟主机，单击**删除 > 确定**，若提示 “**虚拟主机已删除**” ，则该虚拟主机已删除完成。

xigemaAS 管理中心支持批量删除虚拟主机，同时在删除过程中提供虚拟主机删除详情提示列表，列出所选虚拟主机的删除详情。若所选操作目标的操作进度都显示为 “**虚拟主机已删除**”，表明所选虚拟主机已删除完成。

注：

- 至少保留一个虚拟主机。
- 若当前虚拟主机上已配置了应用的负载均衡，则删除虚拟主机会一并删除虚拟主机上应用的负载配置。

3.3.3 集群管理

集群管理主要包括集群的创建、删除、配置、启动、停止、重启以及集群成员的管理等功能。便于用户对配置相同，并部署相同应用的应用服务器进行集中管理。实现从客户端到多个应用服务器之间的请求分发功能，同时保留持久性消息的完整性。

创建集群

通过 xigemaAS 管理中心可以直接创建包含多个应用服务器的集群，并针对该集群进行应用部署等操作，无需手动定义和配置大量应用服务器。

1. 单击**集群 > 创建**，进入**“集群创建”**页面。在**“基本信息”**面板输入集群的基本信息。

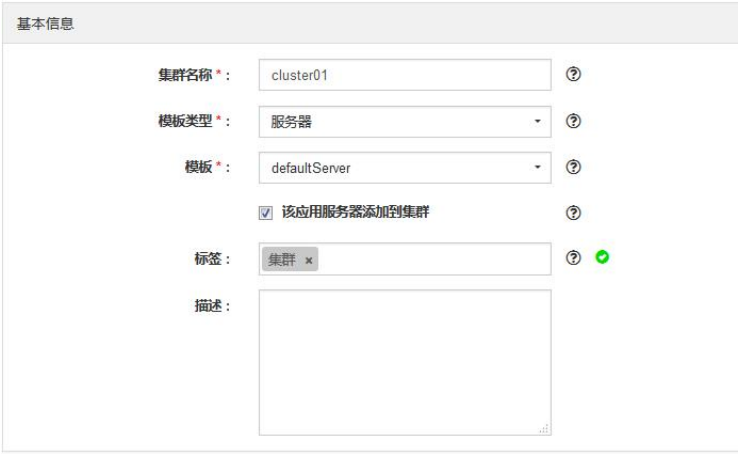


图 78: 集群创建基本信息面板

   a. 在**集群名称**栏输入要创建的集群名称。
   b. 在**模板类型**栏选择集群模板类型，默认模板类型为内置模板。选择完成后在**模板**栏选择相应类型的模板，默认模板为 DefaultServer。

      xigemaAS 管理中心支持的模板类型包括内置、集群、服务器三个类型。若选择服务器模板类型，可以将模板服务器添加为当前集群的集群成员。
   c. 可选： 在**标签**、**描述**栏分别输入当前集群的标签和描述。
2. 在**“集群成员”**面板新增、删除、批量创建集群成员。

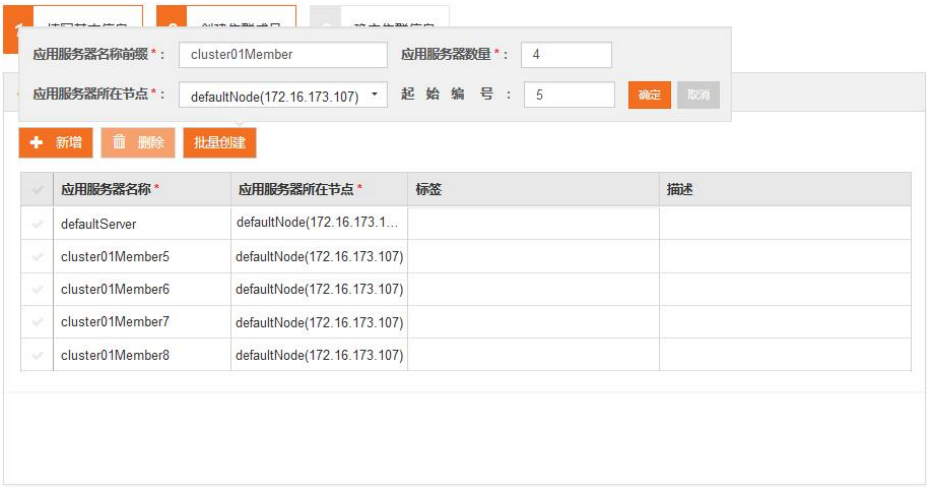


图 79: 集群创建成员信息面板

- 新增集群成员：单击**新增**，在集群成员列表中输入应用服务器名称，标签、描述，从**应用服务器所在节点**下拉列表中选择新增应用服务器的目标节点。
- 删除集群成员：从集群成员列表中勾选需要删除的集群成员，单击**删除**即可。
- 批量创建集群成员：单击**批量创建**，在批量创建窗口输入应用服务器名称前缀、应用服务器数量，并选择应用服务器所在节点。单击**确定**，批量创建的集群成员显示在集群成员列表中，说明已创建成功。用户可以根据需要输入集群成员的起始编号，首次创建集群成员时，编号最小值为1；多次批量创建时，编号必须大于此前批量创建的集群成员的数量。

注：一个集群中必须至少创建一个集群成员。

3. 在**“确认集群信息”**面板确认创建的集群信息是否填写正确。若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若所有集群成员的操作进度均为**“服务器已创建”**，则完成该集群的创建。

   集群创建成功后，可继续创建新的集群或返回集群列表查看已创建的集群。在集群列表中通过集群名称、集群成员名称、集群状态、标签以及部署在该集群上的应用名称可以查询已创建的集群。

有关创建集群时界面元素的含义描述和操作规范，请参阅创建集群（见第 1894 页）。

删除集群

通过 xigemaAS 管理中心可以删除不再需要进行管理的集群，集群删除后，集群中的集群成员被一并删除。

1. 单击**集群**，进入集群管理页面。
2. 从集群列表中勾选需要删除的集群，单击**删除**按钮，当集群中所有的集群成员全部删除后，该集群删除完成。

   xigemaAS 管理中心支持批量删除集群，同时在删除过程中提供集群删除详情提示列表，列出所选集群的删除详情。若所选操作目标的操作进度都显示为**“应用服务器已删除”**，表明所选集群的集群成员已被删除完成。

   注：若当前集群上已部署应用，那么，这个集群即为当前应用的一个部署目标，若该应用只有一个部署目标，则删除集群会一并删除集群上的应用；若应用有多个部署目标，删除一个或多个部署目标后，如果该应用仍存在部署目标，则不会删除这个应用。

3. 可选： 若删除出现异常，可以通过强制删除功能解除 xigemaAS 管理中心对所选集群的管理。在集群列表中选择需要强制删除的集群，单击**强制删除 > 确定**。

   强制删除功能可以将集群从 xigemaAS 管理中心集群管理列表中移除，但仍可能存在残留文件需要手动清理。

配置集群

通过编辑集群基本信息、HttpSession 信息以及 `server.xml` 配置文件来配置集群。

单击**集群**进入集群管理页面。从**“集群列表”**中选择需要配置的集群，单击**集群名称**，进入该集群详情页面。

- 单击**基本信息 > 编辑**，进入集群基本信息编辑页面。

图 80: 集群基本信息编辑面板

1. 在**“基本信息”**面板可以输入标签和描述信息，以标识和分类服务器，其他项可查看不可编辑。
2. 编辑完成后，单击**保存 > 确定**，完成集群基本信息的编辑操作。

- 单击 **`HttpSession` > 编辑**，进入 HttpSession 信息编辑页面。

图 81: HttpSession 信息编辑面板

1. 选择 Session 共享方式，xigemaAS 管理中心支持关系数据库和 Redis 数据库两种 Session 共享方式。
2. 选择 Session 存储数据源，根据已选共享方式，从下拉列表中选择 Session 存储数据源。

- 单击 **“server.xml 配置” > 编辑**，进入 `server.xml` 配置文件编辑页面。

图 82: server.xml 配置面板

xigemaAS 管理中心提供源窗格和设计窗格两种方式编辑 `server.xml` 配置文件。

1. 单击**“源”**，可以在功能部件管理器中查看并编辑 `server.xml` 配置文件。
2. 单击**“设计”**，切换到设计面板。左侧面板列出当前集群已配置元素目录，右侧面板可以添加或移除子代。
3. 单击**设计 > 添加子代 > 添加**，将要添加的配置元素添加到配置文件。单击左侧目录中的配置元素，可以对该元素详细配置信息进行查看和编辑。在左侧目录中选择要移除的元素，单击**移除 > 确定**，可以将该元素从 `server.xml` 文件中移除。
4. 配置完成后，单击**保存**，完成 `server.xml` 配置文件的编辑。

👉 注：

- server.xml 配置文件修改后，需要重启集群使配置生效。
- 同一集群中集群成员的 server.xml 配置文件相同。集群成员的 server.xml 配置文件只可查看不可编辑，可以通过编辑所属集群的 server.xml 配置文件来同步配置集群成员。

集群的常用操作

集群常用操作包括启动集群、停止集群以及重启集群。

- 启动集群：集群处于停止状态、未知状态、部分启动状态或部分未知状态，方可启动集群。

  在集群列表中选择要启动的集群，单击**启动**按钮。

  xigemaAS 管理中心提供启动详情列表，便于集中查看当前集群中各集群成员启动详情。若集群成员全部启动成功，该集群状态更新为“启动”。
- 停止集群：集群处于启动状态、未知状态、部分启动状态或部分未知状态，方可停止集群。

  在集群列表中选择要停止的集群，单击**停止**按钮。

  xigemaAS 管理中心提供停止详情列表，便于集中查看当前集群中各集群成员停止详情。若集群成员已全部停止，该集群状态更新为“停止”。
- 重启集群：集群处于启动状态、停止状态、未知状态、部分启动状态或部分未知状态，方可重启集群。

  在集群列表中选择要重启的集群，单击**重启**按钮。

  xigemaAS 管理中心提供重启详情列表，便于集中查看当前集群中各集群成员重启详情。若集群成员全部重启成功，该集群状态更新为“启动”。

集群的状态是影响集群启动、停止、重启的重要因素。有关服务器状态的更多信息，请参阅集群状态（见第 1896 页）。

集群成员管理

xigemaAS 管理中心支持集群成员的新增、删除、启动、停止、重启和日志下载功能。

- 单击**集群**进入集群管理页面，在集群总列表的集群成员列，单击**详情 > 编辑**进入所选集群的成员信息编辑页面。

图 83: 集群成员编辑页面

  - 新增集群成员：单击**新增**，在成员列表中输入服务器名称，选择所属节点，单击**保存 > 确定**，完成集群成员的新增。
  - 删除集群成员：在集群成员列表中选择要删除的集群成员，单击**删除**，列表中该项被移除，单击**保存 > 确定**，则完成该集群成员的删除。
- 在集群管理页面，单击**详情**进入所选集群的集群成员详情页面，查看当前集群的成员信息。

图 84: 集群成员信息详情页面

  - 启动集群成员：集群成员处于停止状态或未知状态时，方可启动。

    在集群成员列表中选择要启动的集群成员，单击**启动**按钮。

    若提示**服务器已启动**，该集群成员状态更新为“启动”。若启动出现异常，该集群成员状态更新为“未知”。
  - 停止集群成员：集群成员处于启动状态或未知状态时，方可停止。

    在集群成员列表中选择要停止的集群成员，单击**停止**按钮。

若提示**服务器已停止**，该集群成员状态更新为“停止”。若停止出现异常，该集群成员状态更新为“未知”。

- 重启集群成员：不受集群成员状态限制。

  在集群成员列表中选择要重启的集群成员，单击**重启**按钮。

  若提示**服务器已重启**，该集群成员状态更新为“启动”。若重启失败，该集群成员状态更新为“未知”。
- 查询集群成员：可以通过服务器名称、服务器所在节点名称、服务器状态以及标签来查询已创建的集群成员。
- 日志下载：在不勾选集群成员的状态下，单击**日志下载**，xigemaAS 管理中心将下载当前集群的集群日志到本地；若从集群成员列表勾选需要查看日志的集群成员，单击**日志下载**，xigemaAS 管理中心将下载所选集群成员的日志到本地。

集群日志

通过 xigemaAS 管理中心可以下载集群日志、集群成员日志。

1. 集群日志：
   a. 单击**集群**，进入集群管理页面。
   b. 从集群列表中勾选需要查看日志的集群，单击**集群名称 > 基本信息**，进入集群详情页面。
   c. 单击**日志下载**，下载并查看当前集群日志。
2. 集群成员日志：
   a. 单击**集群 > 详情**，进入集群成员信息页面。
   b. 从集群成员列表中勾选需要查看日志的集群成员，单击**日志下载**，下载并查看当前集群成员日志；单击**日志删除**，删除当前集群日志。

3.3.4 应用管理

xigemaAS 管理中心可以对应用服务器或集群上部署的应用进行集中管理，包括应用的部署和卸载、启动、停止和重启等操作。

部署应用

应用服务器或集群创建成功后，可以作为应用的部署目标。一个应用可以选择多个应用服务器或集群作为部署目标。同时，当 Web 服务器添加到 xigemaAS 管理中心后，可以为 Web 应用和企业应用配置负载均衡。

请确认已满足以下前置条件：

- 应用部署所在应用服务器已成功创建，即存在部署目标。
- 所要部署的应用已放置在本地主机目录或者 xigemaAS 管理中心所在主机目录。

1. 在页面左上角，单击**部署**。进入**“选择部署文件”**页面。

图 85: 选择部署文件面板

a. 选择**应用类型**。若部署的应用类型是 Web 应用，可以填写**应用上下文**。

xigemaAS 管理中心支持的应用类型包括：

- Web 应用
- 企业应用
- EJB 应用
- 连接器应用
- OSGi 应用

b. 选择应用部署方式。对于 Web 应用和企业应用，xigemaAS 管理中心支持本地上传、远程部署以及目录部署三种方式；针对其他应用，仅支持本地上传和远程部署。

根据所选部署方式，选择应执行的操作：

- 本地上传或远程部署：

  1. 单击**选择文件 > 开始上传**，若提示**上传成功**，则所选文件已上传。

- 目录部署：

  在部署Web 应用和企业应用时，如果选择**目录部署**，

  1. 在下方显示应用目录输入框，输入应用所在目录，如 D:\installAS\wlp\samples\examples。该应用必须是解压后文件。
  2. 填写**应用上下文**和**应用名称**。

  注：对于目录部署：

  - xigemaAS 管理中心运行用户必须对应用所在目录有读权限。
  - 如果多点部署（如集群），要求应用存在于各个部署服务器的同一目录下。

2. 配置部署属性。

图 86: 配置部署属性面板

a. 可选： 选择私有共享库、公有共享库、以及类加载优先顺序。若应用类型为 OSGi 应用，选择共享库和类加载优先顺序。

b. 可选： 在**标签**、**描述**栏分别输入该应用的标签和描述信息，用于标识和分类管理应用。

3. 选择部署目标，设置负载均衡。

a. 在**“可选部署目标”**列表中选择应用部署目标，添加到**已选部署目标**。

图 87: 选择部署目标面板

b. 可选： 若当前应用类型为 Web 应用或企业应用，可以为应用配置负载均衡。在**“负载均衡设置”**面板输入 Web 服务器端应用上下文，选择配置负载均衡的虚拟主机。

图 88: 设置负载均衡面板

4. 确认部署信息。确认部署应用相关信息是否填写正确。若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若提示 “**应用已部署**” ，则该应用已成功部署。

   应用部署成功后，可继续部署新的应用或返回应用列表查看已部署的应用。在应用列表中通过应用名称、应用类型、应用状态、标签可以查询已部署的应用。

有关部署应用时界面元素的含义描述和操作规范，请参阅应用部署（见第 1897 页）。

卸载应用

通过 xigemaAS 管理中心可以卸载不需要的应用。

1. 单击**应用**，进入应用管理页面。
2. 从应用列表中勾选需要卸载的应用，单击**卸载 > 确定**按钮，若提示**应用已卸载**，则完成该应用的卸载。

   xigemaAS 管理中心支持批量卸载应用，同时在卸载过程中提供应用卸载详情提示列表，列出所选应用的卸载详情。若所选操作目标的操作进度都显示为 “**应用已卸载**” ，表明所选应用已被卸载完成。

   注：若应用的部署目标为集群，卸载集群上的应用后，集群中所有集群成员上的应用都将被卸载。

编辑应用

应用部署成功后，可以通过编辑应用为应用添加或移除部署目标。

1. 单击**应用**，进入应用管理页面。从应用列表中选择需要查看详情或编辑信息的应用，单击**应用名称**，进入当前应用详情页面。

   应用详情页面支持应用基本信息查看，应用的启动、停止、重启、卸载，以及查看 Web 应用的访问链接。切换到**“部署信息”**面板，可以查看当前应用部署目标。
2. 单击**编辑**，进入应用编辑页面。

   a. 在**“基本信息”**面板，支持私有共享库、公有共享库、类加载优先顺序的选择，以及标签和描述的编辑。

      对于部署方式为目录部署的应用，还可以修改应用位置。修改应用位置就是更改应用。

   b. 在**“部署目标”**面板，可以为当前应用添加或移除部署目标。在**“可选部署目标”**中选择要添加的部署目标，单击 将应用服务器或集群添加到已选部署目标；在**“已选部署目标”**中选择要移除的部署

目标，单击将部署目标移除。操作完成后，单击**保存 > 确定**，弹出**“应用部署目标改变任务详情提示”**窗，当所有部署目标的操作进度均为 **“应用已部署”** ，则完成部署目标的添加和移除。

图 89: 应用部署信息编辑页面

c. 在**“负载均衡配置信息面板”**，可以为当前应用配置负载均衡。在**“可选虚拟主机”**中选择要添加的虚拟主机，单击将其添加到已选虚拟主机；在**“已选部署目标”**中选择要移除的虚拟主机，单击将目标移除。

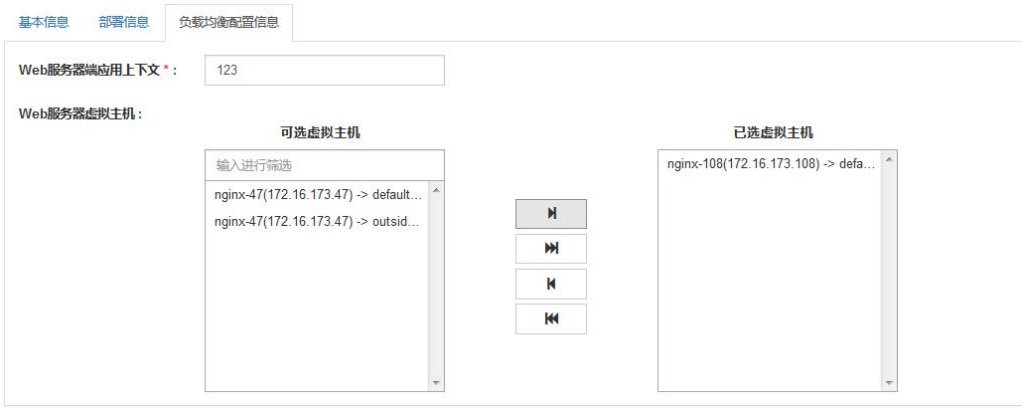


图 90: 应用负载均衡配置信息页面

3. 编辑完成后，单击**保存 > 确定**，若提示 **“应用已更新”** ，则当前应用更新成功。

更新应用

xigemaAS 管理中心提供三种更新应用的方式，具体包括：替换整个应用，替换或添加单个文件，添加、替换或删除多个文件。

1. 单击**应用**，进入应用管理页面。从应用列表中勾选需要更新的应用，单击**更新**按钮，弹出应用更新面板。

图 91: 应用更新面板

2. 选择更新方式。
   - 替换整个应用：更新的应用必须与原始部署包的格式相同，更新完成后会重启应用。
     1. 上传应用部署包。选择应用部署包所在位置，单击**选择文件 > 开始上传 > 确定**，若提示 “**应用已更新**”，则完成所选应用的更新。
   - 替换或添加单个文件：
     1. 静态文件更新：
        - 若只更新静态文件，则勾选**静态文件更新**，更新完成后立即生效，不需要重启应用。
        - 若更新整个应用，则不勾选**静态文件更新**，应用更新完成后，将自动重启应用使更新生效。
     2. 输入文件所在目录：输入要替换或者添加的文件所在目录。

        👉 注：
        - 必须是相对路径，即文件相对于应用部署包的相对路径；
        - 确保所填目录下不存在其他同名文件。
     3. 上传待更新文件：选择文件所在位置，单击**选择文件 > 开始上传 > 确定**，若提示 “**应用已更新**”，则完成所选应用的更新。
   - 添加、替换或删除多个文件：

     👉 注：若要更新的是 .zip 类型的应用，方可选择此方式。

     - 添加：添加应用部署包中没有的文件。
     - 替换：替换应用部署包中已存在的文件。
     - 删除：在 /META-INF/vsettan-delete.properties 文件中，输入要删除的文件路径。

     1. 静态文件更新：
        - 若只更新静态文件，则勾选**静态文件更新**，更新完成后立即生效，不需要重启应用。
        - 若更新整个应用，则不勾选**静态文件更新**，应用更新完成后，将自动重启应用使更新生效。
     2. 上传待更新文件：选择文件所在位置，单击**选择文件 > 开始上传 > 确定**，若提示 “**应用已更新**”，则完成所选应用的更新。

若要替换位于远程主机上的应用 hello.war 中 jsp 子目录下的文件 hello.jsp，那么在 xigemaAS 管理中心上需要进行如下操作：

1. 单击**应用**，在应用管理列表中勾选 hello.war，单击**更新**，弹出应用 hello.war 的更新面板。

2. 在更新方式中选择**替换或添加单个文件**。
3. 在文件所在目录处输入 /hello.war/jsp 或 /hello.war/jsp/。
4. 选择待更新文件：选择**远程文件系统**，单击**选择文件**，选择文件名为 hello.jsp 的文件,单击**确定**。

若提示 “**应用已更新**”，则完成应用 hello.war 中的文件 hello.jsp 的替换。

应用的常用操作

应用的常用操作包括启动应用、停止应用以及重启应用。

应用的启动、停止、重启操作需要应用所在应用服务器或集群处于“启动”状态，方可进行操作。

- 启动应用：应用处于停止状态、未知状态、部分启动状态或部分未知状态时，方可启动应用。

  在应用列表中选择要启动的应用，单击**启动**按钮。

  若提示**应用已启动**，该应用状态更新为“启动”。若启动失败，该应用状态更新为“未知”。若启动的应用部署在多个应用服务器上，当启动过程中存在启动失败的情况时，该应用状态更新为“部分未知”。
- 停止应用：应用处于启动状态、未知状态、部分启动状态或部分未知状态时，方可停止应用。

  在应用列表中选择要停止的应用，单击**停止**按钮。

  若提示**应用已停止**，该应用状态更新为“停止”。若停止失败，该应用状态更新为“未知”。
- 重启应用：应用处于启动状态、未知状态、部分启动状态或部分未知状态时，方可重启应用。

  在应用列表中选择要重启的应用，单击**重启**按钮。

  若提示**应用已重启**，该应用状态更新为“启动”。若重启失败，该应用状态更新为“未知”。若批量重启后失败，该应用状态更新为“部分未知”或“部分启动”。

应用的状态是影响应用常用操作的重要因素，有关应用状态的更多信息，请参阅应用状态（见第 1899 页）。

3.3.5 资源管理

xigemaAS 管理中心提供对 xigemaAS 应用服务器上的资源（包括共享库、连接池、数据源以及 JMS 资源）进行统一管理的功能。

共享库管理

通过 xigemaAS 管理中心可以对共享库进行集中管理，包括新增、删除、配置、查询共享库以及查看共享库信息详情。

新增共享库

xigemaAS 管理中心支持对共享库的管理，包括共享库的新增、删除、编辑、查询及详情查看。共享库创建成功后，在 xigemaAS 管理中心资源仓库 repository/domain/config/library-serverConfig/1.0 目录下会生成 library-serverConfig-1.0.xml 文件。

1. 单击**资源管理 > 共享库 > 新增**，进入**“共享库新增”**页面。
2. 在**“基本信息”**面板输入共享库的基本信息。

图 92: 共享库基本信息面板

a. 在**共享库名称**栏输入共享库的名称。
b. 可选： 在**标签**和**描述**栏分别输入标签和共享库描述信息。

3. 在**“资源文件”**面板中，单击**新增**，从资源文件添加窗口中选择共享库资源的上传方式。

图 93: 共享库资源文件上传

a. 若上传方式为**本地上传**，单击**选择文件**，从登录 xigemaAS 管理中心的浏览器所在主机上选择资源文件，单击**开始上传 > 确定**，将文件上传至 xigemaAS 管理中心资源仓库中。
b. 若上传方式为**仓库资源**，可直接从 xigemaAS 管理中心资源仓库中勾选需要的资源文件，单击**确定**，完成资源文件的选择。

4. 用户可以根据需要在**“共享库资源目录列表”**和**“共享库资源文件列表”**中添加资源文件。

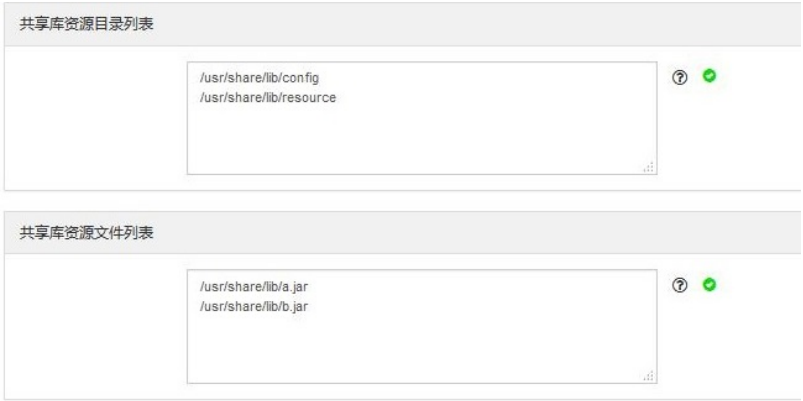


图 94: 添加共享库资源

5. 单击**保存 > 确定**。若提示“**共享库** *sharedLibraryName* **创建成功**”，说明已成功创建共享库。

共享库创建成功后，可继续创建新的共享库或返回共享库列表查看新增共享库。有关新增共享库时界面元素的含义描述和操作规范，请参阅新增共享库（见第 1904 页）。

删除共享库

删除共享库，即清除数据库中的共享库数据、共享库相关的资源文件配置、资源仓库中的资源文件以及 library-serverConfig-1.0.xml 文件中的共享库配置片段。

删除共享库之前，需确保所选共享库未被应用或数据源引用，否则将无法删除。

1. 单击**资源管理 > 共享库**，进入共享库管理页面。
2. 从共享库列表中勾选需要删除的共享库，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除共享库，同时在删除过程中提供共享库删除详情提示列表，列出所选共享库的删除详情。若所选操作目标的操作进度都显示为“**共享库已删除**”，表明已完成所选共享库的删除。

编辑共享库

共享库添加至 xigemaAS 管理中心后，可以通过编辑共享库新增或删除资源文件。

1. 单击**资源管理 > 共享库**，进入共享库管理页面。
2. 从共享库列表中选择需要编辑的共享库，单击**共享库名称 > 编辑**，进入共享库编辑页面。或在共享库列表中，单击需要编辑的共享库的操作图标，直接进入共享库编辑页面。相关数据项的填写要求，请参阅新增共享库（见第 1904 页）。
3. 在共享库编辑页面，可以动态新增或删除资源文件。
   a. 新增资源文件：在**“资源文件”**面板，单击**新增**，选择上传方式。若选择本地上传，单击**选择文件**，从 登录 xigemaAS 管理中心使用的浏览器所在主机选择资源文件，单击**开始上传 > 确定**；若选择仓库资源，从资源仓库中选择资源文件，单击**确定**，完成资源文件的上传。也可以在**“共享库资源目录列表”**或者**“共享库资源文件列表”**中直接添加资源目录或文件。
   b. 删除资源文件：在**“资源文件”**面板，从资源文件列表中勾选需要删除的文件，单击**删除**，即完成资源文件的删除。若所选文件在**“共享库资源目录列表”**或者**“共享库资源文件列表”**中，可以直接删除该文件或目录。
4. 编辑完成后，单击**保存**，完成共享库的编辑操作。

连接池管理

通过 xigemaAS 管理中心可以对连接池进行集中管理，包括新增、删除、配置、查询连接池以及查看连接池信息详情。

新增连接池

xigemaAS 管理中心支持对连接池的管理，包括连接池的新增、删除、编辑、查询及详情查看。

1. 单击**资源管理 > 连接池 > 新增**，进入**“新增连接池”**页面。
2. 在**“基本信息”**面板输入连接池的基本信息。

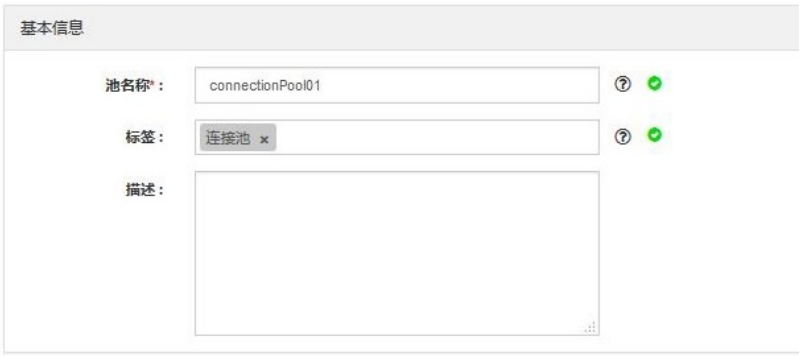


图 95: 连接池基本信息面板

a. 在**池名称**一栏输入连接池的名称。

👉 注：连接池名称保存后不可更改。

b. 可选： 在**标签**和描述栏分别输入标签和连接池的描述信息。

3. 在**“池基本配置”**面板输入连接池的基本配置信息。

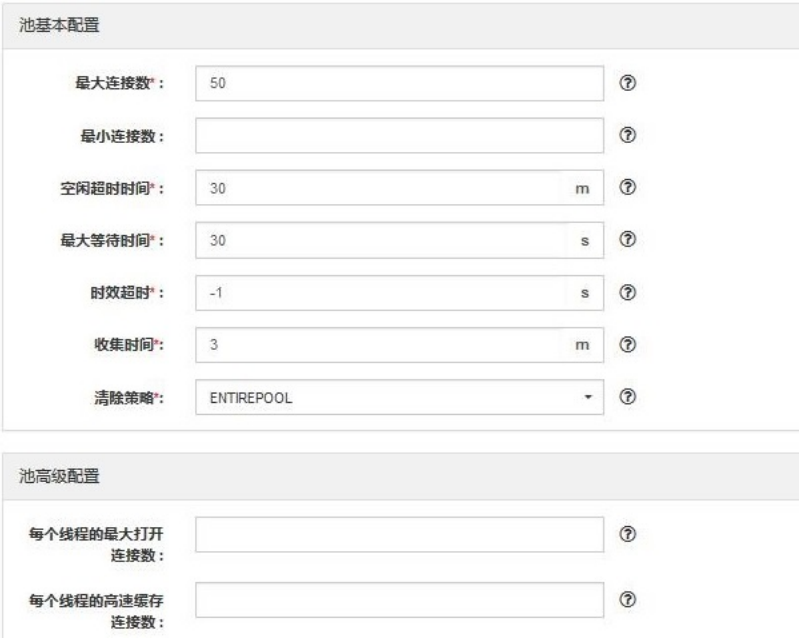


图 96: 连接池配置面板

a. 在**最大连接数**和**最小连接数**栏中输入连接池的最大和最小连接数，其中，最小连接数为可选项。

b. 在**空闲超时时间**、**最大等待时间**、**时效超时**、**收集时间**栏中，输入相应的时间数值。

c. 在**清除策略**栏中，从下拉列表中选择清除策略。

可选清除策略具体包括：

- VALIDATEALLCONNECTIONS：当检测到失效连接时，会测试连接并关闭出现错误的连接。
- FAILINGCONNECTIONONLY：当检测到失效连接时，会关闭出现错误的连接。
- ENTIREPOOL：当检测到失效连接时，会将池中的所有连接都标记为失效，且当这些连接不再使用时，会关闭连接。

4. 可选： 在**“池高级配置”**面板中配置连接池的高级属性。
   a. 在**每个线程的最大打开连接数**一栏输入线程的最大打开连接数。
   b. 在**每个线程的高速缓存连接数**输入线程的高速缓存连接数。
5. 单击**保存 > 确定**。若提示**连接池***connectionPoolName***创建成功**，说明已成功添加连接池到 xigemaAS 管理中心。

连接池创建成功后，可继续创建新的连接池或返回连接池列表查看已创建的连接池。有关新增连接池时界面元素的含义描述和操作规范，请参阅新增连接池（见第 1904 页）。

删除连接池

删除连接池的操作将会删除数据库中和资源仓库中的连接池信息。

删除连接池之前，需确保该连接池未被数据源引用。否则将无法删除该连接池。

1. 单击**资源管理 > 连接池**。
2. 从连接池列表中勾选需要删除的连接池，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除连接池，同时在删除过程中提供连接池删除详情提示列表，列出所选连接池的删除详情。若所选操作目标的操作进度都显示为 **“连接池已删除”** ，表明已完成所选连接池的删除。

   👉 注：defaultConnectionPool 为默认连接池，不可删除。

编辑连接池

可以对已添加至 xigemaAS 管理中心的连接池进行配置编辑。

1. 单击**资源管理 > 连接池**，进入连接池管理页面。
2. 从连接池列表中选择需要编辑的连接池，单击**连接池名称 > 编辑**，进入连接池编辑页面，编辑需要修改的配置项。或在连接池列表中，单击连接池的操作图标，直接进入连接池编辑页面。相关数据项的填写要求，请参阅新增连接池（见第 1904 页）。

   👉 注：连接池名称不可更改。

3. 单击**保存**，完成连接池的配置更改。

数据源管理

通过 xigemaAS 管理中心可以对 JDBC 数据源和 Redis 数据源进行集中管理，部署在 xigemaAS 上的应用程序能够使用 xigemaAS 提供的数据源与数据库完成数据交互。

JDBC 数据源

xigemaAS 管理中心支持对 JDBC 数据源的管理。通过 xigemaAS 管理中心为其管理的应用服务器和集群统一配置 JDBC 数据源，使部署在 xigemaAS 服务器上的应用程序能够使用 xigemaAS 提供的数据源与数据库完成数据交互，方便用户的使用。

需要注意的是，如果在处于启动状态的服务器上对 JDBC 数据源进行新增、删除和修改操作，则在保存时用户进行确认后，配置文件 server.xml 将动态生效，而无需重启服务器。

新增 JDBC 数据源

新增 JDBC 数据源，是使用 xigemaAS 管理中心进行 JDBC 管理的前提条件。

确保 xigemaAS 管理中心已存在部署目标，以及包含数据库驱动的共享库。

1. 单击**资源管理 > 数据源 > JDBC > 新增**，进入**“JDBC 数据源新增”**页面。
2. 配置基本属性。
   a. 在**“基本信息”**面板中输入 JNDI 名称，选择连接池名称（即已添加到 xigemaAS 管理中心的连接池）、资源类型、数据库类型、数据库驱动类型和数据库驱动库（即已添加到 xigemaAS 管理中心的共享库）。用户可以根据需要选填标签和描述信息，用以分类识别数据源。有关资源类型、数据库类型、数据库驱动类型和数据源的类名称的更多信息，请参阅数据源及数据库类型（见第 1912 页）。
   b. 在**“数据库信息”**面板中，输入数据库所在主机的主机名、端口号、数据库的名称、用户名和密码。

   以 xigemaDB 为例，为 JDBC 数据源配置基本属性：

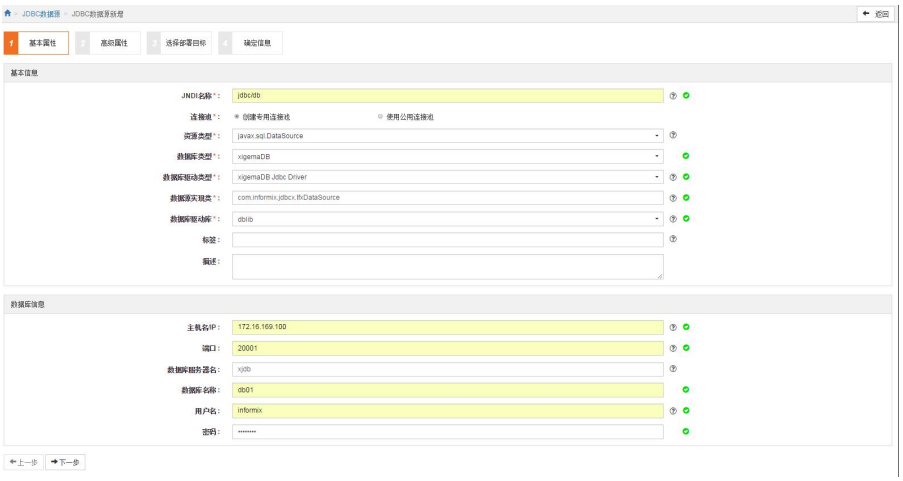


图 97: JDBC 数据源基本属性编辑页面

   c. 基本属性配置完成后，单击**下一步**。
3. 配置高级属性。
   a. 在**“高级属性”**面板中，从下拉菜单中选择合适的共享连接匹配方式，填写每个连接高速缓存语句的最大值。
   b. 选择事务隔离级别，输入 SQL 语句查询的超时时间，勾选是否启用补充 JDBC 跟踪、同步查询和交易超时、参与事务、登记滚动 API、登记供应商 API、启用连接强制类型转换，并选择清除数据库中的连接或将连接返回到连接池中时是提交还是回滚。有关各数据项的含义，请参阅新增 *JDBC* 数据源（见第 1906 页）。

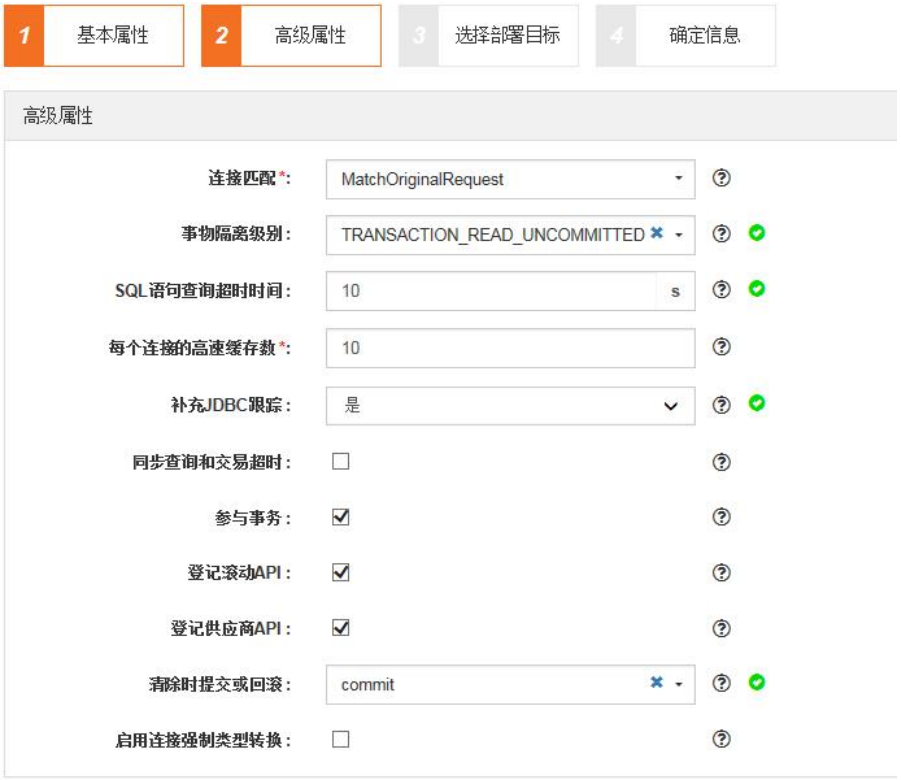


图 98: JDBC 数据源高级属性编辑页面

c. 高级属性配置完成后，单击**下一步**。

4. 选择部署目标。

a. 在**“可选部署目标”**中选择要添加的部署目标，单击 将应用服务器或集群添加到已选部署目标。若选择集群为部署目标，则集群中所有集群成员均为当前 JDBC 数据源的部署目标。

图 99: 部署目标选择面板

b. 部署目标选择完成后，单击**下一步**。

5. 确认所填信息。确认新增数据源相关信息是否填写正确，若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若数据源创建详情列表中所有操作目标的操作进度均提示 **“数据源已创建”**，则该数据源已添加至 xigemaAS 管理中心。

删除 JDBC 数据源

删除 JDBC 数据源，将解除 xigemaAS 管理中心对所选 JDBC 数据源的管理关系。

1. 单击**资源管理 > 数据源 > JDBC**，进入“JDBC 数据源”管理页面。
2. 从 JDBC 数据源列表中勾选需要删除的数据源，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除数据源，同时在删除过程中提供数据源删除详情提示列表，列出所选数据源的删除详情。若所选操作目标的操作进度都显示为“**数据源已删除**”，则完成所选数据源的删除。

编辑 JDBC 数据源

JDBC 数据源新增成功后，可以通过编辑 JDBC 数据源修改数据源属性、添加或移除部署目标、配置连接池。

1. 单击**资源管理 > 数据源 > JDBC**，进入“JDBC 数据源”页面。
2. 从 JDBC 数据源列表中选择需要编辑的 JDBC 数据源，单击**JNDI 名称 > 编辑**，进入 JDBC 数据源编辑页面，编辑需要修改的配置项。或在 JDBC 数据源列表中，单击 JDBC 数据源的操作图标，直接进入 JDBC 数据源编辑页面。相关数据项的填写要求，请参阅新增 *JDBC* 数据源（见第 1906 页）。

   注：

   - JNDI 名称不可更改。
   - JDBC 的部署目标变更时，如果部署目标上已配置 Session 共享并应用了当前数据源，则无法删除该 JDBC 数据源。
3. 单击**保存**，完成 JDBC 数据源的配置更改。

Redis 数据源

xigemaAS 管理中心支持对 Redis 数据库的管理，包括 Redis 数据源的新增、删除、编辑、查询和详情查看。

新增 Redis 数据源

xigemaAS 管理中心 提供对 Redis 单节点和集群的数据源支持。

确保 xigemaAS 管理中心已存在部署目标、连接池、包含 redis 数据库驱动的共享库。

1. 单击**资源管理 > 数据源 > Redis > 新增**，进入“Redis 数据源新增”页面。
2. 配置数据源。

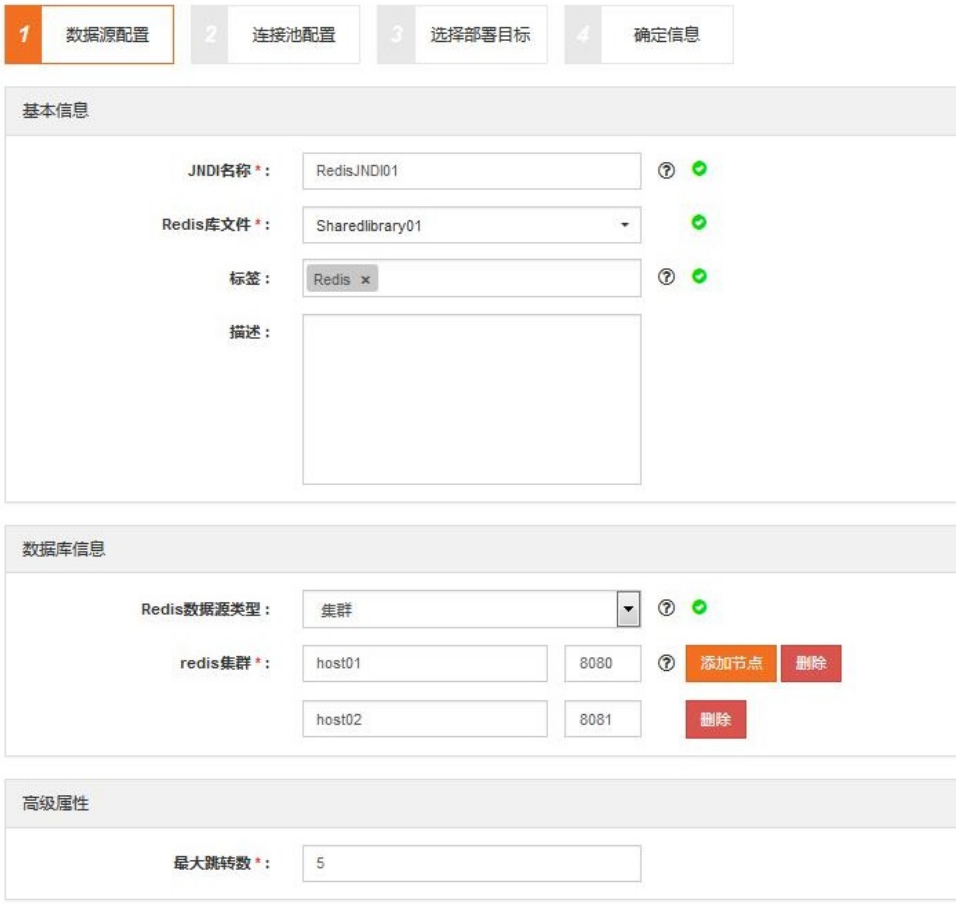


图 100: 数据源配置面板

a. 在**“基本信息”**面板中输入 JNDI 名称，选择 Redis 库文件（即已添加到 xigemaAS 管理中心的共享库）。用户可以根据需要选填标签和描述信息，用以分类识别数据源。

b. 在**“数据库信息”**面板中，选择 Redis 数据源类型。若选择单点类型，需要输入 Redis 主机名称或 IP 地址、 Redis 服务端口号、密码；若选择集群类型，可以添加多个节点，输入 Redis 集群的主机名称或 IP 地址、端口号。

c. 若选择单点类型，在**“高级属性”**面板，输入数据库名称、Socket 通信超时时间、连接超时时间、客户端名称；若选择集群类型，在**“高级属性”**面板，输入最大跳转数，默认最大跳转数为 5。

d. 数据源配置完成后，单击**下一步**。

3. 配置 Redis 连接池。

图 101: 连接池配置面板

a. 在**“基本属性”**面板中分别输入连接池最大连接数、连接池最大空闲数和连接池最小空闲数 。

b. 在**“高级属性”**面板中，输入相关数据项。

c. 连接池配置完成后，单击**下一步**。

4. 选择部署目标。

a. 从**“可选部署目标”**中选择要添加的部署目标，单击 将应用服务器或集群添加到已选部署目标。

图 102: 部署目标选择面板

b. 部署目标选择完成后，单击**下一步**。

5. 确认所填信息。确认新增数据源相关信息是否填写正确，若信息填写有误，返回相应步骤修改信息；若信息填写无误，单击**完成 > 确定**，若数据源创建详情列表中所有操作目标的操作进度均提示 **"数据源已创建"** ，则该数据源已添加至 xigemaAS 管理中心。

Redis 数据源创建成功后，可继续创建新的数据源或返回数据源列表查看已创建的数据源。有关新增 Redis 数据源时界面元素的含义描述和操作规范，请参阅新增 *Redis* 数据源（见第 1908 页）

删除 Redis 数据源

删除 Redis 数据源，将解除 xigemaAS 管理中心对所选 Redis 数据源的管理关系。

1. 单击**资源管理 > 数据源 > Redis**，进入**"Redis 数据源"**管理页面。
2. 从 Redis 数据源列表中勾选需要删除的数据源，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除数据源，同时在删除过程中提供数据源删除详情提示列表，列出所选数据源的删除详情。若所选操作目标的操作进度都显示为 **"数据源已删除"** ，表明已完成所选数据源的删除。

编辑 Redis 数据源

可以对已添加至 xigemaAS 管理中心的 Redis 数据源的信息进行配置编辑。

1. 单击**资源管理 > 数据源 > JDBC**，进入 **"Redis 数据源"**管理页面。
2. 从 Redis 数据源列表中选择需要编辑的 Redis 数据源，单击 **JNDI 名称 > 编辑**，进入 Redis 数据源编辑页面，编辑需要修改的配置项。或在 Redis 数据源列表中，单击 Redis 数据源的操作图标，直接进入 Redis 数据源编辑页面。相关数据项的填写要求，请参阅新增 *Redis* 数据源（见第 1908 页）。

   注：

   - JNDI 名称不可更改。
   - Redis 的部署目标变更时，如果部署目标上已配置 Session 共享并应用了当前数据源，则无法删除该 Redis 数据源。
3. 单击**保存**，完成 Redis 数据源的配置更改。

JMS 管理

通过 xigemaAS 管理中心可以对 JMS 资源进行集中管理。包括JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂、JMS 队列、JMS 主题以及 JMS 激活规范。

JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂

通过 xigemaAS 管理中心可以对 JMS 连接工厂、JMS 队列连接工厂、以及 JMS 主题连接工厂进行新增、删除、配置、查询和信息详情查看等操作。

- JMS 连接工厂用于创建与 JMS 目标的相关 JMS 提供程序的连接，以便进行点到点的消息传递和发布/预订消息传递。
- JMS 队列连接工厂用于创建与 JMS 队列的关联 JMS 提供程序的连接，以便进行点到点的消息传递。
- JMS 主题连接工厂用于创建与 JMS 目标的关联 JMS 提供程序的连接，以便进行发布/预订消息传递。

新增 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂

新增到 xigemaAS 管理中心的 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂会显示在管理列表中，以便进行集中管理。

请确保已部署连接器类型的应用。

新增 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂的操作相同，以新增 JMS 连接工厂为例介绍相关操作步骤。

1. 单击 **JMS > 连接工厂 > 新增**，进入 JMS 连接工厂的新增页面。
2. 在 **“基本信息”** 面板输入 JMS 连接工厂的基本信息。

图 103: JMS 连接工厂基本信息面板

   a. 在 **JNDI 名称**栏输入要新增的 JMS 连接工厂的 JNDI 名称。
   b. 在 **JMS 资源适配器**栏，从下拉列表中选择已部署的连接器类型的应用。
   c. 选择连接池，xigemaAS 管理中心可以为 JMS 连接工厂创建专用连接池，也可以直接选择已创建的公用连接池。
   d. 可选： 在**标签**、**描述**栏分别输入 JMS 连接工厂的标签和描述信息，用于标识和分类管理 JMS 连接工厂。
3. 在 **“属性信息”** 面板可以对 JMS 连接工厂的属性值进行编辑。

图 104: JMS 连接工厂属性信息面板

4. 单击**保存 > 确定**，若提示 “JMS **资源创建成功**”，说明已成功创建 JMS 连接工厂。

JMS 连接工厂新增完成后，需要重启应用所在部署目标。在 JMS 连接工厂列表中的**“部署目标”**列单击**详情**，进入应用部署目标详情页面，选择需要重启的应用服务器，单击**重启**按钮，使配置生效。

有关新增 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂时界面元素的含义和操作规范，请参阅新增 *JMS* 资源 （见第 1910 页）。

删除JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂

通过 xigemaAS 管理中心可以集中删除不需要进行管理的 JMS 连接工厂、JMS 队列连接工厂、或JMS 主题连接工厂。

删除 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂的操作相同，以删除 JMS 连接工厂为例介绍相关操作步骤。

1. 单击 **JMS > 连接工厂**，进入 JMS 连接工厂管理页面。
2. 从 JMS 连接工厂管理列表中勾选需要删除的 JMS 连接工厂，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除 JMS 连接工厂，同时在删除过程中提供 JMS 连接工厂删除详情提示列表，列出所选 JMS 连接工厂的删除详情。若所选操作目标的操作进度都显示为 “JMS **资源已删除**”，表明 JMS 连接工厂删除成功。

编辑JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂

xigemaAS 管理中心支持对 JMS 连接工厂、JMS 队列连接工厂、以及JMS 主题连接工厂基本信息的编辑和修改，并且在编辑的同时可以配置所用的连接池。

编辑 JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂的操作相同，以编辑 JMS 连接工厂为例介绍相关操作步骤。

1. 单击 **JMS > 连接工厂**，进入 JMS 连接工厂管理页面。
2. 从 JMS 连接工厂管理列表中选择需要编辑基本信息或配置所用连接池的 JMS 连接工厂，单击所选 JMS 连接工厂的名称，进入所选 JMS 连接工厂的详情页面。

   JMS 连接工厂详情页面支持 JMS 连接工厂基本信息以及所用连接池配置信息的查看。
3. 单击**编辑**按钮，进入JMS 连接工厂的编辑页面。
   - 在**“基本信息”**面板，可以重新选择应用，并且修改当前 JMS 连接工厂的标签和描述信息。
   - 在**“连接池”**面板，可以编辑 JMS 连接工厂所用连接池的基本配置和高级配置。如下图所示：

图 105: 配置 JMS 连接工厂所用连接池

4. 编辑完成后，单击**保存 > 确定**，若提示"**保存成功**"，则 JMS 连接工厂信息更新完成。

JMS 队列、JMS 主题

通过 xigemaAS 管理中心可以配置由已安装的资源适配器提供的 JMS 队列或 JMS 主题类型的一个或多个实例。

新增 JMS 队列、JMS 主题

新增到 xigemaAS 管理中心的 JMS 队列和 JMS 主题会显示在管理列表中，以便进行集中管理。

请确保已部署连接器类型的应用。

新增 JMS 队列和 JMS 主题的操作相同，以新增 JMS 队列为例介绍相关操作步骤。

1. 单击 **JMS > 队列 > 新增**，进入 JMS 队列新增页面。
2. 在 **"基本信息"** 面板输入 JMS 队列的基本信息。

图 106: JMS 队列基本信息面板

a. 在 **JNDI 名称**栏输入要新增的 JMS 队列的 JNDI 名称。

b. 在 **JMS 资源适配器**栏，从下拉列表中选择已部署的连接器类型的应用。

c. 可选： 在**标签**、**描述**栏分别输入 JMS 队列的标签和描述信息，用于标识和分类管理 JMS 队列。

3. 在**“属性信息”**面板可以对 JMS 队列的属性值进行编辑。

图 107: JMS 队列属性信息面板

4. 单击**保存 > 确定**，若提示 “JMS **资源创建成功**”，说明已成功创建 JMS 队列。

JMS 队列和 JMS 主题新增完成后，需要重启应用所在部署目标。在 JMS 队列和 JMS 主题列表中的**“部署目标”**列单击**详情**，进入应用部署目标详情页面，选择需要重启的应用服务器，单击**重启**按钮，使配置生效。

有关新增 JMS 队列、JMS 主题时界面元素的含义和操作规范，请参阅新增 *JMS* 资源 （见第 1910 页）。

删除 JMS 队列、JMS 主题

通过 xigemaAS 管理中心可以集中删除不需要进行管理的 JMS 队列、JMS 主题。

删除 JMS 队列和 JMS 主题的操作相同，以删除 JMS 队列为例介绍相关操作步骤。

1. 单击 **JMS > 队列**，进入 JMS 队列管理页面。
2. 从 JMS 队列管理列表中勾选需要删除的 JMS 队列，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除 JMS 队列，同时在删除过程中提供 JMS 队列删除详情提示列表，列出所选 JMS 队列的删除详情。若所选操作目标的操作进度都显示为“JMS **队列已删除**”，表明 JMS 队列删除成功。

编辑 JMS 队列、JMS 主题

xigemaAS 管理中心支持对 JMS 队列和 JMS 主题基本信息的编辑。

编辑 JMS 队列和 JMS 主题的操作相同，以编辑 JMS 队列为例介绍相关操作步骤。

1. 单击 **JMS > 队列**，进入 JMS 队列管理页面。
2. 从 JMS 队列管理列表中选择需要编辑基本信息或属性值的 JMS 队列，单击所选 JMS 队列的名称，进入所选 JMS 队列的详情页面。

   JMS 队列详情页面支持 JMS 队列基本信息以及属性信息的查看。
3. 单击**编辑**按钮，进入JMS 队列的编辑页面。
   - 在**“基本信息”**面板，可以修改当前 JMS 队列的标签和描述信息。
   - 在**“属性信息”**面板，可以修改当前 JMS 队列的属性值。
4. 编辑完成后，单击**保存 > 确定**，若提示“**保存成功**”，则 JMS 队列信息更新完成。

JMS 激活规范

通过 xigemaAS 管理中心可以配置由已安装的资源适配器提供的符合 JavaEE 连接器体系结构（JCA）规范的 JMS 激活规范。

新增 JMS 激活规范

新增到 xigemaAS 管理中心的 JMS 激活规范会显示在管理列表中，以便集中管理。

请确保已将包含消息驱动 Bean 的应用程序部署到应用服务器，并且在该应用程序的部署目标上，同时部署了相应的 JMS 资源适配器应用程序。

1. 单击 **JMS ＞ 激活规范 ＞ 新增**，进入 JMS 激活规范新增页面。
2. 在**“基本信息”**面板输入 JMS 激活规范的基本信息。

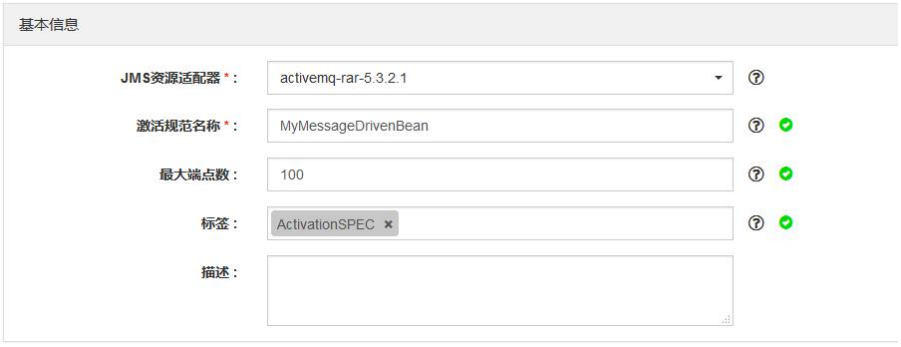


图 108: JMS 激活规范基本信息面板

   a. 在 **JMS 资源适配器**栏，从下拉列表中选择已部署的连接器类型的应用。
   b. 在**激活规范名称**栏输入当前 JMS 激活规范的名称。
   c. 在**最大端点数**栏输入要分派至的最大端点数。
   d. 可选： 在**标签**、**描述**栏分别输入 JMS 激活规范的标签和描述信息，用于标识和分类管理 JMS 激活规范。

3. 在**“属性信息”**面板编辑 JMS 激活规范的属性值。

图 109: JMS 激活规范属性信息面板

4. 单击**保存 ＞ 确定**，若提示 “JMS **资源创建成功**”，说明JMS 激活规范已成功创建 。

JMS 激活规范新增完成后，需要重启应用所在部署目标。在 JMS 激活规范列表中的**“部署目标”**列单击**详情**，进入应用部署目标详情页面，选择需要重启的应用服务器，单击**重启**按钮，使配置生效。

有关新增 JMS 激活规范时界面元素的含义和操作规范，请参阅新增 *JMS* 资源 （见第 1910 页）。

删除 JMS 激活规范

通过 xigemaAS 管理中心可以删除不需要进行管理的 JMS 激活规范。

1. 单击 **JMS > 激活规范**，进入 JMS 激活规范管理页面。
2. 从 JMS 激活规范管理列表中勾选需要删除的 JMS 激活规范，单击**删除 > 确定**。

   xigemaAS 管理中心支持批量删除 JMS 激活规范，同时在删除过程中提供 JMS 激活规范删除详情提示列表，列出所选 JMS 激活规范的删除详情。若所选操作目标的操作进度都显示为"JMS **激活规范已删除**"，表明 JMS 激活规范删除成功。

编辑 JMS 激活规范

xigemaAS 管理中心支持对 JMS 激活规范基本信息的编辑。

1. 单击 **JMS > 激活规范**，进入 JMS 激活规范管理页面。
2. 从 JMS 激活规范管理列表中选择需要编辑基本信息或属性值的 JMS 激活规范，单击所选 JMS 激活规范的名称，进入所选 JMS 激活规范的详情页面。

   JMS 激活规范详情页面支持 JMS 激活规范基本信息以及属性信息的查看。
3. 单击**编辑**按钮，进入JMS 激活规范的编辑页面。
   - 在**“基本信息”**面板，可以修改当前 JMS 激活规范的激活规范名称、最大端点数、标签和描述信息。
   - 在**“属性信息”**面板，可以修改当前 JMS 激活规范的属性值。
4. 编辑完成后，单击**保存 > 确定**，若提示"**保存成功**"，则 JMS 激活规范信息更新完成。

3.3.6 系统管理

xigemaAS 管理中心系统管理功能支持对系统参数的编辑和维护。

参数配置

xigemaAS 管理中心提供参数配置功能以配置管理中心的 IP 地址。

1. 单击**系统管理 > 参数配置**进入参数配置页面。
2. 选择 xigemaAS 管理中心 IP 地址，可选的 IP 地址为当前 xigemaAS 管理中心所在主机（包括物理计算机及虚拟机）的 IP 地址。单击**确认**，完成管理中心 IP 地址的选择。

   注：通过参数配置功能变更 xigemaAS 管理中心 IP 地址后，若应用服务器不可达，xigemaAS 管理中心仍会保持变更前的 IP 地址不变。待应用服务器可达后，变更各应用服务器上保存的 xigemaAS 管理中心 IP 地址。变更后，不影响管理中心各模块的正常工作。

数据备份

xigemaAS 管理中心自动将数据库进行备份，当操作不当导致误删数据时，可以恢复数据，有效防范因误操作等原因对业务数据造成的不可逆的影响。

通过 xigemaAS 管理中心可以对数据备份功能进行定制，根据需要来自定义数据备份的频率、每次备份的具体时间、备份的数量、以及备份数据的存储地址，以便根据实际需要灵活选择备份数据进行数据恢复（见第 1868 页）。

1. 单击**系统管理 > 数据备份**进入数据备份参数设置页面，如下图所示：

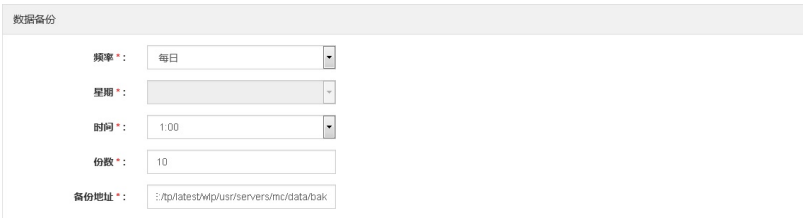


图 110: 数据备份参数设置面板

2. 从下拉列表中选择数据备份的频率和时间，设置数据备份的频率和开始备份的具体时间。xigemaAS 管理中心默认每日凌晨1:00进行数据备份。
3. 输入数据备份的数量，xigemaAS 管理中心默认自动备份10份数据。

   若数据备份频率为**每日**，则 xigemaAS 管理中心默认自动保留10日内的业务数据；若数据备份频率为**每周**，则 xigemaAS 管理中心默认自动保留10周内的业务数据。

   注：若备份的数据份数超过设置的份数，会按时间顺序自动清除最早的数据，使备份数据的份数始终保持在已设置的份数。
4. 输入备份数据存储的地址。xigemaAS 管理中心默认备份数据存储地址为 {wlp_install_dir}/usr/servers/mc/data/bak，其中 {wlp_install_dir} 为 xigemaAS 的安装目录。
5. 单击**确认**，完成数据备份的参数设置。

数据恢复

xigemaAS 管理中心提供自动备份数据功能，数据丢失时，可以使用备份数据进行数据恢复。

数据恢复前，请先停止 xigemaAS 管理中心。

类 UNIX 系统环境下执行以下命令停止 xigemaAS 管理中心：

```
./server stop mc
```

Windows 系统环境下执行以下命令停止 xigemaAS 管理中心：

```
server stop mc
```

1. 进入数据备份目录，默认备份数据存放地址为 {wlp_install_dir}/usr/servers/mc/data/bak，其中 {wlp_install_dir} 为 xigemaAS 的安装目录。
2. 从已备份数据中选择需要进行恢复的数据，解压后与 xigemaAS 管理中心当前数据（wlp_install_dir}/usr/servers/mc/data/mcdb）进行替换。
3. 启动 xigemaAS 管理中心。

审计日志配置

配置审计日志的清理策略以自动清理审计日志。

1. 在左侧导航栏中，单击**系统管理 > 日志配置**。

   进入**“日志配置”**页面：

2. 配置清理策略。

   审计日志的清理策略包括删除、归档和不清理三种。在定义的时间间隔后，审计日志将自动删除或归档。如果选择归档，则要指定归档目录，即归档后审计日志文件的存储位置；归档文件为 Excel 格式，文件名称为归档日期，如 2019-08-05.xlsx。通过存储的审计日志文件，用户可以查询并追踪资源的操作记录。如果选择不清理，则日志永久保存。

   该时间间隔默认是 60 天。该值必须大于0。单位：天。

3.3.7 用户管理

用户管理功能用于管理所有用户以及这些用户的基本信息。

管理用户

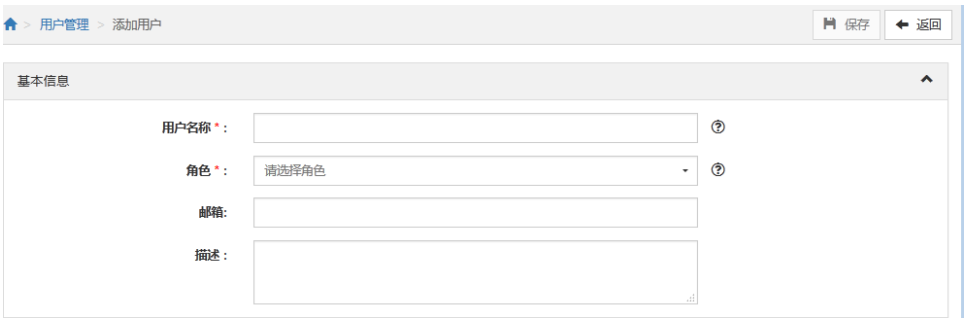
通过用户管理功能，系统管理员可以创建用户、删除用户、修改用户基本信息以及重置用户密码。

xigemaAS 管理中心的用户角色包括系统管理员和普通用户，二者的区别在于权限不同：

- 系统管理员：可执行任何操作。
- 普通用户：可执行部分操作。不允许该类用户执行如下功能：
  - 节点管理，包括节点增加、删除、安装 xigemaAS、卸载 xigemaAS。
  - 系统管理。普通用户无法使用“日志配置”功能。
  - 用户管理 。
  - 审计日志。

新增用户

1. 单击**用户管理**，进入**“用户管理”**页面。
2. 在页面左上角，单击**新增**按钮，进入用户新增页面。



3. 填写用户基本信息。

   注：用户名是用户登录的唯一标识，保存后将不可修改。

4. 在页面右上角，单击**保存**按钮。
5. 在弹出的提示修改页面，单击**确定**。
6. 在弹出的提示修改成功页面，若要继续添加用户，单击**继续添加**；确认完成新增，单击**返回列表**。

用户新增成功。对于新增用户，其密码默认与用户名相同。用户登录后，可以自行修改密码。

修改用户基本信息

可以修改除用户名之外的所有用户基本信息。

1. 单击**用户管理**，进入**“用户管理”**页面。
2. 找到目标用户名所在行，单击其“操作”列中的 图标。
3. 修改用户基本信息。
4. 在页面右上角，单击**保存**按钮。
5. 在弹出的提示修改页面，单击**确定**。
6. 在弹出的提示修改成功页面，单击**确定**。

删除用户

1. 单击**用户管理**，进入**“用户管理”**页面。
2. 有两种方法可删除用户。
   - 方法一：
     1. 找到目标用户名所在行，单击其“操作”列中的 图标。
     2. 单击**确定**。
   - 方法二：
     1. 找到目标用户名所在行，选中其第一列中置灰的 图标。
     2. 在页面左上角，单击**删除**按钮。
3. 在弹出的提示删除页面，单击**确定**。
4. 在弹出的提示删除成功页面，单击**确定**。

重置密码

重置密码就是将密码还原成与用户名一致。

1. 单击**用户管理**，进入**“用户管理”**页面。
2. 找到目标用户名所在行，单击其“操作”列中的 图标。
3. 在弹出的提示重置页面，单击**确定**。
4. 在弹出的提示重置成功页面，单击**确定**。

修改当前用户密码

修改当前用户密码和注销用户。

1. 修改登录密码：
   a. 单击 > **密码修改**，进入**“密码修改”**页面。

b. 输入新密码，再次输入新密码进行确认。完成后单击**保存**，若提示**密码修改成功**则完成新密码的设置，跳转到 xigemaAS 管理中心登录页面。

2. 忘记密码后修改登录密码：

a. 在 /home/wlp/usr/servers/mc 目录下打开 server.xml 配置文件，找到如下配置片段：

```
#other configurations
 <basicRegistry id="MC_USER">
     <user name="admin" password="{sm4}VnUIDL6DvvLAR5+3irh1BQ==" />
 </basicRegistry>
#other configurations
```

b. 修改 password 的值为当前密码。

c. 重启 xigemaAS 管理中心并使用修改后的密码登录。

3. 注销用户：单击 > **注销**，注销当前用户。

3.3.8 审计日志

xigemaAS 管理中心审计日志功能支持记录对象相关的管理操作并提供对历史操作的查询。

只有角色为系统管理员的用户才能使用审计日志功能。

自动记录审计日志

审计日志功能支持自动记录操作目标类型列表中所有对象的操作，包括这些对象的增加、删除、启动、停止、编辑和重启等操作。

审计日志功能自动记录历史操作信息，并支持将这些信息保存到指定的日志文件中。用户依据这些记录，可以执行操作查询，进而进行对象变动跟踪、相关数据分析和安全审计。

查询审计日志

通过审计日志查询历史操作信息。

审计日志默认保留最近 60 天的操作记录，详细记录操作相关的上下文信息，并支持从操作时间、操作用户、对象类型等维度来查询相关历史操作。

1. 单击**审计日志**。进入审计日志页面。

该页面默认展示最近操作。

2. 查询。可按查询条件查找相关日志。

a. 输入查询条件，如开始时间、结束时间等。

b. 单击**搜索**。

3.4 参考

本章主要介绍 xigemaAS 管理中心涉及到的术语、图标、列表、界面上各数据项的填写要求，并对节点上 xigemaAS 的安装状态、服务器状态、集群状态和应用状态的转换流程进行详细说明。

3.4.1 xigemaAS 管理中心术语

本节对 xigemaAS 管理中心中出现的术语进行解释。

标签

tag

一种对 xigemaAS 管理中心的对象和资源进行识别、分类的关键词。利用标签对节点、服务器、集群、应用、和资源进行分组管理和检索。

节点

node

主机上的一个 xigemaAS 物理安装，由 xigemaAS 管理中心管理。

本机节点

local node

在 xigemaAS 管理中心所在主机上的节点，无需远程认证。

远程节点

remote node

在远程主机上的节点，须填写认证信息。

xigemaAS Agent

xigemaAS Agent 内置了服务器 mc-agent 和应用 agent.war，面向管理中心提供 HTTPS 服务。Agent 安装完成后，可以基于 Agent 管理 Web 服务器。同时，通过 Agent 还可以监控被其管理的应用服务器，若进程被意外关闭，Agent 会进行故障恢复，自动启动该应用服务器。

主机

host

物理计算机或虚拟机。

SSH

secure shell

建立在应用层基础上的安全协议。利用 SSH 协议可以有效防止远程管理的信息泄露问题。xigemaAS 管理中心通过 SSH 协议对注册的远端节点进行管理。

服务器

server

包括 xigemaAS 管理中心管理的应用服务器和 Web 服务器。

应用服务器

application server

节点上的 xigemaAS 服务器实例。

Web 服务器

Web server

可通过 xigemaAS 管理中心在本地或远程主机上安装 Web 服务器。可选 Web 服务器 为 Nginx、Apache 和 THS。

虚拟主机

virtual host

Web 服务器上的虚拟主机，用于设置应用的负载均衡。

平滑重启

reload configuration

对于 Web 服务器 Nginx，重新加载 配置文件 nginx.conf。

对于 Web 服务器 Apache，重新加载配置文件 httpd.conf。

模板

template

创建应用服务器和集群时所使用的配置模板。

集群

cluster

一组具有相同配置并且部署了相同应用的应用服务器集合。同一集群下的各个 xigemaAS 服务器，除端口等需要根据本地机器的情况进行定制外，其他部分的配置（如配置功能部件，部署应用）必须完全相同。

集群成员

cluster member

属于某个集群的 xigemaAS 服务器。

会话

Session

服务器端使用的一种记录客户端状态的机制。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上，这就是一次会话过程。客户端浏览器再次访问时只需要从该会话中查找该客户端的状态。

HttpSession

HttpSession 是 Java 平台对 Session 机制的实现规范，代表的是服务器和客户端 的一次会话过程。

Session 共享

集群成员共享一个 HttpSession 对象。

Session 存储数据源

存储 HttpSession 对象的数据源。

资源

resource

xigemaAS 管理中心中的资源包括共享库、连接池、数据源、以及 JMS。

资源仓库

resource warehouse

建立在 xigemaAS 管理中心端，负责存储 xigemaAS 安装包、服务器和集群的配置信息、应用文件、应用程序部署包、共享库资源文件等资源的存储仓库。

共享库

shared library

可被多个应用程序使用的资源文件。

私有共享库

private library

库引用的列表，库类实例是此类加载器特有的，与来自其他类加载器的类实例无关。

公有共享库

common library

库引用的列表，会与其他类加载器共享库类实例。

类加载优先顺序

delegation

控制父类加载器是用在此类加载器之前还是之后。若选择父优先，那么在搜索类路径之前，授权给直接父代；若选择子优先，那么在搜索类路径之后，授权给直接父代。

连接池

connection pool

连接池允许多个客户端使用缓存起来的、共享的、可重用的连接对象以连接数据库。

数据源

Data Source

数据源中存储了所有建立数据库连接的信息。通过提供正确的数据源名称，可以找到相应的数据库连接。

3.4.2 xigemaAS 管理中心所用图标

xigemaAS 管理中心使用图标来表示不同的对象、操作和状态。

对象图标

下表列出了 xigemaAS 管理中心对象图标。

图标	含义
	节点
	服务器
	应用服务器
	Web 服务器
	集群
	应用
	资源管理
	共享库
	连接池
	数据源
	JDBC
	Redis
	JMS
	JMS 连接工厂
	JMS 队列连接工厂
	JMS 主题连接工厂
	JMS 队列
	JMS 主题
	JMS 激活规范
	用户管理
	审计日志
	系统管理
	参数配置
	数据备份
	日志配置

图标	含义
	登录用户管理
	帮助

操作图标

可以在列表中通过操作图标对相应对象执行操作。下表列出了 xigemaAS 管理中心的操作图标及相关操作。

图标	含义
	1. 注册 ： 注册节点。
	2. 创建 ： 创建服务器、集群。
	3. 部署：部署应用。
	4. 新增：新增 Web 服务器、虚拟主机、共享库、连接池、JDBC、Redis、资源文件、集群成员。
	5. 添加：添加子代、系统属性、JVM 选项。
	环境检查：对节点、Web 服务器进行环境检查。
	安装 xigemaAS：为当前节点安装 xigemaAS。
	卸载 xigemaAS：卸载当前节点上的 xigemaAS。
	启动：启动服务器、集群、集群成员、应用。
	停止：停止服务器、集群、集群成员、应用。
	重启：重启服务器、集群、集群成员、应用。
	1. 下载 Agent。
	2. 下载日志：下载服务器、集群、集群成员日志。
	链接：当前应用访问链接。
	子模块：查看当前应用子模块详情。
	查看服务器：查看当前节点上创建的服务器。
	查看应用：查看当前服务器、集群或虚拟主机上的应用。
	虚拟主机管理
	平滑重启
	更新：应用更新
	编辑

图标	含义
	删除、强制删除
	保存
	返回
	搜索
	定制列表

- 有关应用服务器常用操作的更多信息，请参见应用服务器的常用操作（见第 1828 页）。
- 有关 Web 服务器常用操作的更多信息，请参见 *Web* 服务器的常用操作（见第 1837 页）。
- 有关集群常用操作的更多信息，请参见集群的常用操作（见第 1843 页）。
- 有关应用常用操作的更多信息，请参见应用的常用操作（见第 1851 页）。

状态图标

下表列出了节点、服务器、集群及应用的状态图标。

图标	含义
	1. 已安装：当前节点已安装 xigemaAS。
	2. 启动：当前服务器或集群处于启动状态。
	3. 启动：当前应用处于启动状态。
	1. 部分启动：当前集群处于部分启动状态。
	2. 部分启动：当前应用处于部分启动状态。
	1. 启动中：当前服务器或集群处于启动中状态。
	2. 启动中：当前应用处于启动中状态。
	1. 停止中：当前服务器或集群处于停止中状态。
	2. 停止中：当前应用处于停止中状态。
	1. 卸载中：当前集群处于卸载中状态。
	2. 卸载中：当前应用处于卸载中状态。
	安装中：当前节点正在安装 xigemaAS，处于安装中状态。
	卸载中：当前节点正在卸载 xigemaAS，处于卸载中状态。
	未安装：当前节点未安装 xigemaAS，处于未安装状态。
	安装失败：当前节点安装 xigemaAS 失败。

图标	含义
	卸载失败：当前节点卸载 xigemaAS 失败。
	1. 停止：当前服务器处于停止状态。
	2. 停止：当前集群处于停止状态。
	3. 停止：当前应用处于停止状态。
	1. 未知：当前服务器处于未知状态。
	2. 未知：当前集群处于未知状态。
	3. 未知：当前应用处于未知状态。
	1. 部分未知：当前集群处于部分未知状态。
	2. 部分未知：当前应用处于部分未知状态。

- 有关节点上 xigemaAS 安装状态的更多信息，请参见*xigemaAS 安装状态*（见第 1883 页）。
- 有关应用服务器状态的更多信息，请参见应用服务器状态（见第 1887 页）。
- 有关 Web 服务器状态的更多信息，请参见 *Web 服务器状态*（见第 1892 页）。
- 有关集群状态的更多信息，请参见集群状态（见第 1896 页）。
- 有关应用状态的更多信息，请参见应用状态（见第 1899 页）。

3.4.3 xigemaAS 管理中心列表定制

xigemaAS 管理中心使用列表管理节点、服务器、集群、应用、共享库、连接池、JDBC 数据源、Redis 数据源、用户管理和审计日志。通过列表可以浏览目标信息，使用列表定制功能，用户可以自定义列表显示项，操作便捷，直观展示不同用户关注的不同信息。

节点管理列表

操作序列：单击**节点**，进入节点管理页面。单击**定制列表**自定义节点管理列表的列表项。

默认列表项：节点名称、主机名/ IP、端口、用户名、xigemaAS 安装目录、代理类型、xigemaAS 安装状态。

自定义项：xigemaAS 用户目录、认证方式、Java 运行时 (JRE) 目录、标签、描述、注册时间、最新修改时间。

应用服务器管理列表

操作序列：单击**服务器 > 应用服务器**，进入应用服务器管理页面。单击**定制列表**自定义应用服务器管理列表的列表项。

默认列表项：应用服务器名称、应用服务器所在节点、所属集群、应用服务器目录、状态、最近启停时间。

自定义项：HTTP 端口、HTTPS 端口、Java 运行时 (JRE) 目录、标签、描述、创建时间、最新修改时间。

特定应用服务器应用列表

操作序列：单击**服务器 > 应用服务器**，进入应用服务器管理页面。在应用服务器管理列表操作栏，单击 进入特定应用服务器应用列表页面。单击**定制列表**自定义特定应用服务器应用列表的列表项。

默认列表项：应用名称、应用类型、应用状态、应用最近启停时间。

自定义项：应用上下文、应用位置、标签、描述、部署时间、最新修改时间。

Web 服务器管理列表

操作序列：单击 **服务器 > Web 服务器**，进入 Web 服务器管理页面。单击**定制列表**自定义 Web 服务器管理列表的列表项。

默认列表项：Web 服务器名称、服务器类型、主机名或 IP 地址、用户名、执行文件路径、代理类型、状态、最近启停时间。

自定义项：配置文件路径、日志目录、认证方式、Java 运行时（JRE）目录、标签、描述、创建时间、最新修改时间。

特定 **Web** 服务器负载应用列表

操作序列：单击**服务器 > Web 服务器**，进入 Web 服务器管理页面。在Web 服务器管理列表操作栏，单击 进入特定 Web 服务器负载应用列表页面。单击**定制列表**自定义特定 Web 服务器应用列表的列表项。

默认列表项：应用名称、应用类型、应用状态、应用最近启停时间、负载虚拟主机。

自定义项：应用上下文、标签、描述、部署时间、最新修改时间。

虚拟主机管理列表

操作序列：单击**服务器 > Web 服务器 >** ，进入虚拟主机管理页面。

默认列表项：虚拟主机名称、域名、监听信息。

自定义项：标签、描述、创建时间、最新修改时间。

特定虚拟主机负载应用列表

操作序列：单击**服务器 > Web 服务器 >** ，进入虚拟主机管理页面。在虚拟主机管理列表操作栏，单击 进入特定虚拟主机负载应用列表页面。

默认列表项：应用名称、应用类型、应用状态、应用最近启停时间。

自定义项：应用上下文、标签、描述、部署时间、最新修改时间。

集群管理列表

操作序列：单击**集群**，进入集群管理页面。单击**定制列表**自定义集群管理列表的列表项。

默认列表项：集群名称、状态、集群成员。

自定义项：标签、描述、创建时间、最新修改时间。

集群成员详情列表

操作序列：单击**集群 > 详情**，进入集群成员详情页面。单击**定制列表**自定义集群成员详情列表的列表项。

默认列表项：服务器名称、所属节点、服务器目录、状态、最近启停时间、标签。

自定义项：HTTP 端口、HTTPS 端口、Java 运行时 (JRE) 目录、标签、描述、创建时间、最新修改时间。

特定集群应用列表

操作序列：单击**集群**，进入集群管理页面。在集群管理列表操作栏，单击 进入特定集群应用列表页面。单击**定制列表**自定义特定集群应用列表的列表项。

默认列表项：应用名称、应用类型、应用状态。

自定义项：应用上下文、标签、描述、部署时间、最新修改时间。

应用管理列表

操作序列：单击**应用**，进入应用管理页面。单击**定制列表**自定义应用管理列表的列表项。

默认列表项：应用名称、类型、状态、部署目标。

自定义项：应用上下文、应用位置、标签、描述、部署时间、最新修改时间。

应用部署目标详情列表

操作序列：单击**应用 > 详情**，进入应用部署目标详情页面。单击**定制列表**自定义应用部署目标详情列表的列表项。

默认列表项：服务器名称、节点名称、所属集群、服务器状态、应用状态、应用最近启停时间。

自定义项：应用位置、最新修改时间。

共享库管理列表

操作序列：单击**资源管理 > 共享库**，进入共享库管理页面。单击**定制列表**自定义共享库管理列表的列表项。

默认列表项：共享库名称、资源文件、引用应用、应用数据源、标签。

自定义项：描述、创建时间、最新修改时间。

连接池管理列表

操作序列：单击**资源管理 > 连接池**，进入连接池管理页面。单击**定制列表**自定义连接池管理列表的列表项。

默认列表项：连接池名称、标签、最新修改时间。

自定义项：描述、创建时间。

JDBC 数据源管理列表

操作序列：单击**资源管理 > 数据源 > JDBC**，进入 JDBC 数据源管理页面。单击**定制列表**自定义 JDBC 数据源管理列表的列表项。

默认列表项：JNDI 名称、数据库类型、数据库驱动类型、部署目标、标签。

自定义项：描述、创建时间、最新修改时间。

JDBC 数据源部署目标详情列表

操作序列：单击**资源管理 > 数据源 > JDBC > 详情**，进入 JDBC 数据源部署目标详情页面。单击**定制列表**自定义 JDBC 数据源部署目标详情列表的列表项。

默认列表项：服务器名称、服务器所在节点、所属集群、状态。

自定义项：服务器目录、最近启停时间、最新修改时间。

Redis 数据源管理列表

操作序列：单击**资源管理 > 数据源 > Redis**，进入 Redis 数据源管理页面。单击**定制列表**自定义 Redis 管理列表的列表项。

默认列表项：JNDI 名称、部署目标、标签、最新修改时间。

自定义项：描述、创建时间。

Redis 数据源部署目标详情列表

操作序列：单击**资源管理 > 数据源 > Redis > 详情**，进入 Redis 数据源部署目标详情页面。单击**定制列表**自定义 Redis 数据源部署目标详情列表的列表项。

默认列表项：服务器名称、服务器所在节点、所属集群、状态。

自定义项：服务器目录、最近启停时间、最新修改时间。

JMS 资源管理列表

JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂、JMS 队列、JMS 主题以及 JMS 激活规范管理列表的默认列表项和自定义项相同，操作序列相似，以 JMS 连接工厂为例描述列表定制项如下：

操作序列：单击 **JMS > 连接工厂**，进入 JMS 连接工厂管理页面。单击**定制列表**自定义 JMS 连接工厂管理列表的列表项。

默认列表项：JNDI 名称、JMS 资源适配器、部署目标、标签。

自定义项：描述、创建时间、最新修改时间。

用户管理

操作序列：单击 **用户管理**，进入用户管理页面。单击 **定制列表** 自定义用户管理列表的列表项。

默认列表项：用户名、角色、邮箱、创建人和操作。

自定义项：用户名、角色、邮箱、创建人。

审计日志

操作序列：单击 **审计日志**，进入审计日志查询页面。单击**定制列表**自定义审计日志列表的列表项。

默认列表项：操作描述、操作目标类型、操作目标名称、操作用户、时间。

自定义项：操作描述、操作目标类型、操作目标 ID、操作目标名称、操作用户、时间。

3.4.4 节点管理

节点管理主要是指通过 xigemaAS 管理中心集中管理节点。主要涉及节点的注册、删除、环境检查，以及在该节点上安装、卸载 xigemaAS 等操作。

xigemaAS 管理中心提供默认节点(defaultNode) 。该节点上已安装 xigemaAS，并且已创建应用服务器(defaultServer) ，用户可以直接进行应用部署等操作。默认节点不可删除或编辑，用户可以在默认节点上继续创建新的应用服务器。

注册节点

了解注册节点时界面元素的含义描述和操作规范，以帮助您正确注册节点。

“认证信息”

创建新的认证

不使用已注册节点的认证信息，为当前节点创建新的认证信息。

使用已有认证

使用已注册节点的认证信息，包括代理类型、认证方式、主机名/ IP、SSH 用户名和端口号。

代理类型：**SSH**

注册在远程主机上的节点，须填写认证信息。

代理类型：本机

注册在 xigemaAS 管理中心所在主机上的节点，无需远程认证。

主机名

节点所在远程主机的计算机名。由字母、数字、短划线 (-)和句点 (.) 组成，长度不超过128个字符。

IPv4

节点所在远程主机的 IPv4 地址。

SSH 用户名

登录远程主机使用的用户名。由字母、数字、下划线 (_)、句点 (.) 组成，长度不超过128个字符。root 是超级管理员用户账户，请慎重使用。

SSH 端口号

节点所在主机用于 SSH 通信的端口号。

认证方式

xigemaAS 管理中心支持两种 SSH 认证方式：

- 用户名/密码认证。

  需填写登录节点所在主机使用的密码。密码以密文的方式呈现。
- 证书认证。

  请确保已为 xigemaAS 管理中心生成了公、私钥证书文件，并已将公钥添加到待注册节点的受信文件中。

  SSH 证书文件，即用于 SSH 认证的私钥文件。需填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径。

默认认证方式为**用户名/密码认证**方式。

👉 注：

- 标有 * 的选项为必填项。
- 请确保 xigemaAS 管理中心与远程主机之间的网络连接，并且信息填写正确。以下内容保存后不可修改：
  - 节点名称

- 主机名
- IPv4
- SSH 用户名

“基本信息”

节点名称

节点的唯一标识。节点名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.) ；大小写字母敏感；长度在3~50个字符之间。

标签

节点在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

xigemaAS 安装目录

${wlp.install.dir}，xigemaAS 的安装路径（即 wlp 的父目录）。一个目录下只允许安装一个 xigemaAS。同时，需校验用户对该目录具备写权限。

Java 运行时 (JRE) 目录

Java 运行时（JRE）在节点上的安装路径。JRE 必须存在，且版本为1.7或以上。**Java 运行时 (JRE) 目录**保存后，将作为该节点上服务器环境变量 JAVA_HOME 的值。对 JRE 安装路径的修改，不会影响已创建服务器的 JAVA_HOME 环境变量，但会作为此后创建服务器环境变量 JAVA_HOME 的值。

描述

节点的描述信息。长度不超过512个字符。

xigemaAS 安装状态

根据 xigemaAS 在安装和卸载中的不同状态和结果，xigemaAS 管理中心定义了不同的安装状态，以便了解节点安装和卸载的实际情况。

xigemaAS 安装状态之间的转换流程如下:

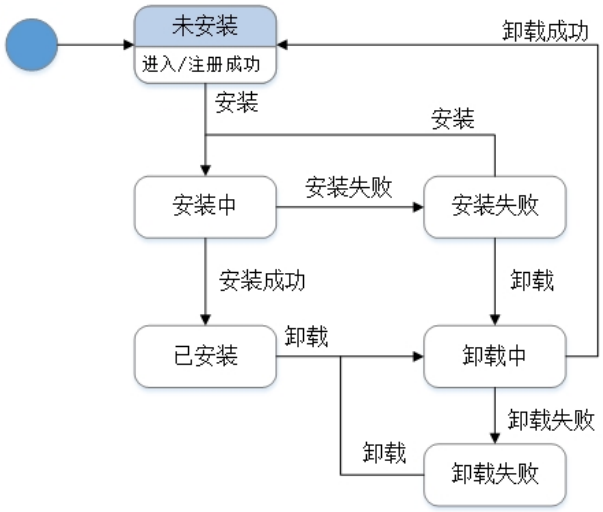


图 111: xigemaAS 安装状态转换图

节点上 xigemaAS 的安装状态具体包括：

未安装

节点已注册但未安装 xigemaAS。

安装中

节点上正在安装 xigemaAS，该节点 xigemaAS 处于安装中状态。

已安装

节点上已成功安装 xigemaAS，该节点 xigemaAS 处于已安装状态。

卸载中

节点上正在卸载 xigemaAS，该节点 xigemaAS 处于卸载中状态。

安装失败

节点环境检查失败，以及当管理中心所在主机和服务器之间网络不通时，节点上安装 xigemaAS 失败，该节点 xigemaAS 处于安装失败状态。

卸载失败

当前用户不具备卸载文件权限，以及当管理中心所在主机和服务器之间网络不通时，节点上卸载 xigemaAS 失败，该节点 xigemaAS 处于卸载失败状态。

3.4.5 服务器管理

服务器管理主要包括对应用服务器的管理和 Web 服务器的管理。

服务器管理除创建、删除、启动、停止、重启等常用操作外，还支持服务器指标监控，配置文件编辑等功能。

应用服务器管理

应用服务器管理主要是指通过 xigemaAS 管理中心集中管理节点上的应用服务器。主要涉及应用服务器的创建、删除、启动、停止、重启等操作。

xigemaAS 管理中心提供默认应用服务器(defaultServer)，用户可以直接进行应用部署等操作。默认应用服务器支持编辑、删除、启动、停止、重启、配置、监控等操作，并且可以作为应用服务器模板添加到集群中。

👉 注：默认应用服务器使用的端口号为“9080”，若该端口被占用，则 xigemaAS 管理中心不会生成默认应用服务器。

创建应用服务器

了解创建应用服务器时界面元素的含义描述和操作规范，以帮助您正确创建应用服务器。

基本信息

应用服务器所在节点

应用服务器所在节点，该节点上必须已安装 xigemaAS。

应用服务器名称

应用服务器名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)，长度在3~50个字符之间。

👉 注：请确保应用服务器名称填写正确，保存后不可修改。

模板类型

应用服务器的模板类型。包括 xigemaAS 管理中心的内置模板，已创建的应用服务器的应用服务器模板、以及已创建集群的集群模板。默认模板类型为内置模板。

模板

应用服务器的配置模板。

1. 若模板类型为“内置”，可以选择以下模板：
   - DefaultServer，支持 JSP 2.2 以及 Servlet 3.0 规范，是 xigemaAS 管理中心提供的默认模板。
   - WebProfile-JAVA6，支持 Java EE 6 Web Profile 所有规范。
   - WebProfile-JAVA7，支持 Java EE 7 Web Profile 所有规范。
   - FullProfile-JAVA7，支持 Java EE 7 Full Profile 所有规范。
2. 若模板类型为“应用服务器”，可以选择在 xigemaAS 管理中心已创建的应用服务器的应用服务器模板。
3. 若模板类型为“集群”，可以选择在 xigemaAS 管理中心已创建集群的集群模板。

标签

应用服务器在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

用户名

应用服务器的用户名，默认为admin，不可修改。

密码

应用服务器的密码，默认为admin。

描述

应用服务器的描述信息，长度不超过512个字符。

“系统属性”

名称

系统属性的名称。xigemaAS 管理中心支持的默认系统属性包括HTTP_PORT、HTTPS_PORT、MC_MEMBER_PORT、IIOP_PORT、IIOPS_PORT，默认系统属性不可删除，且名称不可编辑。

值

可以输入 (0,65535] 区间内的整数，但不能与其他系统属性值重复。若输入的值已被占用，则不能保存。

描述

系统属性的描述。默认系统属性的描述不可编辑。

👉 注：

- 标有 * 的选项为必填项。
- 在同一个节点上不能创建多个同名应用服务器。
- ${server.config.dir}，即创建的某个应用服务器的目录，如 wlp/usr/servers/server1。

有关创建应用服务器的具体操作步骤，请参阅创建应用服务器（见第 1807 页）。

配置应用服务器

了解配置应用服务器时界面元素的含义描述和操作规范，以帮助您正确配置应用服务器。

“基本信息”

修改密码

应用服务器的密码，默认为admin。若应用服务器是集群成员，则使用默认的应用服务器密码，并且不可修改。

标签

应用服务器在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

服务器描述

应用服务器的描述信息，长度不超过512个字符。

值

可以输入 (0,65535] 区间内的整数，但不能与其他系统属性值重复。若输入的值已被占用，则不能保存。默认系统属性的名称不能编辑和删除。

👉 注：置灰项为可查看不可编辑状态。

“JVM 配置”

JAVA_HOME

JAVA_HOME 的绝对路径，不能包含中文及以下字符：*<>:?"|；长度不超过512个字符。

-Xms（堆最小值）、**-Xmx**（堆最大值）

堆最小值不能超过堆最大值。

-XX:PermSize（永久域最小值）、**-XX:MaxPermSize**（永久域最大值）

永久域最小值不能超过永久域最大值。

JVM 属性

可以在 **“ JVM 选项”**面板添加并描述 JVM 属性。

有关配置应用服务器的具体操作步骤，请参阅配置应用服务器（见第 1823 页）。

应用服务器状态

根据应用服务器在启动、停止、重启等操作中的不同状态和结果，xigemaAS 管理中心定义了不同的应用服务器状态，以便了解应用服务器运行的实际情况。

应用服务器各状态之间的转换流程如下：

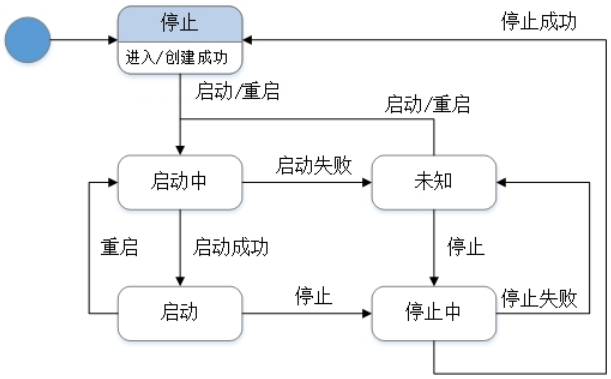


图 112: 应用服务器状态转换图

应用服务器的状态具体包括：

启动

应用服务器启动成功或重启成功，该应用服务器处于启动状态。

停止

应用服务器已停止，该应用服务器处于停止状态。

未知

应用服务器启动失败、重启失败、停止失败、以及当管理中心所在主机和应用服务器之间网络不通时，该应用服务器处于未知状态。

启动中

应用服务器正在启动或重启，该应用服务器处于启动中状态。

停止中

应用服务器正在停止，该应用服务器处于停止中状态。

Web 服务器管理

Web 服务器管理主要是指通过 xigemaAS 管理中心集中管理本机 Web 服务器、远程 Web 服务器以及创建的虚拟主机。主要涉及Web 服务器以及虚拟主机的新增、删除、启动、停止、重启等操作。

新增 Web 服务器

了解新增 Web 服务器时界面元素的含义描述和操作规范，以帮助您正确添加 Web 服务器。

认证信息

创建新的认证

为当前 Web 服务器创建新的认证信息。创建新的认证需要选择代理类型并填写认证信息。

使用已有认证

使用已通过认证的 Web 服务器的认证信息作为当前 Web 服务器的认证信息。

代理类型：**SSH**

远程主机上的 Web 服务器，须填写认证信息。xigemaAS 管理中心默认新增远程 Web 服务器。

代理类型：本机

xigemaAS 管理中心所在主机上的 Web 服务器，无需远程认证。

主机名

Web 服务器所在远程主机的计算机名。由字母、数字、短划线 (-)和句点 (.) 组成，长度不超过128个字符。

IPv4

Web 服务器所在远程主机的 IPv4 地址。

SSH 用户名

登录远程主机使用的用户名。由字母、数字、下划线 (_)、句点 (.) 组成，长度不超过128个字符。`root` 是超级管理员用户账户，请慎重使用。

SSH 端口号

Web 服务器所在主机用于 SSH 通信的端口号。用 (0,65535] 区间内的整数表示。

认证方式

xigemaAS 管理中心支持两种 SSH 认证方式：

- 用户名/密码认证。

  需填写登录节点所在主机使用的密码。密码以密文的方式呈现。
- 证书认证。

  请确保已为 xigemaAS 管理中心生成了公、私钥证书文件，并已将公钥添加到待注册节点的受信文件中。

  SSH 证书文件，即用于 SSH 认证的私钥文件。需填写 SSH 认证的私钥文件在 xigemaAS 管理中心所在主机的绝对路径。

默认认证方式为**用户名/密码认证**方式。

基本信息

Web 服务器名称

Web 服务器的唯一标识，保存后不可修改。Web 服务器名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)；大小写字母敏感；长度在3~50个字符之间。

Web 服务器类型

xigemaAS 管理中心支持的 Web 服务器类型包括 Apache 和 Nginx。

若 Web 服务器类型为 Apache，则需要选择**操作系统类型**操作系统类型，包括类 Unix 系统和 Windows 系统。

若选择添加 Windows 系统下的 Apache，还需要填写 **Windows 服务名称**，即Apache 注册为 Windows 服务时的 Windows 服务名称。

配置文件路径

Web 服务器核心配置文件的完整路径。例如：

- Nginx (Linux)：url/local/nginx/conf/nginx.conf
- Apache (Linux)：url/local/apahce/conf/httpd.conf

可执行文件路径

Web 服务器可执行文件的完整路径，保存后不可修改。例如：

- Nginx (Linux)：url/local/nginx/sbin/nginx
- Apache (Linux)：url/local/apahce/bin/httpd
- Apache (Windows)：C:/Apache24/bin/httpd.exe

日志文件路径

Web 服务器日志文件的所在目录。例如：

- Nginx (Linux)：url/local/nginx/logs
- Apache (Linux)：url/local/apahce/logs

Java 运行时 (JRE) 目录

Java 运行时（JRE）在 Web 服务器上的安装路径。JRE 必须存在，且版本为1.6或以上。**Java 运行时 (JRE) 目录**保存后，将作为该 Web 服务器环境变量 JAVA_HOME 的值。对 JRE 安装路径的修改，不会影响已添加的 Web 服务器的 JAVA_HOME 环境变量，但会作为此后新增的 Web 服务器环境变量 JAVA_HOME 的值。

标签

Web 服务器在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

描述

Web 服务器的描述信息，长度不超过512个字符。

配置 Web 服务器

编辑 Web 服务器配置文件时（包括在线配置 Web 服务器和线下编辑 Web 服务器配置文件），可以添加指令、参数、模块等高级配置，但不能针对由 xigemaAS 管理中心管理的各功能在配置文件中的内容进行更改。

通过编辑 Web 服务器配置文件，可以对虚拟主机管理和负载均衡设置两个模块功能进行配置，配置编辑的注意事项如下：

可编辑内容

添加模块、指令：可以在已有的 server{} 块、location{} 块内添加，也可以在配置文件中添加其他模块。

添加参数：可以在手动配置的指令中编辑参数，也可以在 xigemaAS 管理中心自动配置的指令上添加额外的参数，但不可以对 xigemaAS 管理中心自动配置的参数进行修改。

不可编辑内容

通过 xigemaAS 管理中心新增虚拟主机、配置负载均衡时，会对配置文件的内容进行修改并添加相应配置信息。这些配置信息由 xigemaAS 管理中心识别、记录和管理，请勿手动编辑，以防出现不可预知的错误。

👉 注：xigemaAS 管理中心不对手动配置添加的模块、指令、参数进行识别或管理。配置保存后，如果需要修改，单击**服务器 > Web 服务器 > Web 服务器名称 > 配置信息**进入 Web 服务器配置页面，手动编辑配置信息进行修改。

Nginx

针对虚拟主机管理和负载均衡配置两个模块，编辑 Nginx 配置文件 nginx.conf 的相关示例如下：

- 不可编辑内容：
  - 通过 xigemaAS 管理中心新增虚拟主机（见第 1838 页）时，对 nginx.conf 配置文件添加的相应配置信息；
  - 通过 xigemaAS 管理中心部署应用（见第 1813 页）时，若部署的应用类型是 Web 应用或企业应用，可以为应用配置负载均衡。配置完成后，对 nginx.conf 配置文件添加的相应配置信息。

如下示例包括添加虚拟主机以及配置负载均衡两部分的配置内容，其中粗体部分为配置负载均衡后，xigemaAS 管理中心在 nginx.conf 中添加的配置片段，这两部分均只能由 xigemaAS 管理中心添加，不能手动编辑。

```
#other configurations
http{
#other configurations
   server{
        #server_block_name:defaultVirtualHost  (Warning:this comment is
 used by xigemaAS Management Center, please do not modify this line.)
        listen 172.16.173.108:90;
        server_name vsettan.com vsettan.cn;
    location examples/{
            proxy_pass http://up4examples#1/examples/;
         }
     }
#other configurations
   upstream up4examples#1{
      sticky;
      server 172.16.173.107:9080
    }
#other configurations
}
#other configurations
```

👉 注：

1. 不允许直接在配置文件中添加虚拟主机。
2. 不允许修改、删除、拷贝此条注释：

```
#server_block_name:defaultVirtualHost  (Warning:this comment is
 used by xigemaAS Management Center, please do not modify this
 line.)
```

- 可编辑内容：
  - 添加模块、指令：在上述示例中，向已有的 location{} 块内添加 "expires 30m;" 指令，向已有的 server{} 块内添加 "keepalive_timeout 65;" 指令；在 server{} 块内添加其他 location{} 块。
  - 添加参数：在上述示例中，向 "listen 172.16.173.108:90" 指令上添加 ssl 参数，将该端口设置为 ssl 端口。

编辑后的配置片段示例如下（其中粗体部分由 xigemaAS 管理中心自动配置，不可进行手动配置）：

```
#other configurations
http{
#other configurations
   server{
        #server_block_name:defaultVirtualHost  (Warning:this comment is
 used by xigemaAS Management Center, please do not modify this line.)
        listen 172.16.173.108:90 ssl;
        server_name vsettan.com vsettan.cn;
          keepalive_timeout 65;
        location examples/{
            expires 30m;
            proxy_pass http://up4examples#1/examples/;
         }
         location=/50x.html{
          root html;
         }
      }

#other configurations
   upstream up4examples#1{
      sticky;
      server 172.16.173.107:9080
    }
#other configurations
}
#other configurations
```

Apache

针对虚拟主机管理和负载均衡配置两个模块，编辑 Apache 配置文件 httpd.conf 的相关示例如下：

- 不可编辑内容：
  - 通过 xigemaAS 管理中心新增虚拟主机（见第 1838 页）时，对 httpd.conf 配置文件添加的相应配置信息；
  - 通过管理中心部署应用（见第 1813 页）时，若部署的应用类型是 Web 应用或企业应用，可以为应用配置负载均衡。配置完成后，对 httpd.conf 配置文件添加的相应配置信息。

如下示例包括添加虚拟主机以及配置负载均衡两部分的配置内容，这两部分均只能由 xigemaAS 管理中心添加，不能手动编辑。

```
#other configurations
Listen 172.16.173.20:8889
<VirtualHost 172.16.173.20:8889>
    #virtual_host_name:outsideVirtualHost-1 (Warning: This comment is used
 by xigemaAS Management Center, please do not modify this line.)
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass /demoAppWebServerContextRoot/balancer://balancer_demoApp/
demoAppContextRoot/
    ProxyPassReverse /demoAppWebServerContextRoot/balancer://
balancer_demoApp/demoAppContextRoot/
</VirtualHost>
<Proxy balancer://balancer_demoApp>
    BalancerMember http://172.16.173.19:9080
    BalancerMember http://172.16.173.20:9080
</Proxy>
#other configurations
```

注:

1. 不允许直接在配置文件中添加虚拟主机。
2. 不允许修改、删除、拷贝此条注释：

```
#virtual_host_name:outsideVirtualHost-1 (Warning: This comment
 is used by xigemaAS Management Center, please do not modify this
 line.)
```

- 可编辑内容:添加模块、指令、参数。详细信息请参考示例 *1*。

Web 服务器状态

根据 Web 服务器在启动、停止、重启等操作中的不同状态和结果，xigemaAS 管理中心定义了不同的 Web 服务器状态，以便了解 Web 服务器运行的实际情况。

Web 服务器各状态之间的转换流程如下：

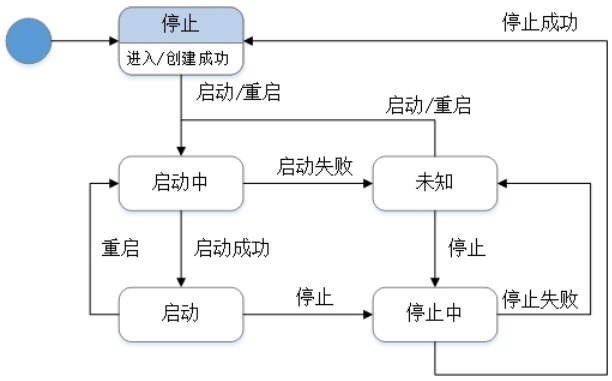


图 113: Web 服务器状态转换图

Web 服务器的状态具体包括：

启动

Web 服务器启动成功或重启成功，该Web 服务器处于启动状态。

停止

Web 服务器已停止，该Web 服务器处于停止状态。

未知

Web 服务器启动失败、重启失败、停止失败、以及当管理中心所在主机和Web 服务器之间网络不通时，该Web 服务器处于未知状态。

启动中

Web 服务器正在启动或重启，该Web 服务器处于启动中状态。

停止中

Web 服务器正在停止，该Web 服务器处于停止中状态。

新增虚拟主机

了解新增虚拟主机的含义描述和操作规范，以帮助您正确添加虚拟主机。

基本信息

Web 服务器

显示当前虚拟主机所在 Web 服务器的名称和 IP 地址，不可编辑。

虚拟主机名称

虚拟主机的唯一标识，保存后不可修改。虚拟主机名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)；大小写字母敏感；长度在3~50个字符之间。

域名

访问虚拟主机的域名，可以输入多个域名。

标签

虚拟主机在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

描述

虚拟主机的描述信息，长度不超过512个字符。

“监听信息”

监听地址

虚拟主机的主机名或 IP 地址，由字母数字字符 (A-Z、a-z、0-9)、下划线 (_) 和句点 (.) 组成

监听端口

虚拟主机的端口。监听端口不可重复，用 (0,65535] 区间内的整数表示。

3.4.6 集群管理

集群管理主要是指通过 xigemaAS 管理中心集中管理由多个应用服务器组成的集群。主要涉及集群的创建、删除、配置、启动、停止、重启等操作。

创建集群

正确创建集群，可以简化应用服务器的初始配置和后续管理，提高工作效率。

基本信息面板

集群名称

集群名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)；长度在3~50个字符之间。

👉 注：集群名称唯一，不可重复。请确保集群名称填写正确，保存后不可修改。

模板类型

集群模板的类型。包括 xigemaAS 管理中心的内置模板，已创建服务器的服务器模板、以及已创建集群的集群模板。默认模板类型为内置模板。

模板

集群的配置模板。

1. 若模板类型为“内置”，可以选择以下模板：
   - DefaultServer，支持 JSP 2.2 以及 Servlet 3.0 规范，是 xigemaAS 管理中心提供的默认模板。
   - WebProfile-JAVA6，支持 Java EE 6 Web Profile 所有规范。
   - WebProfile-JAVA7，支持 Java EE 7 Web Profile 所有规范。
   - FullProfile-JAVA7，支持 Java EE 7 Full Profile 所有规范。
2. 若模板类型为“应用服务器”，可以选择在 xigemaAS 管理中心已创建的应用服务器的应用服务器模板。
3. 若模板类型为“集群”，可以选择在 xigemaAS 管理中心已创建集群的集群模板。

标签

集群在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

描述

集群的描述信息，长度不超过512个字符。

集群成员面板

应用服务器名称

应用服务器名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)，长度在3~50个字符之间。

👉 注：同一节点下的应用服务器名称不允许重复。请确保应用服务器名称填写正确，保存后不可修改。

应用服务器名称前缀

批量创建集群成员时，服务器名称由“服务器前缀+集群成员编号”组成。

应用服务器数量

批量创建集群成员时，需要创建成员的数量。

应用服务器所在节点

服务器所在节点，该节点上必须已安装 xigemaAS。集群中的集群成员可以创建于不同节点。

起始编号

批量创建集群成员时，集群成员的起始编号。首次创建时，编号最小值为1；多次批量创建时，编号必须大于此前批量创建的集群成员的数量。

标签

应用服务器在 xigemaAS 管理中心上的分组标识。可自定义或选取现有标签，单个标签长度不超过20个字符，标签个数不超过5个。

描述

应用服务器的描述信息，长度不超过512个字符。

👉 注：一个集群成员只能从属于一个集群。

有关创建集群的具体操作步骤，请参阅创建集群（见第 1808 页）。

集群状态

根据集群在启动、停止、重启等操作中的不同状态和结果，xigemaAS 管理中心定义了不同的集群状态，以便了解集群运行的实际情况。

集群各状态之间的转换流程如下：

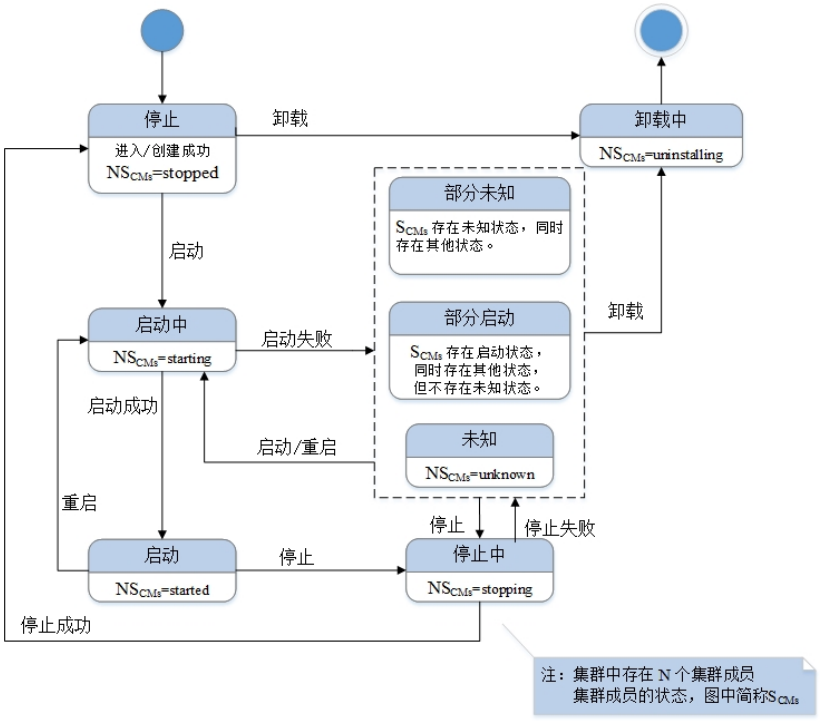


图 114: 集群状态转换图

集群的状态具体包括：

启动

集群中的所有集群成员均启动成功，该集群处于启动状态。

停止

集群中的所有集群成员均已停止，该集群处于停止状态。

未知

集群中的所有集群成员均处于未知状态。

集群成员的未知状态可能由以下几种情况触发：

- 集群成员停止失败；
- 集群成员启动或重启失败；
- 管理中心所在主机和集群成员之间网络不通。

启动中

集群中的所有集群成员均正在启动或重启，则该集群处于启动中状态。

停止中

集群中的所有集群成员均正在停止，则该集群处于停止中状态。

卸载中

集群中的所有集群成员均正在卸载，则该集群处于卸载中状态。

部分未知

集群中的集群成员存在未知状态，且同时存在其他状态。

部分启动

集群中的集群成员存在启动状态，且同时存在其他状态，但不存在未知状态。

3.4.7 应用管理

应用管理主要是指通过 xigemaAS 管理中心集中管理部署目标上的应用。主要涉及应用的部署、卸载、启动、停止以及重启等操作。

应用部署

了解应用部署时界面元素的含义和操作规范，以帮助您正确部署应用。

应用类型

要部署的应用类型，xigemaAS 管理中心支持的应用类型包括：

- Web 应用
- 企业应用
- EJB 应用
- 连接器应用
- OSGi 应用

注：

- 若部署的应用类型为 EJB 应用，目标服务器的 Java 运行时版本须为 JRE 1.7 及以上。若使用默认模板创建服务器并且部署 EJB 应用，启动服务器后，xigemaAS 管理中心会自动在 server.xml 配置文件中添加 ejb-3.2 功能部件。因此，若目标服务器处于启动状态，则需要重启目标服务器使配置生效。
- 若部署的应用类型为连接器应用，且该应用的 JCA 版本为 JCA 1.7及以上，则需要将该应用部署在 Java 运行时版本为 JRE 1.7 及以上的部署目标。

部署方式

目标应用的所在位置。xigemaAS 管理中心支持**本地上传**、**远程部署**和**目录部署**。

- 本地上传：从登录 xigemaAS 管理中心的浏览器所在主机上选择应用。
- 远程部署：从 xigemaAS 管理中心所在主机上选择应用。
- 目录部署：从部署目标（应用服务器）所在主机上选择应用的解压后文件。但该部署方式仅适用于Web 应用和企业应用。

注：上传的应用必须与之前选择的应用类型匹配。

应用上下文

应用类型为 Web 应用时，可以输入应用上下文，默认应用上下文为当前应用名称。应用上下文与应用一一对应，不可重复。

应用名称

应用名称可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)。

私有共享库

库引用的列表，库类实例是此类装入器特有的，与来自其他类装入器的类实例无关。应用类型为 Web 应用、企业应用、EJB 应用和连接器应用时，从下拉列表中选择私有共享库。

公有共享库

库引用的列表，会与其他类装入器共享库类实例。应用类型为 Web 应用、企业应用、EJB 应用和连接器应用时，从下拉列表中选择公有共享库。

共享库

应用类型为 OSGi 应用时，从下拉列表中选择共享库。

类加载优先顺序

父优先，子优先。若选择父优先，那么在搜索类路径之前，授权给直接父代；若选择子优先，那么在搜索类路径之后，授权给直接父代。

标签

应用在 xigemaAS 管理中心的分组标识，单个标签长度不超过20个字符，标签个数不超过5个。

描述

应用的描述信息，长度不超过512个字符。

可选部署目标

应用部署的目标应用服务器或集群。可选部署目标中列出当前 xigemaAS 管理中心已创建的应用服务器和集群。若选择集群为部署目标，则该应用会部署到集群中的所有集群成员上。

Web 服务器端应用上下文

应用类型为 Web 应用时，若选择了配置负载均衡的虚拟主机，则需要输入 Web 服务器端应用上下文。

可选虚拟主机

应用类型为 Web 应用时，可以设置负载均衡。可选虚拟主机中列出当前 xigemaAS 管理中心已创建的虚拟主机。

注:

- 标有 * 的选项为必填项。
- 不同应用，应用名称不可以相同。
- 同一应用，可以部署在多个应用服务器和集群上，但不能在同一应用服务器和集群上重复部署。
- 同一应用，可以选择多个虚拟主机设置负载均衡；同一虚拟主机，可以为多个应用配置负载均衡。

有关部署应用的具体操作步骤，请参阅部署应用（见第 1813 页）

应用状态

根据应用在启动、停止、重启等操作中的不同状态和结果，xigemaAS 管理中心定义了不同的应用状态，以便了解应用运行的实际情况。

应用各状态之间的转换流程如下：

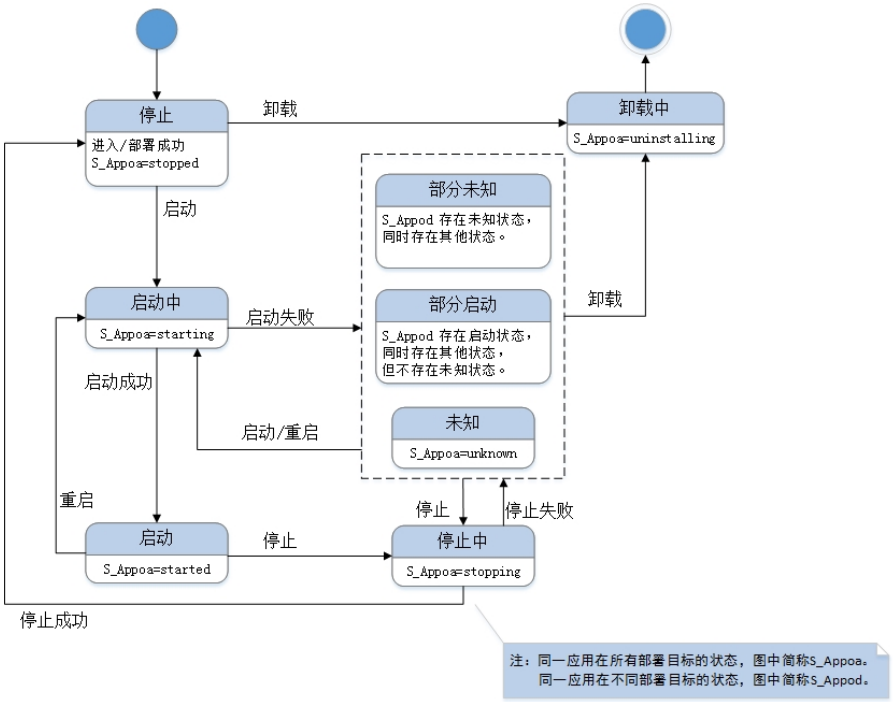


图 115: 应用状态转换图

应用的状态具体包括：

启动

同一应用在所有部署目标上均启动成功，则该应用处于启动状态。

停止

同一应用在所有部署目标上均已停止，则该应用处于停止状态。

未知

同一应用在所有部署目标上均处于未知状态。

应用的未知状态可能由以下几种情况触发：

- 应用停止失败；
- 应用启动或重启失败；
- 管理中心所在主机和服务器之间网络不通。

启动中

同一应用在所有部署目标上均正在启动或重启，则该应用处于启动中状态。

停止中

同一应用在所有部署目标上均正在停止，则该应用处于停止中状态。

卸载中

同一应用在所有部署目标上均正在卸载，则该应用处于卸载中状态。

部分未知

同一应用在不同部署目标上存在未知状态，且同时存在其他状态。

部分启动

同一应用在不同部署目标上存在启动状态，且同时存在其他状态，但不存在未知状态。

应用列表

xigemaAS 管理中心提供应用列表便于用户管理和查询应用，包括应用总列表、应用部署目标详情列表、特定应用服务器应用列表、特定集群应用列表、特定 Web 服务器应用列表、以及特定虚拟主机应用列表。

应用总列表

在 xigemaAS 管理中心主页单击**应用**，进入应用管理页面，查看应用总列表。

应用总列表列出了 xigemaAS 管理中心管理的所有应用的相关信息，如下图所示：

图 116: 应用总列表

应用总列表页面可以进行以下操作：

部署、卸载、启动、停止、重启、查询、定制列表、编辑应用、访问 Web 应用链接以及查看应用部署目标详情。

访问 Web 应用链接：若当前应用类型为 Web 应用，则可以单击，查看当前 Web 应用的访问链接地址以及所在的应用服务器名称。

应用部署目标详情列表

在 xigemaAS 管理中心主页单击**应用**，从应用总列表中选择需要查看其部署目标详情的应用，单击**详情**，进入应用部署目标详情页面。

应用部署目标详情列表列出了当前应用的所有部署目标的相关信息，如下图所示：

图 117: 应用部署目标详情列表

应用部署目标详情列表页面可以进行以下操作：

启动、停止、重启、查询、定制列表以及子模块详情查看。

子模块详情查看：若当前应用类型为 Web 应用或 EJB 应用，并且应用状态为“启动”，则可以单击，查看当前应用的子模块详情，如下图所示：

图 118: 子模块详情列表

特定应用服务器应用列表

在 xigemaAS 管理中心主页单击**服务器 > 应用服务器**，进入应用服务器管理页面。从应用服务器列表中选择要查看应用部署情况的应用服务器，单击**查看应用**，进入当前应用服务器的应用列表页面。

特定应用服务器应用列表列出了当前应用服务器上部署的所有应用的相关信息，如下图所示：

图 119: 特定应用服务器应用列表

特定应用服务器应用列表可以进行以下操作：

启动、停止、重启、查询、定制列表。

特定集群应用列表

在 xigemaAS 管理中心主页单击**集群**，进入集群管理页面。从集群列表中选择要查看应用部署情况的集群，单击**查看应用**，进入当前集群的应用列表页面。

注：集群处于启动状态、停止状态或未知状态下，方可查看特定集群应用列表。

特定集群应用列表列出了当前集群上部署的所有应用的相关信息，如下图所示：

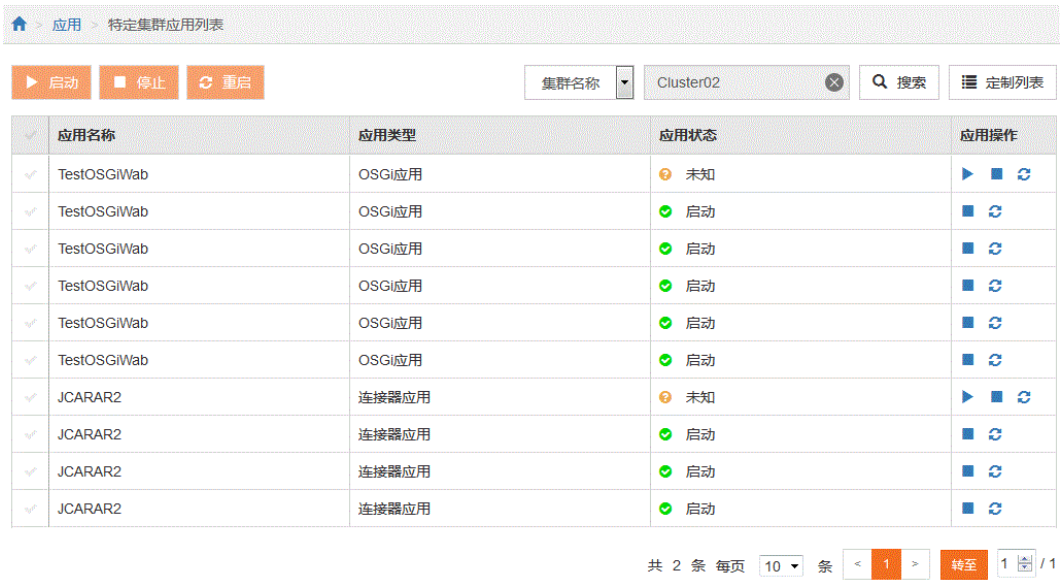


图 120: 特定集群应用列表

特定集群应用列表可以进行以下操作：

启动、停止、重启、查询、列表定制。

特定 Web 服务器应用列表

在 xigemaAS 管理中心主页单击**服务器 > Web 服务器**，进入 Web 服务器管理页面。从 Web 服务器列表中选择要查看应用详细信息的 Web 服务器，单击**查看负载应用**，进入当前 Web 服务器的应用列表页面。

特定 Web 服务器应用列表列出了当前 Web 服务器上所有应用的相关信息，如下图所示：

图 121: 特定 Web 服务器应用列表

特定 **Web** 服务器应用列表可以进行以下操作：

启动、停止、重启、查询、定制列表。

特定虚拟主机应用列表

在 xigemaAS 管理中心主页单击**服务器 > Web 服务器 >** ，进入虚拟主机管理页面。从虚拟主机列表中选择要查看应用详细信息的虚拟主机，单击**查看负载应用**，进入当前虚拟主机的应用列表页面。

特定虚拟主机应用列表列出了当前虚拟主机上所有应用的相关信息，如下图所示：

图 122: 特定虚拟主机应用列表

特定虚拟主机应用列表可以进行以下操作：

启动、停止、重启、查询、定制列表。

3.4.8 资源管理

资源管理主要是指通过 xigemaAS 管理中心集中管理共享库、连接池、数据源以及 JMS 资源。主要涉及资源的新增、删除、编辑等操作。

xigemaAS 管理中心提供默认连接池(defaultConnectionPool)，该连接池不支持编辑、删除操作。

新增共享库

了解新增共享库时界面元素的含义描述和操作规范，以帮助您正确创建共享库。

基本信息

共享库名称

共享库的名称唯一，不允许重复，保存后不可修改。可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-) 或句点 (.)，长度在3~50个字符之间。

标签

单个标签长度不超过20个字符，标签个数不超过5个。

描述

长度不超过512个字符。

资源文件

本地文件

从登录 xigemaAS 管理中心使用的浏览器所在主机上选择资源文件。所选文件类型必须为 JAR 文件。

仓库资源

从 xigemaAS 管理中心已上传资源仓库中选择资源文件。

共享库资源目录列表

用于配置需要被加载的资源可批量配置资源目录，每行配置一个资源目录。文件路径可以为绝对路径或包含 xigemaAS 自定义环境变量的路径。

例： /usr/share/lib/config

${share.lib.dir}/lib/resources

共享库资源文件列表

用于配置需要被加载的资源可批量配置资源文件，每行配置一个资源文件。文件路径可以为绝对路径或包含 xigemaAS 自定义环境变量的路径。

例： /usr/share/lib/a.jar

${share.lib.dir}/lib/b.jar

有关新增共享库的具体操作步骤，请参阅新增共享库（见第 1851 页）。

新增连接池

了解新增连接池时界面元素的含义描述和操作规范，以帮助您正确创建连接池。

基本信息

池名称

连接池名称唯一，不允许重复。可以使用 Unicode 字母数字字符 (A-Z a-z 0-9)、下划线 (_)、短划线 (-)、加号 (+) 和句点 (.) 命名，但第一个字符不可以是短划线 (-)或句点 (.)，长度在3~50个字符之间。

标签

单个标签长度不超过20个字符，标签个数不超过5个。

描述

连接池的描述信息。长度不超过512个字符。

池基本配置

最大连接数

池的最大物理连接数。只能输入数字，最小值为0。默认值为50。值为0时意味着不受限制。

最小连接数

池中要保留的最小物理连接数。只能输入数字，最小值为0。无默认值。时效超时可以覆盖此值。

空闲超时时间

池维护期间可废弃未使用或空闲的连接之前的时间量。只能输入数字，最小值为-1。默认值为30（分钟）。值为-1时会禁用此超时。

最大等待时间

连接请求超时之前的时间量。只能输入数字，最小值为-1。默认值为30（秒）。值为-1时会禁用此超时。

时效超时

池维护可以废弃物理连接之前的时间量。只能输入数字，最小值为-1。默认值为-1。值为-1时会禁用此超时。

收集时间

两次运行池维护线程之间的时间量。只能输入数字，最小值为-1。默认值为3（分钟）。值为-1时会禁用此超时。

清除策略

指定在池中检测到旧连接时要销毁哪些连接。

- VALIDATEALLCONNECTIONS：当检测到失效连接时，会测试连接并关闭出现错误的连接。
- FAILINGCONNECTIONONLY：当检测到失效连接时，会关闭出现错误的连接。
- ENTIREPOOL：当检测到失效连接时，会将池中的所有连接都标记为失效，且当这些连接不再使用时，会关闭连接。

池高级配置

每个线程的最大打开连接数

只能输入数字，最小值为0。无默认值。

每个线程的高速缓存连接数

只能输入数字，最小值为0。无默认值。

有关新增连接池的具体操作步骤，请参阅新增连接池（见第 1853 页）。

新增 JDBC 数据源

了解新增 JDBC 数据源时界面元素的含义描述和操作规范，以帮助您正确创建 JDBC 数据源。

基本信息

JNDI 名称

JNDI 的唯一标识。不能包含中文。保存后不可修改。

连接池名称

已通过 xigemaAS 管理中心创建的连接池的名称。

资源类型

数据源类型。可选值为：

- javax.sql.XADataSource：数据源的基本形式。不提供互操作性来增强连接池，并且不能作为具备两阶段功能的资源来参与涉及多个资源的事务；
- javax.sql.ConnectionPoolDataSource：数据源已对连接池启用。能够作为具备两阶段功能的资源来参与涉及多个资源的事务；
- javax.sql.DataSource：数据源已对连接池启用。能够作为具备两阶段功能的资源参与涉及多个资源的事务。

数据库类型

- xigemaDB
- Oracle
- MySQL
- IBM DB2
- IBM Informix
- Microsoft SQL Server
- Sybase
- Derby
- 达梦
- 神通
- 人大金仓
- PostgreSql

数据库驱动类型

已选数据库类型所支持的 JDBC 驱动程序实现。新增 JDBC 数据源时，需要 xigemaAS 管理中心已有包含数据库驱动的共享库。

数据源实现类

已选数据库驱动类型中数据源的具体实现类。数据库类型、数据库驱动类型与数据源类名称的对应关系，请参阅数据源及数据库类型（见第 1912 页）。

数据库驱动库

数据源使用的具体 JDBC 驱动程序实现库文件。下拉列表中列出已通过 xigemaAS 管理中心创建的共享库供用户选择。

标签

对 JDBC 数据源进行标识和分类的关键词，便于检索 JDBC 数据源。单个标签长度不超过20个字符，标签个数不超过5个。

描述

JDBC 数据源描述信息。长度不超过512个字符。

数据库信息

主机名

数据库所在主机的主机名。由字母、数字、短划线 (-)和句点 (.) 组成，长度不超过128个字符。

端口

用 (0,65535] 区间内的整数表示。

数据库名称

可以使用Unicode 字母数字字符 (A-Z a-z 0-9)、加号（+）、句点（.）和短划线（-）命名，但第一个字符不可以是短划线（-）或句点（.）。长度在1~50字节之间。

用户名

登录数据库的用户名。由字母、数字、下划线 (_)、句点 (.) 组成，长度不超过128个字符。

密码

登录数据库的密码。

连接 **URL**

根据表单中的数据库类型、主机名、端口和数据库名称自动生成。

高级属性

连接匹配

共享连接的匹配方式。

- MatchOriginalRequest：共享连接时，根据原始连接请求进行匹配。
- MatchCurrentState：共享连接时，根据连接的当前状态进行匹配。

事务隔离级别

默认事务隔离级别。

- TRANSACTION_REPEATABLE_READ：不可进行脏读取和不可重复读取；可以进行幻象读取。
- TRANSACTION_READ_COMMITTED：不可进行脏读取；可以进行不可重复读取和幻象读取。
- TRANSACTION_SERIALIZABLE：脏读取、不可重复读取和幻象读取受到阻止。
- TRANSACTION_READ_UNCOMMITTED：可以进行脏读取、不可重复读取和幻象读取。
- TRANSACTION_SNAPSHOT：Microsoft SQL Server JDBC 驱动程序和 DataDirect Connect for JDBC 驱动程序的快照隔离。

SQL 语句查询超时时间

SQL 语句的默认查询超时时间。在 JTA 事务中，syncQueryTimeoutWithTransactionTimeout 可以覆盖此默认值。只能输入数字，最小值为0。

每个连接的高速缓存数

每个连接高速缓存语句的最大数目。只能输入数字，最小值为0。默认值为10。

补充 **JDBC** 跟踪

补充在 bootstrap.properties 中启用 JDBC 驱动程序跟踪时记录的 JDBC 驱动程序跟踪。

同步查询和交易超时

将 JTA 事务中的剩余时间（如果有）用作 SQL 语句的缺省查询超时。

参与事务

支持参与由应用程序服务器管理的事务。

登记滚动 **API**

使用结果集滚动接口时尝试事务登记。

登记供应商 **API**

使用供应商接口时尝试事务登记。

清除时提交或回滚

确定当关闭数据库工作单元(AutoCommit=false) 中可能存在的连接或将其返回到池中时如何清除这些连接。

- commit：通过提交来清除连接。
- rollback：通过回滚来清除连接。

启用连接强制类型转换

可以从数据源获得的连接强制类型转换为 JDBC 供应商连接实现所实现的接口类。

有关新增 JDBC 数据源的具体操作步骤，请参阅新增 *JDBC* 数据源（见第 1855 页）。

新增 Redis 数据源

了解新增 Redis 数据源时界面元素的含义描述和操作规范，以帮助您正确创建 Redis 数据源。

数据源基本信息配置

JNDI 名称

JNDI 的唯一标识。不能包含中文。保存后不可修改。

Redis 库文件

Redis 数据源使用的 JDBC 驱动程序实现库文件。新增 Redis 数据源时，需要 xigemaAS 管理中心已有包含 Redis 数据库驱动的共享库。下拉列表中列出已添加至 xigemaAS 管理中心的共享库供用户选择。

标签

对 Redis 数据源进行标识和分类的关键词，便于检索 Redis 数据源。单个标签长度不超过20个字符，标签个数不超过5个。

描述

Redis 数据源描述信息。长度不超过512个字符。

数据源相关数据库信息配置

Redis 数据源类型

- 单点：若选择单点类型，数据库信息需要输入 Redis 主机名称或 IP 地址、Redis 服务端口号、密码，高级属性需要输入数据库名称、Socket 通信超时时间、连接超时时间、客户端名称；
- 集群：若选择集群类型，数据库信息需要输入 Redis 集群主机名称或 IP 地址、端口号，高级属性需要输入最大跳转数，默认最大跳转数为 5。

Redis 主机名称或 **IP** 地址

由字母、数字、短划线 (-)和句点 (.) 组成，长度不超过128个字符。

Redis 服务端口号

用 (0,65535] 区间内的整数表示。

密码

登录 Redis 数据库的密码。

数据源配置—高级属性

数据库

Redis 数据库的名称。只能为整数。默认为0。

Socket 通信超时时间

只能输入数字，最小值为0。默认值为2000毫秒。

连接超时时间

通信超时时间。只能输入数字，最小值为0。默认值为2000毫秒。

客户端名称

不能包含中文。

最大跳转数

只能输入数字，最小值为0。默认值为5。

连接池基本属性配置

连接池最大连接数

只能输入整数。值为负数表示连接池连接数无限制。默认值为8。

连接池最大空闲数

只能输入数字，最小值为0。默认值为8。

连接池最小空闲数

只能输入数字，最小值为0。

连接池高级属性配置

连接耗尽时是否阻塞

默认为是（true）。

- true：阻塞直到超时。
- false：抛出异常。

逐出策略类名

由字母、数字、短划线 (-)和句点 (.) 组成，但第一个字符不可以是短划线（-）或句点（.）。

是否启用连接池的 **JMX** 管理功能

默认为是（true）。

JMX 名称前缀

由字母、数字、短划线 (-)和句点 (.) 组成，但第一个字符不可以是短划线（-）或句点（.）。

是否启用后进先出

默认为是（true）。

获取连接时的最大等待毫秒数

只能输入数字，最小值为-1。默认值为-1。如果该值为负数， 则会导致阻塞。当勾选了连接耗尽时是否阻塞时，超时将会抛出异常。

逐出连接的最小空闲时间

逐出连接的最小空闲时间 。只能输入数字，最小值为0。默认为60000毫秒

每次逐出检查时，逐出的最大数目

只能输入数字，最小值为0。默认值为-1。

空闲多久后逐出

当空闲时间大于该值且空闲连接大于最大空闲数时，直接逐出，而不再根据逐出连接的最小空闲时间进行判断。只能输入数字，最小值为0。默认值为1800000 毫秒。

获取连接时是否检查有效性

默认为否（false）。

归还连接时是否检查有效性

默认为否（false）。

在空闲时是否检查有效性

空闲时检查池内连接的有效性。默认为否（false）。

逐出扫描的时间间隔

若值为负数，表示不运行逐出线程。只能输入数字，最小值为0。默认为30000毫秒。

有关新增 Redis 数据源的具体操作步骤，请参阅新增 *Redis* 数据源（见第 1858 页）。

新增 JMS 资源

了解新增 JMS 资源（包括JMS 连接工厂、JMS 队列连接工厂、JMS 主题连接工厂、JMS 队列、JMS 主题、JMS 激活规范）时界面元素的含义描述和操作规范，以帮助您正确创建 JMS 资源。

基本信息配置

JNDI 名称

资源的 JNDI 表示，保存后不可修改。

JMS 资源适配器

JMS 资源适配器由 JMS 消息服务提供商提供。通过“应用部署”功能将其部署到 xigemaAS 服务器上后，方可创建 JMS 资源 。

连接池

创建专用连接池：选择此项后，xigemaAS 管理中心会为当前 JMS 资源创建专用连接池。 JMS 连接工厂新增完成后，可以在当前 JMS 连接工厂详情页面查看或修改该连接池的配置信息。

使用共用连接池：从已新增到 xigemaAS 管理中心的连接池中选择要使用的连接池。多个 JMS 连接工厂可以使用一个连接池，xigemaAS 管理中心建议不同的 JMS 连接工厂使用不同的连接池。

激活规范名称

JMS 激活规范用于定义消息监听器，包括要监听的目标主题或队列，以及用于接收目标主题或队列中的消息的消息驱动 Bean。

为使 JMS 激活规范在监听到目标队列或主题中的消息后，能将其传递给相应的消息驱动 Bean，请按照以下格式定义激活规范名称：

- EJB 应用或 Web 应用：应用名称/消息驱动 Bean 名称，例如：MyEjbApp/myMDB
- 企业应用：应用名称/模块名称/消息驱动 Bean 名称，例如：MyEarApp/mdbModule/myMDB

在应用程序中开发消息驱动 Bean 时，可以通过以下方式定义并配置消息驱动 Bean：

- 使用 ejb-jar.xml 定义消息驱动 Bean 时，可以通过定义 ejb-name 属性来定义消息驱动 Bean 的名称；
- 使用 @MessageDriven 注解定义消息驱动 Bean 时，可以通过 name 属性来定义消息驱动 Bean 名称。若未定义 name 属性，则消息驱动 Bean 的名称默认为消息驱动 Bean 的类名（不包括包名）。

注：请确保已为包含消息驱动 Bean 的应用程序的部署目标配置相应的激活规范，即应用程序的部署目标是激活规范部署目标的一个子集。

最大端点数

用于设置消息监听的最大端点数（缺省值为500，最小值为1），会将消息并行传递给这些端点。 增大该数值会提高性能，但是同时也会增大在指定时间使用的线程数

标签

对 JMS 资源进行标识和分类的关键词，便于检索 JMS 资源。单个标签长度不超过20个字符，标签个数不超过5个。

描述

JMS 资源描述信息。长度不超过512个字符。

属性信息配置

destination

此属性可用于指定 destionaType 的对象名称，这些对象定义 JMS 队列或主题，端点（消息驱动的 bean）可从该队列或主题接收消息。

destination type

目标的类型。数据类型为 javax.jms.Queue 和 javax.jms.Topic，缺省值为 javax.jms.Queue。

数据源及数据库类型

数据源的新增，需要选择数据库类型和数据库驱动类型，选择完成后 xigemaAS 管理中心自动为您匹配数据源类名称。

数据源类型

xigemaAS 中的数据源有以下三种类型

- javax.sql.DataSource
- javax.sql.XADataSource
- javax.sql.ConnectionPoolDataSource

数据库类型、数据库驱动类型与数据源实现类对应关系

表 81: 数据库类型、数据库驱动类型与数据源实现类对应关系

数据库类型	数据库驱动类型	数据源实现类
xigemaDB	xigemaDB JDBC Driver	com.informix.jdbcx.IfxDataSource
		com.informix.jdbcx.IfxXADataSource
		com.informix.jdbcx.IfxConnectionPoolDataSource
DB2	DB2 UDB for iSeries (Native)	com.ibm.db2.jdbc.app.UDBDataSource
		com.ibm.db2.jdbc.app.UDBXADataSource
		com.ibm.db2.jdbc.app.UDBConnectionPoolDataSource
	DB2 UDB for iSeries (Toolbox)	com.ibm.as400.access.AS400JDBCDataSource
		com.ibm.as400.access.AS400JDBCXADataSource
		com.ibm.as400.access.AS400JDBCConnectionPoolDataSource
	DB2 Using IBM JCC Driver	com.ibm.db2.jcc.DB2DataSource
		com.ibm.db2.jcc.DB2XADataSource
		com.ibm.db2.jcc.DB2ConnectionPoolDataSource
Oracle	Oracle JDBC Driver	oracle.jdbc.pool.OracleDataSource
		oracle.jdbc.xa.client.OracleXADataSource
		oracle.jdbc.pool.OracleConnectionPoolDataSource
	Oracle JDBC Driver UCP	null (oracle.ucp.jdbc.PoolDataSourceImpl)
		oracle.ucp.jdbc.PoolXADataSourceImpl
		oracle.ucp.jdbc.PoolDataSourceImpl
	DataDirect Oracle	com.ddtek.jdbcx.oracle.OracleDataSource
Informix	Informix JDBC Driver	com.informix.jdbcx.IfxDataSource
		com.informix.jdbcx.IfxXADataSource

数据库类型	数据库驱动类型	数据源实现类
		com.informix.jdbcx.IfxConnectionPoolDataSource
	Informix Using IBM JCC Driver	com.ibm.db2.jcc.DB2DataSource
		com.ibm.db2.jcc.DB2XADataSource
		com.ibm.db2.jcc.DB2ConnectionPoolDataSource
MySQL	MySql JDBC Driver	com.mysql.jdbc.jdbc2.optional.MysqlDataSource
		com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
		com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource
Derby	Derby Client 40	org.apache.derby.jdbc.ClientDataSource40
		org.apache.derby.jdbc.ClientXADataSource40
		org.apache.derby.jdbc.ClientConnectionPoolDataSource40
	Derby Embedded 40	org.apache.derby.jdbc.EmbeddedDataSource40
		org.apache.derby.jdbc.EmbeddedXADataSource40
		org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource40
SQL Server	DataDirect SQLServer	com.ddtek.jdbcx.sqlserver.SQLServerDataSource
	Microsoft SQL Server JDBC Driver	com.microsoft.sqlserver.jdbc.SQLServerDataSource
		com.microsoft.sqlserver.jdbc.SQLServerXADataSource
		com.microsoft.sqlserver.jdbc.SQLServerConnectionPoolDataSource
Sybase	Sybase JDBC 2 Driver	com.sybase.jdbc2.jdbc.SybDataSource
		com.sybase.jdbc2.jdbc.SybXADataSource
		com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource
	Sybase JDBC 3 Driver	com.sybase.jdbc3.jdbc.SybDataSource
		com.sybase.jdbc3.jdbc.SybXADataSource
		com.sybase.jdbc3.jdbc.SybConnectionPoolDataSource
	Sybase JDBC 4 Driver	com.sybase.jdbc4.jdbc.SybDataSource
		com.sybase.jdbc4.jdbc.SybXADataSource
		com.sybase.jdbc4.jdbc.SybConnectionPoolDataSource
达梦	Da Meng JDBC Driver	dm.jdbc.xa.DmdbXADataSource
		dm.jdbc.pool.DmdbDataSource
		dm.jdbc.pool.DmdbConnectionPoolDataSource

数据库类型	数据库驱动类型	数据源实现类
人大金仓	kingbase	com.kingbase.Driver
神通	oscar	com.oscar.Driver
PostgreSql	PostgreSql JDBC Driver	org.postgresql.ds.PGPoolingDataSource

索 引

G

H

I

J

Q

R

S

T

W

X

Y

Z