Sun Java System Application Server Enterprise Edition 8.2 管理指南

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

文件号码 820–0847
2007 年 3 月

070430@17039

目录

图

表

示例

前言

《Sun Java System Application Server Enterprise Edition 8.2 管理指南》介绍 Application Server 的系统管理，包括配置、监视、安全性、资源管理和 Web 服务管理。

此前言包含有关整个 Sun Java™ System Application Server 文档集的信息和约定。

Application Server 文档集

Application Server 文档集介绍了部署规划和系统安装。独立 Application Server 文档的统一资源定位器 (Uniform Resource Locator, URL) 为 `http://docs.sun.com/app/docs/coll/1310.4`。Sun Java Enterprise System (Java ES) Application Server 文档的 URL 为 `http://docs.sun.com/app/docs/coll/1310.3` 和 `http://docs.sun.com/app/docs/coll/1386.2`。有关 Application Server 的说明，请按照下表列出的顺序参阅各本书。

表 P–1 Application Server 文档集中的书籍

书名	说明
发行说明	软件和文档的最新信息。其中包括以表格形式对所支持的硬件、操作系统、Java Development Kit (JDK™) 和数据库驱动程序所做的全面概述。
快速入门指南	如何开始使用 Application Server 产品。
Installation Guide	安装软件及其组件。
部署规划指南	评估系统需求和企业状况，确保以最适合您的站点的方式部署 Application Server。此外还介绍了部署服务器时应该注意的常见问题。
Developer's Guide	创建和实现要在 Application Server 上运行的 Java 2 Platform, Enterprise Edition（J2EE™ 平台）应用程序，这些应用程序遵循针对 J2EE 组件和 API 的开放式 Java 标准模型。其中包括有关开发者工具、安全性、调试、部署和创建生命周期模块的信息。
J2EE 1.4 Tutorial	使用 J2EE 1.4 平台技术和 API 开发 J2EE 应用程序。
管理指南	从管理控制台配置、管理和部署 Application Server 子系统和组件。
High Availability Administration Guide	有关高可用性数据库的安装后配置和管理说明。

表 P–1 Application Server 文档集中的书籍 （续）

书名	说明
Administration Reference	编辑 Application Server 配置文件 `domain.xml`。
Upgrade and Migration Guide	将应用程序迁移到新的 Application Server 编程模型，特别是从 Application Server 6.x 和 7 进行迁移。该指南还介绍了可导致与产品规范不兼容的相邻产品发行版和配置选项之间的差异。
Performance Tuning Guide	调节 Application Server 以提高性能。
Troubleshooting Guide	解决 Application Server 问题。
Error Message Reference	解析 Application Server 错误消息。
Reference Manual	可用于 Application Server 的实用程序命令，以手册页样式编写。其中包括 `asadmin` 命令行界面。

相关文档

Application Server 可以单独购买，也可以作为 Java ES 的组件购买，Java ES 是支持分布在网络或 Internet 环境中的企业应用程序的软件基础结构。如果购买的是作为 Java ES 组件的 Application Server，则您应熟悉 `http://docs.sun.com/coll/1286.2` 和 `http://docs.sun.com/coll/1382.2` 中的系统文档。有关 Java ES 及其组件的所有文档的 URL 为 `http://docs.sun.com/prod/entsys.5` 和 `http://docs.sun.com/prod/entsys.5?l=zh`。

对于其他 Sun Java System 服务器文档，请参见以下文档：

- Message Queue 文档
- Directory Server 文档
- Web Server 文档

此外，以下资源可能会有用：

- J2EE 1.4 规范 (`http://java.sun.com/j2ee/1.4/docs/index.html`)
- J2EE 1.4 教程 (`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`)
- J2EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

默认路径和文件名

下表介绍了在本书中使用的默认路径和文件名。

表 P–2 默认路径和文件名

占位符	说明	默认值

install-dir	表示 Application Server 的安装基目录。	Solaris™ 平台上的 Sun Java Enterprise System (Java ES) 的安装：
		`/opt/SUNWappserver/appserver`
		Linux 平台上的 Java ES 的安装：
		`/opt/sun/appserver/`
		其他 Solaris 和 Linux 的安装（非 root 用户）：
		user's home directory`/SUNWappserver`
		其他 Solaris 和 Linux 的安装（root 用户）：
		`/opt/SUNWappserver`
		Windows 的所有安装：
		SystemDrive`:\Sun\AppServer`

domain-root-dir	表示包含所有域的目录。	Solaris 平台上的 Java ES 的安装：
		`/var/opt/SUNWappserver/domains/`
		Linux 平台上的 Java ES 的安装：
		`/var/opt/sun/appserver/domains/`
		所有其他安装：
		install-dir`/domains/`

domain-dir	表示域的目录。 在配置文件中，您可能会看到 *domain-dir* 显示为以下内容： `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-dir*
instance-dir	表示服务器实例的目录。	*domain-dir*`/`*instance-dir*

印刷约定

下表描述了本书中使用的印刷约定。

表 P–3 印刷约定

字体	含义	示例
AaBbCc123	命令、文件和目录的名称；计算机屏幕输出	编辑 .login 文件。 使用 ls -a 列出所有文件。 machine_name% you have mail.
AaBbCc123	用户键入的内容，与计算机屏幕输出的显示不同	machine_name% **su** Password:
AaBbCc123	要使用实名或值替换的命令行占位符	删除文件的命令为 rm *filename*。
新词术语强调	新词或术语以及要强调的词（注：某些强调的词在联机状态下以粗体显示。）	**高速缓存**是存储在本地的副本。 请**勿**保存文件。
《书名》	书名	阅读《用户指南》的第 6 章。

符号约定

下表介绍了本书中使用的符号。

表 P–4 符号约定

符号	说明	示例	含义
[ ]	包含可选参数和命令选项。	ls [-l]	-l 选项不是必需的。
{ | }	包含所需命令选项的一组选择。	-d {y|n}	-d 选项要求使用 y 参数或 n 参数。
${ }	表示变量引用。	${com.sun.javaRoot}	引用变量 com.sun.javaRoot 的值。
-	连接需同时按下的多个击键。	Control-A	按住 Ctrl 键的同时按 A 键。
+	连接需连续按下的多个击键。	Ctrl+A+N	按 Ctrl 键，然后松开并依次按后面的键。
→	表示图形用户界面中的菜单项选定。	“文件” → “新建” → “模板”	从“文件”菜单中，选择“新建”。从“新建”子菜单中，选择“模板”。

文档、支持和培训

Sun Web 站点提供了有关以下附加资源的信息：

- 文档 (http://www.sun.com/documentation/)
- 支持 (http://www.sun.com/support/)
- 培训 (http://www.sun.com/training/)

第三方 Web 站点引用

本文档引用了第三方 URL 以提供其他相关信息。

注 – Sun 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他资料，Sun 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的、名义上造成的或连带产生的实际或名义损坏或损失，Sun 概不负责，也不承担任何责任。

Sun 欢迎您提出意见

Sun 致力于提高其文档的质量，并十分乐意收到您的意见和建议。为了共享您的意见，请访问 http://docs.sun.com，并单击 "Send Comments"（发送意见）。在联机表单中，请提供完整的文档标题和文件号码。文件号码是一个七位或九位的数字，可以在书的标题页或文档的 URL 中找到。例如，本书的文件号码为 820-0847。

◆ ◆ ◆ 第 1 章

1 入门

本章介绍 Sun Java System Application Server 管理。Application Server 管理包括许多任务，例如部署应用程序，创建和配置域、服务器实例和资源，控制（启动和停止）域和服务器实例，监视和管理性能，以及诊断和解决问题。本章包含以下几节：

关于 Sun Java System Application Server

Sun Java System Application Server 提供了 Java 2 Platform, Enterprise Edition（J2EE 平台）1.4 兼容的平台，用于开发和提供服务器端 Java 应用程序和 Web 服务。主要功能包括可伸缩的事务管理、容器管理持久性运行时、优秀的 Web 服务、群集、高可用性、安全性以及集成功能。

Application Server 具有以下版本：

- 平台版，为免费软件，用于软件开发以及部门级生产环境。
- 企业版，用于对业务至关重要的服务以及大规模生产环境。它支持通过负载平衡器插件和群集管理实现水平可伸缩性和服务连续性。企业版还支持通过高可用性数据库 (high availability database, HADB) 实现可靠会话状态管理。

本节包括以下主题：

什么是 Application Server？

Application Server 是一种平台，支持从 Web 发布到企业规模事务处理等多项服务，同时使开发者能够建立基于 JavaServer Pages (JSP)、Java Servlet 和 Enterprise JavaBeans (EJB) 技术的应用程序。

Application Server Platform Edition 是一款用于开发、生产部署和再分发的**免费**软件。有关再分发的更多信息，请访问 `http://www.sun.com/software/products/appsrvr/appsrvr_oem.xml`。

Application Server Enterprise Edition 提供了高级群集和故障转移技术。Application Server 基础结构支持部署许多类型的分布式应用程序，并且是建立基于面向服务的体系结构 (Service Oriented Architecture, SOA) 的应用程序的理想基础。SOA 是一种设计方法，旨在最大限度地重新使用应用程序服务。这些功能可以帮助您运行可扩展的且具有高可用性的 J2EE 应用程序。

- **可伸缩性** – 可伸缩性通过群集实现。群集是一组应用服务器实例，它们作为一个逻辑实体一起工作。群集中的每个 Application Server 实例都具有相同的配置，并被部署了相同的应用程序。

  通过将 Application Server 实例添加到群集来提高系统性能，从而实现了水平缩放。可以在不中断服务的情况下将 Application Server 实例添加到群集。HTTP、RMI/IIOP（通过基于 Internet 的 ORB 间协议的远程方法调用）和 JMS（Java(TM) 消息服务）负载平衡系统会将请求分发到群集中正常运行的 Application Server 实例中。
- **高可用性** – 可用性是指故障转移功能。如果某个服务器实例出现故障，则群集中的其他服务器实例便会接管故障实例的会话，并以无缝方式继续为客户机提供服务。会话信息存储在高可用性数据库 (high availability database, HADB) 中。HADB 支持 HTTP 会话和有状态会话 Bean 的持久性。

Application Server 体系结构

本节介绍了图 1–1，该图显示了 Application Server 的高级体系结构。

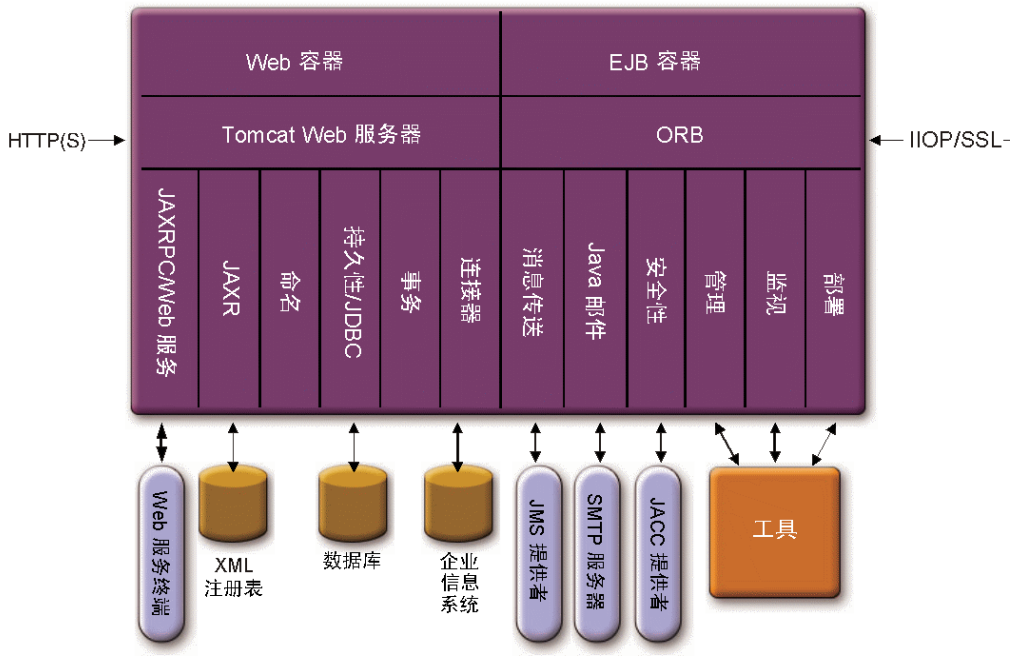

图 1–1 Application Server 体系结构

- **容器**—容器是一种运行时环境，它为 J2EE 组件提供安全性和事务管理等服务。图 1–1 显示了两种 J2EE 容器：Web 和 EJB。Web 组件（如 JSP 页面和 Servlet）在 Web 容器内运行。Enterprise Java Bean 在 EJB 容器内运行。
- **客户机访问**—运行时，浏览器客户机通过 HTTP（在 Internet 中使用的协议）与 Web 服务器进行通信来访问 Web 应用程序。HTTPS 协议用于需要安全通信的应用程序。Enterprise Java Bean 客户机通过 IIOP 或 IIOP/SSL（安全）协议与对象请求代理 (Object Request Broker, ORB) 进行通信。Application Server 具有分别用于 HTTP 协议、HTTPS 协议、IIOP 协议和 IIOP/SSL 协议的侦听器。每个侦听器独占使用特定的端口号。
- **Web 服务**—在 J2EE 平台上，可以部署一个 Web 应用程序，该应用程序提供由 Java API for XML-Based RPC (JAX-RPC) 实现的 Web 服务。J2EE 应用程序或组件还可以是其他 Web 服务的客户机。应用程序通过用于 XML 注册表的 Java API (JAXR) 访问 XML 注册表。
- **用于应用程序的服务**—J2EE 平台旨在使容器为应用程序提供服务。图 1–1 显示了以下服务：
  - 命名—命名和目录服务将对象绑定到名称。J2EE 应用程序通过查找对象的 JNDI 名称来找到对象。JNDI 表示 Java 命名和目录接口 API。

  - 安全性—Java 容器授权约定 (Java Authorization Contract for Containers, JACC) 是一组为 J2EE 容器定义的安全性约定。根据客户机的标识，容器限制对容器的资源和服务的访问。
- **事务管理**—事务是不可分的工作单元。例如，在银行帐户之间转帐是一个事务。事务管理服务用于确保完全完成事务或将事务回滚。

访问外部系统

J2EE 平台使应用程序能够访问应用服务器之外的系统。应用程序通过称为资源的对象连接到这些系统。管理员的职责之一是资源配置。J2EE 平台使得可以通过以下 API 和组件访问外部系统：

- JDBC－数据库管理系统 (database management system, DBMS) 提供了用于存储、组织和检索数据的工具。大多数商业应用程序将数据存储在关系数据库中，这些应用程序通过 JDBC API 访问关系数据库。由于数据库中的信息保存在磁盘上并在应用程序结束之后仍然存在，因此通常将数据库中的信息称为持久性信息。Application Server 包包含 Java DB 数据库管理系统。
- 消息传送－消息传送是软件组件或应用程序之间的一种通信方法。消息传送客户机可以向任何其他客户机发送消息，也可以从任何其他客户机接收消息。应用程序通过 Java 消息传送服务 (Java Messaging Service, JMS) API 访问消息传送提供者。Application Server 包含一个 JMS 提供者。
- 连接器－J2EE 连接器体系结构使 J2EE 应用程序和现有企业信息系统 (Enterprise Information System, EIS) 之间实现了集成。应用程序通过称为连接器或资源适配器的可移植 J2EE 组件访问 EIS。
- JavaMail－应用程序通过 JavaMail API 连接到 SMTP（简单邮件传输协议）服务器以发送和接收电子邮件。
- 服务器管理－图 1-1 的右下角显示了 Application Server 的管理界面。管理工具使用这些界面与 Application Server 进行通信。

管理工具

可以使用各种不同的工具和 API 来管理 Sun Java System Application Server：

- 第 28 页中的 “管理控制台”
- 第 29 页中的 “命令行界面（asadmin 实用程序）”
- 第 29 页中的 “JConsole”
- 第 30 页中的 “Application Server Management Extension (AMX)”

管理控制台

管理控制台是一种基于浏览器的工具，具有易于浏览的界面和联机帮助。要使用管理控制台，管理服务器（也称为域管理服务器或 DAS）必须处于运行状态。您需要管理

服务器主机名和端口号才能启动管理控制台。对于默认管理服务器，默认管理服务器端口号为 4849。您还需要管理用户名和密码才能登录到管理控制台。请查阅本节以获得更多详细信息。

要启动管理控制台，请在 Web 浏览器中键入以下内容：

```
https://hostname:port
```

例如：

```
https://kindness.sun.com:4849
```

如果管理控制台在运行有管理服务器的计算机上运行，则可以指定 `localhost` 作为主机名。

在 Windows 中，从“开始”菜单启动 Application Server 管理控制台。

命令行界面（asadmin 实用程序）

`asadmin` 实用程序是 Sun Java System Application Server 的命令行界面。您可以执行管理控制台所提供的一组相同的管理任务。可以通过 shell 的命令提示符或通过其他脚本或程序调用 `asadmin` 实用程序。`asadmin` 实用程序安装在 *install-dir*`/bin` 目录中。在 Solaris 上，默认的 Sun Java System Application Server 安装根目录为 `/opt/SUNWappserver`。

要启动 `asadmin` 实用程序，请转至 *install-dir*`/bin` 目录并输入以下内容：

```
$ asadmin
```

要列出 `asadmin` 中的可用命令，请使用：

```
asadmin> help
```

也可以在 shell 的命令提示符下发出 `asadmin` 命令：

```
$ asadmin help
```

要查看命令的语法和示例，请在命令名称后面键入 `help`。例如：

```
asadmin> help create-jdbc-resource
```

给定命令的 `asadmin help` 信息将显示该命令的 UNIX 手册页。这些手册页还在 Web 上的《Sun Java System Application Server Enterprise Edition 8.2 Reference Manual》中以 HTML 形式提供。

JConsole

在 Java 2, Platform Standard Edition 5.0 中，引入了 Java 监视和管理控制台 (JConsole)。JConsole 用于监视 Sun Java System Application Server。您可以使用 JConsole 的“远程”选项卡或“高级”选项卡来连接到 Application Server。

- “远程”选项卡：标识用户名、密码、管理服务器主机和 JMS 端口号（默认为 8686），并选择“连接”。
- “高级”选项卡：将 `JMXServiceURL` 标识为 `service:jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi`，并选择“连接”。`JMXServerURL` 列显在 `server.log` 文件中，同时还在域创建命令的命令窗口中输出。

Application Server Management Extension (AMX)

Application Server **M**anagement e**X**tension 是一个 API，显示 Application Server 的所有配置，并将 JMX 管理的 Bean当作实现 AMX 接口的、易于使用的客户机端动态代理进行监视。

有关使用 Application Server Management Extension 的更多信息，请参见《Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide》中的第 16 章 “Using the Java Management Extensions (JMX) API”。

Application Server 命令和概念

Sun Java System Application Server 由一个或多个域组成。域是管理边界或上下文。每个域都具有一个与其关联的管理服务器（也称为域管理服务器或 DAS），并且由零个或多个独立实例和/或群集组成。每个群集都具有一个或多个同类服务器实例。服务器实例是在一个物理计算机上运行 Application Server 的一个 Java 虚拟机 (Java Virtual Machine, JVM)。域中的服务器实例（无论为独立实例还是群集实例）可以在不同的物理主机上运行。

本节包含以下主题：

域

域是一组共同进行管理的实例。但是，一个应用服务器实例只能属于一个域。除了管理边界以外，域还提供了基本的安全性结构，凭借此结构，不同的管理员可以管理应用服务器实例的特定组（域）。通过将服务器实例分组到单独的域中，不同的组织和管理员可以共享一个 Application Server 安装。每个域都有自己的独立于其他域的配置、日志文件和应用程序部署区域。如果更改某个域的配置，其他域的配置不会受到影响。

Sun Java System Application Server 安装程序会创建默认管理域（名为 `domain1`）。它还会创建关联的域管理服务器（名为 `server`）。您必须提供管理服务器端口号。默认管理服务器端口号为 4849。安装程序还将查询管理用户名和密码。安装之后，还可以创建其他管理域。

域管理服务器 (Domain Administrative Server, DAS)

每个域都有自己的域管理服务器 (Domain Administrative Server, DAS)，该服务器具有唯一的端口号。管理控制台与特定 DAS 进行通信以管理关联的域。每个管理控制台会话都可用于配置和管理特定的域。

域管理服务器 (Domain Administrative Server, DAS) 是专门指定的应用服务器实例，用于托管管理应用程序。DAS 会对管理员进行验证，接受来自管理工具的请求，并与域中的服务器实例进行通信以执行请求。有时，DAS 称为管理服务器或默认服务器。将它称为默认服务器是因为它是在 Sun Java System Application Server 安装时创建并可用于部署的唯一服务器实例。DAS 只是一个具有附加管理功能的服务器实例。

每个管理控制台会话都可用于配置和管理单个域。如果创建了多个域，则必须启动另外一个管理控制台会话以管理其他域。为管理控制台指定 URL 时，请确保使用与要管理的域关联的 DAS 的端口号。

群集

群集是命名的服务器实例集合，它们共享一组相同的应用程序、资源和配置信息。服务器实例只能属于一个群集。通过在多台计算机之间分配负载，群集有助于实现服务器实例负载平衡。通过进行实例级故障转移，群集还有助于实现高可用性。从管理角度看，群集表示一个虚拟化的实体，对群集的操作（如部署应用程序）将对构成群集的所有实例都起作用。

节点代理

需要对域中每个节点使用轻量代理（例如，仅托管 JMX 运行时），以便于对实例进行远程生命周期管理。它的主要用途是按照 DAS 指示启动、停止和创建服务器实例。节点代理还可用作监视程序并重新启动出现故障的进程。与 DAS 相同，节点代理应仅用于特定的管理操作，并且不应期望它具有高可用性。但是，节点代理是一个“始终运行”的组件，并且必须将其配置为通过本机 O/S 节点引导（例如，Solaris/Linux `inetd` 或作为 Windows 服务）启动。DAS 不需要节点代理。

服务器实例

服务器实例是在单个节点上托管 J2EE 1.4 Application Server 的单个 J2EE 兼容的 Java 虚拟机。每个服务器实例在域中都具有唯一的名称。群集服务器实例是群集的成员，并从其父群集中接收所有的应用程序、资源和配置，从而确保群集中的所有实例是同类实例。非群集服务器实例不属于群集，并且具有一组独立的应用程序、资源和配置。

应用服务器实例构成了应用程序部署的基础。每个实例都属于单个域。每个服务器实例（DAS 除外）必须包含对节点代理名称的引用，该名称定义实例将要驻留的计算机。

如果拓扑中包含远程服务器实例（DAS 之外的服务器实例），请创建节点代理以管理和改善远程服务器实例。节点代理负责创建、启动、停止和删除服务器实例。使用命令行界面命令可以设置节点代理。图 1–2 详细显示了一个应用服务器实例。

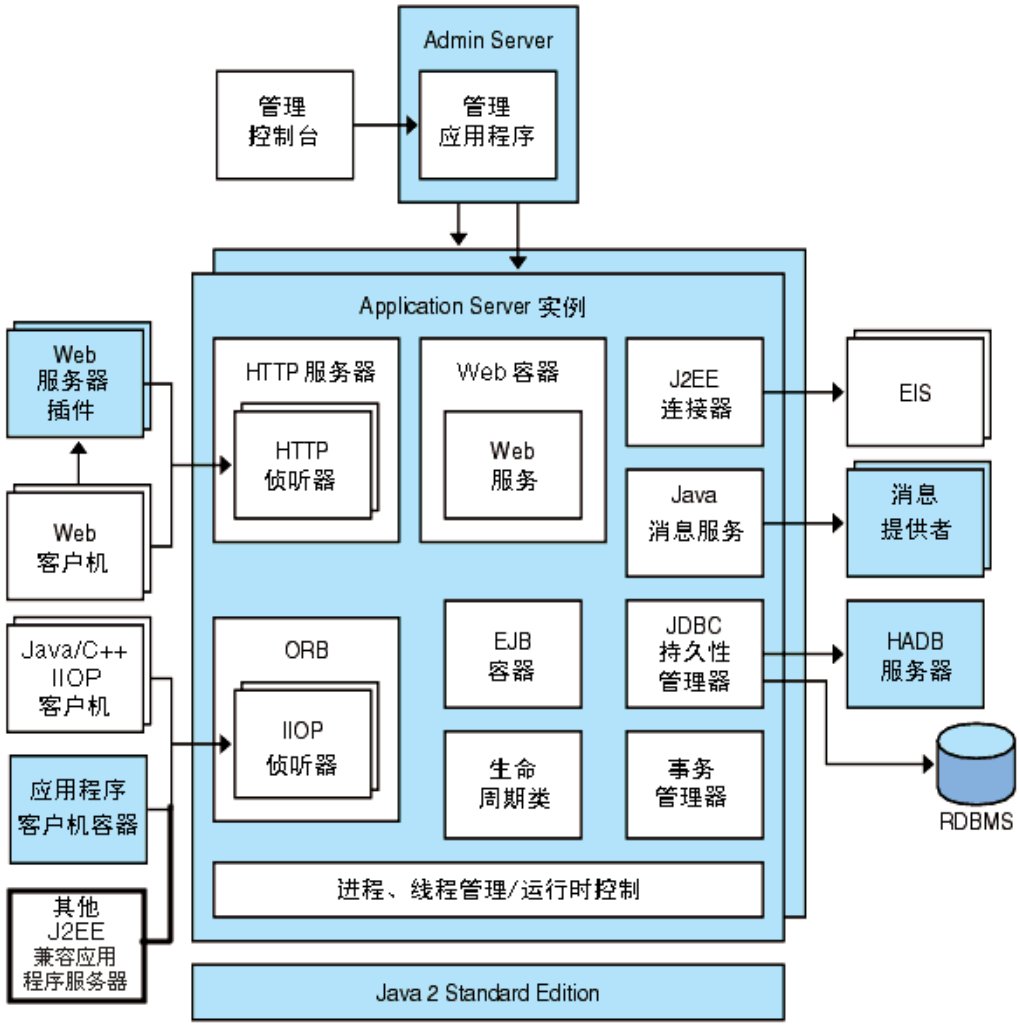

图 1–2　Application Server 实例

Sun Java System Application Server 在安装时将创建一个称为 server 的应用服务器实例。对于很多用户来说，一个应用服务器实例即可满足他们的需要。但是，根据用户环境的不同，可能需要创建一个或多个附加应用服务器实例。例如，在开发环境中，可以使用不同的应用服务器实例来测试不同的 Application Server 配置，或者比较和测试不同的应用程序部署。由于添加或删除应用服务器实例比较容易，因此可以使用这些实例创建临时的沙箱区来进行试验。

此外，还可以为每个应用服务器实例创建虚拟服务器。在一个已安装的应用服务器实例中，您可以提供公司或个人域名、IP 地址和某些管理功能。对于用户，就好比他们

拥有自己的 Web 服务器，但无需进行硬件和基础服务器的维护。这些虚拟服务器不能跨应用服务器实例使用。有关虚拟服务器的更多信息，请参见第 12 章。

在操作部署中，很多情况下都可以使用虚拟服务器代替多个应用服务器实例。但是，如果虚拟服务器不能满足需求，您也可以使用多个应用服务器实例。停止应用服务器实例后，它将不再接受新的连接，只是等待所有未完成的连接完成。如果您的计算机崩溃或脱机，则服务器将退出，并且正在处理的所有请求都将丢失。

Application Server 命令

Application Server 的管理包括创建、配置、控制和管理域、群集、节点代理和服务器实例等任务。本节包括以下主题：

创建域

域是使用 `create-domain` 命令创建的。以下示例命令将创建名为 `mydomain` 的域。管理服务器在端口 1234 上进行侦听，管理用户名为 `hanan`。该命令提示输入管理密码和主密码。

```
$ asadmin create-domain --adminport 1234 --adminuser hanan mydomain
```

要为 `mydomain` 域启动管理控制台，请在浏览器中输入以下 URL：

`http://主机名:1234`

对于前面的 create-domain 示例，域的日志文件、配置文件和部署的应用程序现在位于以下目录中：

domain-root-dir/mydomain

要在其他位置创建域的目录，请指定 --domaindir 选项。要查看完整的命令语法，请键入 asadmin help create-domain。

删除域

使用 asadmin delete-domain 命令可以删除域。只有可以管理域的操作系统用户（即超级用户）可以成功地执行此命令。例如，要删除名为 mydomain 的域，请键入以下命令：

```
$ asadmin delete-domain mydomain
```

列出域

使用 asadmin list-domains 命令可以找到在计算机上创建的域。要列出默认 *domain-root-dir* 目录中的域，请键入以下命令：

```
$ asadmin list-domains
```

要列出在其他目录中创建的域，请指定 --domaindir 选项。

启动域

启动域时，将启动管理服务器和应用服务器实例。启动应用服务器实例之后，应用服务器实例将持续运行、侦听并接受请求。必须单独启动各个域。

要启动域，请键入 asadmin start-domain 命令并指定域名。例如，要启动默认域 (domain1)，请键入以下命令：

```
$ asadmin start-domain --user admin domain1
```

如果只有一个域，则可以省略域名。要查看完整的命令语法，请键入 asadmin help start-domain。如果省略了密码数据，系统将提示您提供此数据。

在 Windows 上启动默认域

在 Windows“开始”菜单中，依次选择“程序”-> "Sun Microsystems" -> "Application Server" ->“启动管理服务器”。

停止域

停止域将关闭该域的管理服务器和应用服务器实例。停止域时，服务器实例将停止接收新的连接，然后等待所有未完成的连接完成。由于服务器实例必须完成其关闭进程，因此该进程需要几秒钟时间。当域停止时，管理控制台或大多数 asadmin 命令都无法使用。

要停止域，请键入 asadmin stop-domain 命令并指定域名。例如，要停止默认域 (domain1)，请键入以下命令：

```
$ asadmin stop-domain domain1
```

如果只有一个域，则域名是可选键入项。要查看完整的语法，请键入 asadmin help stop-domain。

在 Windows 上停止默认域

在“开始”菜单中，依次选择“程序”-> "Sun Microsystems" -> "Application Server" ->“停止管理服务器”。

重新启动域

重新启动服务器与重新启动域相同。要重新启动域或服务器，请停止然后再启动域。

创建群集

群集是使用 create-cluster 命令创建的。以下示例将创建名为 mycluster 的群集。管理服务器主机为 myhost，服务器端口为 1234，管理用户名为 admin。该命令提示输入管理密码。

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

要查看完整的语法，请键入 asadmin help create-cluster。

启动群集

群集是使用 start-cluster 命令启动的。以下示例将启动名为 mycluster 的群集。该命令提示输入管理密码。

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

myhost 为管理服务器主机，1234 为管理端口，admin 为管理用户名。

要查看完整的语法，请键入 asadmin help start-cluster。启动群集时，将启动群集中的所有服务器实例。无法启动没有实例的群集。

停止群集

群集是使用 stop-cluster 命令停止的。以下示例将停止名为 mycluster 的群集。该命令提示输入管理密码。

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

myhost 为管理服务器主机，1234 为管理端口，admin 为管理用户名。

要查看完整的语法，请键入 asadmin help stop-cluster。停止群集时，将停止群集中的所有服务器实例。无法停止没有实例的群集。

创建节点代理

节点代理是使用 create-node-agent 命令创建的。以下示例将创建名为 mynodeagent 的节点代理。管理服务器主机为 myhost，管理服务器端口为 1234，管理用户名为 admin。该命令提示输入管理密码。

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

要查看完整的语法，请键入 asadmin help create-node-agent。

启动节点代理

使用 start-node-agent 命令并指定节点代理名称可以启动节点代理。例如，要启动节点代理 mynodeagent，请键入以下命令：

```
$ asadmin start-node-agent --user admin mynodeagent
```

要查看完整的语法，请键入 asadmin help start-node-agent。

停止节点代理

使用 stop-node-agent 命令并指定节点代理名称可以停止节点代理。例如，要停止节点代理 mynodeagent，请键入以下命令：

```
$ asadmin stop-node-agent mynodeagent
```

要查看完整的语法，请键入 asadmin help stop-node-agent。

创建实例

服务器实例是使用 create-instance 命令创建的。以下示例将创建名为 myinstance 的实例。管理服务器主机为 myhost，管理服务器端口为 1234，管理用户名为 admin。该命令提示输入管理密码。

以下示例将创建名为 myinstance 的群集服务器实例。该命令提示输入管理密码。

```
$ asadmin create-instance --host myhost --port 1234
--user admin --cluster mycluster --nodeagent mynodeagent myinstance
```

myhost 为管理服务器主机，管理端口为 1234，管理用户名为 admin，此服务器实例所属的群集为 mycluster，管理此服务器实例的节点代理为 mynodeagent。

要查看完整的语法，请键入 asadmin help create-instance。

要创建独立服务器实例，请不要指定 --cluster 选项。

以下示例将创建名为 myinstance 并由名为 mynodeagent 的节点代理管理的独立服务器实例。

```
$ asadmin create-instance --host myhost --port 1234
--user admin --nodeagent mynodeagent myinstance
```

启动实例

服务器实例是使用 start-instance 命令启动的。以下示例将启动名为 myinstance 的服务器实例。该命令提示输入管理密码。

```
$ asadmin start-instance --host myhost --port 1234 --user admin  myinstance
```

管理服务器主机为 myhost，管理端口为 1234，管理用户名为 admin。服务器实例 myinstance 可以为群集实例，也可以为独立实例。

要查看完整的语法，请键入 asadmin help start-instance。

停止实例

服务器实例是使用 stop-instance 命令停止的。以下示例将停止名为 myinstance 的实例。该命令提示输入管理密码。

```
$ asadmin stop-instance --host myhost --port 1234 --user admin  myinstance
```

管理服务器主机为 myhost，管理端口为 1234，管理用户名为 admin。服务器实例 myinstance 可以为群集实例，也可以为独立实例。

要查看完整的语法，请键入 asadmin help stop-instance。

重新启动实例

要重新启动服务器实例，请停止然后再启动实例。

重新创建域管理服务器

要进行镜像和提供域管理服务器 (Domain Administration Server, DAS) 的工作副本，您必须拥有以下设备：

- 一台包含原始 DAS 的计算机 (machine1)。
- 一台包含群集的计算机 (machine2)，该群集具有运行应用程序并满足客户机需要的服务器实例。该群集是使用第一台计算机上的 DAS 配置的。
- 一台备份计算机 (machine3)，当第一台计算机崩溃时，需要在该备份计算机上重新创建 DAS。

注 – 必须对第一台计算机上的 DAS 进行备份。使用 `asadmin backup-domain` 来备份当前域。

▼ 迁移 DAS

以下步骤用于将域管理服务器从第一台计算机 (machine1) 迁移到第三台计算机 (machine3)。

1. **在第三台计算机上安装应用服务器，方法与在第一台计算机上安装相同。**
   为了可以在第三台计算机上正确地恢复 DAS 并且不会发生路径冲突，您必须执行此操作。

   a. **使用命令行（交互式）模式来安装应用服务器管理软件包。要激活交互式命令行模式，请使用 `console` 选项调用安装程序：**

      `./`*`bundle-filename`* `-console`

      要使用命令行界面进行安装，您必须具有超级用户权限。

   b. **要安装默认域，请取消选择该选项。**
      只有具有相同体系结构并具有**完全**相同的安装路径（即都使用相同的 *install-dir* 和 *domain-root-dir*）的两台计算机才支持备份域的恢复。

2. **将第一台计算机上的备份 ZIP 文件复制到第三台计算机上的 *domain-root-dir* 中。也可以通过 FTP（文件传输协议）方式传输文件。**

3. **执行 `asadmin restore-domain` 命令以将 ZIP 文件恢复到第三台计算机：**

   ```
   asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip domain1
   ```

   可以备份任何域。但是，在重新创建域时，域名称应与原始域名称相同。

4. **将第三台计算机上的 *domain-root-dir*`/domain1/generated/tmp` 目录的权限更改为与第一台计算机上相同目录的权限相匹配。**
   该目录的默认权限为：`?drwx------?`（或 700）。

   例如：

   ```
   chmod 700 domain-root-dir/domain1/generated/tmp
   ```

   以上示例假定您备份的是 `domain1`。如果备份的是其他名称的域，应使用要备份的域的名称替换上面的 `domain1`。

5. **更改第三台计算机的 `domain.xml` 文件中的主机属性值：**

6 **更新第三台计算机上的** *domain-root-dir*`/domain1/config/domain.xml`**。**

例如，搜索 `machine1` 并将其替换为 `machine3`。这样，您就可以将：

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

更改为：

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

7 **将：**

```
<jms-service... host=machine1.../>
```

更改为：

```
<jms-service... host=machine3.../>
```

8 **在 machine3 上启动已恢复的域：**

```
asadmin start-domain --user admin-user --password admin-password domain1
```

9 **在 machine2 上更改节点代理下的 DAS 主机属性值。**

10 **更改 machine2 中** *install-dir*`/nodeagents/`*nodeagent*`/agent/config/das.properties` **中的** `agent.das.host` **属性值。**

11 **在 machine2 上重新启动节点代理。**

注 – 使用 `asadmin start-instance` 命令启动群集实例，可以使这些实例与已恢复的域同步。

更改管理员密码

要重新设置管理员密码，必须停止域中的所有节点代理。这将停止所有关联的服务器实例。此时，所有的服务器实例和节点代理均停止，只有域管理服务器 (Domain Administration Server, DAS) 在运行。

现在，您可以按如下方式更改管理用户的密码：

1. 使用命令行界面更改管理密码：

   ```
   asadmin update-file-user --authrealmname admin-realm ... --userpassword
   newpassword <admin-user-name>
   ```

2. 使用管理控制台更改管理密码：

   选择管理服务器的配置节点>“安全性”>“领域”>“管理领域”>“编辑文件领域用户”。

   将显示一条指示您已经成功更改了管理员密码的消息。

3. 按如下方式使用新密码重新启动域管理服务器 (Domain Admin Server, DAS)：

使用命令行界面：`asadmin start-domain --user admin --password newpassword domain1`

假设进行了以下配置：域中具有两个节点代理 (`i1na`, `c1-na`) 以及三个实例（`c1i1` 和 `c1i2` 位于名为 `c1` 的相同群集中，`i1` 为独立服务器实例）。

4. 使用新密码重新启动代理节点而不启动实例。

   例如：

   ```
   asadmin start-node-agent --user admin --password newpassword
   --startinstances=false i1-na asadmin
   ```

   ```
   asadmin start-node-agent --user admin --password newpassword
   --startinstances=false c1-na
   ```

5. 重新启动服务器和群集。

   ```
   asadmin start-node-agent --user admin --password newpassword ... c1
   ```

   ```
   asadmin start-node-agent --user admin --password newpassword i1
   ```

Application Server 配置

Sun Java System Application Server 配置存储在 `domain.xml` 文件中。`domain.xml` 是一个表示 Application Server 配置状态的文档。它是给定管理域的中心系统信息库。此文档包含 Application Server 域模型的 XML 表示。`domain.xml` 的内容由以域 DTD 形式表示的规范控制。

本节包括以下主题：

- 第 41 页中的 “更改 Application Server 配置”
- 第 42 页中的 “Application Server 中的端口”
- 第 42 页中的 “更改 J2SE 软件”

更改 Application Server 配置

在进行以下任何配置更改时，需要重新启动服务器以使更改生效：

- 更改 JVM 选项
- 更改端口号
- 管理 HTTP 服务、IIOP 服务和 JMS 服务
- 管理线程池

有关说明，请参见第 36 页中的 “重新启动域”。

在进行以下任何配置更改时，如果启用了动态配置，则**不**需要重新启动服务器即可使更改生效：

- 部署和取消部署应用程序

- 添加或删除 JDBC、JMS 和连接器资源和池
- 更改日志记录级别
- 添加文件领域用户
- 更改监视级别
- 启用和禁用资源和应用程序

请注意，`asadmin reconfig` 命令已过时，不再需要此命令。配置更改将被动态应用到服务器。

Application Server 中的端口

下表介绍了 Application Server 的端口侦听器。

表 1–1 使用端口的 Application Server 侦听器

侦听器	默认端口号	说明
管理服务器	4849	通过管理控制台和 `asadmin` 实用程序访问域的管理服务器。对于管理控制台，请在浏览器的 URL 中指定端口号。远程执行 `asadmin` 命令时，请使用 --port 选项指定端口号。
HTTP	8080	Web 服务器侦听端口上的 HTTP 请求。要访问已部署的 Web 应用程序和服务，客户机应连接到此端口。
HTTPS	8181	为安全通信配置的 Web 应用程序在单独的端口上进行侦听。
IIOP	3700	企业 Bean（EJB 组件）的远程客户机通过 IIOP 侦听器访问 Bean。
IIOP、SSL	3820/3890	另一个端口由为安全通信配置的 IIOP 侦听器使用。
IIOP、SSL 和相互验证		另一个端口由为双向（客户机和服务器）验证配置的 IIOP 侦听器使用。
JMX	8686	另一个端口由 JMX 连接器使用以与 DAS 进行通信。

更改 J2SE 软件

Application Server 依赖于 Java 2 Standard Edition (J2SE) 软件。安装 Application Server 时，指定 J2SE 软件的目录。有关更改 J2SE 软件的说明，请参见第 17 章。

◆ ◆ ◆ 第 2 章

2 部署应用程序

本章说明如何在 Application Server 上部署（安装）J2EE 应用程序。本章包含以下几节：

部署生命周期

安装 Application Server 并启动域之后，您可以部署（安装）J2EE 应用程序和模块。在部署过程中和更改应用程序时，应用程序或模块可能会经过以下阶段：

1. 初始部署

   部署应用程序或模块之前，请启动域。

   将应用程序或模块部署（安装）到特定的独立服务器实例或群集。由于应用程序和模块封装在归档文件中，因此在部署期间应指定归档文件名。默认情况下，部署到默认服务器实例 `server`。

   如果部署到服务器实例或群集，应用程序或模块将存在于域的中心系统信息库中，并由部署为目标的任何群集或服务器实例所引用。

   您还可以使用 `asadmin deploy` 命令（而非管理控制台）将其部署到域。如果将应用程序或模块只部署到域，则应用程序或模块将存在于域的中心系统信息库中，但要在您添加引用之后才会有服务器实例或群集引用该应用程序或模块。

   部署是动态的：部署应用程序或模块后，无需重新启动服务器实例即可使用应用程序或模块。如果重新启动了服务器实例，所有已部署的应用程序和模块仍将处于部署状态并且可用。

2. 启用或禁用

   默认情况下，将启用已部署的应用程序或模块，这表示可以运行它并且可以通过客户机对其进行访问（如果应用程序或模块已部署到可访问的服务器实例或群集）。要禁止访问，请禁用应用程序或模块。在部署之后，已禁用的应用程序或模块并未从域中被卸载，而且可以很容易地将其启用。

3. 添加或删除已部署应用程序或模块的目标

   部署后，应用程序或模块将存在于中心系统信息库中，并可以被多个服务器实例和/或群集引用。最初，作为目标部署到的服务器实例或群集将引用应用程序或模块。

   在部署应用程序或模块后，要更改引用应用程序或模块的服务器实例和群集，请使用管理控制台更改应用程序或模块的目标，或使用 `asadmin` 工具更改应用程序引用。由于应用程序本身存储在中心系统信息库中，因此添加或删除目标将添加或删除不同目标上同一版本的应用程序。但是，可以在一个目标上启用而在另一个目标上禁用部署到多个目标的应用程序，因此即使应用程序被一个目标引用，也只有在该目标上启用它时用户才能对其进行使用。

4. 重新部署

   要替换已部署的应用程序或模块，请将其重新部署。重新部署将自动取消部署先前已部署的应用程序或模块，并将其替换为新的应用程序或模块。

   当通过管理控制台重新部署时，重新部署的应用程序或模块将部署到域中，并且所有引用该应用程序或模块的独立或群集服务器实例将自动接收新的版本（如果已启用动态重新配置）。如果使用 `asadmin deploy` 命令来重新部署，请将 `domain` 指定为目标。

   对于生产环境，请使用滚动升级（升级应用程序而不中断服务）。

5. 取消部署

   要卸载应用程序或模块，请取消部署应用程序或模块。

自动部署

自动部署功能使您能够通过将预封装的应用程序或模块复制到 *domain-dir*`/autodeploy` 目录来部署该应用程序或模块。

例如，将名为 `hello.war` 的文件复制到 *domain-dir*`/autodeploy` 目录。要取消部署应用程序，请从 `autodeploy` 目录中删除 `hello.war` 文件。

也可以使用管理控制台或 `asadmin` 工具来取消部署应用程序。在这种情况下，归档文件将保留。

注 – 自动部署仅适用于默认服务器实例。

自动部署功能适用于开发环境。它与会话持久性（一种生产环境功能）不兼容。如果启用了自动部署，则不要启用会话持久性。

部署未封装的应用程序

此功能适用于高级开发者。

使用目录部署仅部署到默认服务器实例 (server)。您不能使用它来部署到群集或独立服务器实例。

包含未封装的应用程序或模块的目录有时称为展开的目录。目录的内容必须与相应的 J2EE 归档文件的内容匹配。例如，如果部署某一目录中的 Web 应用程序，则该目录的内容必须与相应的 WAR 文件的内容相同。有关必需的目录内容的信息，请参见相应的规范。

您可以直接在展开的目录中更改部署描述符文件。

如果您的环境被配置为使用动态重新装入，则还可以从目录中动态重新装入已部署的应用程序。有关更多信息，请参见第 184 页中的 “配置高级设置” 。

使用部署规划

此功能适用于高级开发者。

部署规划是指仅包含特定于 Application Server 的部署描述符的 JAR 文件。有关这些部署描述符（例如 `sun-application.xml`）的说明，请参见 Application Server Developer's Guide。部署规划是 JSR 88: J2EE Application Deployment 实现的一部分。使用部署规划可以部署不包含特定于 Application Server 的部署描述符的应用程序或模块。

要使用部署规划进行部署，请指定 `asadmin deploy` 命令的 `--deploymentplan` 选项。例如，以下命令将根据 `mydeployplan.jar` 文件中指定的规划来部署 `myrosterapp.ear` 文件中的企业应用程序。

```
$ asadmin deploy --user admin ---deploymentplan mydeployplan.jar myrosterapp.ear
```

在企业应用程序 (EAR) 的部署规划文件中，`sun-application.xml` 文件位于根目录下。根据以下语法来存储每个模块的部署描述符：*module-name.sun-dd-name*，其中 *sun-dd-name* 取决于模块类型。如果模块包括 CMP 映射文件，则该文件命名为 *module-name*`.sun-cmp-mappings.xml`。`.dbschema` 文件存储在根级别目录下，并用井号 (#) 替换每个正斜杠符号 (/)。下面列出的内容显示了企业应用程序 (EAR) 的部署规划文件的结构。

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
```

```
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-ejb-jar.xml
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

在 Web 应用程序或模块文件的部署规划中，特定于 Application Server 的部署描述符位于根级别目录下。如果独立 EJB 模块包括 CMP Bean，则部署规划包括位于根级别目录的 `sun-cmp-mappings.xml` 和 `.dbschema` 文件。在下面列出的内容中，部署规划描述了 CMP Bean。

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 sun-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

使用 deploytool 实用程序

为软件开发者设计的 `deploytool` 实用程序可以封装和部署 J2EE 应用程序和模块。有关如何使用 `deploytool` 的说明，请参见《The J2EE 1.4 Tutorial》。

J2EE 归档文件的类型

软件供应商将应用程序或模块封装到了归档文件中。要部署应用程序或模块，请指定归档文件名。归档文件的内容和结构是按照 J2EE 平台的规范定义的。J2EE 归档文件的类型包括：

- Web 应用程序归档文件 (Web Application Archive, WAR)：WAR 文件由 Servlet 和 JSP 等 Web 组件以及静态 HTML 页面、JAR 文件、标记库和实用程序类组成。WAR 文件的扩展名为 `.war`。
- EJB JAR：EJB JAR 文件包含一个或多个企业 Bean（用于 EJB 技术的组件）。EJB JAR 文件还包括企业 Bean 所需的任何实用程序类。EJB JAR 文件的扩展名为 `.jar`。
- J2EE 应用程序客户机端 JAR：该 JAR 文件包含通过 RMI/IIOP 访问服务器端组件（如企业 Bean）的 J2EE 应用程序客户机端的代码。在管理控制台中，J2EE 应用程序客户机端被称为“应用程序客户机端”。J2EE 应用程序客户机端 JAR 文件的扩展名为 `.jar`。
- 资源适配器归档文件 (Resource Adapter Archive, RAR)：RAR 文件保存资源适配器。资源适配器是按照 J2EE 连接器体系结构规范定义的，它是允许企业 Bean 和 Web 组件和应用程序客户机端访问资源和外部企业系统的可移植组件。资源适配器经常称为连接器。RAR 文件的扩展名为 `.rar`。
- 企业应用程序归档文件 (Enterprise Application Archive, EAR)：EAR 文件包含一个或多个 WAR 文件、EJB JAR 文件、RAR 文件或 J2EE 应用程序客户机端 JAR 文件。EAR 文件的扩展名为 `.ear`。

软件供应商可以将应用程序汇编为一个 EAR 文件或多个独立的 WAR 文件、EJB JAR 文件和应用程序客户机端 JAR 文件。在管理工具中，用于所有类型文件的部署页面和命令都是类似的。

命名约定

在给定域中，已部署的应用程序和模块的名称必须是唯一的。

- 如果使用管理控制台进行部署，请在“应用程序名称”字段中指定名称。
- 如果使用 `asadmin deploy` 命令进行部署，则应用程序或模块的默认名称为要部署的 JAR 文件的前缀。例如，如果部署 `hello.war` 文件，则 Web 应用程序的名称为 `hello`。要覆盖默认名称，请指定 `--name` 选项。

在一个应用程序中，不同类型的模块可以具有相同的名称。部署应用程序时，将使用 `_jar`、`_war` 和 `_rar` 后缀来命名保存各个模块的目录。相同类型的模块在一个应用程序内必须具有唯一的名称。此外，数据库架构文件名在一个应用程序内必须是唯一的。

建议将类似 Java 包的命名模式用于模块文件名、EAR 文件名、在 `ejb-jar.xml` 文件的 `<module-name>` 部分找到的模块名以及在 `ejb-jar.xml` 文件的 `<ejb-name>` 部分找到的 EJB 名称。使用这种类似软件包的命名模式可以确保不会发生名称冲突。这种命名方式的好处不仅适用于 Application Server，也适用于其他 J2EE 应用服务器。

EJB 组件的 JNDI 查找名也必须是唯一的。建立一致的命名约定可能会有帮助。例如，将应用程序名和模块名附加到 EJB 名称中是一种确保名称唯一的方式。在这种情况下，`mycompany.pkging.pkgingEJB.MyEJB` 将是模块 `pkgingEJB.jar`（该模块封装在应用程序 `pkging.ear` 中）中 EJB 的 JNDI 名称。

请确保软件包和文件名称中不包含空格或操作系统视为非法的字符。

◆ ◆ ◆ 第 3 章

JDBC 资源

JDBC 资源（数据源）为应用程序提供了连接数据库的方法。通常，管理员要为部署在域中的应用程序访问的每个数据库创建 JDBC 资源。（但是，可以为一个数据库创建多个 JDBC 资源。）

本章包括以下几个部分：

创建 JDBC 资源

为了存储、组织和检索数据，大多数应用程序都采用了关系数据库。Java EE 应用程序通过 JDBC API 访问关系数据库。

JDBC 资源（数据源）为应用程序提供了连接数据库的方法。创建 JDBC 资源之前，请先创建 JDBC 连接池。

要创建 JDBC 资源，需要指定标识资源的唯一 JNDI 名称。JDBC 资源的 JNDI 名称应在 `java:comp/env/jdbc` 子上下文中。例如，工资单数据库资源的 JNDI 名称可以为 `java:comp/env/jdbc/payrolldb`。由于所有资源 JNDI 名称都位于 `java:comp/env` 子上下文中，因此在管理控制台中指定 JDBC 资源的 JNDI 名称时仅需输入 `jdbc/`*name*。例如，对于工资单数据库，可以指定 `jdbc/payrolldb`。

要使用管理控制台创建 JDBC 资源，请选择 “资源” > “JDBC 资源” 。指定资源设置，如下所示：

- JNDI 名称：指定唯一名称。JNDI 名称在分布式计算环境中组织和查找组件的方式与图书馆中卡片目录组织和表示书籍位置的方式类似。因此，JNDI 名称成为访问 JDBC 资源的一种重要方法。按照约定，该名称应以 `jdbc/` 字符串开头。例如：`jdbc/payrolldb`。请不要忘记正斜杠。
- 池名称：选择要与新 JDBC 资源关联的连接池。
- 说明：键入资源的简短说明。
- 状态：如果要使资源不可用，请取消选中“已启用”复选框。默认情况下，创建资源之后立即可以使用资源（已启用）。

要使用命令行实用程序创建 JDBC 资源，请使用 `create-jdbc-resource` 命令。

创建 JDBC 连接池

要创建 资源，请指定与其关联的连接池。多个 JDBC 资源可以指定一个连接池。

连接池是用于特定数据库的一组可重复使用的连接。由于每创建一个新的物理连接都会耗费时间，因此服务器维护了可用连接池以提高性能。应用程序请求连接时可以从池中获取一个连接。应用程序关闭连接时，连接将返回到池中。

创建连接池时，您实际上是在定义特定数据库连接的各个方面。创建连接池之前，您必须首先安装并集成 JDBC 驱动程序。连接池的属性可能会随数据库供应商的不同而有所不同。有一些属性是通用的，如数据库名称 (URL)、用户名和密码。

必须输入某些特定于 JDBC 驱动程序和数据库供应商的数据。继续创建之前，请先收集以下信息：

- 数据库供应商名称
- 资源类型，如 `javax.sql.DataSource`（仅本地事务）和 `javax.sql.XADataSource`（全局事务）
- 数据源类名：如果 JDBC 驱动程序具有资源类型和数据库的数据源类，则“数据源类名”字段的值是必需的。
- 必需的属性，如数据库名称 (URL)、用户名和密码

连接池是用于特定数据库的一组可重复使用的连接。使用管理控制台创建连接池时，管理员实际上是在定义到特定数据库的连接的各个方面。

创建连接池之前，您必须首先安装并集成 JDBC 驱动程序。设置“创建连接池”页面时，必须输入特定于 JDBC 驱动程序和数据库供应商的特定数据。继续创建之前，请先收集以下信息：

- 数据库供应商名称
- 资源类型，如 `javax.sql.DataSource`（仅本地事务）和 `javax.sql.XADataSource`（全局事务）

- 数据源类名
- 必需的属性，如数据库名称 (URL)、用户名和密码

定义所安装的 JDBC 驱动程序指定的常规设置值。这些设置是 Java 编程语言中的类名或接口名称。

参数	说明
数据源类名	实现 `DataSource` 和/或 `XADataSource` API 的特定于供应商的类名。该类位于 JDBC 驱动程序中。
资源类型	选项包括 `javax.sql.DataSource`（仅本地事务）、`javax.sql.XADataSource`（全局事务）和 `java.sql.ConnectionPoolDataSource`（本地事务，性能可能会提高）。

此外，还必须定义一组位于池中的物理数据库连接。应用程序请求连接时，将从池中删除该连接；而应用程序释放该连接之后，连接将返回到池中。

参数	说明
初始和最小池大小	池中连接的最小数目。该值还确定了首次创建池或应用服务器启动时被置于池中的连接的数目。
最大池大小	池中连接的最大数目。
池大小调整数量	当池向最小池大小方向收缩时，将成批调整大小。此值确定批处理中的连接数目。如果将该值设置得过大，会延迟连接回收；如果将该值设置得过小，则会降低效率。
空闲超时	连接在池中保持空闲的最长时间（以秒为单位）。一旦超过此时间，即从池中删除该连接。
最长等待时间	在达到连接超时之前，请求连接的应用程序所等待的时间。由于默认等待时间过长，应用程序可能会出现无限期挂起的情况。

（可选）应用服务器可以在将连接传送给应用程序之前验证连接。如果由于网络出现故障或数据库服务器崩溃造成数据库不可用，此验证将允许应用服务器自动重新建立数据库连接。连接验证会带来额外开销，并会导致性能稍有下降。

参数	说明
连接验证	选中“需要”复选框以启用连接验证。

参数	说明
验证方法	应用服务器可以使用三种方法来验证数据库连接：自动提交、元数据和表。 自动提交和元数据－应用服务器通过调用 `con.getAutoCommit()` 和 `con.getMetaData()` 方法来验证连接。但是，由于许多 JDBC 驱动程序高速缓存了这些调用的结果，因此这两种方法无法始终提供可靠的验证。请与驱动程序供应商进行核实，以确定这些调用是否被高速缓存。 表－应用程序将查询指定的数据库表。表必须存在并且可以访问，但不要求表的行数。请不要使用包含许多行的现有表或经常访问的表。
表的名称	如果从“验证方法”组合框中选择了表，请在此指定数据库表的名称。
一旦失败	如果选中标有“关闭所有连接”的复选框，则单个连接失败时，应用服务器将关闭池中的所有连接，然后重新建立这些连接。如果未选中此复选框，则仅当要使用各个连接时才会重新建立连接。

由于许多用户通常可以并行访问一个数据库，因此可能出现一个事务在更新数据而另一个事务尝试读取同一数据的情况。事务的隔离级别定义了正在更新的数据对于其他事务的可见程度。有关隔离级别的详细资料，请参见数据库供应商的文档。

参数	说明
事务隔离	使您可以为该池的连接选择事务隔离级别。如果不指定此参数，连接将使用 JDBC 驱动程序提供的默认隔离级别进行操作。
保证隔离级别	该项仅在指定了隔离级别的情况下才适用。如果选中“保证”复选框，则从池中获取的所有连接都具有相同的隔离级别。例如，如果上次使用连接时通过编程方式（使用 `con.setTransactionIsolation`）更改了连接的隔离级别，此机制会将状态更改回指定的隔离级别。

JDBC 资源和连接池如何协同工作

为了存储、组织和检索数据，大多数应用程序都采用了关系数据库。Java EE 应用程序通过 JDBC API 访问关系数据库。应用程序必须获得一个连接之后才可以访问数据库。

以下是运行时应用程序连接到数据库时所发生的情况：

1. 应用程序通过 JNDI API 进行调用获取与数据库关联的 JDBC 资源（数据源）。

   给定资源的 JNDI 名称、命名和目录服务定位 JDBC 资源。每个 JDBC 资源指定一个连接池。
2. 通过 JDBC 资源，应用程序获得一个数据库连接。

   应用服务器秘密地从与该数据库相对应的连接池中检索物理连接。池定义了数据库名称 (URL)、用户名和密码等连接属性。

3. 由于已将应用程序连接到数据库，所以该应用程序可以读取和修改数据库中的数据以及将数据添加到数据库中。

   应用程序通过对 JDBC API 进行调用来访问数据库。JDBC 驱动程序将应用程序的 JDBC 调用转换为数据库服务器的协议。

4. 访问数据库完成之后，应用程序将关闭该连接。

   应用服务器将连接返回连接池。连接返回连接池之后，下一个应用程序就可以使用该连接。

设置数据库访问

要设置数据库访问，您必须首先安装支持的数据库产品。有关 Application Server 支持的数据库产品列表，请参见发行说明。然后，必须为数据库产品安装 JDBC 驱动程序，并使域的服务器实例能够访问该驱动程序的 JAR 文件。接下来，创建数据库、数据库的连接池以及指向连接池的 JDBC 资源。

安装 JDBC 驱动程序之后，必须对其进行集成，以便该驱动程序可将应用程序的 JDBC 调用转换为数据库服务器的协议。要将 JDBC 驱动程序集成到管理域中，请执行以下操作之一：

- 使通用类加载器可以访问该驱动程序。
  1. 将驱动程序的 JAR 文件和 ZIP 文件复制到 *domain-dir*`/lib` 目录或将其类文件复制到 *domain-dir*`/lib/ext` 目录。
  2. 重新启动该域。
- 使系统类加载器可以访问驱动程序。
  1. 在管理控制台中，针对所需的配置选择 JVM 设置。
  2. 标识驱动程序的 JAR 文件的全限定路径名。
  3. 保存设置并重新启动服务器。

持久性管理器资源

向后兼容需要此功能。要在 Application Server 7 版上运行，使用容器管理的持久性 Bean（一种 EJB 组件）的应用程序需要持久性管理器资源。建议改为使用 JDBC 资源。

要使用管理控制台创建持久性管理器资源，请选择“资源”节点。“持久性管理器”节点 > “持久性管理器”页面。要使用命令行实用程序，请使用 `create-persistence-resource` 命令。

◆ ◆ ◆ 第 4 章

配置 Java 消息服务资源

本章介绍了如何为使用 Java 消息服务 (Java Message Service, JMS) 的应用程序配置资源。它包含以下小节：

关于 JMS 资源

Application Server 中的 JMS 提供者

Application Server 通过将 Sun Java System Message Queue（以前的 SUN ONE Message Queue）软件集成到 Application Server 中，实现了 Java 消息服务 (Java Message Service, JMS) API。对于基本的 JMS API 管理任务，请使用 Application Server 管理控制台。对于高级任务（包括管理 Message Queue 群集），请使用 *MQ-install-dir*`/imq/bin` 目录中提供的工具。

有关管理 Messge Queue 的详细信息，请参见 Message Queue Administration Guide。

JMS 资源

Java 消息服务 (Java Message Service, JMS) API 使用两种被管理对象：

- 连接工厂，允许应用程序以编程方式创建其他 JMS 对象的对象。
- 目的地，充当消息的系统信息库

这些对象是以管理方式创建的，创建对象的方式特定于每个 JMS 实现。在 Application Server 中，执行以下任务：

- 通过创建连接工厂资源来创建连接工厂
- 通过创建两个对象来创建目的地：
  - 物理目的地
  - 引用物理目的地的目的地资源

JMS 应用程序使用 JNDI API 来访问连接工厂和目的地资源。通常，JMS 应用程序至少使用一个连接工厂和一个目的地。要了解所需创建的资源，请仔细研究应用程序或向应用程序开发者进行咨询。

连接工厂分为三种类型：

- `QueueConnectionFactory` 对象，用于点对点通信
- `TopicConnectionFactory` 对象，用于发布-订阅通信
- `ConnectionFactory` 对象，可用于点对点通信和发布-订阅通信；建议将这些对象用于新的应用程序

目的地有两种类型：

- `Queue` 对象，用于点对点通信
- `Topic` 对象，用于发布-订阅通信

《*The J2EE 1.4 Tutorial*》中有关 JMS 的章节提供了这两种通信和 JMS 其他方面的详细信息（请参见 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`）。

创建资源的顺序并不重要。

对于 J2EE 应用程序，请在 Application Server 部署描述符中指定连接工厂和目的地资源，如下所示：

- 在 `resource-ref` 或 `mdb-connection-factory` 元素中指定连接工厂 JNDI 名称。
- 在消息驱动的 Bean 的 `ejb` 元素和 `message-destination` 元素中指定目的地资源 JNDI 名称。
- 在企业 Bean 部署描述符的 `message-driven` 元素或 `message-destination-ref` 元素内，在 `message-destination-link` 元素中指定物理目的地名称。此外，还应在 `message-destination` 元素中指定该物理目的地名称。（`message-destination-ref`

元素替换了在新的应用程序中过时的 resource-env-ref 元素。）在 Application Server 部署描述符的 message-destination 元素中，将物理目的地名称与目的地资源名称链接起来。

JMS 资源与连接器资源之间的关系

Application Server 通过使用名为 jmsra 的系统资源适配器实现 JMS。用户创建 JMS 资源时，Application Server 会自动创建连接器资源，这些资源将显示在管理控制台树视图的“连接器”节点下。

对于用户创建的每个 JMS 连接工厂，Application Server 都将为其创建连接器连接池和连接器资源。对于用户创建的每个 JMS 目的地，Application Server 都将为其创建管理对象资源。用户删除 JMS 资源时，Application Server 将自动删除连接器资源。

可以通过使用管理控制台的“连接器”节点（而不是“JMS 资源”节点）来为 JMS 系统资源适配器创建连接器资源。有关详细信息，请参见第 7 章。

JMS 连接工厂

JMS 连接工厂是允许应用程序以编程方式创建其他 JMS 对象的对象。这些受管对象将实现 ConnectionFactory、QueueConnectionFactory 和 TopicConnectionFactory 接口。使用 Application Server 管理控制台，可以创建、编辑或删除 JMS 连接工厂。创建新的 JMS 连接工厂时还将为工厂创建连接器连接池并创建连接器资源。

要使用命令行实用程序管理 JMS 连接工厂，请使用 create-jms-resource、list-jms-resources 或 delete-jms-resource 命令。

JMS 目的地资源

JMS 目的地充当消息的系统信息库。使用管理控制台，可以创建、修改或删除 JMS 目的地资源。要创建新的 JMS 目的地资源，请选择“资源”>“JMS 资源”>“目的地资源”。在“目的地资源”页面中，可以指定以下各项：

- 资源的 JNDI 名称。建议的做法是对 JMS 资源使用命名子上下文前缀 jms/。例如：jms/Queue。
- 资源类型，可以为 javax.jms.Topic 或 javax.jms.Queue。
- 目的地资源的其他属性。有关所有这些设置和其他属性的更多详细信息，请参阅管理控制台联机帮助。

要使用命令行实用程序管理 JMS 目的地，请使用 create-jms-resource 或 delete-jms-resource 命令。

提示 – 要为 asadmin create-jms-resource 命令指定 addresslist 属性（格式为 host:mqport,host2:mqport,host3:mqport），请使用 \\ 转义 :。例如，host1\\:mqport,host2\\:mqport,host3\\:mpqport。

有关使用转义符的更多信息，请参见 asadmin(8) 手册页。

JMS 物理目的地

在生产阶段，始终需要创建物理目的地。但是，在开发和测试阶段，不需要执行此步骤。应用程序首次访问目的地资源时，Message Queue 会自动创建目的地资源的 Name 属性指定的物理目的地。该物理目的地是临时的，并且将在 Message Queue 配置属性指定的时间段后过期。

要在管理控制台中创建物理目的地，请选择“配置”>“物理目的地”。在“创建物理目的地”页面中，指定物理目的地的名称并选择目的地类型（可以为 topic 或 queue）。有关“物理目的地”页面中字段和属性的更多详细信息，请参阅管理控制台联机帮助。

要进行生产，务必创建物理目的地。但是，在开发和测试阶段，不需要执行此步骤。应用程序首次访问目的地资源时，Message Queue 会自动创建目的地资源的 Name 属性指定的物理目的地。该物理目的地是临时的，并且将在 Message Queue 配置属性指定的时间段后过期。

要使用命令行实用程序管理 JMS 物理目的地，请使用 create-jmsdest、flush-jmsdest 或 delete-jmsdest 命令。

JMS 提供者

配置 JMS 提供者的常规属性

使用管理控制台中的“JMS 服务”页面配置所有 JMS 连接都要使用的属性。在管理控制台中，选择“配置”>“Java 消息服务”。在“JMS 服务”页面中，可以控制以下常规 JMS 设置。

- 选择“启动超时”时间间隔，此间隔表示在异常中止启动之前 Application Server 等待 JMS 服务启动所用的时间。
- 选择 JMS 服务类型，它用来确定您要管理本地主机还是远程主机上的 JMS 服务。
- 指定“启动变量”以自定义 JMS 服务启动。
- 选中“重新连接”复选框以指定连接中断时，JMS 服务是否尝试重新连接至消息服务器（或 AddressList 中的地址列表）。

- 指定“重新连接时间间隔”（秒数）。此设置适用于对 `AddressList` 中每个地址的尝试及对该列表中连续地址的尝试。如果该时间间隔太短，则代理将没有时间恢复。如果该时间间隔太长，则重新连接可能会指明这是不可接受的延迟。
- 指定重新连接尝试次数。在此字段中，键入客户机运行时尝试连接（或重新连接）`AddressList` 列表中每个地址的次数。到达这个值后，客户机运行时将尝试连接列表中的下一个地址。
- 选择默认 JMS 主机。
- 在“地址列表行为”下拉式列表中，选择是按 `AddressList` 中的地址顺序 (`priority`) 还是按随机顺序 (`random`) 来尝试连接。
- 在“地址列表重复”字段中，键入为了建立（或重新建立）连接 JMS 服务在 `AddressList` 中重复的次数。
- 在“MQ 模式”和“MQ 服务”字段中，输入 Message Queue 地址模式名称和 Message Queue 连接服务名称（如果要使用非默认模式或服务）。

所有这些属性的值也可以在运行时更新。但是，只有那些在属性更新之后创建的连接工厂才会获取已更新的值。现有连接工厂将继续保持原始属性值。此外，为了使几乎所有值都生效，需要重新启动应用服务器。唯一无需重新启动应用服务器即可更新的属性是默认 JMS 主机。

要使用命令行实用程序管理 JMS 提供者，请使用 `set` 或 `jms-ping` 命令。

访问远程服务器

将提供者和主机更改到远程系统将使所有 JMS 应用程序在远程服务器上运行。要在使用本地服务器的同时使用一个或多个远程服务器，请使用 AddressList 属性创建连接工厂资源从而创建访问远程服务器的连接。

外部 JMS 提供者

JMS 通用资源适配器是 Java EE Connector 1.5 资源适配器，它可以包含外部 JMS 提供者（例如 IBM Websphere MQ、Tibco EMS 和 Sonic MQ 等）的 JMS 客户机库，从而将任何 JMS 提供者与 Java EE 1.4 应用服务器（例如 Sun Java System Application Server）进行集成。适配器是 `.rar` 归档文件，可以使用 Java EE 1.4 应用服务器的管理工具进行部署和配置。

配置 JMS 通用资源适配器

可以使用应用服务器的管理工具来部署和配置 JMS 通用资源适配器。本节介绍如何为 Sun Java System Application Server 配置 JMS 通用资源适配器。总体而言，可以配置资源适配器以指明 JMS 提供者是否支持 XA。还可以指明 JMS 提供者可以使用哪种集成模式。资源适配器支持两种集成模式。第一种模式使用 JNDI 作为集成方法。在这种情况

下，在 JMS 提供者的 JNDI 树下设置受管对象，并查找这些对象以供通用资源适配器使用。如果此模式不适于集成，还可以使用 JMS 受管对象 javabean 类的 Java 反射作为集成模式。您可以使用 Sun Java System Application Server 的管理控制台或 CLI 来配置此资源适配器。这与配置任何其他资源适配器相同。

配置通用资源适配器

在部署资源适配器之前，应使应用服务器能够访问 JMS 客户机库。对于某些 JMS 提供者，客户机库也可以包括本地库。在此类情况下，也应使应用服务器 JVM 能够访问这些本地库。

1. 按照与部署连接器模块相同的方式部署通用资源适配器。

   有关执行此操作的步骤，请参阅管理控制台联机帮助。在部署过程中，请确保将通用资源适配器的位置指定为 `install-dir/lib/addons/resourceadapters/genericjmsra/genericra.rar`。此外，还必须指定第 60 页中的 “资源适配器属性”部分中介绍的属性。

2. 创建连接器连接池。

   有关执行此操作的步骤，请参阅管理控制台联机帮助。在“新建连接器连接池”页面中，从“资源适配器”组合框中选择 `genericra`。此外，在“连接定义”组合框中选择 `javax.jms.QueueConnectionFactory`。还应指定第 63 页中的 “ManagedConnectionFactory 属性”部分中介绍的属性。

3. 创建连接器资源。

   有关执行此操作的详细过程，可以参阅管理控制台联机帮助。在“新建连接器资源”页面中，选择在上一步骤中创建的连接池。

4. 创建受管对象资源。

   有关执行此操作的详细过程，可以参阅管理控制台联机帮助。在“新建管理对象资源”页面中，从“资源适配器”中选择 `genericra`，从“资源类型”中选择 `javax.jms.Queue`。单击“下一步”，然后在下一个页面中单击“添加属性”。在“其他属性”表中，指定名为 `DestinationProperties` 的新属性，并将其值设置为 `Name\\=clientQueue`。有关更多属性的信息，请参见第 64 页中的 “受管对象资源属性”部分。

5. 在 Sun Java System Application Server 中，对安全性策略进行以下更改。

   - 修改 `sjsas_home/domains/domain1/config/server.policy`，向其中添加 `java.util.logging.LoggingPermission "control"`
   - 修改 `sjsas_home/lib/appclient/client.policy`，向其中添加 `permission javax.security.auth.PrivateCredentialPermission "javax.resource.spi.security.PasswordCredential * \"*\"","read";`

资源适配器属性

下表列出了创建资源适配器时要使用的属性。

属性名称	有效值	默认值	说明
ProviderIntegrationMode	javabean/jndi	javabean	确定资源适配器与 JMS 客户机之间的集成模式。
ConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.ConnectionFactory	无	JMS 客户机的 javax.jms.ConnectionFactory 实现的类名。在 ProviderIntegrationMode 为 javabean 时使用。
QueueConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.QueueConnectionFactory	无	JMS 客户机的 javax.jms.QueueConnectionFactory 实现的类名。在 ProviderIntegrationMode 为 javabean 时使用此属性。
TopicConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.TopicConnectionFactory	无	JMS 客户机的 javax.jms.TopicConnectionFactory 实现的类名称。在将 ProviderIntegrationMode 指定为 javabean 时使用此属性。
XAConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.XAConnectionFactory	无	JMS 客户机的 javax.jms.ConnectionFactory 实现的类名。在 ProviderIntegrationMode 指定为 javabean 时使用此属性。
XAQueueConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.XAQueueConnectionFactory	无	JMS 客户机的 javax.jms.XAQueueConnectionFactory 实现的类名。在 ProviderIntegrationMode 指定为 javabean 时使用此属性。
XATopicConnectionFactoryClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.XATopicConnectionFactory	无	JMS 客户机的 javax.jms.XATopicConnectionFactory 实现的类名。在 ProviderIntegrationMode 为 javabean 时使用此属性。
TopicClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.Topic	无	JMS 客户机的 javax.jms.Topic 实现的类名。在 ProviderIntegrationMode 为 javabean 时使用此属性。
QueueClassName	可用于应用服务器类路径中的类的名称，例如： com.sun.messaging.Queue	无	JMS 客户机的 javax.jms.Queue 实现的类名。在将 ProviderIntegrationMode 指定为 javabean 时使用此属性。
SupportsXA	True/false	FALSE	指定 JMS 客户机是否支持 XA。

属性名称	有效值	默认值	说明
ConnectionFactoryProperties	以逗号分隔的名称值对。	无	此属性指定 JMS 客户机的 javabean 属性名称以及 ConnectionFactory 的值。仅当 ProviderIntegrationMode 为 javabean 时才需要此属性。
JndiProperties	以逗号分隔的名称值对。	无	此属性指定连接到 JMS 提供者的 JNDI 时使用的 JNDI 提供者属性。仅当 ProviderIntegrationMode 为 jndi 时才使用此属性。
CommonSetterMethodName	方法名	无	此属性指定某些 JMS 供应商在设置其受管对象属性时使用的常见设置方法 (setter method) 名。仅当 ProviderIntegrationMode 为 javabean 时才使用此属性。在 Sun Java System Message Queue 中，此属性名为 setProperty。
UserName	JMS 用户的名称	无	连接到 JMS 提供者时使用的用户名。
Password	JMS 用户的密码。	无	连接到 JMS 提供者时使用的密码。

属性名称	有效值	默认值	说明
RMPolicy	ProviderManaged 或 OnePerPhysicalConnection	ProviderManaged	事务管理器使用 XAResource 的 isSameRM 方法来确定两个 XAResource 所表示的资源管理器实例是否相同。 将 RMPolicy 设置为 ProviderManaged（默认值）时，JMS 提供者将负责确定通用资源适配器中的 RMPolicy 和 XAResource 包装器仅将 isSameRM 调用委托给消息队列提供者的 XA 资源实现。这应该适用于大多数消息队列产品。 某些 XAResource 实现（例如 IBM MQ Series）在每个物理连接中依赖于一个资源管理器。如果在单个事务中与同一队列管理器同时进行入站和出站通信（例如，当 MDB 向目的地发送响应时），这会导致出现问题。 将 RMPolicy 设置为 OnePerPhysicalConnection 时，通用资源适配器中 XAResource 包装器实现的 isSameRM 将检查两个 XAResource 是否使用同一物理连接，然后再委托给被包装的对象。有关此属性的其他信息，请参阅 Glassfish Web 站点上 Issue Tracker 数据库中的问题 5。

ManagedConnectionFactory 属性

创建连接器连接池时，将指定 ManagedConnectionFactory 属性。可以在 ManagedConnectionFactory 中覆盖创建资源适配器时指定的所有属性。下面给出了仅可用于 ManagedConnectionFactory 中的其他属性。

属性名称	有效值	默认值	说明
ClientId	有效的客户机 ID	无	由 JMS 1.1 规范指定的 ClientID。

属性名称	有效值	默认值	说明
ConnectionFactoryJndiName	JNDI 名称	无	JMS 提供者的 JNDI 树中绑定的连接工厂的 JNDI 名称。管理员应在 JMS 提供者本身中提供所有连接工厂属性（clientID 除外）。仅当 ProviderIntegratinMode 为 jndi 时才使用此属性名称。
ConnectionValidationEnabled	true/false	FALSE	如果设置为 true，资源适配器将使用异常侦听器捕捉任何连接异常，并向应用服务器发送 CONNECTION_ERROR_OCCURED 事件。

受管对象资源属性

创建受管对象资源时，将指定本节中的属性。可以在受管资源对象中覆盖所有资源适配器属性。下面给出了仅可用于受管对象资源中的其他属性。

属性名称	有效值	默认值	说明
DestinationJndiName	JNDI 名称	无	JMS 提供者的 JNDI 树中绑定的目的地的 JNDI 名称。管理员应在 JMS 提供者本身中提供所有属性。仅当 ProviderIntegrationMode 为 jndi 时才使用此属性名称。
DestinationProperties	以逗号分隔的名称值对	无	此属性指定 JMS 客户机的 javabean 属性名称以及目的地值。仅当 ProviderIntegrationMode 为 javabean 时才需要此属性。

激活规范属性

可以在 Sun 的特定 MDB 部署描述符中将本节中的属性指定为激活配置属性。可以在激活规范中覆盖所有资源适配器属性。下面给出了仅可用于激活规范中的其他属性。

属性名称	有效值	默认值	说明
MaxPoolSize	整数	8	资源适配器为了实现并发消息传送而在内部创建的服务器会话池的最大大小。此大小应该等于 MDB 对象的最大池大小。
MaxWaitTime	整数	3	资源适配器将等待此属性所指定的时间（以秒为单位）后，才能从其内部池中获得服务器会话。如果超过此限制，消息传送将失败。
SubscriptionDurability	Durable 或 Non-Durable	Non-Durable	由 JMS 1.1 规范指定的 SubscriptionDurability。
SubscriptionName		无	由 JMS 1.1 规范指定的 SubscriptionName。
MessageSelector	有效的消息选择器	无	由 JMS 1.1 规范指定的 MessageSelector。
ClientID	有效的客户机 ID	无	ClientID 由 JMS 1.1 规范指定。
ConnectionFactoryJndiName	有效的 JNDI 名称	无	在 JMS 提供者中创建的连接工厂的 JNDI 名称。资源适配器将使用此连接工厂创建连接以接收消息。仅当 ProviderIntegrationMode 配置为 jndi 时才使用。
DestinationJndiName	有效的 JNDI 名称	无	在 JMS 提供者中创建的目的地的 JNDI 名称。资源适配器将使用此目的地创建连接以接收消息。仅当 ProviderIntegrationMode 配置为 jndi 时才使用。
DestinationType	javax.jms.Queue 或 javax.jms.Topic	null	MDB 将侦听的目的地的类型。
DestinationProperties	以逗号分隔的名称-值对	无	此属性指定 JMS 客户机的 javabean 属性名称以及目的地值。仅当 ProviderIntegrationMode 为 javabean 时才需要。
RedeliveryAttempts	整数		当消息在 MDB 中导致运行时异常时，传送消息的次数。
RedeliveryInterval	时间（以秒为单位）		当消息在 MDB 中导致运行时异常时，重复传送之间的时间间隔。

属性名称	有效值	默认值	说明
SendBadMessagesToDMD	true/false	false	指示当超过传送尝试次数时，资源适配器是否应将消息发送到停用消息目的地。
DeadMessageDestinationJndiName	有效的 JNDI 名称。	无	在 JMS 提供者中创建的目的地的 JNDI 名称。这是停用消息的目的地。仅当 ProviderIntegrationMode 为 jndi 时才使用此属性。
DeadMessageDestinationClassName	目的地对象的类名。	无	仅当 ProviderIntegrationMode 为 javabean 时才使用。
DeadMessageDestinationProperties	以逗号分隔的名称值对	无	此属性指定 JMS 客户机的 javabean 属性名称以及目的地值。仅当 ProviderIntegrationMode 为 javabean 时才需要此属性。
ReconnectAttempts	整数		当异常侦听器捕捉到连接错误时进行重新连接的尝试次数。
ReconnectInterval	时间（以秒为单位）		重新连接之间的时间间隔。

◆ ◆ ◆ 第 5 章

5 配置 JavaMail 资源

Application Server 包含 JavaMail API。JavaMail API 是一组用于建立邮件系统模型的抽象 API。API 提供了一个与平台无关以及与协议无关的框架来建立邮件应用程序和消息传送应用程序。JavaMail API 提供了用于读取、撰写和发送电子邮件的工具。服务提供商可实现特定协议。使用 JavaMail API，您可以向应用程序中添加电子邮件功能。通过 JavaMail，可以从 Java 应用程序访问网络或 Internet 上支持 Internet 消息访问协议 (Internet Message Access Protocol, IMAP) 和简单邮件传输协议 (Simple Mail Transfer Protocol, SMTP) 的邮件服务器。它不提供邮件服务器功能；您必须有权访问邮件服务器，才能使用 JavaMail。

要了解有关 JavaMail API 的更多信息，请查阅位于以下位置的 JavaMail Web 站点：`http://java.sun.com/products/javamail/index.html`。

本章包括以下部分：

- 第 67 页中的 “创建 JavaMail 会话”

创建 JavaMail 会话

要配置 JavaMail 以便用于 Application Server 中，请在 Application Server 管理控制台中创建邮件会话。这样，服务器端组件和应用程序就可以使用您为它们指定的会话属性通过 JNDI 访问 JavaMail 服务。创建邮件会话时，您可以在管理控制台中指定邮件主机、传输和存储协议以及默认邮件用户，这样，使用 JavaMail 的组件就不必设置这些属性。具有大量电子邮件用户的应用程序会从中受益，因为 Application Server 将创建一个会话对象，并使任何需要该对象的组件均可通过 JNDI 使用该对象。

要使用管理控制台创建 JavaMail 会话，请选择“资源”>“JavaMail 会话”。指定 JavaMail 设置，如下所示：

- JNDI 名称：邮件会话的唯一名称。请针对 JavaMail 资源使用命名子上下文前缀 `mail/`。例如：`mail/MySession`。
- 邮件主机：默认邮件服务器的主机名。如果未提供特定协议的主机属性，Store 和 Transport 对象的连接方法使用该值。名称必须可以解析为实际的主机名。

- 默认用户：连接到邮件服务器时要提供的用户名。如果未提供特定协议的用户名属性，Store 和 Transport 对象的连接方法使用该值。
- 默认返回地址：默认用户的电子邮件地址，格式为：*username*@*host.domain*。
- 说明：提供组件的描述性语句。
- 会话：如果您不希望此时启用邮件会话，请取消选中“已启用”复选框。

此外，仅在已将邮件提供商重新配置为使用非默认存储或传输协议时，才定义以下高级设置：

- 存储协议：定义要使用的存储对象通信方法。默认情况下，存储协议为 `imap`。
- 存储协议类：提供实现所需存储协议的存储通信方法类。默认情况下，存储协议类为 `com.sun.mail.imap.IMAPStore`。
- 传输协议：标识传输通信方法。默认情况下，传输协议为 `smtp`。
- 传输协议类：定义用于传输类的通信方法。默认情况下，传输协议类为 `com.sun.mail.smtp.SMTPTransport`。
- 调试：选中此复选框以启用附加调试输出（包含此邮件会话的协议跟踪）。如果将 JavaMail 日志级别设置为 `FINE` 或 FINER，将生成调试输出，并且此输出将包含在系统日志文件中。有关设置日志级别的信息，请参见第 138 页中的 “设置自定义日志级别”。
- 其他属性：创建应用程序所需的属性，例如特定于协议的主机或用户名属性。JavaMail API 文档列出了可用属性。

◆ ◆ ◆ 第 6 章

6 JNDI 资源

Java 命名和目录接口 (Java Naming and Directory Interface, JNDI) 是一种应用编程接口 (application programming interface, API)，用于访问不同类型的命名和目录服务。Java EE 组件通过调用 JNDI 查找方法来定位对象。

JNDI 是 Java 命名和 API 目录接口的首字母缩略词。通过对此 API 进行调用，应用程序可以定位资源和其他程序对象。资源是提供到系统（如数据库服务器和消息传送系统）的连接的程序对象。（JDBC 资源有时被称为数据源。）每个资源对象都是由唯一的友好名称所标识，称为 JNDI 名称。Application Server 附带的命名和目录服务将资源对象及其 JNDI 名称绑定在一起。要创建新资源，需要将新的名称-对象绑定输入到 JNDI 中。

本章包括以下几个部分：

J2EE 命名服务

JNDI 名称是便于用户使用的对象名称。这些名称通过 J2EE 服务器提供的命名和目录服务绑定到其对象。由于 J2EE 组件通过 JNDI API 访问此服务，因此对象通常使用其 JNDI 名称。例如，PointBase 数据库的 JNDI 名称为 `jdbc/Pointbase`。Application Server 启动时，将从配置文件中读取信息，并自动将 JNDI 数据库名称添加到名称空间。

J2EE 应用程序客户机端、企业 Bean 以及 Web 组件都需要具有权限，才能访问 JNDI 命名环境。

应用程序组件的命名环境是一种机制，使用它可以在部署或汇编期间自定义应用程序组件的商业逻辑。使用应用程序组件的环境即可对应用程序组件进行自定义，而无需访问或更改应用程序组件的源代码。

J2EE 容器实现 J2EE 应用程序组件的环境，并将该环境作为 JNDI 命名上下文提供给 J2EE 应用程序组件实例。J2EE 应用程序组件的环境的使用方式如下：

- 应用程序组件的商业方法使用 JNDI 接口访问该环境。应用程序组件提供商在部署描述符中声明应用程序组件需要其运行时环境提供的所有环境项。

- 容器实现存储应用程序组件环境的 JNDI 命名上下文。容器还提供了部署者可以用于创建和管理每个应用程序组件的环境的工具。
- 部署者使用容器提供的工具，可以初始化应用程序组件的部署描述符中声明的环境项。部署者可以设置和修改环境条目的值。
- 容器使环境命名上下文在运行时可用于应用程序组件实例。应用程序组件的实例使用 JNDI 接口获取环境项的值。

每个应用程序组件定义了其本身的环境项集合。一个应用程序组件在同一容器内的所有实例共享相同的环境项。不允许应用程序组件实例在运行时修改环境。

命名引用和绑定信息

资源引用是部署描述符中的一种元素，用于标识该资源的组件的编码名称。更具体地说，编码名称引用资源的连接工厂。在下节给出的示例中，资源引用名称为 `jdbc/SavingsAccountDB`。

资源的 JNDI 名称和资源引用名称是不同的。使用此命名方法，您需要在进行部署之前先映射这两个名称，但此方法也用于将组件与资源分离开。由于具有此分离功能，因此如果组件在以后需要访问其他资源，则无需更改名称。这一灵活性使您可以更加容易地从先前存在的组件汇编 J2EE 应用程序。

下表列出了用于 Application Server 所使用的 J2EE 资源的 JNDI 查找及其关联的引用。

表 6–1 JNDI 查找及其关联的引用

JNDI 查找名称	关联的引用
`java:comp/env`	应用程序环境项
`java:comp/env/jdbc`	JDBC 数据源资源管理器连接工厂
`java:comp/env/ejb`	EJB 引用
`java:comp/UserTransaction`	UserTransaction 引用
`java:comp/env/mail`	JavaMail 会话连接工厂
`java:comp/env/url`	URL 连接工厂
`java:comp/env/jms`	JMS 连接工厂和目的地
`java:comp/ORB`	应用程序组件之间共享的 ORB 实例

使用自定义资源

自定义资源访问本地 JNDI 系统信息库，外部资源访问外部 JNDI 系统信息库。这两种类型的资源都需要用户指定的工厂类元素、JNDI 名称属性等。在本节中，我们将讨论如何为 J2EE 资源配置 JNDI 连接工厂资源，以及如何访问这些资源。

在 Application Server 中，您可以创建、删除和列出资源以及 `list-jndi-entities`。

使用外部 JNDI 系统信息库和资源

通常，在 Application Server 上运行的应用程序需要访问存储在外部 JNDI 系统信息库中的资源。例如，一般的 Java 对象可能会以 Java 模式存储在 LDAP 服务器中。外部 JNDI 资源元素允许用户配置此类外部资源系统信息库。外部 JNDI 工厂必须实现 `javax.naming.spi.InitialContextFactory` 接口。

使用外部 JNDI 资源的示例：

```
<resources>
 <!-- external-jndi-resource element specifies how to access J2EE resources
 -- stored in an external JNDI repository. The following example
 -- illustrates how to access a java object stored in LDAP.
 -- factory-class element specifies the JNDI InitialContext factory that
 -- needs to be used to access the resource factory. property element
 -- corresponds to the environment applicable to the external JNDI context
 -- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
 -- designated (in this case the java) object.
 -->
  <external-jndi-resource jndi-name="test/myBean"
      jndi-lookup-name="cn=myBean"
      res-type="test.myBean"
      factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
</resources>
```

◆ ◆ ◆ 第 7 章

7 连接器资源

连接器模块也称为资源适配器，是一个允许应用程序与企业信息系统 (enterprise information system, EIS) 进行交互式操作的 Java 组件。EIS 软件包含各种类型的系统：包括企业资源规划 (enterprise resource planning, ERP)、主机事务处理和非关系数据库。要安装连接器模块，请按照部署其他 Java 模块的方式对其进行部署。

连接器连接池是一组用于特定 EIS 的可重复使用的连接。要创建连接器连接池，请指定与池关联的连接器模块（资源适配器）。

连接器资源是为应用程序提供到 EIS 的连接的程序对象。要创建连接器资源，请指定其 JNDI 名称及其关联的连接池。多个连接器资源可以指定一个连接池。应用程序可通过查找资源的 JNDI 名称定位资源。EIS 的连接器资源的 JNDI 名称通常位于 `java:comp/env/`*eis-specific* 子上下文中。Application Server 9 使用连接器模块（资源适配器）实现 JMS。

本章包括以下内容：

连接器连接池

下表介绍了连接池设置：

参数	说明
初始和最小池大小	池中连接的最小数目。该值还确定了首次创建池或应用服务器启动时被置于池中的连接的数目。
最大池大小	池中连接的最大数目。

参数	说明
池大小调整数量	当池向最小池大小方向收缩时，将成批调整大小。此值确定批处理中的连接数目。将该值设置过大会延迟连接回收；而将该值设置过小则会导致效率太低。
空闲超时	连接在池中保持空闲的最长时间（以秒为单位）。一旦超过此时间，即从池中删除该连接。
最长等待时间	已请求连接的应用程序在达到连接超时之前等待的时间。由于默认等待时间过长，应用程序可能会出现无限期挂起的情况。
一旦失败	选中标记为“关闭所有连接”的复选框之后，如果单个连接失败，应用服务器将关闭池中的所有连接，然后重新建立这些连接。如果未选中此复选框，则只有在使用各个连接时才会重新建立这些连接。
事务支持	使用“事务支持”列表可以为连接池选择事务支持类型。选择的事务支持将以向下兼容方式覆盖与此连接池关联的资源适配器中的事务支持属性。也就是说，它可以支持比资源适配器中指定的事务级别低或与其相同的事务级别，但它不能指定更高的级别。 “事务支持”菜单中的“无”选项表示资源适配器不支持资源管理器本地事务或 JTA 事务，也不实现 `XAResource` 或 `LocalTransaction` 接口。对于 JAXR 资源适配器，您需要从“事务支持”菜单中选择“无”。JAXR 资源适配器不支持本地事务或 JTA 事务。 “本地”事务支持表示资源适配器将通过实现 `LocalTransaction` 接口来支持本地事务。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。 XA 事务支持表示资源适配器将通过实现 `LocalTransaction` 和 `XAResource` 接口来支持资源管理器本地事务和 JTA 事务。XA 事务由事务管理器在资源管理器外部进行控制和调整。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。
连接器验证	如果希望在将连接池传递给应用程序之前对其进行验证，请选中“已启用”复选框。

创建连接池之前，您需要部署与池关联的连接器模块（资源适配器）。您可以使用管理控制台或使用 `asadmin` 命令来部署连接器模块。有关 `asadmin` 命令的信息，请参见 `asadmin`(1M)。

要在管理控制台中查看、创建、编辑或删除连接池，请单击“资源”>“连接器”>“连接器连接池”。您可以向连接器连接池中添加属性（名称-值对）。或者，可以使用以下 `asadmin` 命令创建和删除连接池：

- `create-connector-connection-pool`(1)
- `delete-connector-connection-pool`(1)

连接器资源

连接器资源为应用程序提供到企业信息系统 (Enterprise Information System, EIS) 的连接。每个连接器资源都与一个连接池相关联。要查看、创建、编辑或删除连接器资源，请在管理控制台中单击“资源”>“连接器”>“连接器资源”。或者，可以使用以下 `asadmin` 命令创建和删除连接资源：

- `create-connector-resource`(1)
- `delete-connector-resource`(1)

受管对象资源

受管对象为应用程序提供专用功能，例如访问特定于资源适配器及其相关 EIS 的解析器。要查看、创建、编辑或删除受管对象，请在管理控制台中单击“资源”>“连接器”>“管理对象资源”。

- `create-admin-object`(1)
- `delete-admin-object-1`(1)

◆ ◆ ◆ 第 8 章

J2EE 容器

本章介绍了如何配置服务器中包含的 J2EE 容器。本章包含以下几节：

J2EE 容器的类型

J2EE 容器为 J2EE 应用程序组件提供运行时支持。J2EE 应用程序组件使用容器的协议和方法访问服务器提供的其他应用程序组件和服务。Application Server 提供了应用程序客户机端容器、applet 容器、Web 容器和 EJB 容器。有关显示容器的示意图，请参见第 26 页中的 “Application Server 体系结构”部分。

Web 容器

Web 容器是用于托管 Web 应用程序的 J2EE 容器。Web 容器通过为开发者提供运行 Servlet 和 JavaServer Pages（JSP 文件）的环境来扩展 Web 服务器的功能。

EJB 容器

企业 Bean（EJB 组件）是包含商业逻辑的 Java 编程语言服务器组件。EJB 容器提供对企业 Bean 的本地访问和远程访问。

企业 Bean 分为三种类型：会话 Bean、实体 Bean 和消息驱动 Bean。会话 Bean 表示瞬态对象和进程，并且通常由单个客户机使用。实体 Bean 表示持久性数据，通常保留在数据库中。消息驱动 Bean 用于将消息异步传送到应用程序模块和服务中。

容器负责创建企业 Bean、将企业 Bean 绑定到命名服务以使其他应用程序组件可以访问企业 Bean、确保仅授权的客户机有权访问企业 Bean 的方法、将 Bean 的状态保存到持久性存储、高速缓存 Bean 的状态，以及在必要时激活或钝化 Bean。

配置 J2EE 容器

配置常规 Web 容器设置

在本版本中，管理控制台中没有用于 Web 容器的容器范围的设置。

配置 Web 容器会话

本节介绍了 Web 容器中的 HTTP 会话设置。HTTP 会话是唯一将状态数据写入持久性存储的 Web 会话。

配置会话超时值

可以使用管理控制台设置 HTTP 会话超时值。会话超时值表示 HTTP 会话有效的持续时间。

在管理控制台中，转至“配置”>“Web 容器”>“会话属性”。在“会话超时”字段中，输入会话有效的秒数。

有关设置会话超时值的详细说明，请在管理控制台中单击“帮助”。

配置管理器属性

会话管理器使您可以配置如何创建和销毁会话、会话状态存储位置说以及最大会话数目。

要在管理控制台中更改会话管理器设置，请转至“配置”>“Web 容器”>“管理器属性”。

- 选择要配置的实例：

  要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。

  要配置所有实例的默认设置，请选择 `default-config` 节点。

在“管理器属性”选项卡中，设置以下属性：

- “Reap 时间间隔”的值。“Reap 时间间隔”字段是从存储中删除非活动会话数据之前的秒数。
- “最大会话数”的值。“最大会话数”字段是允许的最大会话数目。
- 设置“会话文件名”的值。“会话文件名”字段是包含会话数据的文件。
- “会话 ID 生成器类名”的值。

  “会话 ID 生成器类名”字段使您可以指定用于生成唯一的会话 ID 的自定义类。每个服务器实例只允许有一个会话 ID 生成器类，并且群集中的所有实例必须使用同一会话 ID 生成器，以防止会话密钥冲突。

  自定义会话 ID 生成器类必须实现 `com.sun.enterprise.util.uuid.UuidGenerator` 接口：

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj);  //obj is the session object
}
```

  类必须位于 Application Server 类路径中。

有关设置管理器属性的详细说明，请在管理控制台中单击“帮助”。

配置存储属性

要指定会话存储数据的保存位置，请在管理控制台中，转至“配置”>“Web 容器”>“存储属性”。

- 选择要配置的实例：

  要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。

  要配置所有实例的默认设置，请选择 `default-config` 节点。
- 设置“Reap 时间间隔”字段

  “Reap 时间间隔”字段是从存储中删除非活动会话数据之前的秒数。
- 指定会话数据的存储目录。

有关设置会话存储属性的详细说明，请在管理控制台中单击“帮助”。

配置虚拟服务器设置

安装 Application Server 时，将为 Application Server 实例创建一个默认虚拟服务器。此虚拟服务器的默认 docroot 在 *instance-dir*domains/domain1/docroot 中创建，此目录将同步到 *instance_name*/docroot。系统将为创建的每个附加 Application Server 实例创建一个虚拟服务器。

配置常规 EJB 设置

本节介绍了以下适用于服务器上所有企业 Bean 容器的设置：

- 第 80 页中的 “会话存储位置”
- 第 80 页中的 “编辑 EJB 池设置”
- 第 81 页中的 “配置 EJB 高速缓存设置”

要覆盖每个容器的默认值，请调整企业 Bean 的 sun-ejb-jar.xml 文件中的值。有关详细信息，请参见 Application Server Developer's Guide。

会话存储位置

“会话存储位置”字段指定在文件系统上存储钝化 Bean 和持久的 HTTP 会话所在的目录。

钝化 Bean 是已将其状态写入到文件系统上的文件中的企业 Bean。通常，钝化的 Bean 已空闲一段时间，并且当前未被客户机访问。

与钝化 Bean 类似，持久的 HTTP 会话是已将其状态写入到文件系统上的文件中的各个 Web 会话。

“提交选项”字段用于指定容器如何高速缓存事务之间的钝化实体 Bean 实例。

“选项 B”用于高速缓存事务之间的实体 Bean 实例，并且是默认选项。“选项 C”用于禁用高速缓存。

编辑 EJB 池设置

容器维护了一个企业 Bean 池，以便在不创建 Bean 来实现性能的情况下响应客户机请求。这些设置仅适用于无状态会话 Bean 和实体 Bean。

如果在使用已部署企业 Bean 的应用程序中遇到性能问题，创建池或增加现有池维护的 Bean 的数目有助于提高应用程序的性能。

默认情况下，容器维护企业 Bean 池。

配置 EJB 高速缓存设置

容器为大多数使用过的企业维护了企业数据高速缓存。这将允许容器更迅速地响应其他应用程序模块对企业 Bean 的数据请求。本节只适用于有状态会话 Bean 和实体 Bean。

被高速缓存的企业处于以下三种状态之一：活动、空闲或钝化。活动企业 Bean 是当前正被客户机访问的企业 Bean。空闲企业 Bean 的数据当前保存在高速缓存中，但没有客户机访问 Bean。钝化 Bean 的数据是被临时存储的，如果客户机请求此 Bean，其数据将被读回高速缓存中。

配置消息驱动 Bean 设置

消息驱动 Bean 的池与第 80 页中的 “编辑 EJB 池设置”中介绍的会话 Bean 的池类似。默认情况下，容器维护消息驱动 Bean 的池。

要调整该池的配置，请执行以下步骤：

配置 EJB 计时器服务设置

计时器服务是由企业 Bean 容器提供的用于安排企业 Bean 使用的通知或事件的持久性和事务性通知服务。所有企业 Bean（有状态会话 Bean 除外）均可从计时器服务接收通知。关闭或重新启动服务器时，服务设置的计时器不会被销毁。

◆ ◆ ◆ 第 9 章

9 配置安全性

安全性是有关数据保护的功能：在存储和传输数据时如何防止对数据进行未经授权的访问或破坏。Application Server 具有基于 J2EE 标准的动态可扩展安全体系结构。内置了多种安全功能，包括密码学、验证和授权以及公钥基本结构。Application Server 是基于 Java 安全模型构建的，该安全模型使用沙箱，应用程序可以在沙箱中安全地运行，而不会给系统或用户带来潜在的危险。本节介绍了以下主题：

了解应用程序和系统安全性

从宽泛意义上讲，应用程序安全性有两种：

- 在**程序安全性**中，开发者编写的应用程序代码负责处理安全性事务。作为管理员，您对这一机制没有任何控制权。由于程序安全性将安全配置硬编码到应用程序中而不是通过 J2EE 容器对其进行管理，因此这种程序安全性的功能常常受到限制。
- 在**声明性安全**中，容器 (Application Server) 通过应用程序的部署描述符处理安全性事务。您可以通过直接编辑部署描述符或使用 `deploytool` 等工具来控制声明安全性。由于可以在完成应用程序开发之后更改部署描述符，因此声明安全性具有更大的灵活性。

除了应用程序安全性以外，还有影响 Application Server 系统中所有应用程序的**系统安全性**。

程序安全性受应用程序开发者的控制，因此本文档不对其进行讨论；声明安全性受应用程序开发者的控制要少一些，本文档中只偶尔涉及到声明安全性。本文档主要针对系统管理员，因此主要讲述了系统安全性。

管理安全性的工具

Application Server 提供了以下用于管理安全性的工具：

- 管理控制台，它是一种基于浏览器的工具，用于配置整个服务器的安全性，管理用户、组和领域以及执行系统范围内的其他安全性任务。有关管理控制台的一般介绍，请参见第 28 页中的 “管理工具”。有关可以使用管理控制台执行的安全性任务的概述，请参见第 95 页中的 “使用管理控制台管理安全性”。
- `asadmin`，一个命令行工具，可以执行管理控制台能够执行的许多任务。您还可以使用 `asadmin` 执行某些使用管理控制台无法执行的任务。您可以从命令提示符或脚本执行 `asadmin` 命令，以自动执行重复任务。有关 `asadmin` 的一般介绍，请参见第 28 页中的 “管理工具”。
- `deploytool`，一个图形化封装和部署工具，用于编辑应用程序部署描述符，从而控制各个应用程序的安全性。由于 `deploytool` 主要针对应用程序开发者，因此本文档未详细介绍如何使用该工具。有关使用 `deploytool` 的说明，请参见该工具的联机帮助以及位于 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html` 的《The J2EE 1.4 Tutorial》。

Java 2 Platform, Standard Edition (J2SE) 提供了两个用于管理安全性的工具：

- `keytool`，一个命令行实用程序，用于管理数字证书和密钥对。使用 `keytool` 可以管理 `certificate` 领域内的用户。
- `policytool`，一个图形实用程序，用于管理系统范围内的 Java 安全策略。作为管理员，您很少会用到 `policytool`。

有关使用 `keytool`、`policytool` 和其他 Java 安全性工具的更多信息，请参见 `http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security` 上的 Java 2 SDK Tools and Utilities。

在 Enterprise Edition 中，还可以使用两个实现网络安全服务 (NSS) 的工具来管理安全性。有关 NSS 的更多信息，请访问 `http://www.mozilla.org/projects/security/pki/nss/`。管理安全性的工具包括：

- `certutil`，一个命令行实用程序，用于管理证书和密钥数据库。
- `pk12util`，一个命令行实用程序，用于以 PKCS12 格式在证书/密钥数据库和文件之间导入和导出密钥及证书。

有关使用 `certutil`、`pk12util` 和其他 NSS 安全性工具的更多信息，请参见 `http://www.mozilla.org/projects/security/pki/nss/tools` 上的 NSS Security Tools。

管理密码安全性

在此版本的 Application Server 中，包含特定域的规范的 domain.xml 文件最初以明文形式包含了 Sun Java System Message Queue 代理的密码。domain.xml 文件中包含此密码的元素为 jms-host 元素的 admin-password 属性。由于在安装期间不能更改此密码，因此它不会对安全性产生很大的影响。

不过，您可以使用管理控制台添加用户和资源，并为这些用户和资源指定密码。部分密码将以明文形式写入 domain.xml 文件，例如用于访问数据库的密码。将这些密码以明文形式保存在 domain.xml 文件中可能会使安全性受到威胁。您可以对 domain.xml 中的任何密码（包括 admin-password 属性或数据库密码）进行加密。有关管理安全性密码的说明，请参见以下主题：

- 第 85 页中的 “在 domain.xml 文件中加密密码”
- 第 86 页中的 “保护具有编码密码的文件”
- 第 86 页中的 “更改主密码”
- 第 86 页中的 “使用主密码和密钥库”
- 第 87 页中的 “更改管理密码”

在 domain.xml 文件中加密密码

要在 domain.xml 文件中加密密码，请执行以下步骤：

1. 在 domain.xml 文件所在的目录（默认情况下，此目录为 *domain-dir*/config）中，运行以下 asadmin 命令：

   ```
   asadmin create-password-alias --user admin alias-name
   ```

   例如，

   ```
   asadmin create-password-alias --user admin jms-password
   ```

   将显示输入密码提示（在本例中为 admin）。有关更多信息，请参阅 create-password-alias、list-password-aliases 和 delete-password-alias 命令的手册页。

2. 删除并替换 domain.xml 中的密码。使用 asadmin set 命令可以完成此操作。用于此目的的 set 命令的示例如下：

   ```
   asadmin set --user admin server.jms-service.jms-host.
   default_JMS_host.admin-password='${ALIAS=jms-password}'
   ```

   注 – 将别名密码括在单引号中，如本例中所示。

3. 重新启动相关域的 Application Server。

保护具有编码密码的文件

某些文件包含需要使用文件系统权限进行保护的编码密码。这些文件包括：

- *domain-dir*/master-password

  此文件包含编码主密码，并且应使用文件系统权限 600 对其进行保护。
- 任何使用 --passwordfile 参数创建的、作为参数传递给 asadmin 的密码文件均应使用文件系统权限 600 进行保护。

更改主密码

主密码 (MP) 是全局性的共享密码。它从不用于验证，也从不会在网络上传输。此密码是整体安全性的要塞点；用户可以选择在需要时手动输入此密码，也可以将其隐藏在文件中。它是系统中最敏感的数据。用户可以通过删除此文件强制系统提示输入 MP。更改主密码时，会将其重新保存在主密码密钥库（Java JCEKS 类型的密钥库）中。

要更改主密码，请执行以下步骤：

1. 停止域的 Application Server。使用 asadmin change-master-password 命令提示输入旧密码和新密码，然后对所有依赖项重新进行加密。例如：

```
asadmin change-master-password>
Please enter the master password>
Please enter the new master password>
Please enter the the new master password again>
```

2. 重新启动 Application Server。

注意 – 此时，不能启动正在运行的服务器实例并且不能重新启动运行服务器实例，除非已更改这些实例所对应的节点代理上的 SMP。如果在更改服务器实例的 SMP 之前重新启动了该服务器实例，它将无法启动。

3. 一次停止一个节点代理及与其相关的服务器。再次运行 asadmin change-master-password 命令，然后重新启动节点代理及其相关服务器。
4. 继续对下一个节点代理执行此过程，直到对所有节点代理均已执行此过程。这样就可以完成滚动更改。

使用主密码和密钥库

主密码是用于安全密钥库的密码。创建新应用服务器域时，会生成新的自签名证书并将其存储在相关密钥库中，此密钥库是使用主密码锁定的。如果主密码不是默认密码，则 start-domain 命令将提示您输入主密码。输入正确的主密码后，便会启动域。

创建与域关联的节点代理时，节点代理会将数据与域同步。执行此操作时，也会同步密钥库。由此节点代理所控制的任何服务器实例都需要打开密钥库。由于此密钥库实际上与域创建进程所创建的密钥库相同，因此只能使用相同的主密码将其打开。不过主密码本身从未同步，这意味着不会在同步过程中将其传输到节点代理，但是需要在本地为节点代理提供主密码。因此在创建和/或启动节点代理时会提示您输入主密码，并且您需要输入创建/启动域时所输入的密码。如果更改了某个域的主密码，则必须执行相同的步骤在与此域关联的每个节点代理上更改此密码。

更改管理密码

第 85 页中的 “管理密码安全性”中介绍了如何对管理密码进行加密。强烈建议您对管理密码进行加密。如果要在加密管理密码之前更改管理密码，请使用 `asadmin set` 命令。用于此目的的 `set` 命令的示例如下：

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

还可以使用管理控制台更改管理密码，步骤如下。

要使用管理控制台更改管理密码，请选择“配置”节点 > “要配置的实例” > “安全性”节点 > “领域”节点 > `admin-realm` 节点，并根据需要编辑领域页面。

关于验证和授权

验证和授权是应用服务器安全性的核心概念。以下主题讨论了与验证和授权相关的内容：

验证实体

验证是一种实体（用户、应用程序或组件）用来确定另一个实体是否是其声明的实体的方法。实体使用**安全凭证**对其自身进行验证。凭证可以是一个用户名和密码、一个数字证书或其他凭证。

通常，验证表示用户使用用户名和密码登录到某个应用程序；也可以指 EJB 从服务器请求资源时，提供安全凭证。通常，服务器或应用程序要求客户机进行验证；另外，客户机也可以要求服务器对其自身进行验证。如果验证是双向的，则称为双向验证。

当实体尝试访问受保护的资源时，Application Server 将使用为该资源配置的验证机制来决定是否授予访问权限。例如，用户可以在 Web 浏览器中输入用户名和密码，如果应用程序顺利完成了对那些凭证的检验，则表示该用户已通过验证。在此会话余下的时间内，该用户将始终与这个经过验证的安全身份相关联。

Application Server 支持四种类型的验证，如第 87 页中的 “验证实体”所述。应用程序在其部署描述符中指定所使用的验证类型。有关使用 `deploytool` 来配置应用程序验证方法的更多信息，请参见位于以下 URL 的《The J2EE 1.4 Tutorial》：`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`。

表 9–1 Application Server 验证方法

验证方法	通信协议	说明	用户凭证加密
基本	HTTP（SSL 可选）	使用服务器的内置弹出式登录对话框。	无，除非使用 SSL。
基于表单	HTTP（SSL 可选）	应用程序提供它自己的自定义登录页面和错误页面。	无，除非使用 SSL。
客户机证书	HTTPS（基于 SSL 的 HTTP）	服务器使用公钥证书来验证客户机。	SSL

检验单点登录

单点登录允许一个虚拟服务器实例中的多个应用程序共享用户验证状态。使用单点登录，登录到一个应用程序的用户也会隐式登录到需要相同验证信息的其他应用程序。

单点登录以组为基础。其部署描述符定义了相同的**组**并使用相同验证方法（基本、表单、摘要或证书）的所有 Web 应用程序共享单点登录。

对于为 Application Server 定义的虚拟服务器，默认情况下已启用单点登录。

对用户进行授权

用户通过验证后，**授权**级别将决定该用户可以执行哪些操作。用户的授权基于其**角色**。例如，人力资源应用程序可以授权管理者查看所有雇员的个人信息，但只允许雇员查看自己的个人信息。有关角色的更多信息，请参见第 89 页中的 “了解用户、组、角色和区域”。

指定 JACC 提供者

JACC（Java 容器授权合同）属于 J2EE 1.4 规范，它为可插拔授权提供者定义了接口。这使得管理员可以设置第三方插件模块来执行授权。

默认情况下，Application Server 提供一个符合 JACC 规范的基于文件的简单授权引擎。还可以指定其他第三方 JACC 提供者。

JACC 提供者使用 Java 验证和授权服务 (JAAS) API。JAAS 允许服务验证并强制对用户进行访问控制。JAAS 实现了 Java 技术版本的标准可插拔验证模块 (PAM) 框架。

审计验证和授权决策

Application Server 可以通过**审计模块**提供对所有验证和授权决策的审计跟踪。Application Server 提供了一个默认的审计模块，还提供了自定义审计模块的功能。有关开发自定义审计模块的信息，请参见 Application Server *Developer's Guide* 中的 *"Configuring an Audit Module"* 一节。

配置消息安全性

消息安全性使服务器可以在消息层执行 Web 服务调用和响应的端对端验证。Application Server 使用 SOAP 层上的消息安全性提供者来实现消息安全性。消息安全性提供者提供了请求和响应消息所需的验证类型等信息。支持的验证类型包括：

- 发件人验证，包括用户名和密码验证。
- 内容验证，包括 XML 数字签名。

该版本附带了两个消息安全性提供者。可以为 SOAP 层的验证配置消息安全性提供者。可以配置的提供者包括 `ClientProvider` 和 `ServerProvider`。

对消息层安全性的支持以（可插入）验证模块的形式集成到 Application Server 及其客户机容器中。默认情况下，Application Server 中的消息层安全性处于禁用状态。

可以为整个 Application Server 或者为特定应用程序或方法配置消息层安全性。第 10 章介绍了如何配置 Application Server 级别的消息安全性。Developer's Guide 的 *"Securing Applications"* 一章介绍了如何配置应用程序级别的消息安全性。

了解用户、组、角色和区域

Application Server 对以下实体强制执行其验证和授权策略：

- 第 90 页中的 “用户”：*Application Server* **中定义的**单个身份。通常，用户是指一个人、一个软件组件（例如企业 Bean），甚至是一种服务。经过验证的用户有时称为**主体**。用户有时称为**主题**。
- 第 90 页中的 “组”：*Application Server* **中定义的**一组用户，按照常见特性进行分类。
- 第 90 页中的 “角色”：**由应用程序定义的**命名授权级别。可以将角色比喻为开锁的钥匙。许多人都可以有此钥匙的复制钥匙。锁不关心谁要造访，而只关心使用的钥匙是否正确。
- 第 91 页中的 “区域”：包含用户和组信息及其关联的安全凭证的系统信息库。领域也称为**安全策略域**。

注 – 尽管用户和组是为整个 Application Server 指定的，但是每个应用程序都需要定义自己的角色。当封装和部署应用程序时，应用程序会指定用户/组和角色之间的映射，如下图所示。

用户

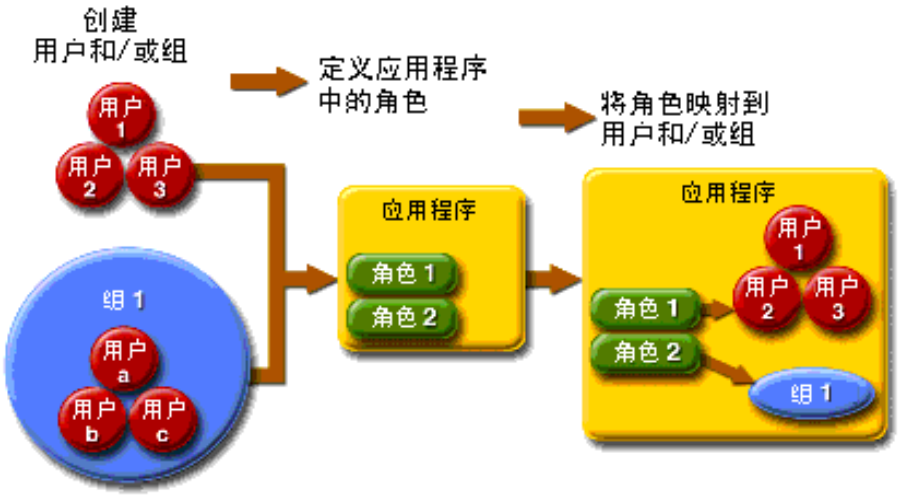

图 9–1 角色映射

用户是已在 Application Server 中定义的单个（或应用程序）身份。用户可以与组关联。Application Server 验证服务可以管理多个领域中的用户。

组

J2EE **组**（或简称组）是按常见特性（例如职务或用户配置文件）分类的用户类别。例如，假定电子商务应用程序的用户属于 `customer` 组，但是大客户可以属于 `preferred` 组。将用户分组可以简化对用户量很大时的访问控制。

角色

角色定义用户可以访问哪些应用程序和每个应用程序的哪些部分，并定义用户可以执行的操作。也就是说，角色决定用户的授权级别。

例如，假定在人事应用程序中，所有雇员均可以访问电话号码和电子邮件地址，但只有管理人员才能访问薪水信息。该应用程序至少需要定义两个角色：`employee` 和 `manager`；仅允许处于 `manager` 角色的用户查看薪水信息。

角色与用户组的不同之处在于，角色在应用程序中定义功能，而用户组是以某一方式相关的一组用户。例如，假定在人事应用程序中有 `full-time`、`part-time` 和 `on-leave` 几个组，但所有这些组中的用户仍是 `employee` 角色。

角色是在应用程序部署描述符中定义的。相反，组是针对整个服务器和区域而定义的。应用程序开发者或部署者在每个应用程序的部署描述符中将角色映射到一个或多个组。

区域

领域也称为**安全策略域**或**安全域**，是服务器定义和强制执行通用安全策略的范围。在实际应用中，区域是服务器存储用户和组信息的系统信息库。

Application Server 预先配置了三个领域：`file`（初始默认领域）、`certificate` 和 `admin-realm`。还可以设置 `ldap`、`solaris` 或自定义领域。应用程序可以在其部署描述符中指定要使用的区域。如果应用程序不指定领域，Application Server 将使用其默认领域。

在 `file` 领域中，服务器将用户凭证存储在本地名为 `keyfile` 的文件中。您可以使用管理控制台来管理 `file` 领域中的用户。.

在 `certificate` 领域中，服务器将用户凭证存储在证书数据库中。使用 `certificate` 领域时，服务器结合使用证书和 HTTPS 协议来验证 Web 客户机。有关证书的更多信息，请参见第 92 页中的 “证书和 SSL 简介”。

`admin-realm` 也是一个 `FileRealm`，它将管理员用户凭证存储在本地名为 `admin-keyfile` 的文件中。您可以使用管理控制台管理此领域中的用户，方法与管理 `file` 领域中的用户相同。

在 `ldap` 领域中，服务器将从轻量目录访问协议 (Lightweight Directory Access Protocol, LDAP) 服务器（例如 Sun Java System Directory Server）中获取用户凭证。LDAP 是一种协议，它使任何人都可以在网络（无论是公共 Internet 还是企业内联网）中查找组织、个人和其他资源（例如文件和设备）。有关管理 `ldap` 领域中的用户和组的信息，请参阅您的 LDAP 服务器文档。

在 `solaris` 领域中，服务器将从 Solaris 操作系统中获取用户凭证。Solaris 9 OS 和更高版本支持此区域。有关管理 `solaris` 领域中的用户和组的信息，请参阅您的 Solaris 文档。

自定义区域是用户凭证的任何其他系统信息库，例如关系型数据库或第三方组件。有关更多信息，请参见管理控制台联机帮助。

证书和 SSL 简介

本节包括以下主题：

关于数字证书

数字证书（或简称证书）是在 Internet 上唯一地标识人员和资源的电子文件。证书使两个实体之间能够进行安全、保密的通信。

证书有很多种类型，例如个人证书（由个人使用）和服务器证书（用于通过安全套接字层 [SSL] 技术在服务器和客户机之间建立安全会话）。有关 SSL 的更多信息，请参见第 93 页中的 “关于安全套接字层”。

证书是基于**公钥加密**的，公钥加密使用数字**密钥**对（很长的数字）对信息进行**加密**或编码，从而使信息只能被目标收件人读取。然后，收件人对信息进行**解密**（解码）即可读取该信息。

一个密钥对包含一个公钥和一个私钥。拥有者对公钥进行分发并使任何人都可以使用该公钥。但是拥有者永远不会分发私钥；私钥始终是保密的。由于密钥与数学相关，因此使用了密钥对中的一个密钥进行加密的数据只能通过密钥对中的另一个密钥进行解密。

证书就好像一本护照：它可以标识持有者并提供其他重要信息。证书由称为**证书授权机构** (Certification Authority, CA) 的可信赖第三方发布。CA 类似于护照申领办公室：它将验证证书持有者的身份并对证书进行签名，以使他人无法伪造或篡改证书。CA 对证书进行签名之后，持有者可以提供该证书作为身份证明并建立经过加密的保密通信。

最重要的是，证书会将拥有者的公钥绑定到拥有者的标识。与护照将照片绑定到其持有者的个人信息类似，证书将公钥绑定到有关其拥有者的信息。

除了公钥以外，证书通常还包括以下信息：

- 持有者的姓名和其他标识，例如使用证书的 Web 服务器的 URL 或个人的电子邮件地址。
- 发布证书的 CA 的名称。
- 失效日期。

数字证书受 X.509 格式的技术规范约束。要检验 `certificate` 领域中某个用户的身份，验证服务将使用 X.509 证书的通用名称字段作为主体名称来检验 X.509 证书。

关于证书链

Web 浏览器已预先配置了一组浏览器自动信任的**根** CA 证书。来自其他证书授权机构的所有证书都必须附带**证书链**，以检验这些证书的有效性。证书链是由一系列 CA 证书发出的证书序列，最终以根 CA 证书结束。

证书最初生成时是一个**自签名**证书。自签名证书是其签发者（签名者）与主题（其公钥由该证书进行验证的实体）相同的证书。如果拥有者向 CA 发送证书签名请求 (CSR)，然后输入响应，自签名证书将被证书链替换。链的底部是由 CA 发布的、用于验证主题的公钥的证书（回复）。链中的下一个证书是验证 CA 的公钥的证书。通常，这是一个自签名证书（即，来自 CA、用于验证其自身的公钥的证书）并且是链中的最后一个证书。

在其他情况下，CA 可能会返回一个证书链。在此情况下，链的底部证书是相同的（由 CA 签发的证书，用于验证密钥条目的公钥），但是链中的第二个证书是由其他 CA 签发的证书，用于验证您向其发送了 CSR 的 CA 的公钥。然后，链中的下一个证书是用于验证第二个 CA 的密钥的证书，依此类推，直至到达自签名的**根**证书。因此，链中的每个证书（第一个证书之后的证书）都需要验证链中前一个证书的签名者的公钥。

关于安全套接字层

安全套接字层 (Secure Socket Layer, SSL) 是用来确保 Internet 通信和事务安全的最常见的标准。Web 应用程序使用 HTTPS（基于 SSL 的 HTTP），HTTPS 使用数字证书来确保在服务器和客户机之间进行安全、保密的通信。在 SSL 连接中，客户机和服务器在发送数据之前都要对数据进行加密，然后由收件人对其进行解密。

当 Web 浏览器（客户机）需要与某个安全站点建立连接时，则会发生 *SSL* **握手**：

- 浏览器将通过网络发送请求安全会话的消息（通常请求以 `https` 而非 `http` 开头的 URL）。
- 服务器通过发送其证书（包括公钥）进行响应。
- 浏览器将检验服务器的证书是否有效，并检验该证书是否是由其证书位于浏览器的数据库中的（并且是可信的）CA 所签发的。它还将检验 CA 证书是否已过期。
- 如果证书有效，浏览器将生成一个一次性的、唯一的**会话密钥**，并使用服务器的公钥对该会话密钥进行加密。然后，浏览器将把加密的会话密钥发送给服务器，这样服务器和浏览器都有一份会话密钥。
- 服务器可以使用其专用密钥对消息进行解密，然后恢复会话密钥。

握手之后，即表示客户机已检验了 Web 站点的身份，并且只有该客户机和 Web 服务器拥有会话密钥副本。从现在开始，客户机和服务器便可以使用该会话密钥对彼此间的所有通信进行加密。这样就确保了客户机和服务器之间的通信的安全性。

SSL 标准的最新版本称为 TLS（传输层安全性）。Application Server 支持安全套接字层 (Secure Socket Layer, SSL) 3.0 和传输层安全性 (Transport Layer Security, TLS) 1.0 加密协议。

要使用 SSL，Application Server 必须拥有接受安全连接的每个外部接口或 IP 地址的证书。只有安装了数字证书之后，大多数 Web 服务器的 HTTPS 服务才能够运行。使用第 99 页中的 “使用 keytool 实用程序生成证书”中介绍的过程来设置您的 Web 服务器可用于 SSL 的数字证书。

关于加密器

加密器是用于加密或解密的加密算法。SSL 和 TLS 协议支持用于服务器和客户机彼此进行验证、传输证书和建立会话密钥的各种加密器。

某些加密器比其他加密器更强大且更安全。客户机和服务器可以支持不同的密码组。从 SSL3 和 TLS 协议中选择加密器。在安全连接期间，客户机和服务器同意在通信中使用它们均已启用的最强大的加密器，因此通常需要启用所有加密器。

使用基于名称的虚拟主机

对安全应用程序使用基于名称的虚拟主机可能会带来问题。这是 SSL 协议本身的设计限制。必须先进行 SSL 握手（客户机浏览器在这时接受服务器证书），然后才能访问 HTTP 请求。这样，在验证之前就无法确定包含虚拟主机名的请求信息，因此也不能将多个证书指定给单个 IP 地址。

如果单个 IP 地址上的所有虚拟主机都需要通过同一证书的验证，则添加多个虚拟主机将不会影响服务器上正常的 SSL 操作。但是请注意，大多数浏览器会将服务器的域名与证书中列出的域名（如果有的话，也主要适用于官方的 CA 签名证书）进行比较。如果域名不匹配，这些浏览器将显示警告。通常在生产环境中，只将基于地址的虚拟主机与 SSL 一起使用。

关于防火墙

防火墙控制两个或多个网络之间的数据流，并管理网络之间的链接。防火墙可以包含硬件和软件元素。本节介绍了一些常用的防火墙体系结构及其配置。此处的信息主要是针对 Application Server 的。有关特定防火墙技术的详细信息，请参阅防火墙供应商提供的文档。

通常，需要对防火墙进行配置，以便客户机访问所需的 TCP/IP 端口。例如，如果 HTTP 侦听器正在端口 8080 上运行，则将防火墙配置为仅允许处理端口 8080 上的 HTTP 请求。同样地，如果为端口 8181 设置了 HTTPS 请求，则必须将防火墙配置为允许处理端口 8181 上的 HTTPS 请求。

如果需要通过 Internet 对 EJB 模块进行直接的 RMI-IIOP 访问，RMI-IIOP 全称为 Remote Method Invocations over Internet Inter-ORB Protocol（通过基于 Internet 的 ORB 间协议的远程方法调用），则还需要打开 RMI-IIOP 侦听器端口，但强烈建议您不要这样做，因为这样可能会破坏安全性。

在双防火墙体系结构中，您必须将外部防火墙配置为允许处理 HTTP 和 HTTPS 事务。您必须将内部防火墙配置为允许 HTTP 服务器插件与防火墙后面的 Application Server 进行通信。

使用管理控制台管理安全性

管理控制台提供了对安全性的以下方面进行管理的方法：

服务器安全性设置

在“安全性设置”页面中，设置整个服务器的属性，包括指定默认区域、匿名角色和默认的主体用户名和密码。

领域和 file 领域用户

第 89 页中的 “了解用户、组、角色和区域”中介绍了领域的概念。

- 创建新领域
- 删除现有领域
- 修改现有领域的配置
- 添加、修改和删除 `file` 领域中的用户
- 设置默认领域

JACC 提供者

第 88 页中的 “指定 JACC 提供者”中介绍了 JACC 提供者。使用管理控制台可以执行以下任务：

- 添加新的 JACC 提供者
- 删除或修改现有 JACC 提供者

审计模块

第 89 页中的 “审计验证和授权决策”中介绍了审计模块。审计是记录重要事件（例如错误或安全漏洞）以便随后进行检查的方法。所有验证事件都被记录到 Application Server 日志中。完整的访问日志提供了 Application Server 访问事件的顺序线索。

使用管理控制台可以执行以下任务：

- 添加新的审计模块
- 删除或修改现有审计模块

消息安全性

第 89 页中的 “配置消息安全性”中介绍了消息安全性的概念。使用管理控制台可以执行以下任务：

- 启用消息安全性
- 配置消息安全性提供者
- 删除或配置现有消息安全性配置或提供者

有关这些任务的详细信息，请参见管理控制台联机帮助。

HTTP 和 IIOP 侦听器安全性

HTTP 服务中的每个虚拟服务器都通过一个或多个 *HTTP* **侦听器**提供网络连接。

Application Server 支持 CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）对象，这类对象使用基于 Internet 对象请求代理间协议 (Internet Inter-Orb Protocol, IIOP) 在网络上进行通信。*IIOP* **侦听器**接受来自 EJB 组件的远程客户机和其他基于 CORBA 的客户机的外来连接。有关 IIOP 侦听器的一般信息，请参见第 134 页中的 “IIOP 侦听器”。

使用管理控制台可以执行以下任务：

- 创建新的 HTTP 或 IIOP 侦听器，并指定该侦听器所使用的安全性。
- 修改现有 HTTP 或 IIOP 侦听器的安全性设置。

管理服务安全性

管理服务确定服务器实例是一个常规实例、一个域管理服务器 (DAS) 还是一个组合。使用管理服务配置 JSR-160 兼容的远程 JMX 连接器，该连接器处理域管理服务器与用于远程服务器实例的节点代理（管理主机上的服务器实例）之间的通信。

使用管理控制台可以执行以下任务：

- 管理管理服务
- 编辑 JMX 连接器
- 修改 JMX 连接器的安全性设置

安全映射

使用管理控制台可以执行以下安全映射任务：

- 将安全映射添加到现有连接器连接池中
- 删除或配置现有安全映射

使用证书和 SSL

关于证书文件

安装 Application Server 时将生成一个适用于内部测试的 JSSE（Java Secure Socket Extension，Java 安全套接口扩展）或 NSS（Network Security Service，网络安全服务）格式的数字证书。默认情况下，Application Server 将其证书信息存储在 *domain-dir*`/config` 目录下的证书数据库中：

- **密钥库文件** `key3.db` 中包含 Application Server 的证书（包括其私钥）。密钥库文件受密码保护。使用 `asadmin change-master-password` 命令可以更改该密码。有关 `certutil` 的更多信息，请参见第 102 页中的 “使用 certutil 实用程序”。

  每个密钥库条目都有唯一的别名。安装后，Application Server 密钥库会有一个别名为 `s1as` 条目。

- **信任库文件** `cert8.db` 中包含 Application Server 的信任证书（包括其他实体的公钥）。对于信任的证书，服务器已确认证书中的公钥属于证书的拥有者。信任的证书通常包括那些证书授权机构 (CA) 的证书。

  在 Platform Edition 的服务器端，Application Server 使用 JSSE 格式，该格式使用 `keytool` 来管理证书和密钥库。在 Enterprise Edition 的服务器端，Application Server 使用 NSS 格式，该格式使用 `certutil` 来管理存储私钥和证书的 NSS 数据库。在两种版本的客户机端（应用程序客户机端或独立客户机端）上均使用 JSSE 格式。

  默认情况下，Application Server 配有一个密钥库和一个信任库，它们将与示例应用程序配合使用并用于开发目的。在用于生产目的时，您可能需要更改证书别名、将其他证书添加到信任库或更改密钥库文件和信任库文件的名称和/或位置。

更改证书文件的位置

为开发提供的密钥库和信任库文件存储在 *domain-dir*`/config` 目录中。

使用管理控制台依次展开 `server-config` 节点 > “JVM 设置” > “JVM 选项” 选项卡，可以添加或修改证书文件新位置的值字段。

```
-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/NSS-database-directory
```

其中，*NSS-database-directory* 是 NSS 数据库的位置。

使用 Java 安全套接口扩展 (Java Secure Socket Extension, JSSE) 工具

使用 `keytool` 可以设置和使用 JSSE（Java Secure Socket Extension，Java 安全套接口扩展）数字证书。在 Platform Edition 和 Enterprise Edition 中，客户机端（应用程序客户机端或独立客户机端）均使用 JSSE 格式。

J2SE SDK 附带了 `keytool`，管理员可以使用它来管理公钥/私钥对和关联证书。还允许用户高速缓存正与其通信的另一方的公钥（以证书形式）。

要运行 `keytool`，必须先配置 shell 环境以使 J2SE `/bin` 目录位于路径中；或者必须在命令行中提供此工具的完整路径。有关 `keytool` 的更多信息，请参见 `http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html` 上的 `keytool` 文档。

使用 keytool 实用程序

以下示例说明了使用 JSSE 工具处理证书的相关用法：

- 使用 RSA 密钥算法在 JKS 类型的密钥库中创建自签名证书。RSA 是 RSA Data Security, Inc. 开发的公钥加密技术。RSA 缩略词分别代表该技术的三位发明者：Rivest、Shamir 和 Adelman。

  ```
  keytool  -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
  -dname  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  第 99 页中的 “使用 keytool 实用程序生成证书”中显示了创建证书的另一个示例。

- 使用默认密钥算法在 JKS 类型的密钥库中创建自签名证书。

  ```
  keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
  ${keystore.pass}
  ```

  第 100 页中的 “使用 keytool 实用程序为数字证书签名”中显示了一个为证书签名的示例。

- 显示 JKS 类型的密钥库中可用的证书。

  ```
  keytool -list -v  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- 显示 JKS 类型的密钥库中的证书信息。

```
keytool  -list -v  -alias ${cert.alias} -keystore ${keystore.file}
-storepass ${keystore.pass}
```

- 将 RFC/文本格式的证书导入 JKS 库。通常，使用 Internet RFC（Request for Comments，请求注释）1421 标准定义的可打印编码格式（而非其二进制编码）来存储证书。此证书格式也称为 *Base 64* **编码**，便于通过电子邮件或某些其他机制将证书导出到其他应用程序。

```
keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- 以 PKCS7 格式从 JKS 类型的密钥库中导出证书。公钥加密标准 #7 加密消息语法标准定义的回复格式包括发布的证书以及支持证书链。

```
keytool  -export -noprompt  -alias ${cert.alias} -file ${cert.file}
-keystore ${keystore.file} -storepass ${keystore.pass}
```

- 以 RFC/文本格式从 JKS 类型的密钥库中导出证书。

```
keytool  -export -noprompt -rfc  -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- 从 JKS 类型的密钥库中删除证书。

```
keytool  -delete -noprompt -alias ${cert.alias}  -keystore ${keystore.file}
-storepass ${keystore.pass}
```

第 101 页中的 “使用 keytool 实用程序删除证书”中显示了从密钥库中删除证书的另一个示例。

使用 keytool 实用程序生成证书

使用 `keytool` 可以生成、导入和导出证书。默认情况下，`keytool` 将在其运行所在的目录中创建一个密钥库文件。

1. 转至要运行证书的目录。

   始终在包含密钥库和信任库文件的目录中生成证书，默认目录为 *domain-dir*`/config`。有关更改这些文件位置的信息，请参见第 97 页中的 “更改证书文件的位置”。

2. 输入以下 `keytool` 命令以在密钥库文件 `keystore.jks` 中生成证书：

```
keytool -genkey -alias keyAlias-keyalg RSA
 -keypass changeit
 -storepass changeit
-keystore keystore.jks
```

   使用任一唯一的名称作为您的 *keyAlias*。如果您已更改密钥库或私钥密码的默认值，请将以上命令中的 `changeit` 替换为新密码。

将显示一个要求您输入姓名、组织和其他信息的提示，keytool 将使用这些信息来生成证书。

3. 输入以下 keytool 命令以将生成的证书导出到文件 server.cer（或 client.cer，如果您愿意）：

```
keytool -export -alias keyAlias-storepass changeit
 -file server.cer
 -keystore keystore.jks
```

4. 如果要求证书授权机构签名的证书，请参见第 100 页中的 “使用 keytool 实用程序为数字证书签名”。
5. 要创建信任库文件 cacerts.jks 并将证书添加到信任库中，请输入以下 keytool 命令：

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
```

6. 如果您已更改密钥库或私钥密码的默认值，请将以上命令中的 changeit 替换为新密码。

   工具将显示有关证书的信息并提示您是否要信任该证书。
7. 键入 yes，然后按 Enter 键。

   然后，keytool 将显示与下面类似的信息：

```
Certificate was added to keystore
[Saving cacerts.jks]
```

8. 重新启动 Application Server。

使用 keytool 实用程序为数字证书签名

创建数字证书之后，拥有者必须为其签名以防止伪造。电子商务站点或身份验证对其很重要的那些站点可以从知名的证书授权机构 (CA) 购买证书。如果无需考虑验证，例如当专用安全通信可以满足全部需求时，则可节省获取 CA 证书所花费的时间和费用并使用自签名证书。

1. 按照 CA Web 站点上的说明进行操作来生成证书密钥对。
2. 下载生成的证书密钥对。

   将证书保存在包含密钥库和信任库文件的目录中，默认目录为 *domain-dir*/config。请参见第 97 页中的 “更改证书文件的位置”。
3. 在 shell 中，转至包含证书的目录。
4. 使用 keytool 将证书导入到本地密钥库和本地信任库（如有必要）。

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
-storepass changeit
```

如果密钥库或私钥密码不是默认密码，请将以上命令中的 `changeit` 替换为新密码。

5. 重新启动 Application Server。

使用 keytool 实用程序删除证书

要删除现有证书，请使用 `keytool -delete` 命令，例如：

```
keytool -delete
 -alias keyAlias
 -keystore keystore-name
 -storepass password
```

使用网络安全服务 (NSS) 工具

在 Enterprise Edition 中，在服务器端使用网络安全服务 (Network Security Service, NSS) 数字证书可以管理存储私钥和证书的数据库。对于客户机端（应用程序客户机端或独立客户机端），均使用第 98 页中的 “使用 Java 安全套接口扩展 (Java Secure Socket Extension, JSSE) 工具”中介绍的 JSSE 格式。

通过网络安全服务 (NSS) 管理安全性的工具包括：

- `certutil`，一个命令行实用程序，用于管理证书和密钥数据库。第 102 页中的 “使用 certutil 实用程序”中显示了一些使用 `certutil` 实用程序的示例。
- `pk12util`，一个命令行实用程序，用于以 PKCS12 格式在证书/密钥数据库和文件之间导入和导出密钥及证书。第 103 页中的 “使用 pk12util 实用程序导入和导出证书”中显示了一些使用 `pk12util` 实用程序的示例。
- `modutil`，一个命令行实用程序，用于管理 `secmod.db` 文件中或硬件令牌中的 PKCS #11 模块信息。第 104 页中的 “使用 modutil 添加和删除 PKCS11 模块”中显示了一些使用 `modutil` 实用程序的示例。

这些工具位于 *install-dir*`/lib/` 目录中。下面的环境变量用于指出 NSS 安全性工具的位置：

- `LD_LIBRARY_PATH =${`*install-dir*`}/lib`
- `${`*os.nss.path*`}`

在示例中，证书通用名 (Common Name, CN) 是客户机或服务器的名称。在 SSL 握手时也会使用 CN，以比较证书名称和生成证书名的主机名。如果证书名称与主机名不匹

配，在 SSL 握手时会产生警告或异常。在某些示例中，使用证书通用名 CN=localhost 是为了方便起见，这样所有用户都可以使用该证书，而不必用他们自己的真实主机名创建一个新证书。

以下各节中的示例说明使用 NSS 工具处理证书的相关用法：

- 第 102 页中的 “使用 certutil 实用程序”
- 第 103 页中的 “使用 pk12util 实用程序导入和导出证书”
- 第 104 页中的 “使用 modutil 添加和删除 PKCS11 模块”

使用 certutil 实用程序

在运行 certutil 之前，请确保 LD_LIBRARY_PATH 指向运行此实用程序所需的库的位置。此位置可以通过 asenv.conf（产品范围的配置文件）中 AS_NSS_LIB 的值来标识。

证书数据库工具 certutil 是一个 NSS 命令行实用程序，可以创建和修改 Netscape Communicator cert8.db 和 key3.db 数据库文件。还可以列出、生成、修改或删除 cert8.db 文件中的证书，以及创建或更改密码、生成新的公钥和私钥对、显示密钥数据库的内容或删除 key3.db 文件中的密钥对。

密钥和证书管理进程通常以在密钥数据库中创建密钥开始，然后在证书数据库中生成和管理证书。位于以下网址的文档说明了使用 NSS（包括 certutil 实用程序的语法）管理证书和密钥数据库
：http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

下表中的每一项都给出了一个使用 NSS 和 JSSE 安全性工具来创建和/或管理证书的示例。

- 生成自签名服务器和客户机证书。在本示例中，CN 的格式必须为 hostname.domain.[com|org|net|...]。

  在本示例中，目录为 *domain-dir*/config。serverseed.txt 和 clientseed.txt 文件可以包含任何随机文本。此随机文本将用于生成密钥对。

```
certutil -S -n $SERVER_CERT_NAME -x -t "u,u,u"
 -s "CN=$HOSTNAME.$HOSTDOMAIN, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25001 -o  $CERT_DB_DIR/Server.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/serverseed.txt
```

  生成客户机证书。此证书也是一个自签名证书。

```
certutil -S -n $CLIENT_CERT_NAME  -x -t "u,u,u"
 -s "CN=MyClient, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25002 -o  $CERT_DB_DIR/Client.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/clientseed.txt
```

- 检验前面的项目中生成的证书。

```
certutil -V -u V -n $SERVER_CERT_NAME -d  $CERT_DB_DIR
certutil -V -u C -n $CLIENT_CERT_NAME -d  $CERT_DB_DIR
```

- 显示可用证书。

```
certutil -L -d  $CERT_DB_DIR
```

- 将 RFC 文本格式的证书导入 NSS 证书数据库。

```
certutil -A -a -n ${cert.nickname} -t ${cert.trust.options}
-f ${pass.file} -i ${cert.rfc.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 以 RFC 格式从 NSS 证书数据库中导出证书。

```
certutil -L -a -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
```

- 从 NSS 证书数据库中删除证书。

```
certutil -D -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 将 NSS 数据库中的证书转换为 JKS 格式

```
certutil -L -a -n ${cert.nickname}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
keytool  -import -noprompt -trustcacerts -keystore ${keystore.file}
-storepass ${keystore.pass} -alias ${cert.alias} -file cert.rfc
```

使用 pk12util 实用程序导入和导出证书

`pk12util` 是一个命令行实用程序，用于以 PKCS12 格式在证书/密钥数据库和文件之间导入和导出密钥和证书。PKCS12 是公钥加密标准 (Public-Key Cryptography Standards, PKCS) #12 个人信息交换语法标准。有关 `pk12util` 实用程序的更多说明，请参见 http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html。

- 将 PKCS12 格式的证书导入 NSS 证书数据库。

```
pk12util -i ${cert.pkcs12.file} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 将 PKCS12 格式的证书导入 NSS 证书数据库令牌模块。

```
pk12util -i ${cert.pkcs12.file} -h ${token.name} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 以 PKCS12 格式从 NSS 证书数据库中导出证书。

```
pk12util -o -n ${cert.nickname} -k ${pass.file} -w${cert.pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 以 PKCS12 格式从 NSS 证书数据库令牌模块中导出证书（用于硬件加速器配置）。

```
pk12util -o -n ${cert.nickname} -h ${token.name} -k ${pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 将 PKCS12 证书转换为 JKS 格式（需要 Java 源代码）：

```
&lt;target name="convert-pkcs12-to-jks" depends="init-common">
   &lt;delete file="${jks.file}" failonerror="false"/>
   &lt;java classname="com.sun.enterprise.security.KeyTool">
      &lt;arg line="-pkcs12"/>
      &lt;arg line="-pkcsFile ${pkcs12.file}"/>
      &lt;arg line="-pkcsKeyStorePass ${pkcs12.pass}"/>
      &lt;arg line="-pkcsKeyPass ${pkcs12.pass}"/>
      &lt;arg line="-jksFile ${jks.file}"/>
      &lt;arg line="-jksKeyStorePass ${jks.pass}"/>
      &lt;classpath>
         &lt;pathelement path="${s1as.classpath}"/>
         &lt;pathelement path="${env.JAVA_HOME}/jre/lib/jsse.jar"/>
      &lt;/classpath>
   &lt;/java>
&lt;/target>
```

使用 modutil 添加和删除 PKCS11 模块

安全性模块数据库工具 `modutil` 是一个命令行实用程序，用于管理 `secmod.db` 文件中或硬件令牌中的 PKCS #11（Cryptographic Token Interface Standard，加密令牌接口标准）模块信息。您可以使用此工具添加和删除 PKCS #11 模块、更改密码、设置默认值、列出模块内容、启用或禁用插槽、启用或禁用 FIPS-140-1 兼容性以及指定加密操作的默认提供者。此工具还可以创建 `key3.db`、`cert7.db` 和 `secmod.db` 安全性数据库文件。有关此工具的更多信息，请参见 `http://www.mozilla.org/projects/security/pki/nss/tools/modutil.html`。

- 添加新的 PKCS11 模块或令牌。

```
modutil -add ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- 从 NSS 库中删除 PKCS11 模块。

```
modutil -delete ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- 列出 NSS 库中的可用令牌模块。

```
modutil -list  -dbdir ${admin.domain.dir}/${admin.domain}/config
```

对 Application Server 使用硬件加密加速器

您可以使用硬件加速器令牌来提高加密性能并提供安全密钥存储功能。此外，还可以通过智能卡为最终用户提供移动安全密钥存储。

当 Sun Java System Application Server 8.1 和 8.2 Standard Edition 或 Enterprise Edition 在 Java 2 Platform, Standard Edition（J2SE 平台）5.0 上运行时，支持针对 SSL 或 TLS 通信使用 PKCS#11 令牌，以及使用网络安全服务 (Network Security Service, NSS) 工具来管理密钥和 PKCS#11 令牌。本节介绍 Application Server 如何提供此支持，并指导您完成相关配置的过程。

J2SE 5.0 PKCS#11 提供者可以轻松与 Application Server 运行时集成。通过这些提供者，您可以使用硬件加速器以及 Application Server 中的其他 PKCS#11 令牌来提高性能，并保护 SSL 或 TLS 通信中固有的私钥。

本节包括以下主题：

关于配置硬件加密加速器

Sun Java System Application Server 8.1 和 8.2 Standard Edition 或 Enterprise Edition 已针对 Sun Crypto Accelerator 1000 (SCA-1000) 和 SCA-4000 进行了测试。

Application Server 与 J2SE 5.0 结合使用时，可以与 PKCS#11 令牌进行通信。Application Server 附带有 NSS PKCS#11 令牌库（对于 NSS 内部 PKCS#11 模块，通常称为 NSS 软令牌）和 NSS 命令行管理工具。有关更多详细信息，请参见第 101 页中的 “使用网络安全服务 (NSS) 工具”。

使用 NSS 工具可以在 PKCS#11 令牌中创建密钥和证书，使用 J2SE PKCS#11 提供者可以在运行时访问令牌密钥和证书。PKCS#11 提供者是用作本地 PKCS#11 库包装器的加密服务提供者。PKCS#11 令牌通常是指所有具有本地 PKCS#11 接口的硬件令牌和软件令牌。硬件令牌是指以物理设备（例如硬件加速器及智能卡）实现的 PKCS#11 令牌。软件令牌是指完全以软件实现的 PKCS#11 令牌。

注 – 如果在 J2SE 1.4.x 平台上运行 Application Server，则仅支持一种 PKCS#11 令牌，即 NSS 软令牌。

对于 Microsoft Windows 环境，将 NSS 库的位置 *AS_NSS* 以及 NSS 工具目录 *AS_NSS_BIN* 添加到 PATH 环境变量中。为简单起见，本节中所述的过程仅使用 UNIX 命令。您应在适当的情况下将 UNIX 变量替换为 Windows 变量。

配置硬件加密加速器包括两个主要过程：

- 第 106 页中的 “配置 PKCS#11 令牌”
- 第 108 页中的 “配置 J2SE 5.0 PKCS#11 提供者”

配置 PKCS#11 令牌

本节介绍如何使用 NSS 安全性工具 `modutil` 来配置 PKCS#11 令牌。可以使用以下过程配置 PKCS#11 令牌。

输入以下命令（在一行中输入全部内容）：

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add moduleName -libfile
 absolute_path_of_pkcs11_library -mechanisms list_of_security_mechanisms
```

其中，*AS_NSS_DB* 是 NSS 数据库目录（如同使用域管理服务器 (Domain Administration Server, DAS) 时的 `AS_DOMAIN_CONFIG`）

例如，要配置硬件加速器令牌，请输入以下命令（在一行中输入全部内容）：

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add "Sun Crypto Accelerator" -libfile
 /opt/SUNWconn/crypto/lib/libpkcs11.so -mechanisms RSA:DSA:RC4:DES
```

在本例中，硬件加速器为 SCA–1000 加密加速器。默认情况下，相应的 PKCS#11 库位于 `/opt/SUNWconn/crypto/lib/libpkcs11.so` 中。

`mechanisms` 必须是可用于令牌中的加密机制的完整列表。要仅使用少数可用的加密机制，请参见第 108 页中的 “配置 J2SE 5.0 PKCS#11 提供者” 。有关所有支持的机制的列表，请参见 NSS Security Tools 站点 `http://www.mozilla.org/projects/security/pki/nss/tools` 上的 `modutil` 文档。

下面的示例假设在安装令牌时指定的令牌名称为 `mytoken`。

要检验硬件加速器配置是否正确，请输入以下命令：

```
modutil -list -dbdir AS_NSS_DB
```

标准输出类似于以下内容：

```
Using database directory /var/opt/SUNWappserver/domains/domain1/config ...

Listing of PKCS#11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS#11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services
```

```
         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB

  2. Sun Crypto Accelerator
        library name: /opt/SUNWconn/crypto/lib/libpkcs11.so
         slots: 1 slot attached
        status: loaded

         slot: Sun Crypto Accelerator:mytoken
        token: mytoken
-------------------------------------------------------------
```

管理密钥和证书

本节介绍几个使用 `certutil` 和 `pk12util` 创建和管理密钥和证书的常用过程。有关`certutil` 和 `pk12util` 的详细信息，请参见第 101 页中的 “使用网络安全服务 (NSS) 工具”以及 NSS Security Tools 站点 `http://www.mozilla.org/projects/security/pki/nss/tools` 上的文档。

注 – 通过在 `java.security` 属性文件（位于 Java 运行时的 *JAVA_HOME*`/jre/lib/security` 目录中）中配置 PKCS#11 提供者，您还可以使用 J2SE `keytool` 实用程序来管理密钥和证书。有关使用 `keytool` 的详细信息，请参见位于 `http://java.sun.com/j2se/1.5.0/docs/guide/secuirty/p11guide.html` 的《Java PKCS#11 Reference Guide》 。

本节包括以下数个主题：

- 第 107 页中的 “列出密钥和证书”
- 第 108 页中的 “使用私钥和证书”

列出密钥和证书

- 要列出已配置的 PKCS#11 令牌中的密钥和证书，请运行以下命令：

  ```
  certutil -L -d AS_NSS_DB [-h tokenname]
  ```

  例如，要列出默认 NSS 软令牌的内容，请键入以下命令：

  ```
  certutil -L -d AS_NSS_DB
  ```

  标准输出类似于以下内容：

  ```
  verisignc1g1                 T,c,c
  verisignc1g2                 T,c,c
  verisignc1g3                 T,c,c
  ```

```
verisignc2g3                  T,c,c
verisignsecureserver          T,c,c
verisignc2g1                  T,c,c
verisignc2g2                  T,c,c
verisignc3g1                  T,c,c
verisignc3g2                  T,c,c
verisignc3g3                  T,c,c
s1as                          u,u,u
```

此输出的左列显示了令牌的名称，右列显示了一组（三个）信任属性，对于 Application Server 证书，这组属性通常为 `T,c,c`。与 J2SE `java.security.KeyStore` API（它仅包含一个信任级别）不同，NSS 技术包含多个信任级别。Application Server 主要关注第一个信任属性，该属性说明此令牌如何使用 SSL。对于该属性：

`T` 表示可信任证书授权机构 (Certificate Authority, CA) 颁发客户机证书。
`u` 表示您可以使用证书（及密钥）进行验证或签名。
属性组合 `u,u,u` 表示数据库中存在私钥。

- 要列出硬件令牌 `mytoken` 的内容，请运行以下命令：

```
certutil -L -d AS_NSS_DB -h mytoken
```

系统将提示您输入硬件令牌的密码。标准输出类似于以下内容：

```
Enter Password or Pin for "mytoken":
mytoken:Server-Cert                                          &#9;u,u,u
```

使用私钥和证书

可以使用 `certutil` 创建自签名证书以及导入或导出证书。要导入或导出私钥，请使用 `pk12util` 实用程序。有关更多详细信息，请参见第 101 页中的 “使用网络安全服务 (NSS) 工具”。

注意 – 在 Application Server 中，请勿直接使用 NSS 工具 `certutil` 和 `modutil` 修改 NSS 密码。如果这样做，Application Server 中的安全性数据可能会被破坏。

配置 J2SE 5.0 PKCS#11 提供者

Application Server 依靠 J2SE PKCS#11 提供者在运行时访问位于 PKCS#11 令牌中的密钥和证书。默认情况下，Application Server 将针对 NSS 软令牌来配置 J2SE PKCS#11 提供者。本节介绍如何覆盖 J2SE PKCS#11 提供者的默认配置。

在 Application Server 中，将针对每个 PKCS#11 令牌生成以下默认 PKCS#11 配置参数。

- 默认 NSS 软令牌的配置：

```
name=internal
library=${com.sun.enterprise.nss.softokenLib}
nssArgs="configdir='${com.sun.appserv.nss.db}'
 certPrefix='' keyPrefix='' secmod='secmod.db'"
slot=2
omitInitialize = true
```

- SCA 1000 硬件加速器的配置：

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
omitInitialize=true
```

这些配置符合《Java PKCS#11 Reference Guide》中所述的语法。

注 – 除了必须唯一之外，对名称参数没有任何要求。某些早期版本的 J2SE 5.0 仅支持字母数字字符。

您可以通过创建自定义配置文件来覆盖默认配置参数。例如，可以在 SCA–1000 中明确禁用 RSA 加密器和 RSA 密钥对生成器。有关禁用 RSA 加密器和 RSA 密钥对生成器的详细信息，请参见 `http://www.mozilla.org/projects/security/pki/nss/tools`。

要创建自定义配置文件，请执行以下操作：

1. 使用以下代码创建名为 *install-dir*`/mypkcs11.cfg` 的配置文件并保存此文件。

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
disabledMechanisms = {
&#9;CKM_RSA_PKCS
&#9;CKM_RSA_PKCS_KEY_PAIR_GEN
}
omitInitialize=true
```

2. 如果有必要，请更新 NSS 数据库。在这种情况下，更新 NSS 数据库以便它禁用 RSA。

   运行以下命令：

   ```
   modutil -undefault "Sun Crypto Accelerator" -dbdir AS_NSS_DB -mechanisms RSA
   ```

   `mechanisms` 列表中的算法名称与默认配置中的算法名称不同。有关 NSS 中有效 `mechanisms` 的列表，请参见 NSS 安全性工具站点 `http://www.mozilla.org/projects/security/pki/nss/tools` 上的 `modutil` 文档。

3. 通过在适当位置添加属性使用该更改来更新服务器，如下所示：

```
&lt;property name="mytoken" value="&InstallDir;/mypkcs11.cfg"/>
```

此属性的位置可以为以下位置之一：

- 如果提供者用于 DAS 或服务器实例，请在相关的 `<security-service>` 下添加此属性。
- 如果提供者用于节点代理，请在 `domain.xml` 文件中相关的 `<node-agent>` 元素下添加此属性。

4. 重新启动 Application Server。

   自定义配置将在重新启动后生效。

更多信息

- Java 2 Standard Edition 的安全性讨论，可以在 `http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html` 中查看到。
- 《The *J2EE 1.4 Tutorial*》的“*Security*”一章，可以在 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html` 中查看到。
- 本管理指南的第 10 章。
- Developer's Guide 中的 *"Securing Applications"* 一章。

◆ ◆ ◆ 第 10 章

10 配置消息安全性

本章中的某些内容假定您已对安全性和 Web 服务概念有了基本的了解。要详细了解这些概念，请在开始阅读本章之前先仔细阅读第 110 页中的 “更多信息”中列出的资源。

本章介绍如何在 Application Server 中为 Web 服务配置消息层的安全性。本章包含以下主题：

消息安全性概述

在**消息安全性**中，安全性信息被插入到消息中以使其通过网络层传输，并和消息一起到达消息目的地。消息安全性与传输层安全性不同（在《The J2EE 1.4 Tutorial》的“Security”一章中有说明），这是因为消息安全性可用于将消息保护从消息传输中分离开，从而使消息在传输后仍保持被保护状态。

Web 服务安全性：SOAP 消息安全性 (WS-Security) 是可交互使用的 Web 服务安全性的国际标准，是在 OASIS 中由所有 Web 服务技术的主要提供商（包括 Sun Microsystems）协作开发的。WS-Security 是一种消息安全性机制，它使用 XML 加密和 XML 数字签名来确保 SOAP 上发送的 Web 服务消息的安全。WS-Security 规范定义了多种安全令牌（包括 X.509 证书、SAML 断言和用户名/密码令牌）的用途，以验证和加密 SOAP Web 服务消息。

可以在 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf 中查看 WS-Security 规范。

了解 Application Server 中的消息安全性

Application Server 将对 WS-Security 标准的支持集成至其 Web 服务客户机和服务器端容器。集成此功能，可以使 Web 服务安全性由代表应用程序的 Application Server 的容器强制执行，并且可以将其应用于保护任何 Web 服务应用程序而无需对应用程序的实现进行更改。Application Server 通过提供工具将 SOAP 层消息安全性提供者和消息保护策略绑定到容器和容器中部署的应用程序来达到该效果。

指定消息安全性职责

在 Application Server 中，第 112 页中的 “系统管理员”和第 112 页中的 “应用程序部署者”角色应承担配置消息安全性的主要责任。尽管通常情况下，其他两个角色中的任意一个在无需更改现有应用程序实现的情况下就可以确保应用程序的安全而不必涉及开发者，但在某些情况下，第 113 页中的 “应用程序开发者”还是会有所作用的。以下小节定义了各种角色的职责：

系统管理员

系统管理员负责：

- 在 Application Server 中配置消息安全性提供者。
- 管理用户数据库。
- 管理密钥库和信任存储文件。
- 在使用加密并且运行 1.5.0 版之前的 Java SDK 时，配置 Java 加密扩展 (JCE) 提供者。
- 安装样例服务器。仅在 `xms` 样例应用程序用于演示消息层 Web 服务安全性的用途时，才执行此操作。

系统管理员使用管理控制台来管理服务器安全性设置，并使用命令行工具来管理证书数据库。在平台版中，证书和专用密钥存储在密钥库中，并通过 `keytool` 进行管理。标准版和企业版将证书和专用密钥存储在 NSS 数据库中，并使用 `certutil` 进行管理。本文档主要针对系统管理员。有关消息安全性任务的概述，请参见第 116 页中的 “为消息安全性配置 Application Server ”。

应用程序部署者

应用程序部署者负责：

- 在上游角色（开发者或汇编者）尚未指定任何所需的特定于应用程序的消息保护策略时，指定这些策略（在应用程序汇编时）。
- 修改特定于 Sun 的部署描述符来为 Web 服务端点和服务引用指定特定于应用程序的消息保护策略信息（即 message-security-binding 元素）。

《Developers' Guide》 的"Securing Applications"一章中讨论了这些安全性任务。有关指向该章的链接，请参见第 110 页中的 "更多信息"。

应用程序开发者

应用程序开发者可以打开消息安全性，但并不是必须要这样做。消息安全性可以由系统管理员设置，从而确保所有 Web 服务的安全；或者由应用程序部署者设置，这适用于绑定到应用程序的提供者或保护策略与绑定到容器的提供者或保护策略不同的情况。

应用程序开发者或汇编者负责：

- 确定应用程序是否需要特定于应用程序的消息保护策略。如果需要，则确保所需策略在应用程序汇编时指定，这可以通过与应用程序部署者沟通来完成。

关于安全令牌和安全机制

WS-Security 规范为使用安全令牌来验证和加密 SOAP Web 服务消息提供了可扩展机制。可以使用与 Application Server 一起安装的 SOAP 层消息安全性提供者来利用用户名/密码和 X.509 证书安全性令牌来对 SOAP Web 服务消息进行验证和加密。利用其他安全令牌（包括 SAML 断言）的其他提供者将与 Application Server 的后续版本一起安装。

关于用户名令牌

Application Server 使用 SOAP 消息中的**用户名令牌**来建立消息**发件人**的验证标识。包含用户名令牌（在嵌入式密码中）的消息的收件人通过确认发件人是否知道用户的秘密（密码），来验证消息发件人是否为经过授权的用户（在令牌中标识）。

使用用户名令牌时，必须在 Application Server 中配置有效的用户数据库。

关于数字签名

Application Server 使用 XML 数字签名将验证标识绑定到消息**内容**中。客户机使用数字签名来建立它们的呼叫者标识，其方法与使用传输层安全性时用基本验证或 SSL 客户机证书验证建立呼叫者标识的方法相似。消息收件人将检验数字签名以验证消息内容的源（可能与消息的发件人不同）。

使用数字签名时，必须在 Application Server 中配置有效的密钥库和信任库文件。有关此主题的更多信息，请参见第 97 页中的 "关于证书文件"。

关于加密

加密的目的是将数据修改为只有目标读者能理解的形式。加密过程通过用加密元素代替原始内容来完成。像公共密钥加密指出的那样，可以使用加密来建立能够阅读消息的一方或多方的标识。

使用加密时，必须先安装支持加密的 JCE 提供者。有关此主题的更多信息，请参见第 118 页中的 “配置 JCE 提供者”。

关于消息保护策略

消息保护策略是针对请求消息处理和响应消息处理而定义的，并根据对源和/或收件人验证的要求来进行表达。源验证策略代表一个请求，即在消息中建立发送了消息或定义了消息内容的实体的标识，以使其可以由消息收件人进行验证。收件人验证策略代表一个请求，即发送消息，以使可以接收消息的实体的标识可以由消息发件人建立。提供者将应用特定的消息安全性机制以使消息保护策略在 SOAP Web 服务消息的上下文中实现。

在给容器配置提供者时，将定义请求和响应的消息保护策略。特定于应用程序的消息保护策略（以 Web 服务端口或操作的粒度级别）也可以在应用程序或应用程序客户机端的特定于 Sun 的部署描述符中进行配置。在任何情况下，如果定义了消息保护策略，则客户机请求和响应的消息保护策略必须与服务器请求和响应的消息保护策略相匹配（二者相当）。有关定义特定于应用程序的消息保护策略的更多信息，请参见《Developer’ s Guide》的“Securing Applications”一章。

消息安全性术语表

以下介绍了此文档中所用到的术语。此外，第 116 页中的 “为消息安全性配置 Application Server ”中还讨论了相关概念。

- 验证层

  验证层是必须执行验证处理的消息层。Application Server 在 SOAP 层强制执行 Web 服务消息安全性。

- 验证提供者

  在此版本的 Application Server 中，Application Server 调用**验证提供者**来处理 SOAP 消息层安全性。

  - **客户机端提供者**可以建立（通过签名或用户名/密码）请求消息的源标识和/或保护（通过加密）请求消息，从而使这些消息只能由其目标收件人查看。客户机端提供者还可以将其容器建立为收到的响应的已授权收件人（通过成功地解密），并验证响应中的密码或签名来验证与此响应相关联的源标识。在 Application Server 中配置的客户机端提供者可用于作为其他服务的客户机来保护服务器端组件（即 Servlet 和 EJB 组件）所发送的请求消息和所接收的响应消息。
  - **服务器端提供者**将其容器建立为收到的请求的已授权收件人（通过成功地解密），并验证请求中的密码或签名以验证与该请求相关联的源标识。服务器端提供者还将建立（通过签名或用户名/密码）响应消息的源标识和/或保护（通过加密）响应消息，以使这些消息只能由其目标收件人查看。**服务器端提供者**仅由服务器端容器调用。

- 默认服务器提供者

默认服务器提供者用于标识服务器提供者，系统将调用该服务器提供者以用于尚未绑定特定服务器提供者的任何应用程序。**默认服务器提供者**有时被称为**默认提供者**。

- 默认的客户机提供者

  默认客户机提供者用于标识客户机提供者，系统将调用该客户机提供者以用于尚未绑定特定客户机提供者的任何应用程序。

- 请求策略

  请求策略定义与验证提供者执行的请求处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

- 响应策略

  响应策略定义与验证提供者执行的响应处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

确保 Web 服务的安全

通过将 SOAP 层消息安全性提供者和消息保护策略绑定到部署有应用程序的容器或绑定到应用程序提供的 Web 服务端点，来确保在 Application Server 中部署的 Web 服务的安全。通过将 SOAP 层消息安全性提供者和消息保护策略绑定到客户机容器或绑定到由客户机应用程序声明的可移植服务引用，从而在 Application Server 的客户机端容器中配置 SOAP 层消息安全性功能。

安装了 Application Server 后，将在 Application Server 的客户机和服务器端容器中配置 SOAP 层消息安全性提供者，这些提供者可由容器或容器中部署的各个应用程序或客户机进行绑定使用。在安装过程中，这些提供者将配置简单的消息保护策略，即如果被绑定到容器或者绑定到容器中的应用程序或客户机，将导致所有请求和响应消息中的内容源由 XML 数字签名进行验证。

可以采用 Application Server 的管理界面来绑定现有提供者以便供 Application Server 的服务器端容器使用，修改由提供者强制执行的消息保护策略，或使用替代的消息保护策略来创建新的提供者配置。可以对应用程序客户机端容器的 SOAP 消息层安全性配置执行类似的管理操作，如第 121 页中的 “启用应用程序客户机端的消息安全性”中所定义。

默认情况下，Application Server 中的消息层安全性处于禁用状态。要为 Application Server 配置消息层安全性，请按照第 116 页中的 “为消息安全性配置 Application Server ”中列出的步骤进行操作。如果您要使 Web 服务安全性用于保护在 Application Server 上部署的所有 Web 服务应用程序，请按照第 120 页中的 “启用消息安全性提供者”中的步骤进行操作。

完成以上步骤（可能包括重新启动 Application Server）后，Web 服务安全性将应用到 Application Server 中部署的所有 Web 服务应用程序。

配置特定于应用程序的Web服务安全性

通过在应用程序的特定于 Sun 的部署描述符中定义 message-security-binding 元素，可以配置特定于应用程序的 Web 服务安全性功能（在应用程序汇编时）。这些 message-security-binding 元素用于将特定提供者或消息保护策略与 Web 服务端点或服务引用相关联，并可以被限定以使这些元素应用到相应端点或引用服务的特定端口或方法。

有关定义特定于应用程序的消息保护策略的更多信息，请参见《*Developer's Guide* 》的"Securing Applications"一章。第 110 页中的 "更多信息"中有指向该章的链接。

确保样例应用程序的安全

Application Server 附带了名为 `xms` 的样例应用程序。`xms` 应用程序具有简单的 Web 服务功能，它由 J2EE EJB 端点和 Java Servlet 端点共同实现。这两个端点共享同一个服务端点接口。服务端点接口定义了单个操作 `sayHello`，此操作将获取一个字符串参数，并将由前置 `Hello` 组成的 `String` 返回到调用参数中。

提供的 `xms` 样例应用程序用来演示 Application Server 的 WS-Security 功能的用途，以确保现有 Web 服务应用程序的安全。样例附带的说明介绍了如何启用 Application Server 的 WS-Security 功能以使其用于确保 `xms` 应用程序的安全。样例还演示如何向应用程序直接绑定 WS-Security 功能（如第 116 页中的 "配置特定于应用程序的 Web 服务安全性"中所述）。

`xms` 样例应用程序安装在以下目录中：*install-dir*`/samples/webservices/security/ejb/apps/xms/`。

有关编译、封装和运行 `xms` 样例应用程序的信息，请参见《Developers' Guide》的"Securing Applications"一章。

为消息安全性配置 Application Server

请求策略配置和响应策略配置的操作

下表显示了消息保护策略配置以及由 WS-Security SOAP 消息安全性提供者针对该配置所执行的最终消息安全性操作。

表 10–1 消息保护策略与 WS-Security SOAP 消息安全性操作的映射

消息保护策略	最终的 WS-Security SOAP 消息保护操作
auth-source="sender"	此消息包含 wsse:Security 标头，此标头包含 wsse:UsernameToken（带有密码）。
auth-source="content"	对 SOAP 消息主体的内容进行签名。此消息包含 wsse:Security，此标头包含显示为 ds:Signature 的消息主体签名。
auth-source="sender" auth-recipient="before-content" 或 auth-recipient="after-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。此消息包含 a wsse:Security 标头，此标头包含 wsse:UsernameToken（带有密码）和 xenc:EncryptedKey。xenc:EncryptedKey 包含用于对 SOAP 消息主体进行加密的密钥。此密钥在收件人的公共密钥中加密。
auth-source="content" auth-recipient="before-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。对 xenc:EncryptedData 进行签名。此消息包含 a wsse:Security 标头，此标头包含 xenc:EncryptedKey 和 ds:Signature。xenc:EncryptedKey 包含用于对 SOAP 消息主体进行加密的密钥。此密钥在收件人的公共密钥中加密。
auth-source="content" auth-recipient="after-content"	对 SOAP 消息主体的内容进行签名、加密，并用最终的 xend:EncryptedData 替换。此消息包含 wsse:Security 标头，此标头包含 xenc:EncryptedKey 和 ds:Signature。xenc:EncryptedKey 包含用于对 SOAP 消息主体进行加密的密钥。此密钥在收件人的公共密钥中加密。
auth-recipient="before-content" 或 auth-recipient="after-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。此消息包含 a wsse:Security 标头，此标头包含 xen:EncryptedKey。xenc:EncryptedKey 包含用于对 SOAP 消息主体进行加密的密钥。此密钥在收件人的公共密钥中加密。
未指定策略。	模块未执行任何安全操作。

配置其他安全性工具

Application Server 使用在其 SOAP 处理层中集成的消息安全性提供者来实现消息安全性。消息安全性提供者取决于 Application Server 的其他安全性工具。

1. 如果使用的 Java SDK 的版本早于 1.5.0，而且使用了加密技术，请配置 JCE 提供者。
2. 第 118 页中的 “配置 JCE 提供者”中说明了如何配置 JCE 提供者。

3. 如果使用了用户名令牌，请在必要时配置用户数据库。使用用户名/密码令牌时，必须配置适当的区域并为该区域配置适当的用户数据库。
4. 管理证书和专用密钥（如有必要）。

完成之后

将 Application Server 的工具配置为供消息安全性提供者使用后，则可启用随 Application Server 一起安装的提供者，如第 120 页中的 “启用消息安全性提供者”中所述。

配置 JCE 提供者

J2SE 1.4.x 中包含的 Java 加密扩展 (JCE) 提供者不支持RSA 加密。由于由 WS-Security 所定义的 XML 加密通常是基于 RSA 加密，因此为了使用 WS-Security 来加密 SOAP 消息，您必须下载并安装支持 RSA 加密的 JCE 提供者。

注 – RSA 是 RSA Data Security, Inc. 开发的公共密钥加密技术。RSA 缩略词分别代表该技术的三位发明者：Rivest、Shamir 和 Adelman。

如果是在 1.5 版的 Java SDK 中运行 Application Server，则已正确配置了 JCE 提供者。如果是在 1.4.x 版的 Java SDK 中运行 Application Server，您可以静态方式将 JCE 提供者添加为 JDK 环境的一部分，如下所示。

1. 下载并安装 JCE 提供者 JAR（Java 归档）文件。

   通过以下 URL 可以获得支持 RSA 加密的 JCE 提供者的列表：`http://java.sun.com/products/jce/jce14_providers.html`。
2. 将 JCE 提供者 JAR 文件复制到 *java-home*`/jre/lib/ext/` 中。
3. 停止 Application Server。

   如果未停止 Application Server 而后来又在这一过程中重新启动了 Application Server，则 Application Server 将无法识别 JCE 提供者。
4. 在任何一种文本编辑器中编辑 *java-home*`/jre/lib/security/java.security` 属性文件。将刚才下载的 JCE 提供者添加到此文件。

   `java.security` 文件包含添加此提供者的详细说明。通常，您需要在具有类似属性的某个位置处添加一行，其格式如下：

   `security.provider.`*n*`=`*provider-class-name*

   在本示例中，*n* 是 Application Server 评估安全性提供者时使用的优先级顺序。为刚才添加的 JCE 提供者将 *n* 设置为 2。

   例如，如果您下载的是 Legion of the Bouncy Castle JCE 提供者，则应添加以下行。

```
security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
```

确保将 Sun 安全性提供者保持在最高优先级，值为 1。

```
security.provider.1=sun.security.provider.Sun
```

将其他安全性提供者的优先级调低，从而使每个优先级上只有一个安全性提供者。

以下是提供所需 JCE 提供者并将现有提供者保持在正确位置的 `java.security` 文件的示例。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsajca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

5. 保存并关闭文件。
6. 重新启动 Application Server。

消息安全性设置

为使用消息安全性而对 Application Server 进行设置的大部分步骤都可以通过使用管理控制台、`asadmin` 命令行工具或通过手动编辑系统文件来完成。通常，建议不要通过编辑系统文件来完成，因为它可能会导致出现无意的更改而使 Application Server 不能正常运行，所以，如有可能，建议优先使用管理控制台来配置 Application Server，其次选用 `asadmin` 工具命令。仅在无管理控制台或等效的 `asadmin` 时，才通过手动编辑系统文件来完成。

对消息层安全性的支持以（可插入）验证模块的形式集成到 Application Server 及其客户机容器中。默认情况下，Application Server 中的消息层安全性处于禁用状态。以下各节提供了启用、创建、编辑和删除消息安全性配置和提供者的详细信息。

- 第 120 页中的 “启用消息安全性提供者”
- 第 120 页中的 “配置消息安全性提供者”
- 第 121 页中的 “创建消息安全性提供者”
- 第 121 页中的 “启用应用程序客户机端的消息安全性”
- 第 121 页中的 “设置应用程序客户机端配置的请求策略和响应策略”
- 第 122 页中的 “更多信息”

大多数情况下，在执行以上列出的管理操作后应重新启动 Application Server。它尤其适用于当执行完操作时，您要将管理更改的效果应用到 Application Server 上已部署的应用程序中的情况。

启用消息安全性提供者

要为在 Application Server 中部署的 Web 服务端点启用消息安全性，必须指定要在服务器端默认使用的提供者。如果为消息安全性启用了默认提供者，您还需要启用要由 Application Server 中部署的 Web 服务客户机所使用的提供者。第 121 页中的 “启用应用程序客户机端的消息安全性”中介绍了有关启用客户机使用的提供者的信息。

要为源于已部署的端点的 Web 服务调用启用消息安全性，您必须指定默认客户机提供者。如果已为 Application Server 启用了默认客户机提供者，则必须确保从 Application Server 中部署的端点中调用的所有服务均已配置为与消息层安全性兼容。

使用命令行实用程序：

- 要指定默认的服务器提供者，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- 要指定默认的客户机提供者，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

配置消息安全性提供者

通常，应重新配置提供者以修改其消息保护策略，当然提供者类型、实现类和特定于提供者的配置属性可能也需要修改。

可以使用命令行实用程序设置响应策略，将以下命令中的文字 `request` 替换为 `response`。

- 要将请求策略添加到客户机并设置验证源，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- 要将请求策略添加到服务器并设置验证源，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- 要将请求策略添加到客户机并设置验证收件人，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- 要将请求策略添加到服务器并设置验证收件人，请使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

创建消息安全性提供者

要使用管理控制台配置现有提供者，请选择“配置”节点>要配置的实例>“安全性”节点>“消息安全性”节点>"SOAP" 节点>“提供者”选项卡。

有关创建消息安全性提供者的更多详细说明，请参见管理控制台联机帮助。

启用应用程序客户机端的消息安全性

必须配置客户机提供者的消息保护策略，以使其等效于将与其进行交互的服务器端提供者的消息保护策略。在安装 Application Server 时已配置（但未启用）的提供者已符合此情况。

要启用客户机应用程序的消息安全性，请修改应用程序客户机端容器特定于 Application Server 的配置。

设置应用程序客户机端配置的请求策略和响应策略

请求策略和响应策略定义与验证提供者执行的请求处理和响应处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

要获得消息安全性，必须既在服务器中也在客户机中启用请求策略和响应策略。在客户机和服务器中配置策略时，请确保请求/响应应用程序级别消息绑定的保护的客户机策略与服务器策略匹配。

要设置应用程序客户机端配置的请求策略，请按第 121 页中的 “启用应用程序客户机端的消息安全性”中所述，修改应用程序客户机端容器特定于 Application Server 的配置。在应用程序客户机端配置文件中，按所示添加 `request-policy` 和 `response-policy` 元素设置请求策略。

提供的其他代码可用作参考。您的安装中的其他代码可能会稍有不同。请勿对其进行更改。

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="install-dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

`auth-source` 的有效值包括 `sender` 和 `content`。`auth-recipient` 的有效值包括 `before-content` 和 `after-content`。第 116 页中的 “请求策略配置和响应策略配置的操作”中包含一个表，用于说明这些值的各种组合的结果。

如果不想指定请求策略或响应策略，请将该元素保留为空，例如：

```
<response-policy/>
```

更多信息

- Java 2 Standard Edition 的安全性讨论，可以在 `http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html` 中查看到。
- 《The *J2EE 1.4 Tutorial*》的“*Security*”一章，可以在 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html` 中查看到。
- 本管理指南的第 9 章。
- 《Developer’s Guide》的“Securing Applications”一章。
- Oasis Web 服务安全性：SOAP 消息安全性 (WS-Security) 规范，可以在 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message -security-1.0.pdf 中查看到。
- OASIS Web Services Security Username Token Profile 1.0，可以在 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token -profile-1.0.pdf 中查看到。
- OASIS Web Services Security X.509 Certificate Token Profile 1.0，可以在 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token -profile-1.0.pdf 中查看到。

- 《XML-Signature Syntax and Processing》文档，可在 http://www.w3.org/TR/xmldsig-core/ 中查看到。
- 《XML Encryption Syntax and Processing》文档，可在 http://www.w3.org/TR/xmlenc-core/ 中查看到。

◆ ◆ ◆ 第 11 章

11 事务

通过将一个或多个步骤包含在不可分的工作单元，事务可确保数据的完整性和一致性。本章包含以下各节：

什么叫事务？

事务是应用程序中一系列严密的操作，所有操作必须成功完成，否则在每个操作中所做的所有更改都会被撤消。例如，将资金从支票帐户转到储蓄帐户中是一项事务，按步骤如下进行：

1. 检查支票帐户是否有足够的资金来支付此转帐操作。
2. 如果支票帐户中有足够的资金，则将该笔资金记入此帐户的借方。
3. 将这些资金记入储蓄帐户的贷方。
4. 将此次转帐记录到支票帐户日志中。
5. 将此次转帐记录到储蓄帐户日志中。

如果这些步骤的任何一个步骤失败，则必须撤消在前面的步骤中所做的所有更改，而且支票帐户和储蓄帐户的状态必须与它们在事务开始之前的状态相同。该事件称为**回滚**。如果所有步骤均成功完成，那么事务即处于**已提交**状态。事务以提交或回滚结束。

J2EE 技术中的事务

J2EE 技术中的事务处理包括以下五个参与者：

- 事务管理器
- Application Server
- 资源管理器
- 资源适配器
- 用户应用程序。

通过实现各自的 API 和功能，每个实体均有助于提高事务处理的可靠性，如下所述：

- 事务管理器提供了支持事务划分、事务资源管理、同步和事务上下文传播所需的服务和管理功能。
- Application Server 提供了支持应用程序运行时环境（包含事务状态管理）所需的基础结构。
- 应用程序可以使用资源管理器（通过资源适配器）访问资源。资源管理器通过实现事务管理器用来就事务关联、事务完成及恢复工作进行通信的事务资源接口来参与分布式事务。例如，关系型数据库服务器就是这样的资源管理器。
- 资源适配器是一个系统级的软件库，应用服务器或客户机可使用它连接到资源管理器。资源适配器通常专用于资源管理器。它以库的形式存在，并在使用它的客户机地址空间中使用。例如，JDBC 驱动程序就是这样的资源适配器。
- 为在应用服务器环境中运行而开发的事务用户应用程序可以使用 JNDI 查找事务数据源及事务管理器（可选）。应用程序可使用 Enterprise Bean 的声明事务属性设置或明确的程序事务划分。

恢复事务

由于服务器崩溃或资源管理器崩溃，事务可能未完成。完成这些被搁置的事务并将其从故障中恢复至关重要。Application Server 旨在在服务器启动时从故障中恢复并完成这些事务。

执行恢复操作时，如果无法访问某些资源，则服务器重新启动操作可能被延迟，因为服务器正在尝试恢复事务。

如果事务跨服务器进行，启动该事务的服务器会联系其他服务器以获得事务的结果。如果无法访问其他服务器，则该事务将使用“试探性决定”字段来确定结果。

您可以使用管理控制台配置 Application Server 以恢复事务。有关执行此操作的详细过程，请参见管理控制台联机帮助。

事务超时值

默认情况下，服务器不会使事务超时。即，服务器无限期地等待事务完成。如果为事务设置了超时值，而事务在配置的时间内未完成，则 Application Server 将回滚此事务。有关执行此操作的详细步骤，请参见管理控制台联机帮助。

事务日志

为了保持被调用资源的数据完整性，同时为了能够从故障中恢复，事务日志将记录有关每个事务的信息。事务日志保存在“事务日志位置”字段指定的目录的 tx 子目录中。用户无法读取这些日志。

密钥点间隔

密钥点操作将压缩事务日志文件。密钥点间隔是日志中密钥点操作之间的事务数量。密钥点操作可以减小事务日志文件的大小。密钥点间隔数越大（例如，2048），事务日志文件也越大，但密钥点操作较少，性能可能更佳。密钥点间隔越小（例如，256），日志文件也越小，而同时由于密钥点操作较为频繁，性能会略微降低。

◆ ◆ ◆ 第 12 章

12 配置 HTTP 服务

HTTP 服务是 Application Server 的一个组件，它提供了用于部署 Web 应用程序的工具和使 HTTP 客户机可以访问已部署 Web 应用程序的工具。HTTP 服务借助于两种相关对象（虚拟服务器和 HTTP 侦听器）来提供这些工具。

本章讨论以下主题：

虚拟服务器

虚拟服务器（有时也称为虚拟主机）是一个允许同一个物理服务器来托管多个域名的对象。同一个物理服务器上托管的所有虚拟服务器共享该物理服务器的 Internet 协议 (Internet Protocol, IP) 地址。虚拟服务器将某个服务器的域名（例如 `www.aaa.com`）与运行 Application Server 的特定服务器关联起来。

注 – 请勿将 Internet 域与 Application Server 的管理域相混淆。

例如，假设您希望在物理服务器上托管以下这些域：

```
www.aaa.com
www.bbb.com
www.ccc.com
```

同时假设 `www.aaa.com`、`www.bbb.com` 和 `www.ccc.com` 分别将 Web 模块 `web1`、`web2` 和 `web3` 与之关联起来。

这意味着以下 URL 将全部由您的物理服务器处理：

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

第一个 URL 将被映射到虚拟主机 `www.aaa.com`，第二个 URL 将被映射到虚拟主机 `www.bbb.com`，第三个 URL 将被映射到虚拟主机 `www.ccc.com`。

另一方面，由于未向 `www.bbb.com` 注册 `web3`，以下 URL 将导致 404 返回码：

```
http://www.bbb.com:8080/web3
```

要使此映射有效，请确保 `www.aaa.com`、`www.bbb.com` 和 `www.ccc.com` 均可解析为物理服务器的 IP 地址。需要向您网络的 DNS 服务器注册这些域名。此外，在 UNIX 系统上，应将这些域添加到 `/etc/hosts` 文件中（如果 `/etc/nsswitch.conf` 文件中的 `hosts` 设置包括 `files`）。

Application Server 启动时，将自动启动以下虚拟服务器：

- 名为 `server` 的虚拟服务器，托管所有用户定义的 Web 模块。
- 名为 `__asadmin` 的虚拟服务器，托管所有与管理相关的 Web 模块（特别是管理控制台）。该服务器是一个受限制的服务器，您不能将 Web 模块部署到该虚拟服务器上。

如果是在非生产环境中开发、测试和部署 Web 服务，通常只需使用 `server` 虚拟服务器。在生产环境中，其他虚拟服务器可以为用户和客户提供托管工具，这样，尽管只有一个物理服务器，但每个用户和客户好像都有自己的服务器。

HTTP 侦听器

每个虚拟服务器都通过一个或多个 HTTP 侦听器来提供服务器与客户机之间的连接。每个 HTTP 侦听器都是包含 IP 地址、端口号、服务器名和默认虚拟服务器的侦听套接字。

HTTP 侦听器必须有唯一的端口号和 IP 地址的组合。例如，HTTP 侦听器可以通过将 IP 地址指定为 0.0.0.0，在计算机的给定端口侦听所有已配置的 IP 地址。HTTP 侦听器还可以为每个侦听器指定唯一的 IP 地址，但使用同一个端口。

由于 HTTP 侦听器是 IP 地址和端口号的组合，因此您可以拥有多个 IP 地址相同而端口号不同的 HTTP 侦听器（例如，1.1.1.1:8081 和 1.1.1.1:8082），或 IP 地址不同而端口号相同的 HTTP 侦听器（例如，1.1.1.1:8081 和 1.2.3.4:8081，如果已将计算机配置为可以响应这两个地址）。

但是，如果 侦听器使用 0.0.0.0 IP 地址侦听某个端口上的所有 IP 地址，则不能使用其他 IP 地址创建用于侦听同一端口上的特定 IP 地址的 HTTP 侦听器。例如，如果 HTTP 侦听器使用 0.0.0.0:8080（端口 8080 上的所有 IP 地址），则其他 HTTP 侦听器不能使用 1.2.3.4:8080。

由于运行 Application Server 的系统通常只能访问一个 IP 地址，因此 HTTP 侦听器通常使用 0.0.0.0 IP 地址和不同的端口号，其中每个端口号用于不同目的。如果系统可以访问多个 IP 地址，则每个地址可以用于不同目的。

默认情况下，Application Server 启动时，具有以下 HTTP 侦听器：

- 两个分别名为 `http-listener-1` 和 `http-listener-2` 的 HTTP 侦听器，这两个侦听器与名为 `server` 的虚拟服务器相关联。名为 `http-listener-1` 的侦听器未启用安全性；而名为 `http-listener-2` 的侦听器启用了安全性。
- 名为 `admin-listener` 的 HTTP 侦听器，该侦听器与名为 `__asadmin` 的虚拟服务器相关联。该侦听器已启用安全性。

所有这些侦听器均使用 IP 地址 0.0.0.0 和在安装 Application Server 过程中被指定为 HTTP 服务器端口号的端口号。如果 Application Server 使用默认端口号值，则 `http-listener-1` 使用端口 8080，`http-listener-2` 使用端口 8181，`admin-listener` 使用端口 4849。

每个 HTTP 侦听器都有一个默认虚拟服务器。当请求 URL 的主机组件与任何 HTTP 侦听器关联虚拟服务器（在虚拟服务器的 `http-listeners` 属性中列出 HTTP 侦听器，即可将虚拟服务器与该 HTTP 侦听器相关联）均不匹配时，HTTP 侦听器会将所有请求 URL 路由到默认虚拟服务器。

此外，还应指定 HTTP 侦听器中的接收器线程数。接收器线程就是等待连接的线程。它们用于接受连接并将其置于队列（称为连接队列）中，以便随后由工作线程拾取。您需要配置足够多的接收器线程，以便在新请求传入时始终有一个可用的线程，但是，线程数目不能过多，否则会占用过多的系统资源。连接队列中既包括接收器线程刚刚接受的新连接，又包括由保持活动连接管理子系统管理的持久性连接。

一组请求处理线程将从连接队列中检索传入的 HTTP 请求并处理这些请求。这些线程将解析 HTTP 头、选择相应的虚拟服务器并通过请求处理引擎以处理请求。如果没有更多要处理的请求，但连接可以保持持久性（通过使用 `HTTP/1.1` 或发送 `Connection: keep-alive` 标头），请求处理线程将假定连接处于空闲状态，并将连接传递给保持活动连接管理子系统。

保持活动子系统将定期轮询此类空闲连接，并将活动的连接排队到连接队列中，以便将来进行处理。请求处理线程将再次从连接队列中检索连接并处理其请求。保持活动子系统是多线程的，可以管理大约数万个连接。通过将大量连接分成较小的子集，使用有效的轮询技术来确定哪些连接已就绪并具有请求，以及哪些连接由于处于空闲状态的时间较长而被视为已关闭（超过允许的保持活动超时的最大值）。

HTTP 侦听器的服务器名就是在重定向过程中由服务器发送给客户机的 URL 中使用的主机名。此属性会影响服务器自动生成的 URL，但不会影响存储在服务器中的目录和文件的 URL。如果服务器使用别名，则此名称通常为别名。如果客户机发送了一个 `Host:` 标头，则在重定向过程中，该主机名将取代 HTTP 侦听器的服务器名值。

要使用不同于原始请求中指定的端口号的端口号，请指定重定向端口。如果出现以下情况之一，就会发生**重定向**：

- 如果客户机尝试访问指定 URL 处已不存在的资源（即该资源已被移动到其他位置），服务器将返回一个指定的响应代码，并在响应的位置标头中包含新的位置，从而将客户机重定向到新位置（而不是返回 404）。
- 如果客户机尝试通过常规 HTTP 端口访问受保护（如 SSL）的资源，则服务器会将此请求重定向到启用了 SSL 的端口。在这种情况下，服务器将在位置响应标头中返回一个新 URL，其中原始的不安全端口已被替换为启用了 SSL 的端口。客户机随后将连接到这个新的 URL。

此外，还应指定是否为 HTTP 侦听器启用安全性以及使用哪种类型的安全性（如使用哪一个 SSL 协议以及哪些加密算法）。

要访问部署在 Application Server 上的 Web 应用程序，请使用 URL `http://localhost:8080/`（或 `https://localhost:8181/`，如果此应用程序是安全的）以及为该 Web 应用程序指定的上下文根路径。要访问管理控制台，请使用 URL `https://localhost:4849/` 或 `https://localhost:4849/asadmin/`（其默认上下文根路径）。

由于虚拟服务器必须指定一个现有的 HTTP 侦听器，并且不能指定其他虚拟服务器已使用的 HTTP 侦听器，因此在创建新的虚拟服务器之前，应至少创建一个 HTTP 侦听器。

◆ ◆ ◆ 第 13 章

13 配置对象请求代理

本章介绍了如何配置对象请求代理 (Object Request Broker, ORB) 和 IIOP 侦听器。本章包含以下几节：

CORBA

Application Server 支持标准的协议和格式集来确保互操作性。这些协议之间的协议是由 CORBA 定义的。

CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）模型的基础是：客户机以远程方法请求形式向分布式对象或服务器发出请求，并通过明确定义的接口从这些对象那里请求服务。远程方法请求携带了有关需要执行的操作的信息，其中包括被调用方法的服务提供商的对象名称（称为对象引用）和参数（如果有）。CORBA 自动处理网络程序任务，如对象注册、对象定位、对象激活、请求多路复用、错误处理、编组和操作分发。

什么是 ORB？

对象请求代理 (Object Request Broker, ORB) 是 CORBA 的核心组件。ORB 提供了识别和定位对象、处理连接管理、传送数据和请求通信所需的框架结构。

CORBA 对象之间从不直接进行通信，对象通过远程桩对运行在本地计算机上的 ORB 发出请求。本地 ORB 使用 Internet Inter-Orb 协议（IIOP 为缩写形式）将该请求传递给其他计算机上的 ORB。然后，远程 ORB 定位相应的对象、处理该请求并返回结果。

使用 RMI-IIOP，应用程序或对象可将 IIOP 用作远程方法调用 (RMI) 协议。Enterprise Bean（EJB 模块）的远程客户机通过 RMI-IIOP 与 Application Server 进行通信。

IIOP 侦听器

IIOP 侦听器是一个侦听套接字，它接收来自 Enterprise Bean 的远程客户机和其他基于 CORBA 的客户机的外来连接。可以为 Application Server 配置多个 IIOP 侦听器。为每个侦听器指定一个端口号、一个网络地址和（可选）多个安全性属性。

使用 ORB

要创建 IIOP 侦听器，请在管理控制台中选择“配置”>"ORB">“IIOP 侦听器”，然后单击“新建”。或者，可以使用以下 asadmin 命令创建 IIOP 侦听器：`create-iiop-listener`(1) and `create-ssl`(1).

要编辑 IIOP 侦听器，请在管理控制台中选择“配置”>"ORB">“IIOP 侦听器”，然后选择要修改的侦听器。修改设置。如果更改了端口号，请重新启动服务器。ORB 使用线程池响应来自通过 RMI-IIOP 进行通信的 Enterprise Bean 的远程客户机和其他客户机的请求。

要删除 IIOP 侦听器，请在管理控制台中选择“配置”>"ORB">“IIOP 侦听器”，然后选择要删除的侦听器。或者，可以使用 `delete-iiop-listener`(1) 命令。

`ORBCommunicationsRetryTimeout` 属性指定 ORB 客户机尝试与不可访问的 ORB 后端建立连接所用的秒数。默认值为 `60` 秒。如果使用此默认设置，则当 ORB 后端不可访问时，您可能会看到日志中存在大量 CORBA 异常，以及很高的网络使用量。

在这种情况下，请将 `ORBCommunicationsRetryTimeout` 设置为较低的值。

第三方 ORB

Sun Java System Application Server 可以与第三方 ORB 软件结合使用。要支持此类第三方 ORB，您需要修改服务器端设置。

要在 Application Server 中实现对第三方 ORB 的支持，您需要编辑文件 `domain.xml` 和 `server.policy`。有关如何配置第三方 ORB 样例的详细说明，请参见 Configuring Sun Java System Application Server for Third-Party ORBs (`http://developers.sun.com/prodtech/appserver/reference/techart/orb.html`)。

◆ ◆ ◆ 第 14 章

14 线程池

Java 虚拟机 (Java Virtual Machine, JVM) 可以支持一次执行多个线程。为了提高性能，Application Server 维护一个或多个线程池。可以将特定的线程池指定给连接器模块和 ORB。

一个线程池可以提供多个连接器模块和企业 Bean。请求线程处理对应用程序组件的用户请求。服务器接收到请求时，它会将请求指定给线程池中的空闲线程。该线程执行客户机的请求并返回结果。例如，如果请求需要使用的系统资源当前正处于忙碌状态，则线程会在允许请求使用该资源前，等待资源回到空闲状态。

指定为来自应用程序的请求预留的最大线程数和最小线程数。线程池在这两个值之间动态调整。指定的最小线程池大小将通知服务器为应用程序请求至少分配该大小的预留线程数。可以将线程数增加到所指定的最大线程池大小。

如果增加可供进程使用的线程数，则该进程可以同时对更多的应用程序进行响应。

在一个资源适配器或应用程序占据 Application Server 中的所有线程时，通过将 Application Server 线程分至几个不同的线程池来避免线程资源缺乏。

本章包含以下主题：

- 第 135 页中的 “配置线程池”

配置线程池

要使用管理控制台创建线程池，请转至“配置”>“线程池”>“当前池”>“新建”。

- 在“线程池 ID”字段中输入线程池的名称。
- 在“最小线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最小数目。

  将此线程池实例化时将预先创建这些线程。
- 在“最大线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最大数目。

这是存在于此线程池中的线程数上限。

- 在“空闲超时”字段中输入数值（以秒为单位），超过此时间段之后将从池中删除空闲线程。
- 在“工作队列的数目”字段中输入由此线程池服务的工作队列总数。
- 重新启动 Application Server。

有关创建线程池的更多详细信息，请在管理控制台中单击“帮助”。

您也可以通过使用 `asadmin` 命令 `create-threadpool`(1) 从命令行中创建线程池。

要使用管理控制台编辑线程池设置，请转至“配置”>“线程池”>“当前池”，然后选择要配置的池。修改选定线程池的值并进行保存，然后重新启动 Application Server。

有关编辑线程池的更多详细信息，请在管理控制台中单击“帮助”。

要使用管理控制台删除线程池，请转至“配置”>“线程池”>“当前池”。选中要删除的线程池名称，然后单击“删除”。

重新启动 Application Server。

您也可以通过使用 `asadmin` 命令 `delete-threadpool`(1) 从命令行中删除线程池。

◆ ◆ ◆ 第 15 章

15 配置日志记录

本章简要介绍了如何配置日志记录和查看服务器日志。它包含以下各节：

日志记录

Application Server 使用 JSR 047 中指定的 Java 2 平台日志记录 API。Application Server 日志信息记录在服务器日志中，通常可以在 *domain-dir*`/logs/server.log` 中找到。

domain-dir`/logs` 目录中除了包含服务器日志外，还包含另外两种日志。`access` 子目录中包含 HTTP 服务访问日志，`tx` 子目录中包含事务服务日志。有关这些日志的信息，请查阅管理控制台联机帮助。

Application Server 组件生成日志记录输出。应用程序组件也可以生成日志记录输出。

应用程序组件可以使用 Apache Commons Logging Library 来记录消息。但是，建议采用平台标准 JSR 047 API 以获得更好的日志配置。

日志记录遵循以下统一格式：

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName-Version|LoggerName|Key Value Pairs|Message|#]
```

例如：

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-e8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

在本示例中，

- `[#` 和 `#]` 标记该记录的开始和结束。

- 垂直条 (|) 用于分隔记录字段。
- 2004-10-21T13:25:53.852-0400 指明了日期和时间。
- **日志级别**为 INFO。日志级别可以是以下任何值：SEVERE、WARNING、INFO、CONFIG、FINE、FINER 和 FINEST。
- **产品名称及版本**为 sun-appserver-ee8.1。
- **日志程序名称**是用于标识日志模块的来源的分层日志程序名称空间，在此示例中为 javax.enterprise.system.core。
- **关键字值对**为关键字名称和值，通常为线程 ID，如 _ThreadID=14;。
- **消息**是日志信息的文本。对于所有 Application Server SEVERE 和 WARNING 消息及多种 INFO 消息，其均以包含模块代码和数值的消息 ID 开头（在此示例中为 CORE5004）。

在以后的版本中，可能会更改或增强日志记录格式。

设置自定义日志级别

本节介绍了如何为使用 java.util.logging 包并访问 Application Server 的日志记录子系统的应用程序配置自定义日志记录级别。

java.util.logging 包提供了分层名称空间，可以在该名称空间中创建日志程序实例。是否将特定日志记录输出到 Application Server 实例的日志文件中取决于日志记录的日志级别以及指定的日志级别。

Application Server 日志程序设置配置可提供二十多种日志记录模块，允许对 Application Server 自身的内部日志记录进行精细控制。还有一个选项可用于创建其他自定义日志模块，它通过指定模块名称和该模块应使用的日志记录级别来创建。

其中重要的一点是日志程序是静态名称，且不提供继承性。因此，如果为自定义日志程序配置了名称 com.someorg.app，在某个应用程序尝试查找日志程序 com.someorg.app.submodule 时，将不会为该应用程序提供继承了 com.someorg.app 中设置的日志程序。相反，com.someorg.app.submodule 将具有日志级别设置为 INFO 或更高级别的默认日志程序。

如果应用程序需要使用日志程序继承，则可以通过编辑用于运行 Application Server 的 Java 运行时的 logging.properties 文件来进行配置。例如，如果将以下条目添加到 logging.properties 文件中，则会导致 com.someorg.app.submodule 在创建时继承相同的 FINE 级别：

```
com.someorg.app.level = FINE
```

有关 Java 日志记录 API 的更多详细信息，请参阅 http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/package-summary.html 中的 Java 文档以及其他 java.util.logging 类。

日志程序名称空间分层结构

Application Server 为它的每个模块都提供了日志程序。下表按照每个日志程序的模块名称和名称空间在管理控制台的“日志级别”页面中的显示方式以字母顺序列出每个日志程序的模块名称和名称空间。“日志级别”页面中未显示表中最后三个模块。

表 15–1 Application Server 日志程序名称空间

模块名称	名称空间
管理	`javax.enterprise.system.tools.admin`
类加载器	`javax.enterprise.system.core.classloading`
CMP	`javax.enterprise.system.container.cmp`
配置	`javax.enterprise.system.core.config`
连接器	`javax.enterprise.resource.resourceadapter`
CORBA	`javax.enterprise.resource.corba`
部署	`javax.enterprise.system.tools.deployment`
EJB 容器	`javax.enterprise.system.container.ejb`
JavaMail	`javax.enterprise.resource.javamail`
JAXR	`javax.enterprise.resource.webservices .registry`
JAX-RPC	`javax.enterprise.resource.webservices.rpc`
JDO	`javax.enterprise.resource.jdo`
JMS	`javax.enterprise.resource.jms`
JTA	`javax.enterprise.resource.jta`
JTS	`javax.enterprise.system.core.transaction`
MDB 容器	`javax.enterprise.system.container.ejb.mdb`
命名	`javax.enterprise.system.core.naming`
节点代理（仅限于 Enterprise Edition）	`javax.ee.enterprise.system.nodeagent`
根目录	`javax.enterprise`
SAAJ	`javax.enterprise.resource.webservices.saaj`
安全性	`javax.enterprise.system.core.security`
服务器	`javax.enterprise.system`
同步（仅限于 Enterprise Edition）	`javax.ee.enterprise.system.tools .synchronization`

表 15–1 Application Server 日志程序名称空间 （续）

模块名称	名称空间
实用程序	javax.enterprise.system.util
验证器	javax.enterprise.system.tools.verifier
Web 容器	javax.enterprise.system.container.web
核心	javax.enterprise.system.core
System Output (System.out.println)	javax.enterprise.system.stream.out
System Error (System.err.println)	javax.enterprise.system.stream.err

◆ ◆ ◆ 第 16 章

16 监视组件和服务

使用监视功能可以观察 Application Server 的服务器实例中所部署的各种组件和服务的运行状态。利用有关运行时组件和进程状态的信息，可以确定性能瓶颈以便进行优化、有助于进行容量规划、预测故障、在发生故障时分析根本原因，以及确保一切运行正常。

启用监视功能会因增加系统开销而使性能降低。

本章包括以下几个部分：

监视概述

要监视 Application Server，请执行以下步骤：

1. 使用管理控制台或 `asadmin` 工具来启用对特定服务和组件的监视功能。
   有关此步骤的更多信息，请参阅第 164 页中的 “启用和禁用监视功能”。
2. 使用管理控制台或 `asadmin` 工具来查看指定服务或组件的监视数据。
   有关此步骤的更多信息，请参阅第 165 页中的 “查看监视数据”。

关于可监视对象的树结构

Application Server 使用树结构来跟踪可监视对象。由于监视对象的树是动态的，因此在实例中添加、更新或删除组件时该树会相应地发生变化。树的根对象为服务器实例名称，例如 `server`。（在平台版中，仅允许使用一个服务器实例。）

以下命令显示了树的顶层：

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

以下各节介绍了这些子树：

- 第 142 页中的 “应用程序树”
- 第 143 页中的 “HTTP 服务树”
- 第 144 页中的 “资源树”
- 第 144 页中的 “连接器服务树”
- 第 144 页中的 “JMS 服务树”
- 第 144 页中的 “ORB 树”
- 第 145 页中的 “线程池树”

应用程序树

以下示意图显示了企业应用程序的各种组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。有关更多信息，请参阅第 145 页中的 “EJB 容器统计信息”。

示例 16–1 应用程序节点树结构

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                  |--- cache (for entity/sfsb) *
   |    |                  |--- pool (for slsb/mdb/entity) *
   |    |                  |--- methods
   |    |                       |---method1 *
   |    |                       |---method2 *
   |    |                  |--- stateful-session-store (for sfsb)*
   |    |                  |--- timers (for s1sb/entity/mdb) *
   |    |--- web-module-1
   |    |        |--- virtual-server-1 *
   |    |                       |---servlet1  *
   |    |                       |---servlet2  *
   |--- standalone-web-module-1
   |    |        |----- virtual-server-2 *
   |    |                       |---servlet3 *
```

示例 16–1 应用程序节点树结构 （续）

```
|    |                        |---servlet4 *
|    |        |----- virtual-server-3 *
|    |                        |---servlet3 *(same servlet on different vs)
|    |                        |---servlet5 *
|--- standalone-ejb-module-1
|    |        |--- ejb2 *
|    |                  |--- cache (for entity/sfsb) *
|    |                  |--- pool (for slsb/mdb/entity) *
|    |                  |--- methods
|    |                       |--- method1 *
|    |                       |--- method2 *
|--- application2
```

HTTP 服务树

以下示意图显示了 HTTP 服务的节点。具有可用的监视信息的节点标有星号 (*)。请参见第 150 页中的 “HTTP 服务统计信息”。

示例 16–2 HTTP 服务示意图（平台版）

```
http-service
         |--- virtual-server-1
              |--- http-listener-1 *
              |--- http-listener-2 *
         |--- virtual-server-2
              |--- http-listener-1 *
              |--- http-listener-2 *
```

示例 16–3 HTTP 服务示意图（企业版）

```
http-service *
             |---connection-queue *
             |---dns *
             |---file-cache *
             |---keep-alive *
             |---pwc-thread-pool *
             |---virtual-server-1*
             |        |--- request *
             |---virtual-server-2*
             |        |--- request *
```

资源树

资源节点保存 JDBC 连接池、连接器连接池等池的可监视属性。以下示意图显示了各种资源组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见第 151 页中的 “JDBC 连接池统计信息” 。

示例 16–4 资源示意图

```
resources
        |---connection-pool1(either connector-connection-pool or jdbc)*
        |---connection-pool2(either connector-connection-pool or jdbc)*
```

连接器服务树

连接器服务节点保存连接器连接池等池的可监视属性。以下示意图显示了各种连接器服务组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见第 152 页中的 “JMS 连接器服务统计信息” 。

示例 16–5 连接器服务示意图

```
connector-service
        |--- resource-adapter-1
        |        |-- connection-pools
        |        |        |-- pool-1 (All pool stats for this pool)
        |        |-- work-management (All work mgmt stats for this RA)
```

JMS 服务树

JMS 服务节点保存连接器连接池等池的可监视属性。以下示意图显示了各种 JMS 服务组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。

示例 16–6 JMS 服务示意图

```
jms-service
        |-- connection-factories [AKA conn. pools in the RA world]
        |        |-- connection-factory-1 (All CF stats for this CF)
        |-- work-management (All work mgmt stats for the MQ-RA)
```

ORB 树

ORB 节点保存连接管理器的可监视属性。以下示意图显示了 ORB 组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见第 153 页中的 “ORB 中连接管理器的统计信息” 。

示例 16–7 ORB 示意图

```
orb
        |--- connection-managers
        |        |--- connection-manager-1 *
        |        |--- connection-manager-1 *
```

线程池树

线程池节点保存连接管理器的可监视属性。以下示意图显示了 ORB 组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见第 153 页中的 “线程池统计信息” 。

示例 16–8 线程池示意图

```
thread-pools
        |    |--- thread-pool-1 *
        |    |--- thread-pool-2 *
```

受监视的组件和服务的统计信息

本节介绍了可用的监视统计信息：

EJB 容器统计信息

下表中介绍了 EJB 统计信息。

表 16–1 EJB 统计信息

属性名称	数据类型	说明
createcount	计数统计信息	调用 EJB 的 create 方法的次数。
removecount	计数统计信息	调用 EJB 的 remove 方法的次数。
pooledcount	范围统计信息	处于汇集状态的实体 Bean 的数目。
readycount	范围统计信息	处于就绪状态的实体 Bean 的数目。
messagecount	计数统计信息	消息驱动 Bean 收到的消息数。
methodreadycount	范围统计信息	处于 MethodReady 状态的有状态或无状态会话 Bean 的数目。
passivecount	范围统计信息	处于 Passive 状态的有状态会话 Bean 的数目。

下表中列出了有关 EJB 方法调用的可用的统计信息。

表 16–2 EJB 方法统计信息

属性名称	数据类型	说明
methodstatistic	时间统计信息	操作被调用的次数；调用期间所花费的总时间等信息。
错误总数	计数统计信息	方法执行时出现异常的次数。如果为 EJB 容器启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 而收集的。
成功总数	计数统计信息	方法成功执行的次数。如果为 EJB 容器启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。
执行时间	计数统计信息	上次成功/不成功尝试执行方法操作所花费的时间 (ms)。如果在 EJB 容器中启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。

下表中列出了有关 EJB 会话存储的统计信息。

表 16–3 EJB 会话存储统计信息

属性名称	数据类型	说明
currentSize	范围统计信息	当前位于存储中的钝化会话数或检查点会话数。
activationCount	计数统计信息	从存储中激活的会话数。
activationSuccessCount	计数统计信息	从存储中成功激活的会话数。
activationErrorCount	计数统计信息	上次成功/不成功尝试执行方法操作所花费的时间 (ms)。如果在 EJB 容器中启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。
passivationCount	计数统计信息	使用此存储钝化（取消激活）的会话数。
passivationSuccessCount	计数统计信息	使用此存储成功钝化的会话数。
passivationErrorCount	计数统计信息	无法使用此存储钝化的会话数。
expiredSessionCount	计数统计信息	此存储删除的过期会话数。
passivatedBeanSize	计数统计信息	由此存储钝化的总字节数，包括总数、最小值和最大值。
passivationTime	计数统计信息	将 Bean 钝化到存储所花费的时间，包括总时间值、最小值和最大值。
checkpointCount（仅限于 EE）	计数统计信息	使用此存储进行会话检查点操作的会话数。
checkpointSuccessCount（仅限于 EE）	计数统计信息	成功进行检查点操作的会话数。
checkpointErrorCount（仅限于 EE）	计数统计信息	无法进行检查点操作的会话数。
checkpointedBeanSize（仅限于 EE）	值统计信息	由该存储进行检查点操作的 Bean 的总数。
checkpointTime（仅限于EE）	时间统计信息	通过检查点操作将 Bean 放入存储中所花费的时间。

下表中列出了有关 EJB 池的可用的统计信息。

表 16–4 EJB 池统计信息

属性名称	数据类型	说明
numbeansinpool	已绑定范围统计信息	相关池中的 EJB 数，决定了该池的更改方式。
numthreadswaiting	已绑定范围统计信息	等待空闲 Bean 的线程数，指出可能的请求拥塞。
totalbeanscreated	计数统计信息	自开始收集数据以来相关池中所创建的 Bean 的数目。
totalbeansdestroyed	计数统计信息	自开始收集数据以来从相关池中销毁的 Bean 的数目。
jmsmaxmessagesload	计数统计信息	为使消息驱动的 Bean 提供服务而一次加载到 JMS 会话中的最大消息数。默认值为 1。仅适用于消息驱动的 Bean 的池。

下表中列出了有关 EJB 高速缓存的可用的统计信息。

表 16–5 EJB 高速缓存统计信息

属性名称	数据类型	说明
cachemisses	已绑定范围统计信息	用户请求未在高速缓存中找到 Bean 的次数。
cachehits	已绑定范围统计信息	用户请求在高速缓存中找到某条目的次数。
numbeansincache	已绑定范围统计信息	高速缓存中 Bean 的数目。这是高速缓存的当前大小。
numpassivations	计数统计信息	钝化的 Bean 数。仅适用于状态会话 Bean。
numpassivationerrors	计数统计信息	挂起时的错误数。仅适用于状态会话 Bean。
numexpiredsessionsremoved	计数统计信息	清除线程删除的过期会话数。仅适用于状态会话 Bean。
numpassivationsuccess	计数统计信息	成功完成挂起的次数。仅适用于状态会话 Bean。

下表中列出了有关计时器的可用的统计信息。

表 16–6 计时器统计信息

统计信息	数据类型	说明
numtimerscreated	计数统计信息	系统中创建的计时器的数目。

表 16–6 计时器统计信息 （续）

统计信息	数据类型	说明
numtimersdelivered	计数统计信息	系统所发送的计时器的数目。
numtimersremoved	计数统计信息	从系统中删除的计时器的数目。

Web 容器统计信息

Web 容器包含在第 142 页中的 “应用程序树”所示的对象树中。系统为每个单独的 Web 应用程序都显示了 Web 容器统计信息。第 149 页中的 “Web 容器统计信息”中显示了有关 Servlet Web 容器的可用的统计信息，第 149 页中的 “Web 容器统计信息”中显示了有关 Web 模块的可用的统计信息。

表 16–7 Web 容器 (Servlet) 统计信息

统计信息	单位	数据类型	说明
errorcount	个	计数统计信息	响应代码大于或等于 400 的情况的累积次数。
maxtime	毫秒	计数统计信息	Web 容器等待请求的最长时间。
processingtime	毫秒	计数统计信息	处理每个请求所需时间的累积值。处理时间是总请时间除以请求计数所得的平均值。
requestcount	个	计数统计信息	到目前为止所处理的请求总数。

第 149 页中的 “Web 容器统计信息”中显示了有关 Web 模块的可用的统计信息。

表 16–8 Web 容器（Web 模块）统计信息

统计信息	数据类型	说明
jspcount	计数统计信息	已装入 Web 模块的 JSP 页面的数目。
jspreloadcount	计数统计信息	已重新装入 Web 模块的 JSP 页面的数目。
sessionstotal	计数统计信息	已为 Web 模块创建的会话总数。
activesessionscurrent	计数统计信息	Web 模块的当前处于活动状态的会话数。
activesessionshigh	计数统计信息	Web 模块的同时处于活动状态的会话最大数。
rejectedsessionstotal	计数统计信息	Web 模块的被拒绝的会话总数。是指由于允许处于活动状态的会话数已达到最大值而未被创建的会话数。

表 16–8 Web 容器（Web 模块）统计信息 （续）

统计信息	数据类型	说明
expiredsessionstotal	计数统计信息	Web 模块的已过期会话的总数。
sessionsize（仅限于 EE）	平均范围统计信息	Web 模块的会话大小。值可以为大、小或平均值，或以字节为单位（用于序列化会话）。
containerlatency（仅限于 EE）	平均范围统计信息	全部请求等待时间中 Web 容器所占的等待时间。值可以是长、短或平均值。
sessionpersisttime（仅限于 EE）	平均范围统计信息	将 HTTP 会话状态保持到 Web 模块的后端存储中所花费的时间（以 ms 为单位，短、长或平均值）。
cachedsessionscurrent（仅限于 EE）	计数统计信息	高速缓存在内存中用于 Web 模块的当前会话数。
passivatedsessionscurrent（仅限于 EE）	计数统计信息	用于 Web 模块的已钝化会话的当前数目。

HTTP 服务统计信息

第 150 页中的 “HTTP 服务统计信息”中显示了有关 HTTP 服务的可用的统计信息。这些统计信息仅适用于平台版。有关 Enterprise Edition 上的HTTP 服务的统计信息，请参见第 158 页中的 “生产 Web 容器 (PWC) 统计信息”。

表 16–9 HTTP 服务统计信息（仅适用于平台版）

统计信息	单位	数据类型	说明
bytesreceived	字节	计数统计信息	每个请求处理器接收到的字节累积值。
bytessent	字节	计数统计信息	每个请求处理器所发送的字节累积值。
currentthreadcount	个	计数统计信息	当前位于侦听器线程池中的处理线程数。
currentthreadsbusy	个	计数统计信息	处理请求的侦听器线程池中当前正在使用的请求处理线程的数目。
errorcount	个	计数统计信息	错误计数的累积值，错误计数是指响应代码大于或等于 400 这类情况发生的次数。

表 16–9 HTTP 服务统计信息（仅适用于平台版） （续）

统计信息	单位	数据类型	说明
maxsparethreads	个	计数统计信息	可以存在的未使用响应处理线程的最大数目。
minsparethreads	个	计数统计信息	可以存在的未使用响应处理线程的最小数目。
maxthreads	个	计数统计信息	侦听器所创建的请求处理线程的最大数目。
maxtime	毫秒	计数统计信息	处理线程的最长时间。
processing-time	毫秒	计数统计信息	处理每个请求所花费时间的累积值。处理时间是总请求处理时间除以请求计数所得的平均值。
request-count	个	计数统计信息	到目前为止所处理的请求总数。

JDBC 连接池统计信息

用于在运行时监视 JDBC 资源，以测量性能并捕获资源使用情况。由于创建 JDBC 连接的成本很高并且常常会导致应用程序出现性能瓶颈问题，因此对 JDBC 连接池释放和创建新连接的方法以及正在等待从特定池中检索连接的线程数的监视是至关重要的。

下表中显示了有关 JDBC 连接池的可用的统计信息。

表 16–10 JDBC 连接池统计信息

统计信息	单位	数据类型	说明
numconnfailedvalidation	个	计数统计信息	从开始时间到上次抽样时间为止在连接池中验证失败的连接总数。
numconnused	个	范围统计信息	提供连接使用情况统计信息。当前正被使用的连接的总数，以及有关使用过的连接的最大数目的信息（高水印）。
numconnfree	个	范围统计信息	上次抽样时池中的空闲连接的总数。
numconntimedout	个	已绑定范围统计信息	开始时间与上次抽样时间之间池中的超时连接总数。
averageconnwaittime	个	计数统计信息	指示尝试与连接器连接池建立连接成功的连接请求的平均等待时间。
waitqueuelength	个	计数统计信息	队列中正在等待处理的连接请求数。

表 16–10 JDBC 连接池统计信息 （续）

统计信息	单位	数据类型	说明
connectionrequestwaittime		范围统计信息	连接请求的最长和最短等待时间。当前值表示连接池处理的上一个请求的等待时间。
numconncreated	毫秒	计数统计信息	自上次复位以来创建的物理连接数。
numconndestroyed	个	计数统计信息	自上次复位以来销毁的物理连接数。
numconnacquired	个	计数统计信息	从池中获取的逻辑连接数。
numconnreleased	个	计数统计信息	释放到池中的逻辑连接数。

JMS 连接器服务统计信息

第 152 页中的 “JMS 连接器服务统计信息”中显示了有关连接器连接池的可用的统计信息。第 152 页中的 “JMS 连接器服务统计信息”中显示了连接器工作管理统计信息。

表 16–11 连接器连接池统计信息

统计信息	单位	数据类型	说明
numconnfailedvalidation	个	计数统计信息	从开始时间到上次抽样时间为止在连接池中验证失败的连接总数。
numconnused	个	范围统计信息	提供连接使用情况统计信息。当前正被使用的连接的总数，以及有关使用过的连接的最大数目的信息（高水印）。
numconnfree	个	范围统计信息	上次抽样时池中的空闲连接的总数。
numconntimedout	个	计数统计信息	开始时间与上次抽样时间之间池中的超时连接总数。
averageconnwaittime	个	计数统计信息	连接池处理连接之前这些连接的平均等待时间。
waitqueuelenght	个	计数统计信息	队列中正在等待处理的连接请求数。
connectionrequestwaittime		范围统计信息	连接请求的最长和最短等待时间。当前值表示连接池处理的上一个请求的等待时间。
numconncreated	毫秒	计数统计信息	自上次复位以来创建的物理连接数。
numconndestroyed	个	计数统计信息	自上次复位以来销毁的物理连接数。

表 16–11 连接器连接池统计信息 （续）

统计信息	单位	数据类型	说明
numconnacquired	个	计数统计信息	从池中获取的逻辑连接数。
numconnreleased	个	计数统计信息	释放到池中的逻辑连接数。

第 152 页中的 “JMS 连接器服务统计信息”中列出了有关连接器工作管理的可用的统计信息。

表 16–12 连接器工作管理统计信息

统计信息	数据类型	说明
activeworkcount	范围统计信息	由连接器执行的工作对象数。
waitqueuelength	范围统计信息	执行前在队列中等待的工作对象数。
workrequestwaittime	范围统计信息	工作对象在被执行前所等待的最长和最短时间。
submittedworkcount	计数统计信息	由连接器模块提交的工作对象数。
rejectedworkcount	计数统计信息	Application Server 拒绝的工作对象数。
completedworkcount	计数统计信息	完成的工作对象数。

ORB 中连接管理器的统计信息

第 153 页中的 “ORB 中连接管理器的统计信息”中列出了有关 ORB 中的连接管理器的可用的统计信息。

表 16–13 ORB 中连接管理器的统计信息

统计信息	单位	数据类型	说明
connectionsidle	个	计数统计信息	提供与 ORB 的空闲连接的总数。
connectionsinuse	个	计数统计信息	提供与 ORB 的正在使用的连接总数。
totalconnections	个	已绑定范围统计信息	与 ORB 的连接总数。

线程池统计信息

下表中显示了有关线程池的可用的统计信息。

表 16–14 线程池统计信息

统计信息	单位	数据类型	说明
averagetimeinqueue	毫秒	范围统计信息	在被处理之前请求在队列中等待的平均时间（以毫秒为单位）。
averageworkcompletion-time	毫秒	范围统计信息	完成分配所花费的平均时间（以毫秒为单位）。
currentnumberofthreads	个	已绑定范围统计信息	当前的请求处理线程数。
numberofavailablethreads	个	计数统计信息	可用的线程数。
numberofbusythreads	个	计数统计信息	处于忙碌状态的线程数。
totalworkitemsadded	个	计数统计信息	到目前为止添加到工作队列中的工作项目总数。

事务服务统计信息

事务服务允许客户机冻结事务子系统，以回滚事务并确定冻结时正在进行的事务。下表中显示了有关事务服务的可用的统计信息。

表 16–15 事务服务统计信息

统计信息	数据类型	说明
activecount	计数统计信息	当前处于活动状态的事务数。
activeids	字符串统计信息	当前处于活动状态的事务的 ID。冻结事务服务后，可以回滚所有类事务。
committedcount	计数统计信息	已提交的事务数。
rolledbackcount	计数统计信息	已回滚的事务数。
state	字符串统计信息	表示事务是否已被冻结。

Java 虚拟机 (JVM) 统计信息

JVM 具有始终处于启用状态的可监视属性。下表中显示了有关 JVM 的可用的统计信息。

表 16–16 JVM 统计信息

统计信息	数据类型	说明
heapsize	已绑定范围统计信息	JVM 内存堆大小的下限或上限驻留内存轨迹。
uptime	计数统计信息	JVM 已运行的时间。

J2SE 5.0 中的 JVM 统计信息

如果 Application Server 被配置为在 J2SE 5.0 或更高版本上运行，则可以从 JVM 中获得其他监视信息。将监视级别设置为“低”以启用这些附加信息的显示。将监视级别设置为“高”还可以查看与系统中每个活动线程相关的信息。有关 J2SE 5.0 中可用的其他监视功能的详细信息，请参见 http://java.sun.com/j2se/1.5.0/docs/guide/management/ 中标题为 "*Monitoring and Management for the Java Platform*" 的文档。

http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage 中讨论了 J2SE 5.0 监视工具。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息” 中显示了有关 J2SE 5.0 中的 JVM 类装入的可用的统计信息。

表 16–17 J2SE 5.0 的 JVM 统计信息 - - 类装入

统计信息	数据类型	说明
loadedclasscount	计数统计信息	当前装入 JVM 的类的数目。
totalloadedclasscount	计数统计信息	自 JVM 开始执行以来已装入的类的总数。
unloadedclasscount	计数统计信息	自 JVM 开始执行以来已从 JVM 中卸载的类的数目。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息” 中显示了有关 J2SE 5.0 中的 JVM 编译的可用的统计信息。

表 16–18 J2SE 5.0 的 JVM 统计信息 - - 编译

统计信息	数据类型	说明
totalcompilationtime	计数统计信息	编译所花费的累积时间（以毫秒为单位）。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息” 中显示了有关 J2SE 5.0 中的 JVM 垃圾回收的可用的统计信息。

表 16–19 J2SE 5.0 的 JVM 统计信息—垃圾收集

统计信息	数据类型	说明
collectioncount	计数统计信息	已发生的收集的总数。
collectiontime	计数统计信息	累积的收集时间（以毫秒为单位）。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息”中显示了有关 J2SE 5.0 中的 JVM 内存的可用的统计信息。

表 16–20 J2SE 5.0 的 JVM 统计信息- -内存

统计信息	数据类型	说明
objectpendingfinalizationcount	计数统计信息	暂挂结束操作的对象的大约数目。
initheapsize	计数统计信息	最初由 JVM 请求的堆的大小。
usedheapsize	计数统计信息	当前正在使用的堆的大小。
maxheapsize	计数统计信息	可用于内存管理的最大内存容量（以字节为单位）。
committedheapsize	计数统计信息	确认可由 JVM 使用的内存容量（以字节为单位）。
initnonheapsize	计数统计信息	最初由 JVM 请求的非堆区域的大小。
usednonheapsize	计数统计信息	当前正在使用的非堆区域的大小。
maxnonheapsize	计数统计信息	可用于内存管理的最大内存容量（以字节为单位）。
committednonheapsize	计数统计信息	确认可由 JVM 使用的内存容量（以字节为单位）。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息”中显示了有关 J2SE 5.0 中的 JVM 操作系统的可用的统计信息。

表 16–21 J2SE 5.0 的 JVM 统计信息- -操作系统

统计信息	数据类型	说明
arch	字符串统计信息	操作系统体系结构。
availableprocessors	计数统计信息	可用于 JVM 的处理器的数目。
name	字符串统计信息	操作系统名称。
version	字符串统计信息	操作系统版本。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息”中显示了有关 J2SE 5.0 中的 JVM 运行的可用的统计信息。

表 16–22 J2SE 5.0 的 JVM 统计信息- -运行

统计信息	数据类型	说明
name	字符串统计信息	代表正在运行的 JVM 的名称
vmname	字符串统计信息	JVM 实现名称。
vmvendor	字符串统计信息	JVM 实现供应商。
vmversion	字符串统计信息	JVM 实现版本。
specname	字符串统计信息	JVM 规范名称。
specvendor	字符串统计信息	JVM 规范供应商。
specversion	字符串统计信息	JVM 规范版本。
managementspecversion	字符串统计信息	由 JVM 实现的管理规范版本
classpath	字符串统计信息	系统类加载器搜索类文件时所使用的类路径。
librarypath	字符串统计信息	Java 库路径。
bootclasspath	字符串统计信息	引导类加载器搜索类文件时所使用的类路径。
inputarguments	字符串统计信息	传递给 JVM 的输入参数。不包括 main 方法的参数。
uptime	计数统计信息	JVM 的正常运行时间（以毫秒为单位）。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息”中显示了有关 J2SE 5.0 中的 JVM ThreadInfo 的可用的统计信息。

表 16–23 J2SE 5.0 的 JVM 统计信息—线程信息

统计信息	数据类型	说明
threadid	计数统计信息	线程 ID。
threadname	字符串统计信息	线程名称。
threadstate	字符串统计信息	线程状态。
blockedtime	计数统计信息	线程进入 BLOCKED 状态以来所经历的时间（以毫秒为单位）。 禁用线程争用监视，则返回 -1。
blockedcount	计数统计信息	线程进入 BLOCKED 状态的总次数。
waitedtime	计数统计信息	线程处于 WAITING 状态所经历的时间（以毫秒为单位）。如果 线程争用监视，则返回 -1。
waitedcount	计数统计信息	线程处于 WAITING 或 TIMED_WAITING 状态的总次数。

表 16–23 J2SE 5.0 的 JVM 统计信息—线程信息 （续）

统计信息	数据类型	说明
lockname	字符串统计信息	一种监视锁的字符串表示形式，该监视锁表明线程被阻塞而无法进或者正等待通过 Object.wait 方法接收通知。
lockownerid	计数统计信息	保存某个对象的监视锁的线程 ID，该线程在该对象上发生阻塞。
lockownername	字符串统计信息	保存某个对象的监视锁的线程名称，该线程在该对象上发生阻塞。
stacktrace	字符串统计信息	与该线程相关的堆栈追踪。

第 155 页中的 “J2SE 5.0 中的 JVM 统计信息”中显示了有关 J2SE 5.0 中的 JVM 线程的可用的统计信息。

表 16–24 J2SE 5.0 的 JVM 统计信息- -线程

统计信息	数据类型	说明
threadcount	计数统计信息	当前的活动守护线程和非守护线程数。
peakthreadcount	计数统计信息	自 JVM 启动或峰复位以来的峰活动线程计数。
totalstartedthreadcount	计数统计信息	自 JVM 启动以来创建和/或启动的线程总数。
daemonthreadcount	计数统计信息	当前的活动守护线程数。
allthreadids	字符串统计信息	所有活动线程 ID 列表。
currentthreadcputime	计数统计信息	如果已启用 CPU 时间测量，则表示当前线程的 CPU 时间（以纳秒单位）。如果已禁用 CPU 时间测量，则返回 -1。
monitordeadlockedthreads	字符串统计信息	处于监视死锁状态的线程 ID 列表。

生产 Web 容器 (PWC) 统计信息

给出了有关 Application Server 企业版 (Enterprise Edition, EE) 中的以下 PWC 组件和服务的可用的统计信息：

- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 虚拟服务器
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 请求
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 文件高速缓存
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 保持活动
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC DNS
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 线程池
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC 连接队列
- 第 158 页中的 “生产 Web 容器 (PWC) 统计信息”， PWC HTTP 服务

第 158 页中的 “生产 Web 容器 (PWC) 统计信息”中列出了有关 PWC 虚拟服务器的统计信息。

表 16–25 PWC虚拟服务器统计信息（仅限于EE）

属性名称	数据类型	说明
id	字符串统计信息	虚拟服务器的ID。
mode	字符串统计信息	虚拟服务器所处的模式。选项包括 unknown 或 active。
hosts	字符串统计信息	由该虚拟服务器提供服务的主机的名称。
interfaces	字符串统计信息	配置了虚拟服务器的接口（侦听器）的类型。

下表中列出了有关PWC请求的可用的统计信息。

表 16–26 PWC请求统计信息（仅限于EE）

属性名称	数据类型	说明
method	字符串统计信息	用于请求的方法。
uri	字符串统计信息	处理的上一个URI。
countrequests	计数统计信息	已处理的请求数。
countbytestransmitted	计数统计信息	传输的字节数；如果此信息不可用，则值为0。
countbytesreceived	计数统计信息	收到的字节数；如果此信息不可用，则值为0。
ratebytesreceived	计数统计信息	在某段服务器定义的时间间隔内数据的接收速率；如果此信息不可用，则值为0。
maxbytestransmissionrate	计数统计信息	在某段服务器定义的时间间隔内数据的最大传输速率；如果此信息不可用，则值为0。
countopenconnections	计数统计信息	当前打开的连接数；如果此信息不可用，则值为0。
maxopenconnections	计数统计信息	同时打开的连接的最大数目；如果此信息不可用，则值为0。
count2xx	计数统计信息	代码为2XX的响应的总数。
count3xx	计数统计信息	代码为3XX的响应的总数。
count4xx	计数统计信息	代码为4XX的响应的总数。
count5xx	计数统计信息	代码为5XX的响应的总数。
countother	计数统计信息	具有其他响应代码的响应总数。

表 16–26 PWC 请求统计信息（仅限于 EE） （续）

属性名称	数据类型	说明
count200	计数统计信息	代码为 200 的响应的总数。
count302	计数统计信息	代码为 302 的响应的总数。
count304	计数统计信息	代码为 304 的响应的总数。
count400	计数统计信息	代码为 400 的响应的总数。
count401	计数统计信息	代码为 401 的响应的总数。
count403	计数统计信息	代码为 403 的响应的总数。
count404	计数统计信息	代码为 404 的响应的总数。
count503	计数统计信息	代码为 503 的响应的总数。

高速缓存信息部分提供了有关文件高速缓存当前的使用方式的信息。下表中列出了有关 PWC 文件高速缓存的统计信息。

表 16–27 PWC 文件高速缓存统计信息（仅限于 EE）

属性名称	数据类型	说明
flagenabled	计数统计信息	指示是否启用了文件高速缓存。禁用时有效值为 0；启用时有效值为 1。
secondsmaxage	计数统计信息	有效高速缓存条目的最长生存期（以秒为单位）。
countentries	计数统计信息	当前的高速缓存条目数。一个高速缓存条目代表一个 URI。
maxentries	计数统计信息	并发高速缓存条目的最大数目。
countopenentries	计数统计信息	与打开的文件关联的条目数。
maxopenentries	计数统计信息	与打开的文件关联的并发高速缓存条目的最大数目。
sizeheapcache	计数统计信息	用于高速缓存内容的堆空间。
maxheapcachesize	计数统计信息	用于高速缓存文件内容的最大堆空间。
sizemmapcache	计数统计信息	内存映射的文件内容所使用的地址空间大小。
maxmmapcachesize	计数统计信息	文件高速缓存用于内存映射的文件内容的最大地址空间大小。
counthits	计数统计信息	高速缓存查找成功的次数。
countmisses	计数统计信息	高速缓存查找失败的次数。
countinfohits	计数统计信息	文件信息查找成功的次数。
countinfomisses	计数统计信息	高速缓存的文件信息丢失的数目。
countcontenthits	计数统计信息	高速缓存的文件内容的命中次数。

表 16–27 PWC 文件高速缓存统计信息（仅限于 EE） （续）

属性名称	数据类型	说明
countcontentmisses	计数统计信息	文件信息查找失败的次数。

本节提供了有关服务器的 HTTP 级保持活动的系统的信息。下表中列出了有关 PWC 保持活动的可用的统计信息。

表 16–28 PWC 保持活动统计信息（仅限于 EE）

属性名称	数据类型	说明
countconnections	计数统计信息	处于保持活动模式的连接数。
maxconnections	计数统计信息	允许同时处于保持活动模式的最大连接数。
counthits	计数统计信息	处于保持活动模式的连接随后进行了有效请求的总次数。
countflushes	计数统计信息	服务器关闭保持活动的连接的次数。
countrefusals	计数统计信息	服务器可能由于有太多的持久性连接而无法将连接传递 到保持活 次数。
counttimeouts	计数统计信息	服务器因客户机连接超时且没有任何活动而终止保持 活动连接的
secondstimeout	计数统计信息	关闭空闲保持活动连接之前经历的时间（以秒为单位）。

DNS 高速缓存高速缓存 IP 地址和 DNS 名称。默认情况下，服务器的 DNS 高速缓存处于禁用状态。一个高速缓存条目代表一个 IP 地址或 DNS 名称查找。下表中列出了有关 PWC DNS 的可用的统计信息。

表 16–29 PWC DNS 统计信息（仅限于 EE）

属性名称	数据类型	说明
flagcacheenabled	计数统计信息	指示是否启用了 DNS 高速缓存。禁用时为 0；启用时为 1。
countcacheentries	计数统计信息	当前位于高速缓存中的 DNS 条目数。
maxcacheentries	计数统计信息	高速缓存中可容纳的 DNS 条目的最大数目。
countcachehits	计数统计信息	DNS 高速缓存查找成功的次数。
countcachemisses	计数统计信息	DNS 高速缓存查找失败的次数。
flagasyncenabled	计数统计信息	指示是否启用了异步 DNS 查找。禁用时为 0；启用时为 1。
countasyncnamelookups	计数统计信息	异步 DNS 名称查找的总数。

表 16–29 PWC DNS 统计信息（仅限于 EE） （续）

属性名称	数据类型	说明
countasyncaddrlookups	计数统计信息	异步 DNS 地址查找的总数。
countasynclookupsinprogress	计数统计信息	正在进行的异步查找的数目。

下表中列出了有关 PWC 线程池的统计信息。

表 16–30 PWC 线程池统计信息（仅限于 EE）

属性名称	数据类型	说明
id	字符串统计信息	线程池 ID。
countthreadsidle	计数统计信息	当前处于空闲状态的请求处理线程数。
countthreads	计数统计信息	当前的请求处理线程的数目。
maxthreads	计数统计信息	可同时存在的请求处理线程的最大数目。
countqueued	计数统计信息	排队等候此线程池处理的请求数。
peakqueued	计数统计信息	队列中同时容纳的最大请求数。
maxqueued	计数统计信息	队列一次可容纳的最大请求数。

连接队列是指请求被处理前容纳这些请求的队列。连接队列的统计信息显示队列中的会话数以及连接被接受前的平均延迟时间。下表中列出了有关 PWC 连接队列的统计信息。

表 16–31 PWC 连接队列统计信息（仅限于 EE）

属性名称	数据类型	说明
id	字符串统计信息	连接队列的 ID。
counttotalconnections	计数统计信息	已接受的连接总数。
countqueued	计数统计信息	当前位于队列中的连接数。
peakqueued	计数统计信息	队列中同时容纳的最大连接数。
maxqueued	计数统计信息	连接队列的最大大小。
countoverflows	计数统计信息	队列太满而无法容纳更多连接的次数。
counttotalqueued	计数统计信息	排队等候的连接总数。由于给定连接可能被多次排队，因此 counttotalqueued 可能会大于或等于 counttotalconnections。

表 16–31 PWC 连接队列统计信息（仅限于 EE） （续）

属性名称	数据类型	说明
tickstotalqueued	计数统计信息	连接在队列中所花费的周期总数。周期是由系统决定的时间单位。
countqueued1minuteaverage	计数统计信息	前 1 分钟内处于排队状态的平均连接数。
countqueued5minuteaverage	计数统计信息	前 5 分钟内处于排队状态的平均连接数。
countqueued15minuteaverage	计数统计信息	前 15 分钟内处于排队状态的平均连接数。

下表中列出了有关 PWC HTTP 服务的统计信息。

表 16–32 PWC HTTP 服务统计信息（仅限于 EE）

属性名称	数据类型	说明
id	字符串统计信息	HTTP 服务的实例名称。
versionserver	字符串统计信息	HTTP 服务的版本号。
timestarted	字符串统计信息	启动 HTTP 服务的时间 (GMT)。
secondsrunning	计数统计信息	HTTP 服务启动以来所经历的时间（以秒为单位）。
maxthreads	计数统计信息	每个实例中的最大工作线程数。
maxvirtualservers	计数统计信息	每个实例中可以配置的最大虚拟服务器数目。
flagprofilingenabled	计数统计信息	是否启用了 HTTP 服务性能探查。有效值为 0 或 1。
flagvirtualserveroverflow	计数统计信息	指示配置的虚拟服务器数目是否超过为 maxvirtualservers 所指定的 果此属性设置为 1，则无法对所有虚拟服务器的统计信息进行跟踪
load1minuteaverage	计数统计信息	前 1 分钟内请求的平均负载。
load5minuteaverage	计数统计信息	前 5 分钟内请求的平均负载。
load15minuteaverage	计数统计信息	前 15 分钟内请求的平均负载。
ratebytestransmitted	计数统计信息	在某个服务器定义的时间间隔内数据的传输速率。如果此信息不可 结果为 0。
ratebytesreceived	计数统计信息	在某个服务器定义的时间间隔内数据的接收速率。如果此信息不可 结果为 0。

启用和禁用监视功能

在管理控制台中，可以通过选择“配置”节点 > “服务器实例”节点 > “监视”页面来配置监视级别，并为要更改监视级别的服务选择值。默认情况下，所有组件和服务的监视功能均处于禁用状态。

有关如何配置监视级别的详细步骤，请查阅管理控制台联机帮助。

在命令行中，使用 `asadmin set` 对各种 Application Server 组件启用监视功能。例如，运行以下命令对 HTTP 服务启用监视功能：

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=HIGH
```

使用 `get` 命令查找当前已对哪些服务和组件启用监视功能。

```
asadmin> get --user admin-user server.monitoring-service.module
-monitoring-levels.*
```

返回：

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.
connector-service = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
server.monitoring-service.module-monitoring-levels.http-service = HIGH
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.jms-service = OFF
server.monitoring-service.module-monitoring-levels.jvm = OFF
server.monitoring-service.module-monitoring-levels.orb = OFF
server.monitoring-service.module-monitoring-levels.thread-pool = OFF
server.monitoring-service.module-monitoring-levels.transaction-service = OFF
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

使用 `set` 命令禁用监视功能。

例如，运行以下命令对 HTTP 服务禁用监视功能：

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=OFF
```

同样，要对其他组件禁用监视功能，请使用 `set` 命令并将监视级别指定为 `OFF`。

查看监视数据

在管理控制台中，可以通过以下方式查看监视数据：导航到独立服务器实例的“监视”页面，然后选择已部署到启用了监视功能的服务器实例上的组件或服务。选定的组件或服务的监视数据显示在“查看”字段下，并附有对可监视属性的说明。

使用管理控制台监视远程应用程序和实例。远程实例必须正在运行并已设置配置才能执行此操作。有关查看监视数据的详细步骤，请查阅管理控制台联机帮助。

要使用命令行实用程序查看监视数据，请使用 `asadmin list` 和 `asadmin get` 命令，后跟可监视对象的带点名称，如下所示：

1. 要查看可监视对象的名称，请使用 `asadmin list` 命令。

   例如，要查看应用程序组件和子系统（已启用对服务器实例上的这些组件和子系统的监视功能）的列表，请在终端窗口中键入以下命令：

   ```
   asadmin> list --user adminuser --monitor server
   ```

   上述命令将返回已启用监视功能的应用程序组件和子系统的列表，例如：

   ```
   server.resources
   server.connector-service
   server.orb
   server.jms-service
   server.jvm
   server.applications
   server.http-service
   server.thread-pools
   ```

2. 要显示已启用监视功能的应用程序组件或子系统的监视统计信息，请使用 `asadmin get` 命令。

   要获得统计信息，请在终端窗口中键入 `asadmin get` 命令，并指定在先前步骤中由 `list` 命令显示的名称。以下示例尝试获取某个特定对象的子系统的所有属性：

   ```
   asadmin> get --user adminuser --monitor server.jvm.*
   ```

   此命令返回以下属性和数据：

   ```
   server.jvm.dotted-name = server.jvm
   server.jvm.heapsize-current = 21241856
   server.jvm.heapsize-description = Provides statistical information about
      the JVM's memory heap size.
   server.jvm.heapsize-highwatermark = 21241856
   server.jvm.heapsize-lastsampletime = 1080232913938
   server.jvm.heapsize-lowerbound = 0
   server.jvm.heapsize-lowwatermark = 0
   server.jvm.heapsize-name = JvmHeapSize
   ```

```
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

了解和指定带点名称

在 `asadmin list` 和 `get` 命令中，指定可监视对象的带点名称。所有子对象都是使用点字符 (.) 作为分隔符来寻址的，因此它们被称为**带点名称**。如果子节点是单元素类型，则只需要使用监视对象类型对对象进行寻址；否则，需要使用 `type.name` 名称格式对对象进行寻址。

例如，`http-service` 就是其中一种有效的可监视对象类型，并且是单元素类型。要对表示实例 `server` 的 `http-service` 的单元素类型子节点进行寻址，则带点名称为：

```
server.http-service
```

再比如，`application` 是一种有效的可监视对象类型，并且是非单元素类型。要对表示应用程序 `PetStore` 的非单元素类型子节点进行寻址，则带点名称为：

```
server.applications.petstore
```

带点名称还可以指定可监视对象中的特定属性。例如，`http-service` 具有一个名为 `bytesreceived-lastsampletime` 的可监视属性。以下名称可对 `bytesreceived` 属性进行寻址：

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

管理员不需要知道 `asadmin list` 和 `get` 命令的有效带点名称。使用 `list` 命令可以显示可用的可监视对象，而使用带有通配符参数的 `get` 命令可以检查任意可监视对象的所有可用属性。

使用具有带点名称的 `list` 和 `get` 命令的基本假设为：

- 如果使用任何具有带点名称且后面**不**跟通配符 (`*`) 的 `list` 命令，则得到的结果为当前节点的直接子节点。例如，`list --user` *adminuser* `--monitor server` 列出了所有属于 `server` 节点的直接子节点。
- 如果使用任何具有带点名称且后面跟有 `.*` 形式的通配符的 `list` 命令，则得到的结果为当前节点的子节点分层树。例如，`list --user` *adminuser* `--monitor server.applications.*` 可列出 `applications` 的所有子节点及其后续子节点等。

- 如果使用任何具有带点名称并且前面或后面带有 **dottedname*、*dotted*name* 或 **dotted name** * 形式的通配符的 `list` 命令，则得到的结果为与所提供的匹配模式创建的正则表达式相匹配的所有节点及其子节点。
- 如果使用后跟 `.*` 或 `*` 的 `get` 命令，则得到的结果为属于要匹配的当前节点的属性集及其值。

有关更多信息，请参见第 172 页中的 “`list` 和 `get` 命令在所有级别上的预期输出” 。

`list` 命令示例

`list` 命令可针对指定的服务器实例名称提供有关当前正在监视的应用程序组件和子系统的信息。使用此命令可以查看某个服务器实例的可监视组件和子组件。有关 `list` 示例的更完整列表，请参见第 172 页中的 “`list` 和 `get` 命令在所有级别上的预期输出” 。

示例 1

```
asadmin> list --user admin-user --monitor server
```

上述命令将返回已启用监视功能的应用程序组件和子系统的列表，例如：

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

还可以列出指定的服务器实例中当前所监视的应用程序。当使用 `get` 命令从某个应用程序中获取特定的监视统计信息时，这会很有用。

示例 2

```
asadmin> list --user admin-user --monitor server.applications
```

返回：

```
server.applications.adminapp
 server.applications.admingui
server.applications.myApp
```

get 命令示例

get 命令将检索以下受监视的信息：

- 组件或子系统中监视的所有属性
- 组件或子系统中监视的特定属性

  如果特定组件或子系统中不存在所请求的属性，将返回一个错误。类似地，如果组件或子系统中所请求的特定属性处于非活动状态，也会返回一个错误。

  有关使用 get 命令的更多信息，请参阅第 172 页中的 “list 和 get 命令在所有级别上的预期输出”。

示例 1

尝试从某个子系统中获取某个特定对象的所有属性：

```
asadmin> get --user admin-user --monitor server.jvm.*
```

返回：

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
   the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

示例 2

尝试获取某个 J2EE 应用程序的所有属性：

```
asadmin> get --user admin-user --monitor server.applications.myJ2eeApp.*
```

返回：

没有与通配符表达式一致的匹配项。CLI137 命令失败。

该 J2EE 应用程序级别上没有暴露可监视的属性，因而显示此回复。

示例 3

尝试获取某个子系统的某个特定属性：

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

返回：

```
server.jvm.uptime-lastsampletime = 1093215374813
```

示例 4

尝试获取某个子系统属性中的某个未知属性：

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

返回：

通过反映相应的统计信息接口 [badname] 没有找到此类属性，CLI137 命令失败。

PetStore 使用示例

下例说明了如何将 asadmin 工具用于监视目的。

用户要检查将样例 PetStore 应用程序部署到 Application Server 之后在该应用程序中调用某个方法的次数。部署该应用程序的实例名称为 server。结合使用 list 和 get 命令即可访问所需的方法统计信息。

1. 启动 Application Server 和 asadmin 工具。
2. 设置一些有用的环境变量，以避免使用每个命令时都输入这些变量：

   ```
   asadmin> export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4849
   ```
3. 列出实例 server 的可监视组件：

   ```
   asadmin> list --user adminuser --monitor server*
   ```

   返回结果（输出类似于以下内容）：

   ```
   server
   server.applications
   server.applications.CometEJB
   ```

```
server.applications.ConverterApp
server.applications.petstore
server.http-service
server.resources
server.thread-pools
```

可监视组件列表包括 thread-pools、http-service、resources 以及所有已部署（并已启用）的 applications。

4. 列出 PetStore 应用程序中的可监视子组件（可以用 -m 代替 --monitor）：

```
asadmin> list -m server.applications.petstore
```

返回：

```
server.applications.petstore.signon-ejb_jar
server.applications.petstore.catalog-ejb_jar
server.applications.petstore.uidgen-ejb_jar
server.applications.petstore.customer-ejb_jar
server.applications.petstore.petstore-ejb_jar
server.applications.petstore.petstore\.war
server.applications.petstore.AsyncSenderJAR_jar
server.applications.petstore.cart-ejb_jar
```

5. 列出 PetStore 应用程序的 EJB 模块 signon-ejb_jar 中的可监视子组件：

```
asadmin> list -m server.applications.petstore.signon-ejb_jar
```

返回：

```
server.applications.petstore.signon-ejb_jar.SignOnEJB
server.applications.petstore.signon-ejb_jar.UserEJB
```

6. 列出 PetStore 应用程序的 EJB 模块 signon-ejb_jar 的实体 Bean UserEJB 中的可监视子组件：

```
asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
```

返回（出于空间考虑，删除了带点名称）：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
```

7. 列出 PetStore 应用程序的 EJB 模块 signon-ejb_jar 的实体 Bean UserEJB 所用方法 getUserName 中的可监视子组件：

```
asadmin> list -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName
```

返回：

在 server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName 没有内容可以列出。要获得以某个字

8. 该方法没有可监视的子组件。获取方法 getUserName 的所有可监视统计信息。

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName.*
```

返回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in milliseconds
   spent during the last successful/unsuccessful attempt to execute the
   operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times an
   operation was called, the total time that was spent during the
   invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
 server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
   that occured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
```

```
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
   successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9. 如果还需要获取执行时间等特定统计信息，请使用如下命令：

```
asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

返回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

list 和 get 命令在所有级别上的预期输出

下表显示了树的各个级别的命令、带点名称及相应的输出。

表 16–33 顶层

命令	带点名称	输出
`list -m`	`server`	`server.applicationsserver.thread-poolsserver.resourcesserver.http-serviceserver.transaction-serviceserver.orb.connection-managersserver.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnectionsserver.jvm`
`list -m`	`server.*`	此节点下的子节点的分层结构。

表 16–33 顶层 （续）

命令	带点名称	输出
get -m	server.*	仅显示一条消息，说明此节点上没有属性。

下表显示了应用程序级别的命令、带点名称以及相应的输出。

表 16–34 应用程序级别

命令	带点名称	输出
list -m	server.applications 或 *applications	appl1app2web-module1_warejb-module2_jar...
list -m	server.applications.* 或 *applications.*	此节点下的子节点的分层结构。
get -m	server.applications.* 或 *applications.*	仅显示一条消息，说明此节点上没有属性。

下表显示了应用程序级别的独立模块和企业应用程序的命令、带点名称以及相应的输出。

表 16–35 应用程序--企业应用程序和独立模块

命令	带点名称	输出
list -m	server.applications.app1 或 *app1 注：仅当已部署了企业应用程序时，此级别才可用。如果部署了独立模块，则此级别不可用。	ejb-module1_jarweb-module2_warejb -module3_jarweb-module3_war...
list -m	server.applications.app1.* 或 *app1.*	此节点下的子节点的分层结构。

表 16–35 应用程序- -企业应用程序和独立模块 （续）

命令	带点名称	输出
get -m	server.applications.app1.* 或 *app1.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.ejb-module1_jar 或 *ejb-module1_jar 或 server.applications.ejb-module1_jar	bean1bean2bean3...
list -m	server.applications.app1.ejb-module1_jar 或 *ejb-module1_jar 或 server.applications.ejb-module1_jar	此节点下的子节点的分层结构。
get -m	server.applications.app1.ejb-module1_jar.* 或 *ejb-module1_jar.* 或 server.applications.ejb-module1_jar.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.ejb -module1_jar.bean1 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	子节点列表： bean-poolbean-cachebean-method
list -m	server.applications.app1.ejb -module1_jar.bean1 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	子节点的分层结构及该节点和所有后续子节点的所有属性的列表。

表 16–35 应用程序--企业应用程序和独立模块 （续）

命令	带点名称	输出
get -m	server.applications.app1.ejb -module1_jar.bean1.* 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	以下属性及其关联值： CreateCount_CountCreateCount_ DescriptionCreateCount_ LastSampleTimeCreateCount_ NameCreateCount_ StartTimeCreateCount_ UnitMethodReadyCount_ CurrentMethodReadyCount_ DescriptionMethodReadyCount_ HighWaterMarkMethodReadyCount_ LastSampleTimeMethodReadyCount_ LowWaterMarkMethodReadyCount_ NameMethodReadyCount_ StartTimeMethodReadyCount_ UnitRemoveCount_CountRemoveCount_ DescriptionRemoveCount_ LastSampleTimeRemoveCount_ NameRemoveCount_StartTimeAttribute RemoveCount_Unit
list -m	server.applications.app1.ejb -module1_jar.bean1.bean-pool 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	没有属性，但会显示一条消息，说明：在 server.applications.appl.ejb-module1_jar.bean1-cache 没有内容可以列出。要获得以某个字符串开头的有效名称，请使用通配符 (*) 字符。例如, 要列出以 server 开头的所有名称，请使用 list server*。
get -m	server.applications.app1.ejb -module1_jar.bean1.bean-pool.* 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	与 EJB 池属性对应的属性和值的列表（如表 1-4 所示）。
list -m	server.applications.app1.ejb -module1_jar.bean1.bean-cache 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.applications.app1.ejb -module1_jar.bean1.bean-cache.* 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	与 EJB 高速缓存属性对应的属性和值的列表（如表 1-5 所示）。
list -m	server.applications.app1.ejb -module1_jar.bean1.bean -method.method1 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。

表 16–35 应用程序- -企业应用程序和独立模块 （续）

命令	带点名称	输出
get -m	server.applications.app1.ejb -module1_jar.bean1.bean -method.method1.* 注：在独立模块中，将不显示包含应用程序名称（本例中为 app1）的节点。	与 EJB 模块属性对应的属性和值的列表（如表 1-2 所示）。
list -m	server.applications.app1.web -module1_war	显示分配给模块的虚拟服务器。
get -m	server.applications.app1.web -module1_war.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.web -module1_war.virtual_server	显示已登记的 servlet 的列表。
get -m	server.applications.app1.web -module1_war.virtual_server.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.web -module1_war.virtual _server.servlet1	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.applications.app1.web -module1_war.virtual _server.servlet1.*	与 Web 容器 (Servlet) 属性对应的属性和值的列表（如表 1-7 所示）。

下表显示了 HTTP 服务级别的命令、带点名称以及相应的输出。

表 16–36 HTTP 服务级别

命令	带点名称	输出
list -m	server.http-service	虚拟服务器列表。
get -m	server.http-service.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.http-service.server	HTTP 侦听器列表。
get -m	server.http-service.server.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.http-service.server .http-listener1	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.http-service.server.*	与 HTTP 服务属性对应的属性和值的列表。

下表显示了线程池级别的命令、带点名称以及相应的输出。

表 16–37 线程池级别

命令	带点名称	输出
list -m	server.thread-pools	thread-pool 名称列表。
get -m	server.thread-pools.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.thread-pools.orb \.threadpool\.thread-pool-1	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.thread-pools..orb \.threadpool\.thread-pool-1.*	与线程池属性对应的属性和值的列表。

下表显示了资源级别的命令、带点名称以及相应的输出。

表 16–38 资源级别

命令	带点名称	输出
list -m	server.resources	池名称列表。
get -m	server.resources.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.resources.jdbc-connection -pool-pool.connection-pool1	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.resources.jdbc-connection -pool-pool.connection-pool1.*	与连接池属性对应的属性和值的列表。

下表显示了事务服务级别的命令、带点名称以及相应的输出。

表 16–39 事务服务级别

命令	带点名称	输出
list -m	server.transaction-service	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
get -m	server.transaction-service.*	与事务服务属性对应的属性和值的列表。

下表显示了 ORB 级别的命令、带点名称以及相应的输出。

表 16–40 ORB 级别

命令	带点名称	输出
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.orb.connection-managers`	ORB 连接管理器的名称。
`get -m`	`server.orb.connection-managers.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.orb.connection-managers.orb \.Connections\.Inbound \.AcceptedConnections`	没有属性，但会显示一条消息，说明“使用带有 --monitor 选项的 get 命令可查看该节点的属性和值”。
`get -m`	`server.orb.connection-managers.orb \.Connections\.Inbound \.AcceptedConnections.*`	与 ORB 连接管理器属性对应的属性和值的列表。

下表显示了 JVM 级别的命令、带点名称以及相应的输出。

表 16–41 JVM 级别

命令	带点名称	输出
`list -m`	`server.jvm`	没有属性，但会显示一条类似于以下内容的消息：使用带有 `--monitor` 选项的 get 命令可查看该节点的属性和值。
`get -m`	`server.jvm.*`	与 JVM 属性对应的属性和值的列表。

使用 JConsole

本节包括以下主题：

- 第 179 页中的“保护 JConsole 与 Application Server 之间的连接”
- 第 179 页中的“将 JConsole 连接到 Application Server 的先决条件”
- 第 180 页中的“将 JConsole 连接到 Application Server”
- 第 180 页中的“将 JConsole 安全连接到 Application Server”

Application Server 的管理（管理和监视）基于 JMX。这意味着被管理组件被表示为 MBean。使用 Java 2 Standard Edition (J2SE) 5.0 可以监视 JVM 并查看 JVM MBean，从而了解其运行情况。为了公开此程序设备，Application Server 提供了一种称为**系统 JMX 连接器服务器**的标准 JMX 连接器服务器配置。启动 Application Server 时，将启动此 JMX 连接器服务器的一个实例，从而向受信客户机公开此程序配置。

Java 监视和管理控制台 (JConsole) 是一种常用的 JMX 连接器，可用于管理 JMX 后端。JConsole (`http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jconsole.html`) 作为

从 J2SE 5.0 开始的标准 JDK 分发的一部分提供。有关 JConsole 的更多信息，请参见http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html

为 Application Server 配置 JConsole 时，Application Server 将成为 JMX 连接器的服务器端，JConsole 将成为 JMX 连接器的首选客户机端。

保护 JConsole 与 Application Server 之间的连接

在如何基于连接的传输层安全性连接到 Application Server 或任何 JMX 连接器服务器端方面，存在细微的差别。如果服务器端是安全的（保证传输层安全性），则只需在客户机端执行较少的配置。

- 默认情况下，平台版 Application Server 的系统 JMX 连接器服务器端是不安全的。
- 默认情况下，企业版 Application Server 的系统 JMX 连接器服务器端是安全的。
- 通信所使用的协议为 RMI/JRMP。如果为 JMX 连接器启用安全性，则使用的协议为基于 SSL 的 RMI/JRMP。

注 – 基于 SSL 的 RMI 不提供其他检查以确保客户机正在与预定服务器进行通信。因此，使用 JConsole 时，始终有可能将用户名和密码发送到恶意主机。能否确保安全性不会降低完全取决于管理员。

当您在计算机（例如 `appserver.sun.com`）上安装平台版域时，将会在 DAS（Domain Administration Server，域管理服务器，即管理服务器或简称域）的 `domain.xml` 中显示以下内容：

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="false"/> <!- – The JSR 160
"system-jmx-connector" – –>
```

JMX 连接器的 `security-enabled` 标志为 *false*。如果运行的是企业版，或者在平台版中为 JMX 连接器启用安全性，则此标志被设置为 *true*。

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="true"/>
...</jmx-connector><!- – The JSR 160 "system-jmx-connector" – –>
```

将 JConsole 连接到 Application Server 的先决条件

JConsole 设置包含两部分：服务器端和客户机端。Application Server 域安装在名为 `appserver.sun.com` 的计算机中，这是一个功能强大的 Solaris 服务器。这是服务器端。

客户机端也安装有 Application Server。假设客户机端是安装有 Java SE 5.0 和 Application Server 的 Windows 计算机。

注 – 仅当对远程计算机上的 Application Server 域启用了安全性（企业版的默认设置）时，才需要在客户机端安装 Application Server。如果仅希望管理上述 Solaris 计算机中的 Application Server 平台版域，则不需要在此客户机上安装 Application Server。

如果服务器端和客户机端位于同一台计算机上，则可以使用 `localhost` 指定主机名。

将 JConsole 连接到 Application Server

本节介绍如何在未对 JMX 连接器启用安全性的情况下将 JConsole 连接到 Application Server。默认情况下，没有在 Application Server 平台版上启用安全性。

1. 在 `appserver.sun.com` 上启动域。
2. 通过运行 *JDK_HOME*`/bin/jconsole` 来启动 JConsole。
3. 在 JConsole 的“连接到代理”选项卡中，输入用户名、密码、主机名和端口（默认为 8686）。

   用户名是指域的管理用户名，密码是指域的管理密码。
4. 单击“连接”。

   在 JConsole 窗口的各个选项卡中，将显示所有的 MBean、虚拟机信息等。

将 JConsole 安全连接到 Application Server

本节介绍如何在对 JMX 连接器启用安全性的情况下将 JConsole 连接到 Application Server。默认情况下，在 Application Server 企业版上启用了安全性。如果已对平台版的 JMX 连接器启用了安全性，请执行此过程。

1. 在客户机（安装有 JConsole）上安装 Application Server。

   需要执行此操作的唯一原因是让 JConsole 知道您所信任的域管理服务器的服务器证书位于何处。要获取此证书，请至少调用一次**远程** `asadmin` 命令；要实现此调用，需要在本地安装 Application Server。
2. 在 `appserver.sun.com` 上启动 Application Server 企业版。

   由于这是企业版域，因此系统 JMX 连接器服务器是安全的。
3. 在本地 Application Server 安装中，运行 *install-dir*`/bin/asadmin list --user admin --secure=true --host appserver.sun.com --port 4849`（其中 4849 是服务器的管理端口）。

   虽然示例选择了 `asadmin list` 命令，但是您可以运行任何远程 `asadmin` 命令。现在，系统将提示您接受 `appserver.sun.com` 的 DAS 所发送的证书。

4. 按 y 以接受 appserver.sun.com 上的域管理服务器所发送的证书。
   在客户机上，服务器证书存储在主目录下名为 .asadmintruststore 的文件中。

   注 – 如果服务器与客户机相同，则不需要执行此步骤。换句话说，如果也在 appserver.sun.com 上运行 JConsole。

5. 通过使用以下 JConsole 命令使 JConsole 知道 DAS 的信任存储位置：
   JDK-dir/bin/jconsole.exe -J-Djavax.net.ssl.trustStore="C:/Documents and Settings/*user*/.asadmintruststore"
   现在，此证书自动得到 JConsole 的信任。
6. 通过运行 *JDK_HOME*/bin/jconsole 来启动 JConsole。
7. 在 JConsole 的“连接到代理”选项卡中，输入用户名、密码、主机名和端口（默认为 8686）。
   用户名是指域的管理用户名，密码是指域的管理密码。
8. 单击“连接”。
   在 JConsole 窗口的各个选项卡中，将显示所有的 MBean、虚拟机信息等。

◆ ◆ ◆ 第 17 章

Java 虚拟机和高级设置

Java 虚拟机 (Java virtual machine, JVM) 是一种解释性计算引擎，负责在已编译的 Java 程序中运行字节代码。JVM 将 Java 字节代码翻译为主机的本机指令。作为 Java 进程的应用服务器需要 JVM 才能运行，并支持在其上运行的 Java 应用程序。JVM 设置是应用服务器配置的一部分。

本章介绍了如何配置 Java 虚拟机 (Java Virtual Machine, JVM) 和其他高级设置。它包含以下各节：

调节 JVM 设置

在配置应用服务器过程中，您可以定义可增强 Java 虚拟机效用的设置。要使用管理控制台更改 JVM 配置，请选择“应用服务器”>“JVM 设置”选项卡，然后定义常规 JVM 设置，如下所示：

- Java 主目录：输入 Java 软件的安装目录名称。Application Server 依赖于 Java SE 软件。

  注 – 如果输入不存在的目录名称，或输入不受支持的 Java EE 软件版本的安装目录名称，则 Application Server 将无法启动。

- Javac 选项：输入 Java 编程语言编译器的命令行选项。当部署 EJB 组件后，Application Server 将运行编译器。
- 调试：要使用 JPDA（Java Platform Debugger Architecture，Java 平台调试器体系结构）设置调试，请选中此“已启用”复选框。

  JPDA 供应用程序开发者使用。
- 调试选项：指定启用调试时传递给 JVM 的 JPDA 选项。

- RMI 编译选项：输入 `rmic` 编译器的命令行选项。当部署 EJB 组件后，Application Server 将运行 `rmic` 编译器。
- 字节码预处理程序：输入以逗号分隔的类名列表。每个类都必须实现 `com.sun.appserv.BytecodePreprocessor` 接口。将按指定顺序调用这些类。

  您可能需要在“字节码预处理程序”字段中输入某些工具（例如，事件探查器）。事件探查器生成用于分析服务器性能的信息。

配置高级设置

要使用管理控制台设置高级应用程序配置，请选择“应用服务器”>“高级”选项卡>“应用程序配置”选项卡，然后设置应用程序配置，如下所示：

- 重新装入：选中此复选框可以启用应用程序的动态重新装入。

  如果启用动态重新装入（默认设置），则在更改应用程序或模块的代码或部署描述符时，可不必重新部署该应用程序或模块。您要做的只是将已更改的 JSP 或类文件复制到应用程序或模块的部署目录中。服务器将定期检查更改，并按照更改自动地、动态地重新部署应用程序。这在开发环境中很有用，因为它允许快速测试代码更改。但在生产环境中，动态重新装入可能会使性能降低。此外，无论何时进行重新装入，该转换时间的会话都将无效。客户机必须重新启动该会话。
- 重新装入轮询时间间隔：定义对应用程序和模块执行代码更改检查和动态重新装入的时间间隔。默认值为 2。
- 管理会话超时：指定处于不活动状态的分钟数，超过该时间后，管理会话将超时。

此外，还应定义部署设置，如下所示：

- 自动部署：选中此复选框可以启用应用程序的自动部署。

  自动部署包括将应用程序或模块文件（JAR、WAR、RAR 或 EAR）复制到特定目录，Application Server 在该目录中自动部署此类文件。
- 自动部署轮询时间间隔：定义对应用程序和模块执行代码更改检查和动态重新装入的时间间隔。默认值为 2。
- 检验器：选中“启用检验器”框可以检验部署描述符文件。此设置为可选设置。
- 预编译：选中“启用预编译”框可以预编译任何 JSP 文件。

◆ ◆ ◆ 第 18 章

18 domain.xml 的带点名称属性

本附录介绍了可用于描述 MBean 及其属性的带点名称属性。`domain.xml` 文件中的每个元素都有相应的 MBean。由于用于使用这些名称的语法涉及使用句点来分隔名称，因此这些名称被称为**带点名称**。

*（星号）可以用于带点名称中的任意位置，它的作用与正则表达式中的通配符类似。使用通配符的好处是它可以折叠带点名称的所有部分。例如，可以将 `this.is.really.long.hierarchy` 之类的长带点名称缩写为 `th*.hierarchy`。不过，`.` 始终用于分隔名称的各部分。`*` 将使您获得带点名称的整个列表。

本附录包括以下主题：

顶层元素

`domain.xml` 文件中的所有顶层元素都必须满足以下条件：

- 每个服务器、配置、群集或节点代理的名称都必须是唯一的。
- 不能将服务器、配置、群集或节点代理命名为 `domain`。
- 不能将服务器实例命名为 `agent`。

下表列出了顶层元素及其相应的带点名称前缀。

元素名	带点名称前缀
applications	`domain.applications`
resources	`domain.resources`
configurations	`domain.configs`

元素名	带点名称前缀
servers	`domain.servers` 该元素包含的所有服务器都可以作为 *server-name* 被访问。其中，*server-name* 是服务器子元素的名称属性值。
clusters	`domain.clusters` 该元素包含的所有群集都可以作为 *cluster-name* 被访问。其中，*cluster-name* 是群集子元素的名称属性值。
node-agents	`domain.node-agents`
lb-configs	`domain.lb-configs`
system-property	`domain.system-property`

有两个可用的别名级别：

1. 利用第一级别的别名可以访问服务器实例或群集的属性而不必通过 `domain.servers` 或 `domain.clusters` 前缀。因此，举例来说，形式为 `server1` 的带点名称将映射到带点名称 `domain.servers.server1`（其中，`server1` 是服务器实例）。
2. 第二级别的别名用于表示群集或独立服务器实例（目标）的配置、应用程序和资源。

下表列出了以服务器名称或群集名称开头的带点名称，这些带点名称被别名化为域下的顶层名称：

带点名称	别名化为	注释
target`.applications.*`	`domain.applications.*`	该别名将解析为仅由 *target* 引用的应用程序。
target`.resources.*`	`domain.resources.*`	该别名将解析为由 *target* 引用的所有 `jdbc-connection-pool`、`connector-connection-pool`、`resource-adapter-config` 和所有其他资源。

下表列出了以服务器名称或群集名称开头的带点名称，这些带点名称在服务器或群集所引用的配置中被别名化的顶层名称。

带点名称	别名化为
target`.http-service`	*config-name*`.http-service`
target`.iiop-service`	*config-name*`.iiop-service`

带点名称	别名化为
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

不能别名化的元素

不应对群集实例进行别名化。要获得群集实例的系统属性，应使用的带点名称属性是：domain.servers.*clustered-instance-name .system-property*，而不是 *clustered-instance-name .system-property*。

asadmin 命令

asadmin get、set 和 list 命令串联使用可以为 Application Server 的抽象分层结构提供导航机制。存在两个分层结构：配置和监视，可以针对它们运行这些命令。list 命令可提供具有只读或可修改属性的管理组件的全限定带点名称。

配置分层结构提供可修改的属性；而监视分层结构中的管理组件的属性完全为只读属性。配置分层结构并不严格基于域的模式文档。使用 list 命令可以访问所需域中的特定管理组件。随后，调用 get 和 set 命令可以获取管理组件的属性的名称和值，或设置这些属性的值。使用通配符 (*) 选项可以获取给定全限定带点名称中的所有匹配。有关 get、set 和 list 命令的使用示例，请参见以下手册页：

get(1)
set(1)
list(1)

◆ ◆ ◆ 第 19 章

19 asadmin 实用程序

Application Server 包含一个称为 asadmin 的命令行管理实用程序。asadmin 实用程序用于启动和停止 Application Server，同时也可用于管理用户、资源和应用程序。

本章包括以下各节：

asadmin 命令用法

使用 asadmin 实用程序可以执行 Application Server 的所有管理任务。您可以使用此 asadmin 实用程序来代替管理控制台。

asadmin 实用程序将调用可标识您希望执行的操作或任务的命令。这些命令区分大小写。短选项参数具有单个破折号 (-)，而长选项参数具有两个破折号 (--)。选项用于控制实用程序执行命令的方式。选项也区分大小写。大多数选项都需要参数值，但是可在功能 ON 或 OFF 之间切换的布尔选项除外。操作数出现在参数值后面，并且以空格、制表符或双破折号 (--) 分隔。asadmin 实用程序将跟在选项及其值后面的任何内容都视为操作数。

示例 19–1 语法示例

asadmin *command* [-short_option] [*short_option_argument*]* [--long_option [*long_option_argument*]* [*operand*]*

```
asadmin create-profiler -u admin --passwordfile password.txt myprofiler
```

要在 Solaris 平台上访问 Application Server asadmin 实用程序命令的手册页，请将 $AS_INSTALL/man 添加到 MANPATH 环境变量中。

您可以通过调用 --help 选项来获取任何 asadmin 实用程序命令的全部用法信息。如果指定一条命令，则会显示此命令的用法信息。使用不带命令的 --help 选项将显示所有可用命令的列表。

示例 19–2 help 命令示例

asadmin --help 显示一般帮助

asadmin *command* --help 显示指定命令的帮助。

本节包括以下主题：

多模式和交互模式

asadmin 实用程序可以用于命令 shell 调用或多命令模式（称为 multimode 命令）。在命令 shell 调用中，从命令 shell 调用 asadmin 实用程序。asadmin 将执行命令，然后退出。在多命令模式中，调用 asadmin 一次后，它将接受多个命令，直至您退出 asadmin 并返回到常规命令 shell 调用。在处于多命令模式时设置的环境变量将用于所有的后续命

令，直至您退出 multimode。可以通过从文件或标准输入（管道）传递先前准备好的命令列表来提供命令。此外，您可以从多模式会话中调用 multimode；当您退出第二个多模式环境之后，将返回到原先的多模式环境。

您还可以在交互或非交互模式下运行 asadmin 实用程序。默认情况下，启用交互模式选项。它提示您提供必需的参数。在任何情况下，您都可以在命令 shell 调用中使用交互模式选项。当您在命令提示符下一次运行一条命令时，以及从某个文件中运行时，您可以在 multimode 中使用交互模式选项。在 multimode 中，来自某个输入流的命令以及通过另一个程序调用的命令无法在交互模式下运行。

本地命令

本地命令可以在不存在管理服务器的情况下执行。但是，用户必须登录到托管域的计算机才能执行命令，并且必须对安装目录和域目录具有访问（权限）。

对于可以在本地或远程执行的命令，如果设置了 --host、--port、--user 和 --passwordfile 选项中的任何一个选项，则无论在环境中还是在命令行中，命令都将以远程模式运行。此外，如果未设置任何本地选项，则无论在命令行中还是在环境中，默认情况下命令都将在本地执行。

远程命令

远程命令始终通过连接到管理服务器并在其中执行命令来执行。需要使用正在运行的管理服务器。所有远程命令都需要以下通用选项：

表 19–1 远程命令所需的选项

短选项	选项	定义
-H	--host	运行域管理服务器的计算机名。默认值为 localhost。
-p	--port	用于管理的 HTTP/S 端口。这是为了管理域而应当将浏览器指向的端口。例如，http://localhost:4848。对于 Platform Edition，默认端口号为 4848。
-u	--user	授权的域管理服务器管理用户名。如果您已经使用 asadmin login 命令通过域验证，则对此特定域执行后续操作时，不需要指定 --user 选项。

表 19–1 远程命令所需的选项 （续）

短选项	选项	定义
	--passwordfile	--passwordfile 选项指定包含特定格式密码条目的文件的名称。密码条目必须具有 AS_ADMIN_ 前缀，后跟采用大写字母的密码名。 例如，要指定域管理服务器密码，请使用具有以下格式的条目：AS_ADMIN_PASSWORD=*password*，其中 *password* 是实际的管理员密码。可以指定的其他密码包括 AS_ADMIN_PASSWORD、AS_ADMIN_USERPASSWORD、AS_ADMIN_ALIASPASSWORD 和 AS_ADMIN_MAPPEDPASSWORD。 所有远程命令都必须通过 --passwordfile 或 asadmin login，或在命令提示符下通过交互方式来指定管理密码才能通过域管理服务器验证。asadmin login 命令只能用来指定管理密码。对于必须为远程命令指定的其他密码，请使用 --passwordfile 或在命令提示符下输入这些密码。 如果您已使用 asadmin login 命令通过域验证，则对此特定域执行后续操作时，不需要通过 --passwordfile 选项指定管理密码。不过，这仅适用于 AS_ADMIN_PASSWORD 选项。您仍需要提供其他密码，例如在个别命令（如 update-file-user）需要时提供 AS_ADMIN_USERPASSWORD。 为了安全起见，指定为环境变量的密码不能通过 asadmin 读取。
-s	--secure	如果设置为 true，则使用 SSL/TLS 与域管理服务器通信。
-I	--interactive	如果设置为 true（默认值），则仅提示必需的密码和用户选项。
-t	--terse	指示任何输出数据都必须非常简明，通常在脚本中避免使用用户友好的句子，而支持使用格式完好的数据。默认值为 false。
-e	--echo	如果设置为 true，将在标准输出中回显命令行语句。默认值为 false。
-h	--help	显示命令的帮助文本。

密码文件

为了安全起见，可以从文件中为命令设置密码，而不要在命令行输入密码。--passwordfile 选项用于指定包含密码的文件。文件的有效内容为：

示例 19–3 密码文件内容

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

Multimode 命令

使用 multimode 命令可以处理 asadmin 命令。命令行界面将提示您输入某个命令，执行此命令，显示此命令的结果，然后提示您输入下一个命令。此外，在此模式下设置的所有 asadmin 选项名称都将用于所有后续命令。您可以设置环境并运行命令，直至通过键入 "exit" 或 "quit" 退出 multimode。您还可以通过从文件或标准输入（管道）传递先前准备好的命令列表来提供命令。您可以从**多模式**会话中调用 multimode；当您退出第二个**多模式**环境之后，将返回到原先的**多模式**环境。

要调用 multimode，请输入 asadmin multimode。

List、Get 和 Set 命令

asadmin list、get 和 set 命令串联使用可以为 Application Server 的点式命名分层结构提供导航机制。有两个分层结构：**配置**和**监视**，可以对它们运行这些命令。list 命令可为具有只读或可修改属性的管理组件提供全限定带点名称。

配置分层结构提供可修改属性，而**监视**分层结构中管理组件的属性完全为只读属性。**配置**分层结构并不严格基于域的模式文档，而**监视**分层结构稍有不同。

使用 list 命令可以访问所需分层结构中的特定管理组件。随后，调用 get 和 set 命令可以立即获取管理组件的属性的名称和值，或设置这些属性的值。使用通配符 (*) 选项可以获取给定全限定带点名称中的所有匹配项。

Application Server 带点名称使用 "."（句点）作为分界符来分隔完整名称的各部分。这与 UNIX 文件系统中使用 "/" 字符来分隔文件绝对路径名的各级别的方式类似。当形成 get、set 和 list 命令可接受的带点名称时，以下规则适用。请注意，特定命令将应用某些附加语义。

- .（句点）始终分隔名称的两个连续部分。
- 名称的一部分通常标识应用服务器子系统和/或其特定实例。例如：web-container、log-service、thread-pool-1 等。
- 如果名称的任一部分本身包含 .（句点），则必须使用前导 \（反斜杠）对其进行转义，以使 "." 不用作分界符。
- *（星号）可以用于带点名称中的任意位置，它的作用与正则表达式中的通配符类似。此外，* 可以折叠带点名称的所有部分。可以将 "<classname>this.is.really.long.hierarchy </classname>" 之类的长带点名称缩写为 "<classname>th*.hierarchy</classname>"。但请注意，. 始终用于分隔名称的各部分。
- 在 Solaris 上，当执行将 * 作为选项值或操作数的命令时，需要使用引号。
- 任何带点名称的顶层开关均为 --monitor 或 -m，此开关将在给定命令行中单独指定。此开关的有无意味着针对应用服务器管理选择两种分层结构的其中一种：监视和配置。

- 如果您碰巧知道确切的完整带点名称（不带任何通配符），则 `list` 和 `get`/`set` 在语义方面稍有不同：
  - `list` 命令将此完整带点名称视为分层结构中某个父节点的完整名称。将此名称提供给 `list` 命令时，此命令仅返回该级别的直接子节点的名称。例如，`list server.applications.web-module` 将列出部署到域或默认服务器的所有 Web 模块。
  - `get` 和 `set` 命令将此完整带点名称视为某个节点（其带点名称本身是您在删除此带点名称的最后一部分时获得的名称）的属性的全限定名称，并获取/设置此属性的值。如果存在此类属性，则为 true。但从不会出现这种情况，因为为了查找分层结构中某个特定节点的属性的名称，必须使用通配符 `*`。例如，`server.applications.web-module.JSPWiki.context-root*` 将返回部署到域或默认服务器的 Web 应用程序的上下文根目录。

`list` 命令是这三个命令的导航功能的起源。如果要 `set` 或 `get` 某个特定应用服务器子系统的属性，则必须知道此子系统的带点名称。可以借助 `list` 命令来查找此子系统的带点名称。以查找某个以 `/` 开头的大型文件系统中特定文件的修改日期（属性）为例，您必须首先找出此文件在文件系统中的位置，然后再查看其属性。因此，用于了解 Application Server 中分层结构的前两个命令为：`* list "*"` 和 `<command>* list * --monitor`。请查阅 `get`、`set` 或 `list` 命令的手册页，以确定这些命令的有序输出。

服务器生命周期命令

服务器生命周期命令是指用于创建、删除、启动或停止域、服务 (DAS) 或实例的命令。

表 19–2 服务器生命周期命令

命令	定义
`create-service`	配置自动引导时 DAS 的启动。在 Solaris 10 上，此命令使用服务管理工具 (Service Management Facility, SMF)。这是本地命令，必须以具有超级用户权限的 OS 级用户身份运行。此命令仅可用于 Solaris 10。创建服务时，用户必须启动、启用、禁用、删除或停止服务。DAS 必须存储在超级用户有权访问的文件夹中。不能将配置存储在网络文件系统中。创建服务，以使它受操作系统级用户（该用户拥有 DAS 配置所在文件夹）控制。要运行此命令，您必须具有 `solaris.smf.*` 授权。
`create-domain`	创建域的配置。域是管理名称空间。每个域都有一种配置，此配置存储在一组文件中。可以在给定的 Application Server 安装中创建任意数量的域，每个域都有一个特定的管理身份。一个域可以独立于其他域存在。任何有权访问给定系统上 `asadmin` 实用程序的用户都可以创建域，并将其配置存储在所选的文件夹中。默认情况下，在 *install_dir*`/domains` 目录中创建域配置。您可以覆盖此位置以将配置存储在其他位置。

表 19–2 服务器生命周期命令 （续）

命令	定义
`delete-domain`	删除命名域。此域必须已存在并且必须已停止。
`start-domain`	启动域。如果未指定域目录，则启动默认 *install_dir*`/domains` 目录中的域。如果存在两个或多个域，则必须指定 *domain_name* 操作数。
`stop-domain`	停止指定域的域管理服务器。
`restore-domain`	从备份目录恢复域下的文件。
`list-domains`	列出域。如果未指定域目录，则列出默认 *install_dir*`/domains` 目录中的域。如果存在多个域，则必须指定 *domain_name* 操作数。
`backup-domain`	备份命名域下的文件。
`list-backups`	显示有关备份系统信息库中所有备份的状态信息。
`shutdown`	正常关闭管理服务器以及所有正在运行的实例。您必须手动启动管理服务器才能再次启动它。

列表和状态命令

列表和状态命令显示已部署组件的状态。

表 19–3 列表和状态命令

命令	定义
`show-component-status`	获取已部署组件的状态。状态为服务器所返回的字符串表示。可能的状态字符串包括 "status of *app-name* is `enabled`" 或 "status of *app-name* is `disabled`"。
`list-components`	列出所有已部署的 Java EE 5 组件。如果未指定 `--type` 选项，则列出所有组件。
`list-sub-components`	列出已部署的模块或已部署应用程序的模块中的 EJB 或 Servlet。如果未标识模块，则列出所有模块。

部署命令

部署命令部署应用程序或获取客户机桩模块。

表 19–4 部署命令

命令	定义
deploy	部署企业应用程序、Web 应用程序、EJB 模块、连接器模块或应用程序客户机端模块。如果组件已部署或已存在，则在 --force 选项设置为 true 时会对其进行强制重新部署。
deploydir	从开发目录直接部署应用程序。部署目录中必须存在符合 Java EE 规范的相应目录分层结构和部署描述符。
get-client-stubs	将 AppClient 独立模块或包含 AppClient 模块的应用程序的客户机桩模块 JAR 文件从服务器获取到本地目录。在执行此命令之前，应部署应用程序或模块。也可以将客户机桩模块作为使用 --retrieve 选项的 deploy 命令的一部分获取。
undeploy	删除指定的已部署组件。

Message Queue 管理命令

Message Queue 管理命令可用于管理 JMS 目的地。

表 19–5 Message Queue 命令

命令	定义
create-jmsdest	创建 JMS 物理目的地。可以使用 create-jms-resource 命令连同物理目的地来创建 JMS 目的地资源，此资源具有指定物理目的地的 Name 属性。
delete-jmsdest	删除指定的 JMS 目的地。
flush-jmsdest	从指定目标的 JMS 服务配置的物理目的地中清除消息。
list-jmsdest	列出 JMS 物理目的地。
jms-ping	检查 JMS 服务（也称为 JMS 提供者）是否启动并运行。启动 Application Server 时，默认情况下会启动 JMS 服务。此外，它仅对 JMS 服务中的默认 JMS 主机执行 ping 操作。当它无法对内置 JMS 服务执行 ping 操作时，将显示错误消息。

资源管理命令

资源命令可用于管理应用程序中使用的各种资源。

表 19–6 资源管理命令

命令	定义
create-jdbc-connection-pool	注册具有指定 JDBC 连接池名称的新 JDBC 连接池。
delete-jdbc-connection-pool	删除 JDBC 连接池。操作数标识要删除的 JDBC 连接池。
list-jdbc-connection-pools	获取已创建的 JDBC 连接池。
create-jdbc-resource	创建新的 JDBC 资源。
delete-jdbc-resource	删除具有指定 JNDI 名称的 JDBC 资源。
list-jdbc-resources	显示已创建的 JDBC 资源的列表。
create-jms-resource	创建 Java 消息服务 (Java Message Service, JMS) 连接工厂资源或 JMS 目的地资源。
delete-jms-resource	删除指定的 JMS 资源。
list-jms-resources	列出现有的 JMS 资源（目的地资源和连接工厂资源）。
create-jndi-resource	注册 JNDI 资源。
delete-jndi-resource	删除具有指定 JNDI 名称的 JNDI 资源。
list-jndi-resources	标识所有的现有 JNDI 资源。
list-jndi-entries	浏览并查询 JNDI 树。
create-javamail-resource	创建 JavaMail 会话资源。
delete-javamail-resource	删除指定的 JavaMail 会话资源。
list-javamail-resources	列出现有的 JavaMail 会话资源。
create-persistence-resource	注册持久性资源。
delete-persistence-resource	删除持久性资源。当您删除持久性资源时，此命令也会删除使用 create-persistence-resource 命令创建的 JDBC 资源。
list-persistence-resources	显示所有持久性资源。
create-custom-resource	创建自定义资源。自定义资源指定实现 javax.naming.spi.ObjectFactory 接口的自定义服务器范围资源对象工厂。
delete-custom-resource	删除自定义资源。
list-custom-resources	列出自定义资源。
create-connector-connection-pool	添加具有指定连接池名称的新连接器连接池。
delete-connector-connection-pool	删除使用操作数 connector_connection_pool_name 指定的连接器连接池。

表 19–6 资源管理命令 （续）

命令	定义
list-connector-connection-pools	列出已创建的连接器连接池。
create-connector-resource	注册具有指定 JNDI 名称的连接器资源。
delete-connector-resource	删除具有指定 JNDI 名称的连接器资源。
list-connector-resources	获取所有连接器资源。
create-admin-object	添加具有指定 JNDI 名称的受管对象。
delete-admin-object	删除具有指定 JNDI 名称的受管对象。
list-admin-objects	列出所有受管对象。
create-resource-adapter-config	为连接器模块创建配置信息。
delete-resource-adapter-config	删除在 domain.xml 中为连接器模块创建的配置信息。
list-resource-adapter-configs	列出 domain.xml 中用于连接器模块的配置信息。
add-resources	创建在指定 XML 文件中命名的资源。*xml_file_path* 是要创建的资源所在的 XML 文件的路径。应在 resources.xml 文件中将 DOCTYPE 指定为 *install_dir*/lib/dtds/sun-resources_1_2.dtd。
ping-connection-pool	测试连接池是否可用于 JDBC 连接池和连接器连接池。例如，如果为希望随后部署的应用程序创建新的 JDBC 连接池，则在部署应用程序之前，将使用此命令对此 JDBC 池进行测试。在对连接池执行 ping 操作之前，必须创建通过验证的连接池，并且确保已启动 Application Server 或数据库。

Application Server 配置命令

配置命令可用于配置 Application Server 的操作。本节包括以下主题：

常用配置命令

这些命令可用于管理 Application Server 组件的配置。

表 19–7 常用配置命令

命令	定义
enable	启用指定的组件。如果组件已启用，则会重新启用它。组件必须已经部署才能启用。如果尚未部署组件，则会返回错误消息。
disable	立即禁用命名组件。组件必须已经部署。如果尚未部署组件，则会返回错误消息。
export	标记变量名称以便自动导出到后续命令的环境中。所有后续命令都使用指定的变量名称值，除非您对它们执行 unset 或退出 multimode。
get	获取属性的名称和值。
set	设置一个或多个可配置属性的值。
list	列出可配置的元素。在 Solaris 上，当执行将 * 作为选项值或操作数的命令时，需要使用引号。
unset	删除一个或多个为多模式环境设置的变量。变量及其关联值将不再存在于环境中。

HTTP、IIOP 和 SSL 侦听器命令

HTTP 和 IIOP 侦听器命令可帮助您管理侦听器。这些命令仅在远程模式下受支持。

表 19–8 IIOP 侦听器命令

命令	定义
create-http-listener	添加新的 HTTP 侦听器。
delete-http-listener	删除指定的 HTTP 侦听器。
list-http-listeners	列出现有 HTTP 侦听器。
create-iiop-listener	创建 IIOP 侦听器。
delete-iiop-listener	删除指定的 IIOP 侦听器。
list-iiop-listeners	列出现有 IIOP 侦听器。
create-ssl	创建并配置选定 HTTP 侦听器、IIOP 侦听器或 IIOP 服务中的 SSL 元素，以便针对此侦听器/服务启用安全通信。
delete-ssl	删除选定 HTTP 侦听器、IIOP 侦听器或 IIOP 服务中的 SSL 元素。

生命周期和审计模块命令

生命周期和审计模块命令可帮助您控制生命周期模块以及用于实现审计功能的可选插件模块。这些命令仅在远程模式下受支持。

表 19–9 生命周期模块命令

命令	定义
`create-lifecycle-module`	创建生命周期模块。生命周期模块提供一种用于在 Application Server 环境中运行短持续时间或长持续时间的基于 Java 的任务的方式。
`delete-lifecycle-module`	删除指定的生命周期模块。
`list-lifecycle-modules`	列出现有生命周期模块。
`create-audit-module`	为实现审计功能的插件模块添加命名审计模块。
`delete-audit-module`	删除命名审计模块。
`list-audit-modules`	列出所有审计模块。

事件探查器和 JVM 选项命令

事件探查器和 JVM 选项命令可用于管理事件探查器并控制这些元素。这些命令仅在远程模式下受支持。

表 19–10 事件探查器和 JVM 选项命令

命令	定义
`create-profiler`	创建事件探查器元素。可以通过 Java 配置中的事件探查器元素将服务器实例绑定到特定事件探查器。更改事件探查器将要求您重新启动服务器。
`delete-profiler`	删除您指定的事件探查器元素。可以通过 Java 配置中的事件探查器元素将服务器实例绑定到特定事件探查器。更改事件探查器将要求您重新启动服务器。
`create-jvm-option`	在 Java 配置或 `domain.xml` 文件的事件探查器元素中创建 JVM 选项。如果为事件探查器创建 JVM 选项，则可以使用这些选项来记录运行特定事件探查器所需的设置。必须重新启动服务器才能使新创建的 JVM 选项生效。
`delete-jvm-option`	从 Java 配置或 `domain.xml` 文件的事件探查器元素中删除 JVM 选项。

虚拟服务器命令

虚拟服务器命令可用于控制这些元素。这些命令仅在远程模式下受支持。

表 19–11 虚拟服务器命令

命令	定义
create-virtual-server	创建命名虚拟服务器。通过 Application Server 中的虚拟功能，可以使单个侦听多个主机地址的 HTTP 服务器进程处理多个 URL 域。如果在两个虚拟服务器上提供了应用程序，则这两个服务器仍共享相同的物理资源池。
delete-virtual-server	删除具有指定虚拟服务器 ID 的虚拟服务器。
list-virtual-server	列出现有虚拟服务器。

线程池命令

线程池命令可用于控制这些元素。这些命令仅在远程模式下受支持。

表 19–12 线程池命令

命令	定义
create-threadpool	创建具有指定名称的线程池。您可以指定池中的最大和最小线程数、工作队列数以及线程的空闲超时。可以使用创建的线程池来处理 IIOP 请求，并使资源适配器处理工作管理请求。所创建的线程池可以用于多个资源适配器。
delete-threadpool	删除具有命名 ID 的线程池。
list-threadpools	列出所有线程池。

事务和计时器命令

事务和计时器命令可用于控制事务和计时器子系统，允许您暂停任何进行中的事务。这些命令仅在远程模式下受支持。

表 19–13 事务命令

命令	定义
freeze-transaction	在所有进行中的事务暂停期间冻结事务子系统。请在回滚任何进行中的事务之前调用此命令。对已经冻结的事务子系统调用此命令不会有任何作用。
unfreeze-transaction	恢复所有已暂停的进行中事务。请对已经冻结的事务调用此命令。
recover-transactions	手动恢复暂挂事务。
rollback-transaction	回滚命名事务。

表 19–13 事务命令 （续）

命令	定义
list-timers	列出特定服务器实例所拥有的计时器。

用户管理命令

这些用户命令可用于管理文件领域验证所支持的用户。这些命令仅在远程模式下受支持。

表 19–14 用户管理命令

命令	定义
create-file-user	在密钥文件中创建具有指定用户名、密码和组的条目。可以创建多个以冒号 (:) 分隔的组。
delete-file-user	删除密钥文件中具有指定用户名的条目。
update-file-user	使用指定 user_name、user_password 和组更新密钥文件中的现有条目。可以输入多个以冒号 (:) 分隔的组。
list-file-users	创建文件领域验证所支持的文件用户的列表。
list-file-groups	列出文件领域验证所支持的用户和组。此命令可列出文件用户中的可用组。

监视数据命令

监视数据命令可用于监视服务器。这些命令仅在远程模式下受支持。

表 19–15 监视数据命令

命令	定义
start-callflow-monitoring	从 Web 容器、EJB 容器和 JDBC 收集数据并使其相关联，以便提供完整的请求调用流/路径。仅当 callflow-monitoring 为 ON 时才收集数据。
stop-callflow-monitoring	禁用收集请求的调用流信息。

规则命令

规则命令可用于管理服务器的规则。这些命令仅在远程模式下受支持。

表 19–16 规则命令

命令	定义
`create-management-rule`	创建新的管理规则，以便对 Application Server 安装和已部署的应用程序进行智能化自管理。
`delete-management-rule`	删除您指定的管理规则。
`list-management-rules`	列出可用的管理规则。

数据库命令

数据库命令可用于启动和停止 Java DB 数据库（基于 Apache Derby）。这些命令仅在本地模式下受支持。

表 19–17 数据库命令

命令	定义
start-database	启动随 Application Server 一起提供的 Java DB 服务器。仅可将此命令用于部署到 Application Server 中的应用程序。
stop-database	停止 Java DB 服务器的进程。Java DB 服务器随 Application Server 一起提供。

诊断和日志记录命令

诊断和日志记录命令可帮助您解决 Application Server 问题。这些命令仅在远程模式下受支持。

表 19–18 诊断和日志记录命令

命令	定义
`generate-diagnostic-report`	生成包含指向应用服务器安装详细信息（例如，应用服务器实例的配置详细信息、日志记录详细信息或进程特定信息）的指针或导航链接的 HTML 报告。

表 19–18 诊断和日志记录命令 （续）

命令	定义
generate-jvm-report	显示给定目标实例（包括域管理服务）的线程（堆栈追踪转储）、类和内存。此命令仅用于应用服务器实例进程。此命令取代了传统的技术（例如，将 ctrl+break 或 kill -3 信号发送到应用服务器进程）。如果目标服务器实例未处于运行状态，则此命令将不会起作用。
display-error-statistics	显示自上次重新启动服务器以来 server.log 中严重级别和警告的汇总列表。
display-error-distribution	显示模块级别的实例 server.log 中的错误分布。™
display-log-records	显示标有给定时间戳的给定模块的所有错误消息。

Web 服务命令

Web 服务命令可用于监视已部署的 Web 服务和管理转换规则。

表 19–19 Web 服务命令

命令	定义
configure-webservice-management	配置已部署的 Web 服务端点的监视或 maxhistorysize 属性。
create-transformation-rule	创建可应用于 Web 服务操作的 XSLT 转换规则。此规则可应用于请求、响应或同时应用于这两者。
delete-transformation-rule	删除给定 Web 服务的 XSLT 转换规则。
list-transformation-rules	按照规则的应用顺序列出给定 Web 服务的所有转换规则。
publish-to-registry	将 Web 服务的辅件发布到注册表服务器。
unpublish-from-registry	从注册表服务器中取消发布 Web 服务的辅件。
list-registry-locations	显示已配置的 Web 服务条目访问点的列表。

安全性服务命令

这些安全性命令可用于控制连接器连接池的安全性映射。这些命令仅在远程模式下受支持。

表 19–20 安全性命令

命令	定义
`create-connector-security-map`	为指定连接器连接池创建安全性映射。如果不存在安全性映射，则会创建新的安全性映射。此外，使用此命令可以将应用程序调用方身份（主体或用户组）映射到基于容器管理事务的方案中适当的企业信息系统 (enterprise information system, EIS) 主体。可以将一个或多个命名的安全性映射与连接器连接池相关联。连接器安全性映射配置支持使用通配符星号 (*) 来表示所有用户或所有用户组。为使此命令成功运行，必须首先创建连接器连接池。EIS 是保存组织数据的任何系统。它可以是主机、消息传送系统、数据库系统或应用程序。
`delete-connector-security-map`	删除指定连接器连接池的安全性映射。
`update-connector-security-map`	修改指定连接器连接池的安全性映射。
`list-connector-security-maps`	列出属于指定连接器连接池的安全性映射。
`create-message-security-provider`	允许管理员为给定消息层（用于为 Application Server 指定参数和属性的 `domain.xml` 文件的 `message-security-config` 元素）创建 `provider-config` 子元素。
`delete-message-security-provide`	允许管理员为给定消息层（用于为 Application Server 指定参数和属性的 `domain.xml` 文件的 `message-security-config` 元素）删除 `provider-config` 子元素。
`list-message-security-providers`	允许管理员为给定消息层（`domain.xml` 的 `message-security-config` 元素）列出所有安全性消息提供者（`provider-config` 子元素）。
`create-auth-realm`	添加命名验证领域。
`delete-auth-realm`	删除命名验证领域。
`list-auth-realms`	列出现有验证领域。

密码命令

密码命令可用于管理密码和确保 Application Server 的安全性。

表 19–21 密码命令

命令	定义
`create-password-alias`	创建密码别名并将其存储在 `domain.xml` 中。别名是格式为 `${ALIAS=password-alias-password}` 的令牌。与别名相对应的密码以加密的形式存储。此命令可以采用安全交互形式（提示用户提供所有信息）和更为脚本友好的形式（在命令行中传播密码）。

表 19–21 密码命令 （续）

命令	定义
delete-password-alias	删除密码别名。
update-password-alias	更新命名目标中的密码别名 ID。
list-password-aliases	列出所有密码别名。
change-admin-password	此远程命令用于修改管理密码。此命令为交互式命令，提示用户提供旧的和新的管理密码（并进行确认）。
change-master-password	此本地命令用于修改主密码。此命令为交互式命令，提示用户提供旧的和新的主密码。此命令在服务器停止时才有效。

检验 domain.xml 命令

XML 检验命令用于检验 domain.xml 文件的内容。

表 19–22 检验 domain.xml 命令

命令	定义
verify-domain-xml	检验 domain.xml 文件的内容。

自定义 MBean 命令

MBean 命令可用于管理和注册自定义 MBean。这些命令仅在远程模式下受支持。

表 19–23 自定义 MBean 命令

命令	定义
create-mbean	创建并注册自定义 MBean。如果目标 MBeanServer (DAS) 未运行，则无法注册 MBean。
delete-mbean	删除自定义 MBean。请确保目标 MBeanServer 处于运行状态。
list-mbeans	列出指定目标的自定义 MBean。

杂项命令

杂项命令可用于管理 Application Server 的各个方面。

表 19–24 杂项命令

命令	定义
`login`	使您登录到某个域中。如果在不同的计算机上（本地）创建了各种应用服务器域，则从其中任何一台计算机中调用 `asadmin` 都可以管理位于其他位置（远程）的域。当您选择某台特定计算机作为管理客户机，并使用它来管理多个域和服务器时，这种方法就尤为方便。用于管理位于其他位置的域的 `asadmin` 命令称为远程命令。使用 `asadmin login` 命令可以轻松管理此类远程域。`login` 命令仅在交互模式下运行。它将提示您提供管理用户名和密码。成功登录后，将在用户的主目录中创建文件 `.asadminpass`。此文件与使用 `--savelogin` 选项时通过 `create-domain` 命令修改的文件是同一个文件。域必须处于运行状态，才能运行此命令。
`version`	显示版本信息。如果此命令无法与具有给定用户/密码和主机/端口的管理服务器通信，此命令将在本地检索版本并显示警告消息。
`help`	显示所有 `asadmin` 实用程序命令的列表。指定命令可以显示此命令的用法信息。
`install-license`	防止未经授权使用 Application Server。允许您安装许可证文件。

索引

主

资

自