管理员指南

Sun™ ONE Application Server

Version 7， Update 1

816-6863-10
2003 年 3 月

目录

关于本指南

本指南介绍了如何配置和管理 Sun™ ONE Application Server 7，适用于公司企业中希望通过万维网将客户机服务器应用程序扩展到更广范围的信息技术管理员。

本前言包括以下各节：

- 本指南的内容
- 本指南的组织结构
- 文档惯例
- 生产线概述
- 使用文档
- 产品支持

本指南的内容

本指南介绍了如何配置和管理 Sun ONE Application Server。配置服务器后，可以通过本指南学习如何维护。

本指南的组织结构

本指南分为四部分和一个综合索引。第一部分 “服务器基础知识和管理全局设置”为产品概述。第二部分 “管理单个服务器实例”介绍了如何使用管理服务器以及其它影响所有服务器实例的服务器功能。

基本熟悉如何使用管理服务器后，可以参阅第三部分 “管理 HTTP 服务器功能和虚拟服务器”获得使用程序和配置式样的信息。

最后的附录是对多种主题进行说明的特定参考内容，其中包括国际化问题、服务器扩展和 Sun ONE Application Server 命令行界面文档。

第一部分：服务器基础知识和管理全局设置

本部分概述了 Sun ONE Application Server，其中包括以下各章：

- 第 1 章 “Sun ONE Application Server 管理快速入门”概述了 Sun ONE Application Server。
- 第 2 章 “设置管理服务器首选项”介绍了如何管理管理服务器。
- 第 3 章 “配置管理域”介绍了如何使用多个域。

第二部分：管理单个服务器实例

本部分详细介绍了有关配置、管理和使用服务器实例的概念和过程。其中包括以下各章：

- 第 4 章 “使用应用程序服务器实例”介绍了如何为 Sun ONE Application Server 配置服务器首选项。
- 第 5 章 “使用日志”介绍了日志的基本概念和 Sun ONE Application Server 中的日志特征及功能。
- 第 6 章 “监视 Sun ONE Application Server”介绍了有关在 Sun ONE Application Server 中可用的监视功能以及简单网络管理协议 (SNMP) 的特征和功能的信息。
- 第 7 章 “配置 Web 服务器插件”介绍了 Sun ONE Application Server 如何处理 HTTP 请求，以及如何配置和使用 Sun ONE Application Server 附带的 Web 服务器插件。
- 第 8 章 “配置 J2EE 容器”介绍了如何配置和使用向 J2EE 应用程序组件 （例如企业 Java Bean [EJB] 和消息驱动的 Bean [MDB]）提供运行时支持的容器。
- 第 9 章 “使用事务服务”介绍了数据库事务以及如何使用和管理它们。
- 第 10 章 “配置命名和资源”介绍了如何配置 J2EE 资源。
- 第 11 章 “使用 JMS 服务”提供了理解和管理通过 Sun ONE Message Queue （本地 JMS 供应商）提供的内置 JMS 服务所需的信息。

- 第 12 章 “为 Corba/IIOP 客户机配置服务器”介绍了如何使用 Sun ONE Application Server 环境中的 RMI/IIOP 协议配置对基于 CORBA 的客户机的支持。
- 第 13 章 “部署应用程序”介绍了如何将应用程序部署到 Sun ONE Application Server。

第三部分：管理 HTTP 服务器功能和虚拟服务器

本部分提供了在管理界面上使用程序和配置样式的相关信息，其中包括以下各章：

- 第 14 章 “配置 HTTP 功能”介绍了为 Sun ONE Application Server 的 HTTP 相关功能配置首选项的方法。
- 第 15 章 “使用虚拟服务器”介绍了如何使用 Sun ONE Application Server 设置和管理虚拟服务器。
- 第 16 章 “管理虚拟服务器内容”介绍了如何配置和管理服务器内容。

第四部分：附录

本节包含多个附录，其中包含用户可能需要查看的参考资料。本节包含以下附录：

- 附录 A “使用命令行界面”提供了使用命令行公用程序代替用户界面屏幕的说明。
- 附录 B “第三方版权声明”包含附加的版权信息。

文档惯例

本节介绍本指南使用的各种惯例：

- 一般惯例
- 有关目录的惯例

一般惯例

本指南使用了以下一般惯例：

- **文件和目录路径**采用 UNIX® 格式 （由正斜杠分隔目录名）。对于 Windows 版本，目录路径相同，但使用反斜杠分隔目录。
- **URL** 的格式如下：

  http://*server.domain*/*path*/*file*.html

  在这些 URL 中， *server* 是应用程序运行所在的服务器的名称； *domain* 是您的 Internet 域名； *path* 是服务器的目录结构； *file* 是单个文件名。 URL 中的斜体项为占位符。
- **字体惯例**包括：
  - 等宽字体用于样例代码和代码列表、 API 和语言元素 （例如，函数名和类名）、文件名、路径名、目录名以及 HTML 标记。
  - *斜体*用于代码变量。
  - *斜体*还用于书名、强调、变量、占位符以及斜体文字。
  - **粗体**用于段落标题或粗体文字。
- 本文档中，多数平台的**安装根目录**都由 *install_dir* 表示。第 21 页上的 “有关目录的惯例”对例外情况进行了说明。

  默认情况下，**多数**平台上的 *install_dir* 位置都位于：
  - Solaris 8 非基于软件包的评估安装：

    user's home directory/sun/appserver7
  - 非捆绑的、基于软件包的 Solaris 安装：

    /opt/SUNWappserver7
  - Windows 的所有安装：

    C:\Sun\AppServer7

对于上面列出的平台，*default_config_dir* 和 *install_config_dir* 与 *install_dir* 相同。有关例外情况及其它信息，请参见第 21 页上的 “有关目录的惯例”。

- 本文档中，**实例根目录**由 *instance_dir* 表示，它是以下内容的缩写：

  default_config_dir/domains/*domain*/*instance*

- 除非特别说明，本手册中所有**特定于 UNIX 的说明**也适用于 Linux 操作系统。

有关目录的惯例

默认情况下，使用基于软件包的 Solaris 8 和 9 安装以及 Solaris 9 捆绑安装时，应用程序服务器文件将分布在多个根目录中。本节介绍了这些目录。

- **对于 Solaris 9 捆绑安装**，本指南使用了以下文档惯例，对应于所提供的各个不同的默认安装目录：
  - *install_dir* 为 /usr/appserver/，它包含安装映像的静态部分。组成应用程序服务器的所有公用程序、可执行文件和库均驻留在此目录中。
  - *default_config_dir* 为 /var/appserver/domains，它是所创建的域的默认位置。
  - *install_config_dir* 为 /etc/appserver/，它包含与安装有关的配置信息，例如，为此安装配置的许可证和管理域主列表。
- **对于基于软件包的非捆绑的 Solaris 8 和 9 安装**，本指南了使用以下文档惯例，对应于所提供的各个不同的默认安装目录：
  - *install_dir* 为 /opt/SUNWappserver7，它包含安装映像的静态部分。组成应用程序服务器的所有公用程序、可执行文件和库均驻留在此目录中。
  - *default_config_dir* 为 /var/opt/SUNWappserver7/domains，它是所创建的域的默认位置。
  - *install_config_dir* 为 /etc/opt/SUNWappserver7/config，它包含与安装有关的配置信息，例如，为此安装配置的许可证和管理域主列表。

生产线概述

Sun ONE Application Server 7 是一个符合 J2EE 1.3 规范的应用程序服务器，它除了支持标准的 HTTP 服务器编程设备以外，还支持新兴的 Java Web 服务标准。提供了三种应用程序服务器版本，可以满足生产和开发环境的多种需求：

- 平台版
- 标准版
- 企业版

平台版

平台版构成了 Sun ONE Application Server 7 生产线的核心。此产品可以直接用于生产，它提供高性能、资源占用少、且符合 J2EE 1.3 规范的运行时环境。在理想情况下，该环境不仅适用于嵌入第三方应用程序，还适用于基本操作部署。除了 Web 服务，平台版还包括经过 Sun ONE Web Server 和 Sun ONE Message Queue 产品认可的内置技术。

平台版部署仅限于单个应用程序服务器实例 （即 Java 平台 [“Java 虚拟机”或 “JVM™”] 的单个虚拟机）。平台版支持多层部署技术，但是网络服务器层代理不执行负载平衡。在平台版中，管理公用程序仅限于本地客户机。

平台版已集成到 Solaris 9 中。

标准版

此版本是本 *《Getting Started Guide》* 的重点。标准版层得到了增强，在平台版的基础上增加了远程管理功能。所包含的已增强管理功能、远程命令行和基于 Web 的管理都是标准版的一部分。此版本还可以通过 Web 服务器层代理将 Web 应用程序通信量分区。标准版支持为每台计算机配置多个应用程序服务器实例 (JVM)。

企业版

企业版具有更高的可用性、负载平衡和能够满足大多数基于 J2EE 应用程序部署要求的群集功能，从而实现对核心应用程序服务器平台的增强。企业版中扩展了标准版的管理功能，以便部署多个实例和多台计算机。

群集支持包括易于配置的克隆应用程序服务器实例组，客户机请求可以平均分摊到各实例上。此版本同时支持外部负载平衡器和基于负载平衡 Web 层的代理。企业版中包括 HTTP 会话、状态会话 Bean 实例和 Java 消息传送服务 (JMS) 资源故障转移。独特的 “Always On” 数据库高可用性技术构成了企业版中 HA 持久性存储的基础。

有关产品的详细信息，请参见 Sun Microsystems Web 站点上的 Sun ONE Application Server 页面。

使用文档

要获得联机文件（可移植文档格式 [PDF] 或超文本标记语言 [HTML] 格式）形式的 Sun ONE Application Server 手册，请访问：

http://docs.sun.com/

下表列出了在 Sun ONE Application Server 手册中说明的任务和概念。左列列出了任务和概念，右列列出了相应的手册。

Sun ONE Application Server 文档指南

有关信息	请参见以下手册
软件和文档的最新信息	*发行说明*
支持的平台和环境	*Platform Summary*
应用程序服务器简介，包括新功能、评估安装信息和结构概述。	*Getting Started Guide*
安装 Sun ONE Application Server 及其各种组件 （样例应用程序、管理界面和 Sun ONE Message Queue）。	*安装指南*
根据 Sun ONE Application Server 7 中的开放式 Java 标准模式创建和执行 J2EE 应用程序。包括有关应用程序设计、开发者工具、安全性、汇编、部署、调试和创建设生命周期模块的常规信息。	*Developer's Guide*

Sun ONE Application Server 文档指南 *（续）*

有关信息	请参见以下手册
根据 Sun ONE Application Server 7 中 Web 应用程序的开放式 Java 标准模式创建和执行 J2EE 应用程序。介绍了 Web 应用程序编程概念和任务，并提供了样例代码、执行程序提示和参考资料。	*Web 应用程序开发者指南* *Developer's Guide to Web Applications*
根据 Sun ONE Application Server 7 中企业 Bean 的开放式 Java 标准模式创建和执行 J2EE 应用程序。介绍了 EJB 编程概念和任务，并提供了样例代码、执行程序提示和参考资料。	*Developer's Guide to Enterprise JavaBeans Technology*
在 Sun ONE Application Server 7 上创建访问 J2EE 应用程序的客户机	*Developer's Guide to Clients*
创建 Web 服务	*Developer's Guide to Web Services*
J2EE 功能，例如 JDBC、JNDI、JTS、JMS、JavaMail、资源和连接器	*Developer's Guide to J2EE Features and Services*
创建自定义 NSAPI 插件	*Developer's Guide to NSAPI*
执行以下管理任务： • 使用管理界面和命令行界面 • 配置服务器首选项 • 使用管理域 • 使用服务器实例 • 监视和记录服务器操作 • 配置 Web 服务器插件 • 配置 Java 消息传送服务 • 使用 J2EE 功能 • 配置对基于 CORBA 的客户机的支持 • 配置数据库连接 • 配置事务管理 • 配置 Web 容器 • 部署应用程序 • 管理虚拟服务器	*管理员指南*

Sun ONE Application Server 文档指南 *（续）*

有关信息	请参见以下手册
编辑服务器配置文件	*Administrator's Configuration File Reference*
为 Sun ONE Application Server 7 操作环境配置和管理安全性。包括有关常规安全性、证书和 SSL/TLS 加密的信息。对基于 HTTP 服务器的安全性也进行了说明。	*Administrator's Guide to Security*
为 Sun ONE Application Server 7 配置和管理 J2EE CA 连接器的服务提供商执行程序。包括有关管理工具和 DTD 的信息，并提供了样例 XML 文件。	*J2EE CA Service Provider Implementation Administrator's Guide*
将应用程序从 Netscape Application Server 2.1 版移植到新的 Sun ONE Application Server 7 编程模式（包括随 Sun ONE Application Server 提供的 Online Bank 应用程序的样例移植）。	*Migrating and Redeploying Server Applications Guide*
使用 Sun ONE Message Queue。	Sun ONE Message Queue 文档位于： http://docs.sun.com/

产品支持

如果您的系统出现问题，请使用以下方法与用户支持中心联系

- 访问联机支持 Web 站点：

  http://www.sun.com/supportraining/

- 拨打随您的维修合同提供的电话号码

联系之前，请准备好以下信息。这可以帮助我们的支持中心人员更好地为您解决问题：

- 问题说明，包括出现问题时的情况及其对操作的影响
- 计算机类型、操作系统版本以及产品版本，包括可能影响该问题的任何修补程序和其它软件
- 用于再现问题的方法的详细步骤
- 任何错误日志或信息转储

服务器基础知识和管理全局设置

第 1 章

Sun ONE Application Server 管理快速入门

本章介绍了管理 Sun ONE Application Server 的基本内容：从何处查找有关服务器的常规信息，如何访问服务器的用户界面以及如何访问联机帮助。

本章包括以下小节：

- 关于 Sun ONE Application Server
- 配置 Solaris 捆绑版
- 使用管理界面
- 使用命令行界面
- 访问管理服务器
- 访问应用程序服务器实例
- 使用 Sun ONE Studio
- 关于配置文件
- 使用许可证命令

关于 Sun ONE Application Server

Sun ONE Application Server 提供了强大的 J2EE 平台进行开发、部署以及管理范围广泛的服务器、客户机和设备的电子商务应用程序服务。主要功能包括事务管理、性能、可伸缩性、安全性和企业应用程序集成。

Sun ONE Application Server 支持的服务从 Web 发布到企业规模的事务处理，同时使开发者可以基于 JavaServer Pages (JSP™)、 Java Servlet 和 Enterprise JavaBeans™ (EJB™) 技术创建应用程序。

它包含以下管理员使用的基本工具：

- 多个管理域，使不同的管理员可以创建和管理自己的应用程序服务器实例集。
- 一个管理服务器，提供管理设备 （每个域一个管理服务器）
- 一个图形用户界面，用于服务器管理 （管理界面）
- 一个命令行界面，可以和管理界面一样用来执行相同的任务

使用这些工具可以执行所有管理功能，其中包括：

- 管理域
- 管理服务器实例
- 部署应用程序
- 监测服务器
- 使用日志文件
- 管理资源
- 管理信息队列服务器
- 使用事务服务
- 使用 Corba/IIOP 客户机
- 配置 Web 服务器插件
- 配置 J2EE 容器
- 管理 HTTP 服务器功能

有关 Sun ONE Application Server 体系结构和功能的详细信息，以及应对 Sun ONE Application Server 采取的初始步骤，请参见 *《Sun ONE Application Server Getting Started Guide》*。

配置 Solaris 捆绑版

本指南介绍了两种针对 Solaris 的 Sun ONE Application Server 安装：Solaris 9 捆绑版和非捆绑版。如果您收到的 Sun ONE Application Server 是作为 Solaris 9 安装的一部分，则您拥有的是 Solaris 捆绑版本。如果您收到的是 Sun ONE Application Server 的单机版，则您拥有的是非捆绑版本。

注意 如果您使用的是 Sun ONE Application Server 的 Solaris 非捆绑版本，或者使用的是 Windows 版本，请跳过此节，继续第 33 页上的“使用管理界面”。

如果您使用的是 Sun ONE Application Server 的 Solaris 9 捆绑版本，开始使用服务器之前，需要执行一些额外的配置步骤。

如果您安装 Sun ONE Application Server 的非捆绑版本，作为安装进程的一部分，将自动创建域、管理服务器和服务器实例。

使用 Solaris 9 捆绑版本时，除了执行其它步骤以外，您必须手动创建这些项目。执行这些初始步骤之后，您就可以充分利用 Sun ONE Application Server 的所有功能，其中包括添加其它管理域和服务器实例。

本节包括以下主题：

- 创建管理域
- 启动管理服务器
- 创建应用程序服务器实例
- 部署应用程序

创建管理域

多个管理域使不同的管理用户可以创建和管理各自的域。域是使用一个系统中安装的二进制通用集创建的实例集。每个域都有一个管理服务器。

创建新域时，要指定以下内容：

- 管理服务器的端口号。安装非捆绑版本时，默认值为 4848。
- 管理用户名和密码。访问管理服务器时需要这些密码，无论您是访问管理界面还是运行命令行界面。
- 域位置。

必须使用命令行界面的 asadmin 公用程序的 create-domain 命令创建域。有关创建管理域的详细信息，请参见第 3 章 “配置管理域”。

启动管理服务器

创建管理域时，将创建管理服务器。管理服务器是 Sun ONE Application Server 的一个特殊实例，它支持管理界面并为命令行界面提供管理设备。

要使用管理界面或使用命令行界面中的多数命令，您必须拥有一个正在运行的管理服务器。有关启动管理服务器的详细信息，请参见第 45 页上的“启动管理服务器”。

创建应用程序服务器实例

创建域并启动管理服务器后，需要创建应用程序服务器实例。每个应用程序服务器实例都拥有自己的 J2EE 配置、J2EE 资源、应用程序部署区域以及服务器配置设置。

可以通过管理界面或通过命令行界面创建应用程序服务器实例。服务器实例是在域的文件夹中创建的。

在非捆绑版本中，安装时创建的服务器实例称为 server1。 server1 将在整个文档的各个示例中经常使用。

有关创建应用程序服务器实例的详细信息，请参见第 72 页上的 “添加应用程序服务器实例”。

部署应用程序

创建域、启动管理服务器和添加应用程序服务器实例后，可以对该实例部署应用程序。详细信息，请参见第 13 章 “部署应用程序”。

使用管理界面

使用管理界面配置服务器的各个方面。本节包括以下主题：

- 访问管理界面
- 使用选项卡
- 使用按钮
- 访问联机帮助
- 退出管理界面

注意 服务器配置的某些方面和相应的管理界面仍会有所变化。在以后发行的产品中，可能将不稳定的界面替换为更清晰、更稳定的版本。多数服务器配置和管理界面将保持不变，或以兼容的方式进行一些更改，但是某些方面可能会有很大的更改。以后发行的产品文档中将清楚地介绍真正进行了哪些不兼容的更改。

访问管理界面

Sun ONE Application Server 的管理界面将在称为管理服务器的 HTTP 服务器上运行。如果使用非捆绑版本，安装 Sun ONE Application Server 时将安装管理服务器。如果安装捆绑版本，则必须创建管理域和管理服务器。详细信息，请参见第 31 页上的 “配置 Solaris 捆绑版”。

要使用管理界面，必须运行管理服务器。有关启动管理服务器的详细信息，请参见第 45 页上的 “启动管理服务器”。

安装 Sun ONE Application Server 或创建域时，您为管理服务器选择了端口号或使用了默认端口 4848。要访问管理界面，请在 Web 浏览器中键入：

http://*hostname.domain*:*port*/

例如：

http://austen.sun.com:4848/

如果从安装了 Sun ONE Application Server 的同一台计算机上访问管理服务器，您可以使用：

http://*localhost*:4848

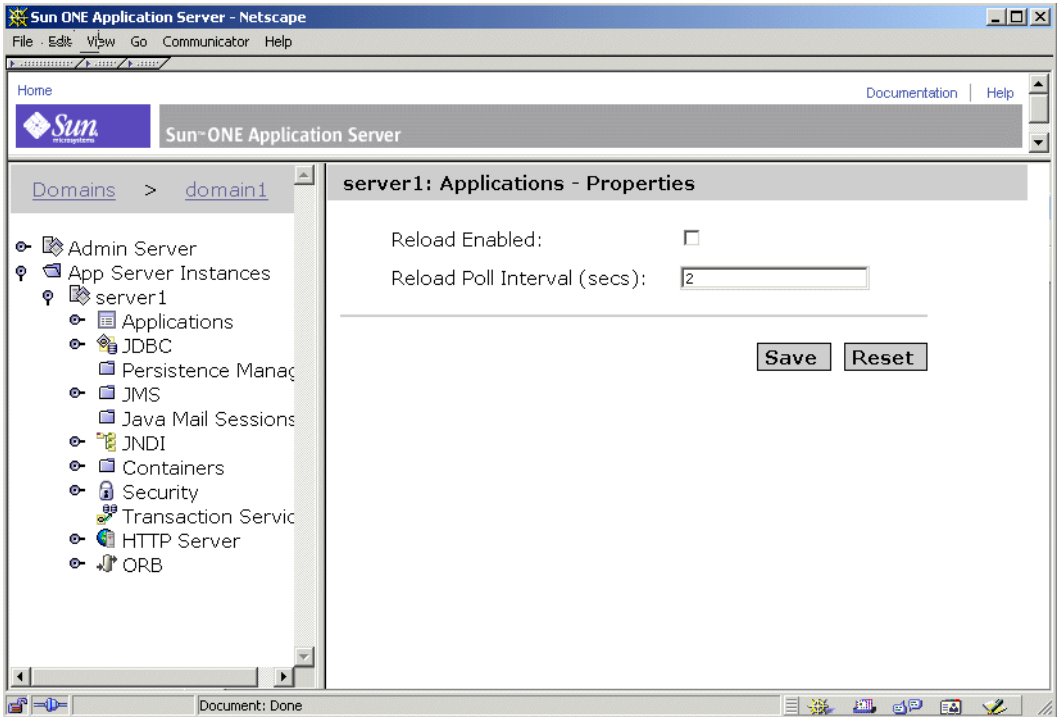
系统将提示您输入用户名和密码，这是安装用户在安装过程中设置的，或是您在创建域和管理服务器时设置的。

只要可以访问浏览器，您便可以从远程位置访问管理界面。您可以通过网络从能够到达服务器的任何计算机对其进行访问。

在 Windows 中，您可以从 “开始”菜单访问管理界面，即从 “开始”菜单中选择 “程序”，然后依次选择 “Sun Microsystems”，“Sun ONE Application Server 7” 以及 “启动管理控制台”。

下图显示了管理界面。

Sun ONE Application Server 管理界面



左侧窗格是可以在 Sun ONE Application Server 中配置的所有项目的树视图。要使用管理界面，请单击左侧窗格中的项目。右侧窗格显示了与该项目关联的页面。

如果左侧窗格中的项目旁边有一个打开/关闭符号，可以单击打开/关闭符号，展开该项目以显示其子项目。如果未展开树项目，则符号如下所示：

关闭的符号

如果已展开树项目，则符号如下所示：

打开的符号

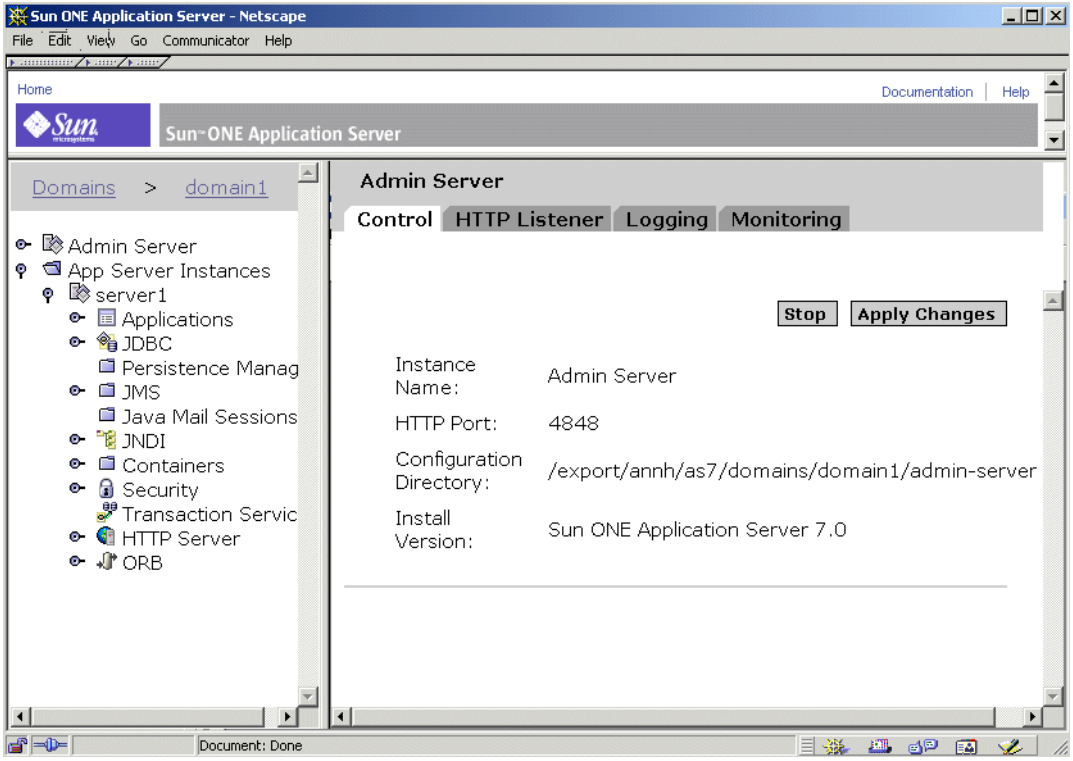
使用选项卡

管理界面中的某些部分包含浏览其它页面可以使用的选项卡。这些选项卡在右侧窗格顶部的单独窗格中显示。

要使用选项卡，请单击选项卡名称。一些选项卡将直接使您进入某个页面，而另一些选项卡在其名称下面显示一个可用页面列表。单击某个页面名称即可转至该页面。

下图显示了带选项卡的管理界面：

带选项卡的管理界面



使用按钮

以下是管理界面提供的标准按钮。

标准管理界面按钮

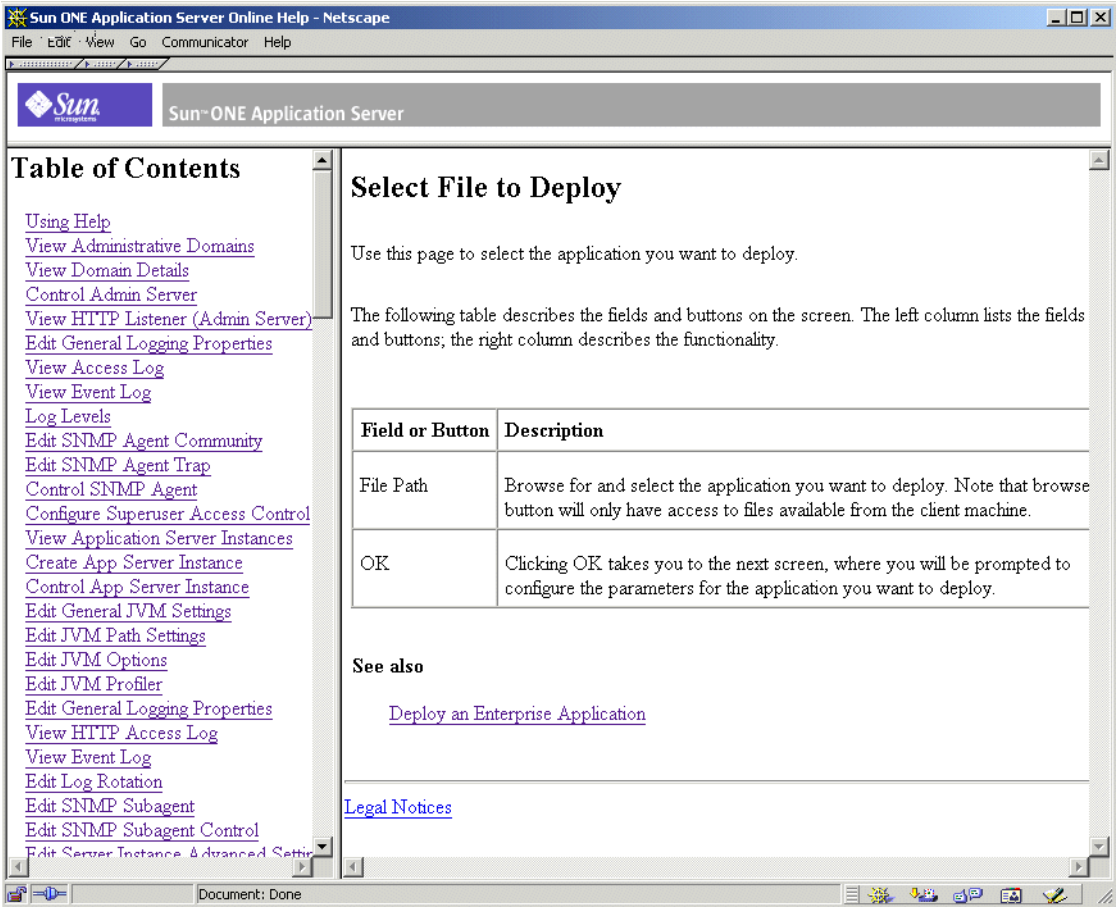
按钮	执行的操作
取消	取消操作且不保存更改，并返回上一页。
删除	删除某个项目。通常单击项目旁边的 “Select”可以指示单击 “Delete”时要删除的内容。
新建	显示创建新项目的页面。例如，单击 “Application Server Instances”页面上的 “New”，将显示 “Create New Instance”页面。
确定	保存输入的信息。如果对 Sun ONE Application Server 实例进行了配置更改，必须应用这些更改才能使其生效。 详细信息，请参见第 73 页上的“应用对应用程序服务器实例所做的更改”。
重置	将页面上的值重置为默认值。
保存	保存输入的信息。如果对 Sun ONE Application Server 实例进行了配置更改，必须应用这些更改才能使其生效。 详细信息，请参见第 73 页上的“应用对应用程序服务器实例所做的更改”。

其它按钮则根据特定屏幕的需要提供。

访问联机帮助

通过单击管理界面顶部标题中的 “Help”按钮，可以在管理界面中访问任何页面的帮助。联机帮助将介绍正在访问的页面的用法，并给出有关如何在页面的字段中进行输入的信息。

联机帮助



您可以使用帮助窗口左侧窗格中的目录浏览其它页面的帮助。有关使用帮助的帮助信息，请参见联机帮助目录中的第一个主题 “Using Help”。

退出管理界面

管理界面中没有明显的退出或注销按钮。要退出，请关闭访问管理界面使用的浏览器。同时，还要关闭可能在计算机中运行的同一个浏览器中的其它实例。

使用命令行界面

Sun ONE Application Server 包含一个命令行界面。可以使用 asadmin 公用程序和与其关联的命令，和在管理界面中一样执行相同的任务集。例如，可以启动和停止应用程序服务器实例、配置服务器和部署应用程序。

可以从 Shell 的命令提示符使用这些命令，也可以从其它脚本和程序调用这些命令。可以使用这些命令自动重复管理任务。

有关使用命令行界面的详细信息以及命令列表，请参见附录 A“使用命令行界面”。

访问管理服务器

管理服务器是 Sun ONE Application Server 的一个特殊实例，它支持管理界面并为管理界面和命令行界面提供管理设备。它可以管理这些界面和工具的配置、部署和监测设备，因此必须运行它以便这些界面和工具正常工作。

要配置管理服务器属性，请参见管理界面。单击左侧窗格中的“Admin Server”，将显示管理服务器的配置设置。

有关管理服务器的详细信息，请参见第 2 章“设置管理服务器首选项”。

访问应用程序服务器实例

您可以拥有多个 Sun ONE Application Server 实例。每个应用程序服务器实例都拥有自己的配置、资源和应用程序部署区域，这些都独立于其它应用程序服务器实例。更改一个应用程序服务器实例的配置，不会更改其它应用程序服务器实例的配置。管理界面为创建的每个应用程序服务器实例包含一个项目。如果使用的是软件的非捆绑版本，则在安装时将创建一个应用程序服务器实例。如果愿意，还可以创建其它实例。

如果使用的是 Solaris 9 捆绑版本，系统不会自动创建应用程序服务器实例。详细信息，请参见第 31 页上的“配置 Solaris 捆绑版”。

有关应用程序服务器实例的详细信息，请参见第 4 章 “使用应用程序服务器实例”。

使用 Sun ONE Studio

要部署应用程序，除了使用管理界面和命令行界面以外，还可以使用 Sun ONE Studio 4。有关 Sun ONE Studio 的详细信息，请参见 *《Sun ONE Studio 4, Enterprise Edition for Java with Application Server 7 Tutorial》*，访问地址如下：http://docs.sun.com。

关于配置文件

通过管理界面或命令行界面对服务器设置进行更改时，管理服务器将更改基础配置文件。这些文件包含管理服务器和单个应用程序服务器实例的设置。

有关文件及其包含的设置的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

使用许可证命令

当您购买 Sun ONE Application Server 时，安装过程中将自动安装相应类型的许可证。有关许可证类型及其限制条件的详细信息，请参见 *《Sun ONE Application Server Installation Guide》*。

Sun ONE Application Server 将包含命令行公用程序，以便安装后管理许可证。

要在安装后升级许可证，对 Windows 计算机可以使用 asadmin 公用程序的 install-license 命令。此命令必须在安装了 Sun ONE Application Server 的本地计算机上运行。安装许可证之前，必须停止所有正在运行的应用程序服务器。

其语法如下：

asadmin install-license

对于基于软件包的 Solaris 的非捆绑安装，许可证安装使用以下语法：

pkgadd -d *full_path* SUNWaslco

例如，

pkgadd -d /*install_dir*/pkg SUNWaslco

对于 Solaris 9 捆绑许可证安装，许可证安装使用以下语法：

pkgadd -d *full_path* SUNWaslc

要获取有关许可证的信息，请使用 display-license 命令，其语法如下：

asadmin display-license [--user *admin_user*] [--password *admin_password*] [--passwordfile *password_file*][--host *localhost*] [--port *admin_port*] [--local=*true/false*]

此命令可以在本地运行，也可以远程运行，这取决于本地选项的值。例如，要从本地计算机运行此命令，并且使用主机和端口号的默认值，其语法如下：

asadmin display-license --local

输出描述了当前安装的许可证类型 （例如， evaluation）、到期日期 （如果有）、许可证允许的每个管理服务器的实例数目以及是否允许进行远程管理。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

第 2 章

设置管理服务器首选项

管理服务器是 Sun ONE Application Server 的特殊实例，用于提供管理界面并为管理界面和命令行界面提供管理设备。此外，还用于管理 Sun ONE Application Server 的配置、部署和监视设备。本章介绍如何配置管理服务器。

本章包括以下主题：

- 关于管理服务器
- 启动管理服务器
- 关闭管理服务器
- 访问管理服务器设置
- 查看管理服务器控制设置
- 向管理服务器应用更改
- 为管理服务器编辑 HTTP 监听器设置
- 设置 SNMP、记录和安全首选项

关于管理服务器

管理服务器是 Sun ONE Application Server 的特殊实例，用于为管理界面和命令行界面提供管理设备。还用于管理这些界面的配置、部署和监视设备，并提供管理界面页面。管理服务器必须处于运行状态，才能使用管理界面并在命令行界面中运行大多数命令。

管理服务器在您安装 Sun ONE Application Server （非捆绑版本）或创建新域时安装。每个域只能有一个管理服务器，但可以创建多个由管理服务器管理的 Sun ONE Application Server 实例。使用管理服务器，可以访问每个应用程序服务器实例的配置设置、部署的应用程序和其它服务器功能。

有关管理域的详细信息，请参见第 3 章 “配置管理域”。

如果使用的是非捆绑版本的 Sun ONE Application Server，则在安装 Sun ONE Application Server 时，应捆绑为管理服务器选择端口号，或使用默认端口号 4848。

如果使用的是 Solaris 9 捆绑版的 Sun ONE Application Server，则必须手动创建域和管理服务器。有关配置 Solaris 9 捆绑版的详细信息，请参见第 31 页上的 “配置 Solaris 捆绑版”。

要访问管理界面，请在 Web 浏览器中键入：

http://*hostname.domain*:*port*/

请注意，此处的 *domain* 不是 Sun ONE Application Server 管理域，而是域名。

例如：

http://austen.sun.com:4848/

如果要从安装了 Sun ONE Application Server 的同一台计算机上访问管理服务器，可以使用：

http://localhost:4848

系统将提示您输入用户名和密码。

启动管理服务器

要启动或重新启动管理服务器，请使用以下主题介绍的方法之一：

- 使用 startserv 脚本
- 使用命令行界面
- 使用 “服务”窗口 (Windows)
- 使用 “开始”菜单 (Windows)

使用 startserv 脚本

要使用启动脚本启动管理服务器，请以超级用户的身份登录 （如果管理服务器运行所在的端口号小于 1024 [UNIX]），否则请以超级用户的身份或使用此服务器的用户帐户登录。在命令行提示处，转至目录：

install_dir/domains/*domain_dir*/admin-server/bin

其中， *install_dir* 是安装此服务器的目录， *domain_dir* 是管理域目录。

对于 UNIX，请键入：

./startserv

您可以在该行结尾使用可选参数 -i。管理服务器通常作为后台进程运行。-i 选项用于禁止管理服务器在后台运行。如果使用 /etc/inittab 运行此服务器，则此选项非常有用。

对于 Windows，请运行 startserv.bat 文件。

注意	如果管理服务器已经运行，则 startserv 命令将失败。必须先停止管理服务器，然后再使用 startserv 命令。此外，如果管理服务器启动失败，则应在尝试重新启动之前中止此进程。

使用命令行界面

命令行界面的 asadmin 公用程序包含 start-domain 命令，用于启动应用程序服务器和所有关联的 Sun ONE Application Server 实例。您只能在本地运行此命令，即从安装 Sun ONE Application Server 的计算机上运行。此命令不需要任何参数。

此外，您还可以使用 start-domain 命令启动管理域中的所有实例。此命令使用以下语法：

asadmin start-domain [--domain *domain-name*]

有关使用命令行界面的详细信息，请参见命令行界面的联机帮助和附录 A “使用命令行界面”。

使用 “服务” 窗口 (Windows)

要使用 Windows 中的 “服务控制面板” 来启动管理服务器，请执行以下步骤：

1. 在 “控制面板” 中，单击 “管理工具”。
2. 单击 “服务”。
3. 滚动服务列表并双击 “Application Server 7.0 Administration Server” 服务。

   为给定的域选择管理服务器。“服务” 窗口中的 “名称” 列将管理服务器的名称显示为 Sun App Server Admin Server (*your_domain_name*:admin-server)。
4. 单击 “启动”。
5. 单击 “确定”。

启动计算机时， Application Server 7.0 Administration Server 服务将自动启动。

使用 “开始” 菜单 (Windows)

要使用 Windows 的 “开始” 菜单启动管理服务器，请执行以下步骤：

1. 从 “开始” 菜单中，选择 “程序”。
2. 选择 “Sun Microsystems”。
3. 选择 “Sun ONE Application Server 7”。
4. 单击 “启动应用程序服务器”。

关闭管理服务器

管理服务器启动后，它将持续运行、监听并接收请求。如果更改管理服务器记录首选项或管理服务器的 HTTP 监听器监听的端口，则可能需要停止并重新启动此服务器。

停止管理服务器时，它将停止接收新的连接。然后，将等待所有未完成的连接全部完成。停止管理服务器后，您将无法使用管理界面或命令行界面。

可以使用以下主题介绍的方法之一来停止管理服务器：

- 使用管理界面关闭
- 使用 stopserv 脚本关闭
- 使用命令行界面关闭
- 使用 “服务”窗口关闭 (Windows)

关闭管理服务器后，可能需要几秒钟的时间完成关闭进程。

使用管理界面关闭

要使用管理界面关闭管理服务器，请执行以下步骤：

1. 在左窗格中，单击 “Admin Server”。
2. 单击 “Control”选项卡。
3. 单击 “停止”。

   单击此链接时，管理服务器将立即关闭，而不会显示其它屏幕。

使用 stopserv 脚本关闭

要手动停止管理服务器，请在命令提示符处转到以下目录：

install_dir/domains/*domain_dir*/admin-server/bin

其中， *install_dir* 是安装此服务器的目录， *domain_dir* 是域目录。

对于 UNIX，请键入：

./stopserv

必须以运行此服务器的用户身份运行此命令。

如果使用 */etc*/inittab 文件重新启动此服务器，则在尝试停止服务器前，必须从 */etc*/inittab 中删除启动此服务器的行，并键入 kill -1。否则，此服务器在停止后将自动重新启动。

对于 Windows，请运行 stopserv.bat 文件。

使用命令行界面关闭

可以使用命令行界面 asadmin 公用程序的 shutdown 命令停止管理服务器。此命令不需要任何参数，既可以在本地运行，也可以远程运行。

此外，您还可以使用命令行界面 asadmin 公用程序的 stop-appserv 命令，停止管理服务器和所有关联的 Sun ONE Application Server 实例。您只能在本地运行此命令，即从安装 Sun ONE Application Server 的计算机上运行。此命令不需要任何参数。

还可以通过使用 stop-domain 命令关闭域来关闭管理服务器。默认情况下，此命令将关闭域中的所有实例 （包括管理服务器）。您还可以对其进行配置，从而关闭域中的所有实例 （管理服务器*除外*）。此命令的语法如下：

asadmin stop-domain [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--adminserv=*true/false*] [--passwordfile *file_name*] [--secure | -s]

如果使用 local 选项，则该命令将在本地运行。如果使用 --adminserv=false 选项，则此命令将不停止管理服务器。但在默认情况下， --adminserv 将设置为 True，因此管理服务器将在默认情况下停止。

有关使用命令行界面的详细信息，请参见命令行界面的联机帮助和附录 A “使用命令行界面”。

使用 “服务”窗口关闭 (Windows)

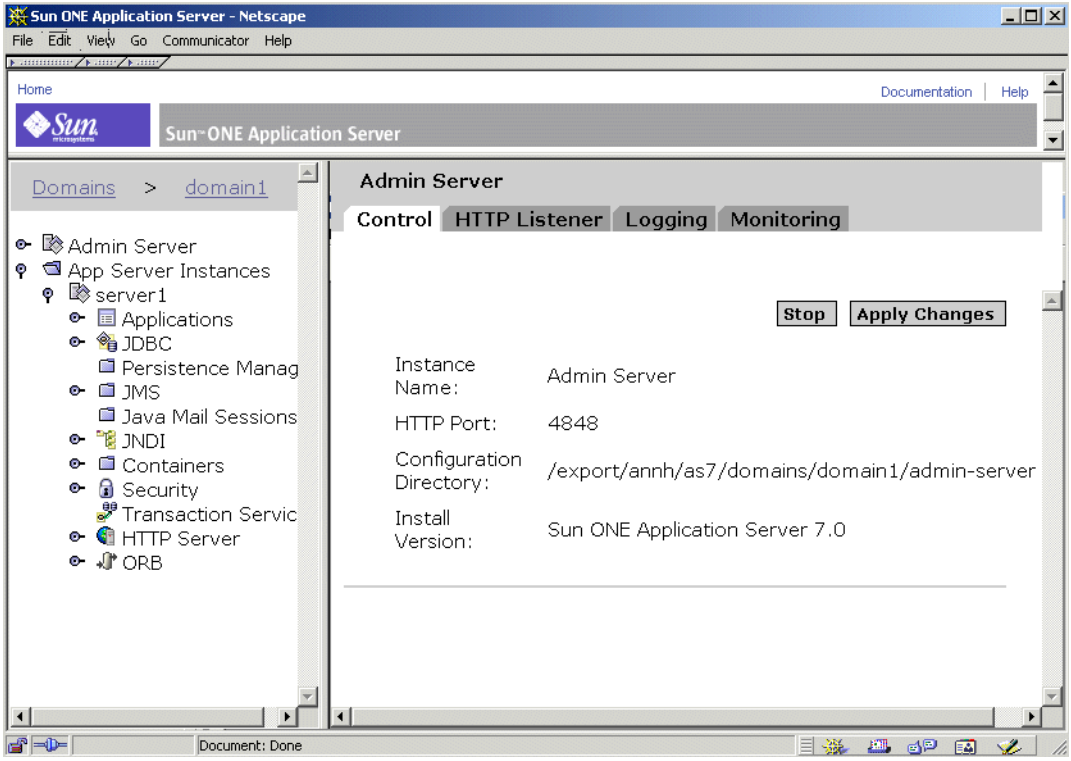
要使用 “服务”窗口关闭管理服务器，请执行以下步骤：

1. 在 “控制面板”中，单击 “管理工具”。
2. 单击 “服务”。
3. 滚动服务列表并双击 “Sun Application Server 7 Admin Server”服务。
4. 单击 “停止”。
5. 单击 “确定”。

访问管理服务器设置

要访问管理服务器设置，请在管理界面的左窗格中单击 “Admin Server”。管理服务器的设置显示在右窗格中，并按选项卡组的形式组织。

管理服务器用户界面



单击某一选项卡可以访问某一特定功能区域的设置。有时，单击某一选项卡可以直接转到某个页面，有时则显示从中进行选择的页面列表。

本章介绍 “Control”和 “HTTP Listener”选项卡。有关监视和 SNMP 设置的信息，请参见第 6 章 “监视 Sun ONE Application Server”。有关记录的信息，请参见第 5 章 “使用日志”。

查看管理服务器控制设置

管理服务器控制设置显示实例名称 (Admin Server)、管理服务器运行所在的端口、包含配置文件的目录和正在运行的 Sun ONE Application Server 软件的版本。

查看这些设置的步骤：

1. 在左窗格中，单击 “Admin Server”。
2. 单击 “Control”选项卡。

向管理服务器应用更改

使用管理界面或命令行界面更改管理服务器的配置信息时，可能需要应用所做的更改才能使它们生效。此操作也称作服务器重新配置。应用更改时，自上次应用更改以来对配置所做的所有更改都将生效。

对需要应用更改的管理服务器配置进行更改后，在左窗格树视图中的 “Admin Server”旁边将显示一个黄色图标。

警告图标

应用配置更改的步骤：

1. 在左窗格中，单击 “Admin Server”。
2. 单击 “Control”选项卡。
3. 单击 “Apply Changes”。

   应用更改后，屏幕将显示一个消息。

为管理服务器编辑 HTTP 监听器设置

管理服务器必须通过 HTTP 监听器接收请求，才能对其进行处理。

对于 Sun ONE Application Server 的非捆绑版本，安装时将自动创建管理服务器的 HTTP 监听器 http-listener-1。此 HTTP 监听器使用 IP 地址 0.0.0.0 和安装过程中指定为管理服务器端口号的端口号。默认端口号为 4848。您不能删除默认的 HTTP 监听器。

对于管理服务器实例 （在域中），只有 HTTP 监听器具有 http-listener-1 ID。也就是说，如果创建管理服务器实例，则只有一个 HTTP 监听器可以始终在 HTTP 或 HTTPS 协议中起作用。（此外，还意味着您不能同时与管理服务器建立 HTTP 和 HTTPS 连接）。有关配置 Solaris 9 捆绑版的详细信息，请参见第 31 页上的 “配置 Solaris 捆绑版”。

您可以在 HTTP 监听器上激活和配置管理服务器的 SSL/TLS 安全设置。

此外，您可以指定 HTTP 监听器中接收方线程 （有时称作接受线程）的数目。接受线程是等待连接的线程，用于接受连接并将其置于队列中以便随后由辅助线程拾取。理想情况下，您需要有足够的接受线程，以便在新请求传入时始终有一个可用的线程，但是，线程数不能过多，否则会占用过多的系统资源。默认线程数为 1。最好是系统上的每个 CPU 都有一个接受线程。如果发现性能下降，则可以调整此值。有关性能的详细信息，请参见 *《Sun ONE Application Server Performance Tuning and Sizing Guide》*。

编辑管理服务器的 HTTP 监听器设置的步骤：

1. 在左窗格中，单击 “Admin Server”。
2. 单击 “HTTP Listeners”选项卡。
3. 进行所需的更改并单击 “OK”。

有关 HTTP 监听器的详细信息，请参见联机帮助。

设置 SNMP、记录和安全首选项

有关 SNMP 设置的信息，请参见第 6 章 “监视 Sun ONE Application Server”。有关记录的信息，请参见第 5 章 “使用日志”。有关安全设置的信息，请参见 *《Sun ONE Application Server Administrator's Guide to Security》*。

第 3 章

配置管理域

本章介绍如何使用 Sun ONE Application Server 设置和管理管理域。

本章包括以下主题：

- 关于管理域
- 配置域
- 重新创建域注册表

关于管理域

管理域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计算机上特定的应用程序服务器实例组（域）。通过如此划分应用程序服务器实例，可以在不同的组织之间共享一台计算机，而且每个组织都拥有自己的管理员。

在 Sun One Application Server 中，每个应用程序服务器实例都是一个域的成员。并不需要有多个域，但是如果需要，作为一个非常有用的功能多个域是受支持的。

管理安全性是使用基础操作系统的安全性机制 （即通过文件权限）为本地命令建立的。远程命令安全性是使用用户名/密码对建立的，以便与特定的管理服务器通信。管理域不使用任何其它安全性结构。

本节包括以下主题：

- 实现管理域
- 目录结构
- 进程/端口结构

实现管理域

域是通过使用文件、操作系统进程和端口来实现的。每个域都有一个唯一的名称。

目录结构

有些文件 （配置文件、可执行文件等等）在安装中为所有的域共享。这里重点介绍那些专用于域的文件。

专用于域的所有文件都共享一个称为域目录的公用根目录，而该目录的名称为域的名称。在域目录下，每个实例一个目录，目录以实例名命名，并且每一个实例目录下面都含有专用于实例的文件。

域目录可以在文件系统的任何位置建立 （符合安全性权限和其它操作系统级别限制）。除非用户进行选择，否则域目录将在默认目录 （称为域目录）下建立。但是，用户也可以选择在任意位置创建域目录。

进程/端口结构

域运行时将占用操作系统进程和端口。具体而言，域中运行的每个实例 （包括该域的管理服务器）都占用一个进程和一个端口。

配置域

可以使用专门用途的命令创建、删除、列出、启动和停止域。

创建、删除和启动域只能在本地完成，而列出和停止域既可以在本地执行，也可以远程执行。

使用删除、启动和停止命令时，您需要分别指定一个域名。如果只有一个域，则不必指定此名称。如果未给定任何域，但配置了多个域，命令将给出错误信息。

本节包括以下主题：

- 创建域
- 删除域
- 列出域
- 启动域
- 停止域

创建域

域是使用 create-domain 命令创建的。此命令仅在本地执行。

语法：

asadmin create-domain [--path *domain_path*] [--sysuser *sys_user*] [--passwordfile *file_name*] --adminport *port_number* --adminuser *admin_user* --adminpassword *password domain_name*

示例：在默认位置创建域

$ asadmin create-domain --adminport 123 --adminuser MyAdmin --adminpassword MyPassword MyDomain

此示例将在默认位置（即域目录）创建名为 MyDomain 的域。管理服务器将监听端口 123，管理用户名将为 MyAdmin，而密码将为 MyPassword。域目录和它下面的文件将属于执行此命令的操作系统用户。此外，操作系统进程将以执行此命令的用户的名义运行。

如果已经存在名为 MyDomain 的域，将返回错误信息。

（请注意，如果不在命令行使用密码（避免可能的安全性问题），用户可以将密码放入一个文件并使用 --passwordfile 选项将它传送进去。

示例：在默认位置以外的位置创建域

```
$ asadmin create-domain --path $HOME --adminport 123 --adminuser MyAdmin
--adminpassword MyPassword MyDomain
```

此示例与第一个示例类似，只是域目录这一次将位于用户的 $HOME 目录下，而不是在默认的域目录下。

示例：为另一个用户创建域 （仅适用于 UNIX）

```
# asadmin  create-domain --user AnotherUser --adminport 123 --adminuser MyAdmin
--adminpassword MyPassword MyDomain
```

此示例与第一个示例类似，只是该域及其文件将由用户 AnotherUser 所有，操作系统进程也是如此

使用 --sysuser 选项，用户可以创建能够由其他用户进行后续管理的域。此选项要求运行 create-domain 命令的用户是超级用户。

UNIX 平台上的用户权限

要使非超级用户能够创建和删除管理域，必须将该用户的 ID 添加到具有域配置文件写入权限的 UNIX 组中：

1. 创建一个 UNIX 组，该组将适用于安装范围内的域配置文件。例如，名为 asadmin 的 UNIX 组。
2. 将位于 /etc/appserver 下的安装范围内的域配置文件设置为由新创建的 UNIX 组所有。

   这些文件名为 domains.bin 或 domains.lck。例如，更改分配给这些文件的组后：

   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.bin

   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
3. 启用新创建的 UNIX 组对这些文件的写入权限。在本示例中，生成的权限将类似于以下格式：

   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.bin

   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
4. 将用户 ID 添加到 UNIX 组中。

或者，如果您不希望为非超级用户提供对安装范围内的各配置文件的写入权限，也可以代表用户创建管理域。创建新管理域期间，请指定 --sysuser 和 --path 选项，以标识将拥有域目录和文件的 UNIX 用户 ID，以及创建管理域的位置。有关示例，请参见第 56 页上的 “示例：为另一个用户创建域 （仅适用于 UNIX）”。

在某个用户 ID 下创建管理域之后，该用户可以创建新的应用程序服务器实例，并对应用程序服务器实例执行多种不同的管理操作。该用户 ID 不必属于拥有管理域配置文件写入权限的 UNIX 组。 UNIX 组中的成员身份仅在创建和删除管理域时需要。

删除域

域是使用 delete-domain 命令删除的。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。此命令仅在本地执行。

语法：

asadmin delete-domain [*domain_name*]

示例：删除域

$ asadmin delete-domain MyDomain

此示例将在本地计算机上删除名为 MyDomain 的域。

列出域

在计算机中创建的域可以使用 list-domains 命令查找到。

此命令既可以在本地执行，也可以远程执行。

语法：

asadmin list-domains [--host *host*] [--port *port*] [--password *password*] [--user *user*]

示例：列出本地计算机中的域

$ asadmin list-domains

domain1 [/opt/ias/build/domains/domain1]

示例：使用远程选项列出本地计算机中的域：

$ asadmin list-domains --user admin --password password --host localhost --port 4848

domain1 [/opt/ias/build/domains/domain1]

启动域

可以使用 start-domain 命令启动域。此命令将启动域的管理服务器和域中的所有其它实例。

此命令只能在本地执行。

语法：

asadmin start-domain [--domain domain_name]

示例：启动计算机中唯一的域：

$ start-domain

Instance domain1:admin-server started

Instance domain1:server1 started

Domain domain1 Started.

停止域

可以使用 stop-domain 命令停止域。用户可以选择停止域中的每个实例，也可以选择停止除管理服务器之外的所有实例，以便能够远程管理该域。

此命令既可以在本地执行，也可以远程执行。

语法：

asadmin stop-domain [--user *admin_user*] [--password *admin_password*] [--host *host_name*] [--port *port_name*] [-- local=false] [--domain *domain_name*] [--adminserv=true] [--passwordfile *file_name*] [--secure | -s]

示例：停止域中除管理服务器实例之外的所有实例

$ asadmin asadmin stop-domain --user admin --password password --host localhost --port 4848 --adminserv=false --domain domain1

DomainStoppedRemotely

重新创建域注册表

要实现各种用途，每个域的详细资料 （其名称、位置、使用的端口等等）都记录在称为域注册表的文件中。

在正常的操作条件下，您不必直接对域注册表进行任何操作，因为对域注册表的任何修改或使用都将由管理系统使用的命令完全封装。但是，由于域注册表是一个文件，它可能会被破坏 （例如脚本产生错误、或注册表被某人无意中删除了等等）；而在这些情况下，您可能必须重新创建该文件。

注意 可以使用 asadmin 命令通过命令行界面访问域注册表。

如果注册表被破坏，请执行以下步骤重新创建注册表：

1. 获取所有域的列表，以及它们所在的目录 （默认目录或非默认目录）。
2. 重命名每个目录 （例如在每个目录名末尾附加 “.bak”）。
3. 使用端口、密码等的默认值，在其初始位置再次创建每个域。
4. 删除每个新的域目录并替换为原来的目录。
5. 对每个域执行 reconfig 命令。这将使域注册表用原有域的值进行更新。

第 2 部分

管理单个服务器实例

第 4 章 “使用应用程序服务器实例”

第 5 章 “使用日志”

第 6 章 “监视 Sun ONE Application Server”

第 7 章 “配置 Web 服务器插件”

第 8 章 “配置 J2EE 容器”

第 9 章 “使用事务服务”

第 10 章 “配置命名和资源”

第 11 章 “使用 JMS 服务”

第 12 章 “为 Corba/IIOP 客户机配置服务器”

第 13 章 “部署应用程序”

第 4 章

使用应用程序服务器实例

本章介绍了如何创建、删除、配置、启动和停止 Sun ONE Application Server 实例。

本章包括以下主题：

- 关于应用程序服务器实例
- 启动和停止应用程序服务器实例
- 在调试模式下启动应用程序服务器实例
- 设置终止超时
- 自动重新启动应用程序服务器实例 (UNIX)
- 手动重新启动应用程序服务器实例 (UNIX)
- 关于 Watchdog
- 添加应用程序服务器实例
- 删除应用程序服务器实例
- 应用对应用程序服务器实例所做的更改
- 查看应用程序服务器实例状态
- 配置 JVM 设置
- 配置日志设置和监视设置
- 更改应用程序服务器实例的高级设置

关于应用程序服务器实例

安装软件的非捆绑版本时， Sun ONE Application Server 将创建一个名为 server1 的应用程序服务器实例。如果需要，可以删除 server1 实例，并创建一个其它名称的新实例。

如果使用的是软件的 Solaris 9 捆绑版本，则必须自行创建服务器实例。有关详细信息，请参见第 31 页上的 “配置 Solaris 捆绑版”。

每个应用程序服务器实例都具有自己的 J2EE 配置、J2EE 资源、应用程序部属区和服务器配置设置，因此对一个应用程序服务器实例所做的更改不会影响其它应用程序服务器实例。一个管理域中可以包含多个应用程序服务器实例。域中的所有服务器实例都具有相同的管理服务器。有关域的详细信息，请参见第 3 章 “配置管理域”。

对于很多用户，一个应用程序服务器实例即可满足需要。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程序服务器实例。例如，在开发环境中，可以使用不同的应用程序服务器实例测试不同的 Sun ONE 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程序服务器实例很容易，因此可以在开发过程中使用这些实例创建临时 “沙盒”区来进行测试。

此外，还可以为每个应用程序服务器实例创建虚拟服务器。在一个已安装的应用程序服务器实例中，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程序服务器实例使用。有关虚拟服务器的详细信息，请参见第 15 章 “使用虚拟服务器”。

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程序服务器实例。但是，如果虚拟服务器不能满足需求，也可以使用多个应用程序服务器实例。

启动和停止应用程序服务器实例

Sun ONE Application Server 实例不会自动启动。启动某个实例后，该实例将一直运行，直到用户停止它。停止应用程序服务器实例后，它将不再接受新的连接，只是等待未完成的连接完成。如果计算机崩溃或脱机，服务器将退出，正在处理的所有请求都将丢失。

用户可以使用以下主题中所述的任意一种方法启动和停止应用程序服务器实例：

- 使用管理界面中的 “Start”和 “Stop”按钮
- 使用 start-instance 和 stop-instance 命令
- 使用 Windows 服务 (Windows)
- 使用 startserv 和 stopserv 脚本

注意	如果在服务器中安装了安全性模块，则需要在启动或停止服务器之前输入相应的密码。

如果安装了服务器证书， Sun ONE Application Server 会在启动之前提示管理员输入密钥数据库密码。如果希望重新启动一个无人看管的 Sun ONE Application Server，则需要将该密码保存在 password.conf 文件中。仅当系统受到充分的保护时才可以执行此操作，只有这样文件和密钥数据库才不会被损坏。有关创建和使用 password.conf 的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

注意	在 UNIX 中，某些 Sun ONE Application Server 安装需要访问的内存和/或文件描述符可能比默认情况下操作系统允许访问的内存和/或文件描述符多。如果无法启动服务器，请使用 ulimit 命令检查由操作系统强加的资源限制。有关详细信息，请参见操作系统的 ulimit 手册页。

使用管理界面中的 “Start” 和 “Stop” 按钮

使用管理界面启动和停止服务器的步骤：

1. 在左侧窗格中的 “App Server Instances” 下，单击要启动或停止的实例的名称。
2. 在右侧窗格中，单击 “Start” 或 “Stop”；或者在 “General” 选项卡中，单击 “Start” 或 “Stop”。
3. 成功启动或停止应用程序服务器实例后，会出现一条消息。

使用 start-instance 和 stop-instance 命令

使用命令行界面公用程序 asadmin，可以从命令提示符或脚本启动和停止应用程序服务器实例。可以使用 start-instance 和 stop-instance 命令。

这些命令使用以下语法：

start-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--debug=*true/false*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

stop-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

这些命令包含一个 local 选项，可以使用该选项启动或停止服务器，而无需通过管理服务器。如果使用 local 选项，则无需指定 host、 port、 user 和 password （或 passwordfile）选项。

有关这些命令的语法的信息，请参见 asadmin 帮助。有关使用 asadmin 的信息，请参见附录 A “使用命令行界面”。

使用 Windows 服务 (Windows)

用户可以使用 Windows 中的 “服务控制面板”启动服务器。

请执行以下步骤：

1. 在 “控制面板”中单击 “管理工具”。
2. 单击 “服务”。
3. 滚动服务列表并双击用于服务器的服务。

   该服务称作 Sun Application Server (*domain_name:instance_name*)。例如， Sun Application Server (domain1:server1)。
4. 单击 “启动”或 “停止”。
5. 单击 “确定”。

使用 startserv 和 stopserv 脚本

要使用 startserv 和 stopserv 脚本，请在命令行提示符下转至以下目录：

instance_dir/bin

其中 *install_dir* 为安装服务器的目录； *domain_dir* 为域目录； *instance_dir* 为要启动的实例的名称。

对于 UNIX，请键入：

./startserv

如果服务器在端口号小于 1024 的端口上运行，请以超级用户身份登录；否则，请以超级用户身份或使用服务器用户帐户登录。

用户可以在该行的末尾使用可选参数 -i。-i 选项在 inittab 模式下运行服务器，因此如果服务器进程终止或崩溃， inittab 会重新启动服务器。此选项还可以防止将服务器置于后台进程。

注意 如果服务器已经运行， startserv 命令将失败。必须先停止服务器，然后再使用 startserv 命令。同样，如果服务器启动失败，则应该在尝试重新启动之前终止该进程。

对于 Windows，请键入：

startserv

要手动停止服务器，请在命令行提示符下转至以下目录：

instance_dir/bin

其中 *install_dir* 为安装服务器的目录， *instance_dir* 为要启动的实例的名称。

对于 UNIX，请键入：

./stopserv

如果使用 /etc/inittab 文件重新启动服务器，则必须在尝试停止服务器之前从 /etc/inittab 文件中删除启动服务器的命令行并键入 kill -1 1。否则，服务器将在停止后自动重新启动。

对于 Windows，请键入：

stopserv

在调试模式下启动应用程序服务器实例

如果开发者需要调试 J2EE 应用程序，则可以在调试模式下运行应用程序服务器实例。

在调试模式下启动服务器的步骤：

1. 访问管理界面，并单击要在调试模式下启动的应用程序服务器实例的名称。
2. 单击 “General”选项卡。
3. 单击 “Run in Debug Mode”旁边的复选框。
4. 重新启动应用程序服务器实例。

调试模式可以更改 JVM 设置。将 “Debug Enabled”设置为 True 后，“Debug Options”将随之改变。有关 JVM 调试选项的详细信息，请参见 http://java.sun.com/products/jpda/doc/conninv.html 上的 Java Platform Debugger Options 文档。

要从命令行界面在调试模式下启动应用程序服务器实例，请使用 asadmin 公用程序的 start-instance 命令并将 debug 选项设置为 True。有关命令语法的详细信息，请参见命令行界面的联机帮助。

设置终止超时

停止应用程序服务器实例后，它将不再接受新的连接，而只是等待所有未完成的连接完成。在 init.conf 文件 （可以在 *instance_dir*/config/ 中找到该文件）中可以配置服务器在超时之前等待的时间。默认情况下，该时间设置为 30 秒。要更改该值，请将下面一行文本添加到 init.conf 文件中：

TerminateTimeout *seconds*

其中 *seconds* 代表服务器在超时之前等待的秒数。

配置此值的优点是：服务器将等待更长的时间，以便于连接完成。但是，由于服务器通常从非响应的客户机打开连接，因此增加终止超时可能会增加服务器关闭所用的时间。

自动重新启动应用程序服务器实例 (UNIX)

可以使用下列方法之一重新启动应用程序服务器实例：

- 自动从 /etc/inittab 文件重新启动。

  请注意，如果所使用的 UNIX 版本不是源自系统 V，则无法使用 /etc/inittab 文件。

- 重新引导计算机时，会自动使用 /etc/rc2.d 中的守护程序重新启动。
- 手动重新启动。请参见第 64 页上的 “启动和停止应用程序服务器实例”和第 73 页上的 “删除应用程序服务器实例”。

本节包括以下主题：

- 关于自动重新启动
- 使用 /etc/inittab 自动重新启动 (UNIX)
- 通过系统 RC 脚本自动重新启动 (UNIX)

关于自动重新启动

由于安装脚本无法编辑 /etc/rc.local 或 /etc/inittab 文件，因此必须使用文本编辑器对其进行编辑。如果不知道如何编辑这些文件，请向系统管理员咨询或参阅系统文档。

通常情况下，不能使用以上的任意文件来启动启用了 SSL 的服务器，因为启动之前服务器会要求输入密码。尽管可以通过将密码以纯文本格式存储在某个文件中来自动启动启用了 SSL 的服务器，但建议*不要*使用这种方法。

警告 将启用了 SSL 的服务器的密码以纯文本格式存储在服务器的 startserv 脚本中会带来很大的安全风险。任何可以访问该文件的用户都有权访问启用了 SSL 的服务器的密码。将启用了 SSL 的服务器的密码保存为纯文本格式之前，请考虑可能带来的安全风险。

服务器的 startserv 脚本、密钥对文件和密钥密码应属于超级用户 （如果该服务器由非超级用户安装，则属于该用户帐户），并且只有所有者具有读写权限。

使用 /etc/inittab 自动重新启动 (UNIX)

要使用 inittab 重新启动服务器，请将下面一行文本添加到 /etc/inittab 文件中：

http:2:respawn:*install_dir*/*path_to_domain_dir*/*instance_dir*/bin/startserv -start -i

其中 *install_dir* 为安装服务器的目录； *path_to_domain_dir* 为域的路径；*instance_dir* 为服务器的目录。

-i 选项可以防止将服务器置于后台进程。

停止服务器之前必须删除此行，否则服务器将自动重新启动。

通过系统 RC 脚本自动重新启动 (UNIX)

如果使用 /etc/rc.local 或系统的等效文件，请将下面一行文本添加到 /etc/rc.local 文件中：

install_dir/*path_to_domain_dir*/*instance_dir*/bin/startserv

将 *install_dir* 替换为安装服务器的目录、将 *path_to_domain_dir* 替换为域的路径、并将 *instance_dir* 替换为应用程序服务器实例的名称。

手动重新启动应用程序服务器实例 (UNIX)

UNIX 中提供了手动重新启动服务器实例的选项。与先停止服务器实例然后再启动该实例的操作不同，重新启动不停止 watchdog 程序。有关 watchdog 的信息，请参见第 71 页上的 “关于 Watchdog”。

注意 如果已通过编辑配置文件对其进行了手动更改，则必须在重新启动服务器之前应用更改，方法是：使用管理界面中的 “Apply Changes” 按钮，或者使用 asadmin reconfig 命令 （其中将 keepmanualchanges 选项设置为 True）。有关应用更改的详细信息，请参见第 73 页上的 “应用对应用程序服务器实例所做的更改”。

在以下主题中介绍了三种重新启动服务器实例的方法：

- 使用 “Restart” 按钮重新启动服务器实例 (UNIX)
- 使用 restart-instance 命令重新启动服务器实例 (UNIX)
- 使用 restartserv 脚本重新启动服务器实例 (UNIX)

使用 “Restart” 按钮重新启动服务器实例 (UNIX)

使用管理界面重新启动服务器实例的步骤：

1. 在左侧窗格中的 “App Server Instances” 按钮下，单击要重新启动的实例的名称。
2. 在右侧窗格中，单击 “Restart”。
3. 成功重新启动应用程序服务器后，会出现一条消息。

使用 restart-instance 命令重新启动服务器实例 (UNIX)

使用命令行界面公用程序 asadmin，可以从命令行或脚本启动和停止应用程序服务器实例。可以使用 restart-instance 命令。此命令使用以下语法：

restart-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--local=*true/false*] [--domain *domain_name*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

此命令包含一个 local 选项，可以使用该选项重新启动服务器实例，而无需通过管理服务器。

有关这些命令的语法的信息，请参见 asadmin 帮助。有关使用 asadmin 的信息，请参见附录 A “使用命令行界面”。

使用 restartserv 脚本重新启动服务器实例 (UNIX)

要使用 restartserv 脚本，请在命令行提示符下转至以下目录：

instance_dir/bin

其中 *install_dir* 为安装服务器的目录； *domain_dir* 为域目录； *instance_dir* 为要启动的实例的名称。

请键入：

./restartserv

如果服务器在端口号小于 1024 的端口上运行，请以超级用户身份登录；否则，请以超级用户身份或使用服务器用户帐户登录。

关于 Watchdog

Watchdog （在 UNIX 中为 appserv-wdog，在 Windows 中为 appservd-wdog.exe）是 Sun ONE Application Server 附带的程序。它用于执行以下任务：

- 启动服务器
- 停止服务器
- 在服务器启动时提示管理员输入信任数据库密码 （如果启用了 SSL/TLS）
- 在服务器出现故障时重新启动服务器

Watchdog 在后台运行，无需用户的介入。请勿对其进行配置或以其它方式进行更改。每个应用程序服务器实例运行一个 Watchdog （包括管理服务器）。

在 UNIX 中，每个 Watchdog 可以为原始应用程序服务器 (appservd) 的处理产生一个进程，该进程又会反过来产生接受请求的 appservd 进程。由于 Watchdog 进程标识将用于启动服务器，因此会显示在 *instance_dir*/logs 中的 pid 日志文件中。

注意 UNIX 平台上的 appservd 进程：用户可能会注意到在 Windows 中为每个应用程序服务器实例启动了一个 appservd 进程，而在 UNIX 系统中却为每个应用程序服务器实例启动了两个 appservd 进程。

在 UNIX 中，第一个 appservd 进程指“原始”进程，第二个 appservd 进程指“工作”进程。工作进程是处理实际应用程序请求的进程，而原始进程则是作为首要的控制器。在以后的应用程序服务器发行版中，将具有可以为每个应用程序服务器实例定义工作进程数目的选项。在产品的最初发行版中，每个应用程序服务器实例只支持一个工作进程。

添加应用程序服务器实例

使用管理界面添加应用程序服务器实例的步骤：

1. 访问管理界面，并在左侧窗格中单击 “App Server Instances”。
2. 单击 “General”选项卡。
3. 在 “Application Server Instances”页面中单击 “New”。
4. 在 “Create New Instance”页面中，提供实例名和端口号。

   对于此管理服务器和域，实例名必须唯一。计算机上的任何其它进程都不能使用该端口号。

   如果使用的是 UNIX 系统，也可以为实例指定 UNIX 用户，使其作为 UNIX 用户运行。
5. 单击 “OK”。

有关详细信息，请参见联机帮助。

要使用命令行界面添加其它应用程序服务器实例，请使用 asadmin 公用程序的 create-instance 命令，该命令使用以下语法：

asadmin create-instance [--user *admin_user*] [--password *admin_password*] [--host *host*] [--port *port*] [--sysuser *sys_user*] [--domain *domain_name*] [--local=*true/false*] [--passwordfile *file_name*] [--secure | -s] --instanceport *instance_port instance_name*

此命令包含一个 local 选项，可以使用该选项重新启动服务器实例，而无需通过管理服务器。 sysuser 选项仅适用于 UNIX 系统。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

删除应用程序服务器实例

用户可以从管理域中删除应用程序服务器实例。由于此操作不能撤消，因此在删除应用程序服务器实例之前请确保不再需要该实例。

使用管理界面从计算机中删除应用程序服务器实例的步骤：

1. 访问管理界面，并单击要删除的应用程序服务器实例的名称。
2. 单击 “General”选项卡。
3. 单击 “Delete”。

有关详细信息，请参见联机帮助。

要使用命令行界面从计算机中删除应用程序服务器实例，请使用 asadmin 公用程序的 delete-instance 命令，该命令使用以下语法：

asadmin delete-instance [--user *admin_user*] [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--domain *domain_name*] [--local=*true/false*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

此命令包含一个 local 选项，可以使用该选项删除服务器实例，而无需通过管理服务器。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

应用对应用程序服务器实例所做的更改

使用管理界面或命令行界面更改配置信息时，系统不会立即应用所做的更改，而是将其保存在 *server_instance*/config/backup 目录的特定文件中。管理界面和命令行界面将显示存储在以上目录的文件中的配置值。应用更改后，更改才会生效。应用更改也称为重新配置服务器。应用更改后，自上次更改应用后对配置所做的更改将生效。请注意，重新启动实例不会自动应用更改。

如果已对要求应用更改的服务器实例配置进行了更改，左侧窗格树视图中的应用程序服务器实例旁边、访问服务器实例时的标题中和服务器实例主页面上将出现一个黄色图标。

警告图标

使用管理界面应用对应用程序服务器实例所做的更改的步骤：

1. 访问管理界面，并单击要重新配置的应用程序服务器实例的名称。
2. 单击 “General” 选项卡。
3. 单击 “Apply Changes”。

   应用更改后，屏幕上将显示一条消息。

要使用命令行界面重新配置应用程序服务器实例，请使用 asadmin 公用程序的 reconfig 命令，该命令使用以下语法：

asadmin reconfig --user *admin_user* [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--passwordfile *file_name*] [--secure | -s] [--discardmanualchanges=*true/false* | --keepmanualchanges=*true/false*] *instance_name*

如果已通过手动编辑的方式对配置文件进行了更改，则在配置过程中必须使用 keepmanualchanges=true 保留编辑的内容 （该选项的默认值为 False）。如果设置了 discardmanualchanges=true，则会放弃手动进行的更改。设置 discardmanulchanges=false （默认值）与设置 keepmanualchanges=true 意义不同。将 discardmanualchanges 选项设置为 False 相当于未指定该选项。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

对于某些属性，需要应用更改并重新启动服务器才能使更改生效。这些属性包括在 init.conf 和 obj.conf 配置文件中设置的所有属性，以及 server.xml 中的某些属性。有关这些文件的信息，请参见 《*Sun ONE Application Server Administrator's Configuration File Reference*》。

如果更改要求重新启动服务器实例，服务器将在服务器实例左侧窗格中的树视图旁边、访问服务器实例时的标题中和服务器实例主页面上显示黄色警告图标来警告用户。标题和页面中的消息表示需要重新启动服务器实例。重新启动服务器实例后，黄色警告图标将消失。

*不*要求重新启动的 server.xml 设置包括：

- 部署、取消部署和重新部署 J2EE 应用程序 （EAR 文件）、 EJB 模块 （JAR 文件）、 Web 模块 （WAR 文件）和连接器 （RAR 文件）。请注意，这些设置也不要求应用更改。

- 启用和禁用 J2EE 应用程序 （EAR 文件）、 EJB 模块 （JAR 文件）、 Web 模块 （WAR 文件）和连接器 （RAR 文件）。
- 创建、更新和删除资源。
- 将 EJB 容器或 MDB 容器的启用监视设置为 True 或 False。
- 对 HTTP 和 Web 容器的特征所做的更改 （也就是在 server.xml 中对 HTTP 服务、 Web 容器及其子元素所做的更改）。

查看应用程序服务器实例状态

使用管理界面，可以查看服务器已启动还是已停止以及基本的应用程序服务器实例设置。

查看应用程序服务器实例状态的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在右侧窗格中，单击 “General”选项卡。

   查看主机名、端口号、安装目录和 Sun ONE Application Server 软件的版本以及服务器是否正在运行。

要使用命令行界面查看应用程序服务器实例的状态，请使用 asadmin 公用程序的 show-instance-status 命令。其状态为正在启动、已启动、正在停止或已停止。该命令使用以下语法：

asadmin show-instance-status --user *admin_user* [--password *admin_password*] [--host *admin_host*] [--port *admin_port*] [--passwordfile *file_name*] [--secure | -s] *instance_name*

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

配置 JVM 设置

可以为应用程序服务器实例配置 Java 虚拟机 (JVM) 设置。这些设置包括 Java 主页的位置、编译程序选项、调试选项和事件探查器信息。配置这些设置的原因之一是为了改善性能。有关性能的详细信息，请参见 *《Sun ONE Application Server Performance and Tuning Guide》*。

本节包括以下主题：

- 配置常规设置
- 配置路径设置
- 配置 JVM 选项
- 配置 JVM 事件探查器
- 使用命令行界面配置 JVM 设置

配置常规设置

在管理界面中配置 JVM 的常规选项的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在右侧窗格中，单击 “JVM” 选项卡。
3. 单击 “General”。
4. 设置 Java 主页。

   Java 主页是安装 Java 开发工具 (JDK) 的目录的路径。 Sun ONE Application Server 支持 Sun JDK 1.4.0_02 或更高版本。
5. 选择是否启用调试并设置调试选项。

   可以从 http://java.sun.com/products/jpda/doc/conninv.html#Invocation 获得调试选项的列表。
6. 选择 RMIC 选项。

   RMIC 选项字段用于显示在部署应用程序时传递给 RMI 编译程序的 RMIC 选项。 -keepgenerated 选项用于保存为占位程序和捆绑程序生成的源代码。有关 RMIC 命令的详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Enterprise Java Beans》*。
7. 单击 “Save”。

配置路径设置

在管理界面中配置 JVM 的路径设置的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在右侧窗格中，单击 “JVM” 选项卡。
3. 单击 “Path Settings”。
4. 为系统的类路径选择后缀。
5. 选择是否忽略环境类路径。

   如果不忽略类路径，系统将读取 CLASSPATH 环境变量并将其附加到 Sun ONE Application Server 类路径。 CLASSPATH 环境变量将被添加到类路径后缀的后面，即类路径的最后端。

   对开发环境而言，应使用类路径。对生产环境而言，应忽略类路径，以防止环境变量产生不好的影响。
6. 设置本地库路径的前缀和后缀。

   本地库路径是应用程序服务器安装本地共享库的相对路径、标准 JRE 本地库路径、 shell 环境设置 （UNIX 中的 LD_LIBRARY_PATH）和在事件探查器元素中指定的任意路径的结构自动合并。由于这是一个合成的路径，因此不会明确地显示在服务器配置中。
7. 单击 “Save”。

配置 JVM 选项

在管理界面中设置 JVM 命令行选项的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在右侧窗格中，单击 “JVM” 选项卡。
3. 单击 “JVM Options”。
4. 要添加 JVM 选项，请在屏幕顶部的文本字段中键入该选项并单击 “Add”。
5. 要删除 JVM 选项，请单击该选项旁边的复选框并单击 “Delete”。
6. 要编辑 JVM 选项，请在 “JVM Option” 字段中编辑文本并单击 “Save”。

有关特定的 JVM 选项的信息，请访问
http://java.sun.com/docs/hotspot/VMOptions.html

配置 JVM 事件探查器

在管理界面中配置 JVM 事件探查器的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在右侧窗格中，单击 “JVM”选项卡。
3. 单击 “Profiler”。
4. 指定事件探查器的名称、类路径和本地库路径，以及是否启用了事件探查器。
5. 要为事件探查器添加 JVM 选项，请在屏幕顶部的文本字段中键入该选项并单击 “Add”。
6. 要删除事件探查器的 JVM 选项，请单击该选项旁边的复选框并单击 “Delete”。
7. 要编辑事件探查器的 JVM 选项，请在 “JVM Option”字段中编辑文本并单击 “Save”。

有关事件探查器的详细信息，请参阅 *《Sun ONE Application Server Developer's Guide》*。

使用命令行界面配置 JVM 设置

要使用命令行界面的 asadmin 公用程序配置 JVM 设置，请使用以下命令：

从实例获取所有属性：

asadmin> get *server_instance*.java-config.*

获取 server1 中名为 classpathprefix 的属性：

asadmin> get server1.java-config.classpathprefix

设置 server1 中名为 classpathprefix 的属性：

asadmin> set server1.java-config.classpathprefix=com.sun

以上示例都假设已在环境变量中设置了用户、密码、主机和端口。要获得属性的完整列表，请参见附录 A “使用命令行界面”。

要使用命令行界面的 asadmin 公用程序设置 JVM 选项，请使用以下命令：

asadmin> create-jvm-options --user *admin_user* [--password *admin_password*] [--host *host*] [--port *port*] [--secure | -s] [--instance *instance_name*] [--profiler=*true/false*] (*jvm_option_name=jvm_option_value*)[:*jvm_option_name=jvm_option_name*]*

asadmin> delete-jvm-options --user *admin_user* [--password *admin_password*] [--host *host*] [--port *port*] [--secure | -s] [--instance *instance_name*] [--profiler=*true/false*] (*jvm_option_name=jvm_option_value*)[:*jvm_option_name=jvm_option_name*]*

注意：可以输入多个 JVM 选项，中间用冒号隔开。如果这些选项由事件探查器使用，请将 --profiler 设置为 True。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

配置日志设置和监视设置

“Logging”和 “Monitoring”选项卡中的设置就是在各章中介绍的日志和监视设置。有关日志的信息，请参见第 5 章 “使用日志”。有关监视和 SNMP 设置的信息，请参见第 6 章 “监视 Sun ONE Application Server”。

更改应用程序服务器实例的高级设置

应用程序服务器实例具有附加的设置，这些设置可以显示实例的语言环境 （用于确定字符集和语言等设置）、到服务器日志文件的路径、到已部署应用程序的目录的路径以及到会话存储目录 （其中存储非活动 Bean 和持久 HTTP 会话）的路径。

此外，可以启用应用程序重新装入和指定重新装入频率的轮询时间间隔。动态应用程序重新装入将自动检查应用程序是否更改，如果已更改，它将自动处理更新后的版本。一般情况下，应该在开发环境 （而不是生产环境）中启用动态重新装入。轮询时间间隔用于指定应用程序服务器检查应用程序更新的时间间隔。

使用管理界面更改应用程序服务器实例设置的步骤：

1. 在左侧窗格中，单击应用程序服务器实例的名称。
2. 在应用程序服务器实例的页面中，单击 “Advanced”选项卡。
3. 在字段中输入所需的值。
4. 单击 “Save”。

要使用 asadmin 公用程序更改服务器实例的高级设置，请使用 get 和 set 命令。当获取服务器实例的所有属性时

从实例获取所有属性：

asadmin get *instance_name*.*

例如：

asadmin get server1.*

获取 server1 中名为 logRoot 的属性：

asadmin get server1.logRoot

为 server1 设置名为 logRoot 的属性：

asadmin set server1.logRoot=/space/log

以上例子都假设已在环境变量中设置了用户、密码、主机和端口。有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

第 5 章

使用日志

本章介绍了 Sun ONE Application Server 的日志特征和功能。此外，还讨论了日志可能使用的各种组件。

本章包括以下主题：

- 关于日志
- UNIX 和 Windows 平台上的日志
- 使用日志级别
- 关于虚拟服务器和日志
- 关于记录器
- 关于客户端日志
- 重定向应用程序和服务器日志输出
- 日志文件管理
- 通过命令行界面配置日志
- 通过管理界面配置日志
- 为错误日志配置指令
- 查看访问日志文件
- 查看事件日志文件
- 设置日志首选项
- 运行日志分析程序
- 查看事件 (Windows 2000 Pro)

关于日志

当用于应用程序时，日志是一种非常有用的调试和诊断工具。它还可以提高开发者的生产效率。应用程序服务器自身的日志输出可以帮助用户标识和诊断服务器配置以及有关部署的问题。

Sun ONE Application Server 中的日志使用 Java 日志 API。 Sun ONE Application Server 收集日志信息并将其存储在名为 access.log 和 server.log 的日志文件中，这两个日志文件位于 logs 目录中。用户也可以将日志定向到自己的日志文件。

除了消息本身以外，日志消息还可以提供更多的信息。所提供的附加信息包括：

- 事件的日期和时间。
- 事件的日志级别。应用程序服务器指定的日志级别 ID 或名称。
- 进程 ID (PID)。应用程序服务器进程的 PID。
- （可选）虚拟服务器 ID (vsid)。生成该消息的 vsid。
- 消息 ID。子系统和四位整数。
- 消息数据。

附加消息信息的种类和顺序取决于日志使用的平台以及该平台启用的日志服务。要启用日志消息的虚拟服务器 ID，请参见第 103 页上的 “配置日志服务”。

UNIX 和 Windows 平台上的日志

本节介绍如何创建日志文件。此外，还包括以下主题：

- server.log 中的默认日志
- 使用 syslog 记录日志
- 使用 Windows 事件日志记录日志

server.log 中的默认日志

在 UNIX 和 Windows 平台上，日志文件都在 server.log （位于 log 子目录中）中创建。来自一个实例的所有服务器组件和虚拟服务器的日志都存储在此文件中。

可以设置整个服务器的默认日志级别，但是，也可以在子系统级别上覆盖某个特定子系统的默认日志级别。还可以将 stdout 和 stderr 重定向到服务器的事件日志，将日志输出定向到操作系统的系统日志。此外，还可以将 stdout 和 stderr 内容定向到服务器的事件日志。默认情况下，日志消息除了发送到指定的服务器日志文件以外，还将发送到 stderr。

另一个可用的功能是使用日志消息记录虚拟服务器 ID。当使用多个虚拟服务器将消息记录到同一个日志文件时，此功能很有用。可以选择将日志消息写入系统日志。执行此操作时，不会在 server.log 文件中进行日志记录，而是使用 UNIX 中的 syslog 日志服务或 Windows 平台上的系统日志服务来生成和管理日志。

还可以使用 server.xml 属性来控制此文件的内容。有关 server.xml 文件的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

server.log 的示例

以下是 server.log 的一个示例。

时间标记类，日志级别，（PID vsid [可选]）：消息 ID：消息

[01/Aug/2002:11:39:31] INFO (1224):CORE1116:Sun ONE Application Server 7.0

[01/Aug/2002:11:39:36] INFO (1224):CORE5076:Using [Java HotSpot™ Server VM, Version 1.4.0_02-20020712] from [Sun Microsystems Inc.]

[01/Aug/2002:11:39:50] INFO (1224):JMS5023:JMS service successfully started.Instance Name = domain1_server1, Home = [D:\install_7_29\imq\bin].

[01/Aug/2002:11:39:53] INFO (1224):CIS0056:Creating TCP ServerConnection at [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:53] INFO (1224):CIS0057:Created TCP ServerConnection at [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:54] INFO (1224):CIS0054:Creating TCP Connection from [-] to [EndPoint [IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

注意 重定向与预编译的 JSP 相关的日志消息：

默认情况下，与预编译的 JSP 相关的日志消息存储在管理服务器日志文件中，该文件位于 {*domain_root*}/{*domain_name*}/admin-server/logs/server.log 中。

因为所有的消息都记录到同一个文件，所以使用预编译的 JSP 部署应用程序时产生的异常或错误消息可能会淹没在公共日志文件的大量消息中。当将多个应用程序部署到给定域下的多个实例时，需要仔细检查管理服务器中的日志消息是否有关于特定应用程序 JSP 的异常信息。这将造成冗余。

因此，最好在服务器实例的 server.log 文件 （而不是管理服务器的 server.log 文件）中记录与使用预编译的 JSP 部署的应用程序的相关消息。

要将日志消息重定向到 Sun ONE Application Server 实例的 server.log 文件，请在管理员界面中更改日志文件的路径。详细信息，请参见配置日志服务。

使用 syslog 记录日志

syslog 适用于要求集中记录日志的稳定的可操作环境。在经常要求日志输出以进行诊断和调试的环境中，单个的服务器实例或虚拟服务器日志可能比较容易管理。

注意

- 服务器实例和管理服务器的所有日志数据都在一个文件中可能难于读取和调试。建议将 syslog 主日志文件仅用于已部署且正在顺利运行的应用程序。
- 日志消息将与 Solaris 守护程序应用程序中的所有其它日志相混合。

通过将 syslog 日志文件与 syslogd 以及系统日志守护程序一起使用，可以将 syslog.conf 文件配置为：

- 将消息记录到相应的系统日志
- 将消息写入系统控制台
- 将日志消息转发到一组用户，或通过网络将其转发到另一台主机上的另一个 syslogd

注意 安装 Sun ONE Application Server 之后，该服务器的日志服务元素属性 use-system-logging 并没有启用。这说明在默认情况下日志不会定向到 UNIX 上的 syslog 或 Windows 平台上的 Windows 事件日志。可以按照《Sun ONE Application Server Configuration File Reference》中所述，启用 server.xml 的 Server 元素中的日志服务元素属性，将日志定向到 syslog 或 Windows 事件日志。在设置 use-system-logging 之前，请参见第 107 页的 “日志文件管理”。

配置 syslog

要提高可管理性和可读性，可以通过配置 /etc 目录中的 syslog.conf 将严重程度不高的消息定向到单独的文件中。

配置 syslog 的步骤：

1. 要将严重程度不高的消息定向到单独的文件中，请将以下命令添加到 Solaris 中的 syslog.conf 文件中：

```
daemon.debug  /var/adm/iasdebug
```

注意 如果将日志消息定向到 Windows 事件日志，将仅记录严重级别为 INFO、 WARNING、 SEVERE、 ALERT 或 FATAL 的消息。

2. 向 syslogd 发出挂起信号。可以使用以下命令完成此操作：

```
kill -HUP <PID syslogd>
```

3. 转到管理界面中的管理服务器并选择 “Write to system log”选项。保存并应用所做的更改。重新启动管理服务器，使更改生效。

以下是一个已配置的 Solaris syslog.conf 文件的示例：

```
#ident"@(#)syslog.conf1.598/12/14 SMI"/* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
```

```
# that match m4 reserved words.Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice/dev/sysmsg
*.err;kern.debug;mail.crit/var/adm/messages
daemon.info;daemon.err;daemon.debug;daemon.alert;daemon.crit;daemon.warning
/var/adm/iaslog
daemon.debug/var/adm/iasdebug
#daemon.notice;                 /var/adm/iaslognotice
#daemon.warning;                /var/adm/iaslogwarning
#daemon.alert;                  /var/adm/iaslogalert
#daemon.err;                    /var/adm/iaslogerr

#*.alert;kern.err;daemon.erroperator
#*.alert                        root
*.emerg                         *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice  ifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debug    ifdef(`LOGHOST', /var/log/syslog, @loghost)
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                /dev/sysmsg
user.err                /var/adm/messages
user.alert              `root, operator'
user.emerg              *
```

)

详细信息，请参见 syslog.conf 手册页。

要使对 syslog.conf 所做的任何更改生效，都需要重新启动 Sun ONE Application Server。

因为将日志记录到 syslog 意味着来自所有 Sun ONE Application Server 以及其它守护程序应用程序的日志都存储在同一个文件中，所以日志消息中增加了以下信息，以便标识来自特定服务器或虚拟服务器实例中专用于 Sun ONE Application Server 的消息：

- 唯一消息 ID
- 时间标记类
- 实例名
- 程序名 （appservd 或 appserv-wdog）
- 进程 ID （应用程序服务器进程的 PID）
- 线程 ID （可选）
- 服务器 ID

可以在 server.xml 文件中为服务器实例和虚拟服务器实例配置日志服务。针对虚拟服务器实例的日志服务配置在第 91 页上的 “关于虚拟服务器和日志”中介绍。针对服务器实例的日志服务配置在第 103 页上的 “通过管理界面配置日志”中介绍。

通过管理界面可以为可用的子系统和组件配置日志级别。

要获得有关 UNIX 操作环境所使用的 syslog 记录机制的详细信息，请在出现终端提示后使用以下手册命令：

man syslog

man syslogd

man syslog.conf

syslog 消息的示例

以下是 syslog 消息的一个示例。

时间标记类，主机名 [instance_name]，[子系统]，[vsid]，消息 ID，日志级别，消息数据

Jul 19 14:33:18 strange /usr/lib/nfs/lockd[164]:[ID 599441 daemon.info] Number of servers not specified.Using default of 20.

Jul 19 14:33:20 strange ntpdate[181]:[ID 558275 daemon.notice] adjust time server 192.18.56.149 offset 0.06702 6 sec

Jul 19 14:38:13 strange xntpd[248]:[ID 204180 daemon.info] synchronisation lost

Jul 19 14:38:47 strange server1 appservd[374]:[ID 702911 daemon.info] INFO (374):CORE1116:Sun ONE Application Server 7.0

Jul 19 14:38:48 strange server1 appservd[374]:[ID 702911 daemon.info] FINE (374):Collecting statistics for up to 1 processes with 128 threads, 200 listen sockets, and 1000 virtual servers

使用 Windows 事件日志记录日志

有关 Windows 操作环境所使用的事件日志机制的详细信息，请在 Windows 帮助系统索引中查找关键字 *事件日志*。

使用日志级别

本节介绍了各种日志级别以及如何为每个 Sun ONE Application Server 子系统指定日志级别。

其中包括以下主题：

- 关于日志级别
- 用于 syslog 配置的日志级别

关于日志级别

Sun ONE Application Server 使用标准的 JDK 1.4 日志级别进行选择性的信息记录。除了标准的 JDK 日志级别以外， Sun ONE Application Server 还添加了一些日志级别，旨在与 server.log 进行更直观的映射并与 Solaris 进行紧密的集成。

当路由到 server.log 时，日志消息还将映射到在第 90 页上的 “映射到 server.log 的 Sun ONE Application Server 日志级别”中定义的日志级别。

注意 管理服务器和默认应用程序服务器实例的 server.log 文件（或 syslog）的默认日志级别为 INFO。当应用程序服务器实例使用默认的日志级别时，将记录错误消息和信息消息。要避免记录此类消息，请在 server.xml 文件或管理服务器和服务器实例的管理界面中将日志级别更改为 WARNING 或 SEVERE。

服务器的默认日志级别可以在 log-service 元素中设置。此操作将影响所有日志级别设置为 “默认”的元素。

可以为每一个启用了日志记录的 Sun ONE Application Server 子系统指定日志级别。压缩运行期间所记录的消息信息总量时，日志级别很有用。要使用的子系统的级别是在 server.xml 文件中指定的。可以从管理界面为选定的子系统指定日志级别，也可以直接编辑 server.xml 文件以设置选定子系统所需的日志级别。

警告 对 server.xml 文件进行手动编辑可能会产生语法错误，从而导致服务器启动失败。手动编辑配置文件的指南将在 *《Sun ONE Application Server Administrator's Configuration File Reference》*中的 “Manually Editing Configuration Files”小节中进行介绍。

第 94 页上的 “JMS 服务的日志级别”图中显示了通过管理界面设置日志级别的示例。要直接在 server.xml 文件中为每个子系统或组件设置日志级别，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

第 89 页上的 “日志级别”表中所描述的日志级别符合 JDK1.4 日志 API 规范的要求。但是，日志级别 ALERT 和 FATAL 是专门用于 Sun ONE Application Server 的，而不能用于 JDK1.4 日志 API 规范。

下表按严重程度升序定义了 Sun ONE Application Server 中的日志级别和消息。表格左侧列中列出了 Sun ONE Application Server 中指定的日志级别，右侧列中提供了每种日志级别的简要说明。

日志级别

日志级别	说明
FINEST FINER FINE	指定调试信息详尽程度的消息。其中 FINEST 最详尽。
CONFIG	与各种静态配置信息相关的消息，可以帮助调试可能与特定配置关联的问题。

日志级别

日志级别	说明
INFO	信息类型的消息，通常与服务器配置或服务器状态相关。这些消息不会指定需要立即采取行动的错误。 例如，可能记录 “已收到配置更改通知；正在消息代理上创建新主题”的消息
WARNING	指定警告的消息。该消息可能伴随有异常情况。
SEVERE	指定相当重要的事件的消息，该事件可能会妨碍应用程序的正常执行。
ALERT*	警告用户采取特定行动的消息。
FATAL*	指定致命错误的消息，建议不要在出现致命错误后执行服务器操作。理论上，出现这条消息后就会发生服务器崩溃。

* 专用于 Sun ONE Application Server 的日志级别。

注意 所有日志级别低于 INFO （FINEST、 FINER、 FINE 和 CONFIG）的消息都可以提供信息来帮助解决有关调试的问题，所以必须按技术支持的建议将其启用。一般不会对日志级别低于 INFO 的消息进行本地化。

用于 syslog 配置的日志级别

下表包含了使用 syslog 时可以在 Sun ONE Application Server 中配置的日志级别列表。左侧列中列出了 Sun ONE Application Server 中指定的日志级别，而右侧列中提供了 syslog 设备中相应的日志级别。

映射到 server.log 的 Sun ONE Application Server 日志级别

Sun ONE Application Server	syslog 级别
FINEST	LOG_DEBUG
FINER	LOG_DEBUG
FINE	LOG_DEBUG
CONFIG	LOG_INFO
INFO （默认）	LOG_INFO
WARNING	LOG_WARNING

映射到 server.log 的 Sun ONE Application Server 日志级别

Sun ONE Application Server	syslog 级别
SEVERE	LOG_ERR
ALERT	LOG_ALERT
FATAL	LOG_CRIT

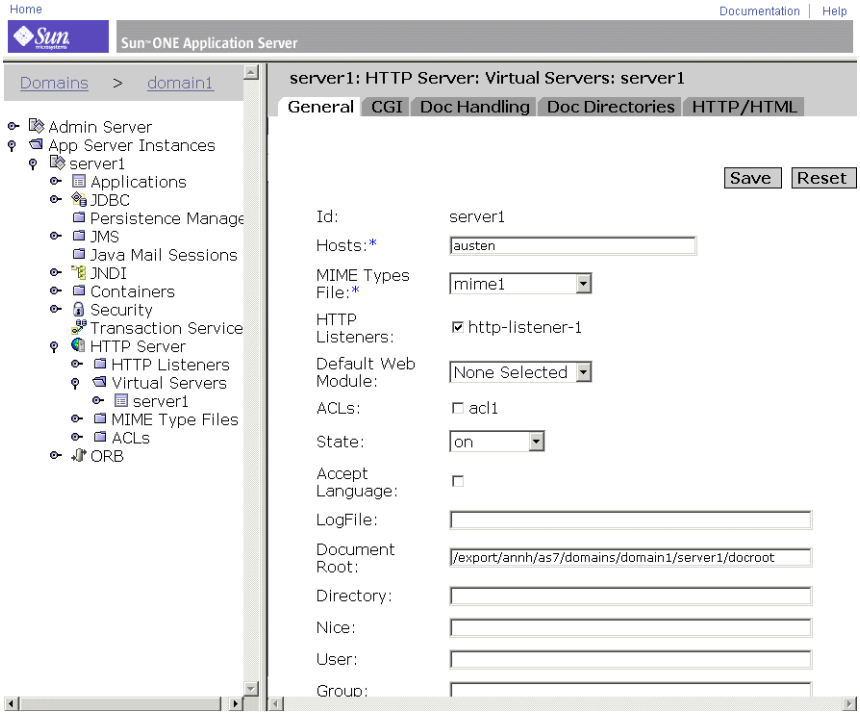
关于虚拟服务器和日志

Sun ONE Application Server 可以拥有虚拟服务器实例。 Sun ONE Application Server 实例中的每个虚拟服务器都具有自己的标识，也可以具有自己的日志文件。每个虚拟服务器的单独日志文件可以用来帮助追踪特定事务和资源的服务器活动。

要通过管理界面为虚拟服务器指定日志文件名，请转至目录树中的“HTTP Server”链接并打开虚拟服务器文件夹下的服务器实例元素，以便在右侧的框中显示“General”选项卡。可以在“Log File”字段中输入此虚拟服务器的日志文件路径和名称。第 92 页上的 “设置虚拟服务器的日志文件名”图中显示了此设置所在的位置。

注意 当启用日志并运行应用程序时，将记录应用程序的日志消息，而不记录虚拟服务器 ID。

设置虚拟服务器的日志文件名



也可以将来自多个虚拟服务器的日志消息定向到一个服务器日志文件。当执行此操作时，用户可能希望启用 log-virtual-server-id （位于 server.xml 文件的 log service 元素中）。这有助于区分来自不同虚拟服务器的日志消息。

<log-service level="FINEST" log-stdout="false" log-stderr="false" echo-log-messages-to-stderr="false" create-console="false" log-virtual-server-id="true" use-system-logging="false">

</log-service>

<http-listener>

<virtual-server-class>

<virtual-server id="server1" http-listeners="http-listener-1" hosts="strange" mime="mime1" state="on" accept-language="false"/>

<virtual-server id="server2" hosts="strange" mime="mime1"/>

</virtual-server-class>

</http-listener>

在此例中， <log-service log-virtual-server-id="true"> 负责将 virtual_server_id 包含在每个日志消息中。这样用户可以区分来自不同虚拟服务器的消息。如果 virtual-server 元素中不包含 “log-file” 属性，所有虚拟服务器上的消息都将记录到一个文件中。

关于记录器

可以在子系统级别上有选择地启用或禁用日志。各子系统的日志控制在 server.xml 文件中指定，如 *《Sun ONE Application Server Configuration File Reference》* 中所述。每个子系统都具有自己的记录器，且符合 JDK1.4 日志 API 规范的要求。

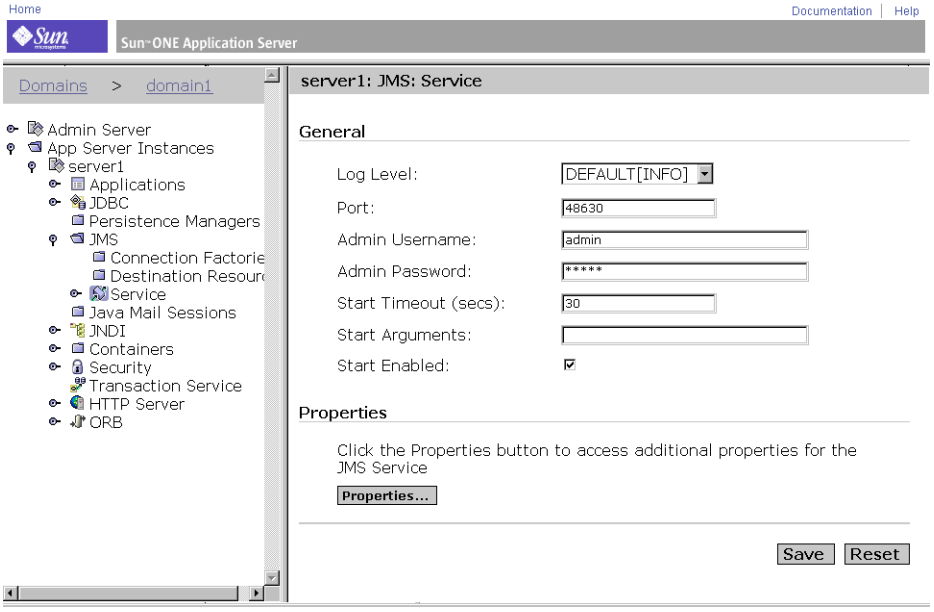
在下表中，左侧的列中定义了子系统，而右侧的列中为每个子系统定义了 server.xml 文件中的元素。

Sun ONE Application Server 中的子系统和位置

子系统	元素
管理服务器	<admin-service>
EJB 容器	<ejb-container>
Web 容器	<web-container>
MDB 容器	<mdb-container>
Sun ONE Message Queue （JMS 服务）	<jms-service>
安全服务	<security-service>
Java 事务服务 (JTS)	<transaction-service>
对象请求代理 (ORB)	<iiop-service>
默认处理程序[1]	<log-service>

1. 默认处理程序是指与所有 server.xml 条目相关联的默认记录器，这些条目不与特定子系统 （例如公用程序类）相关联。

JMS 服务的日志级别



注意 在 Windows 平台上，如果选择将日志发送到 Windows server.log，则只有级别为 INFO、WARNING、SEVERE、ALERT 或 FATAL 的消息会记录到 Windows 事件日志中。

第 89 页上的 “日志级别”表中按严重程度升序定义了为 Sun ONE Application Server 中的消息所提供的日志级别。这些日志级别符合 JDK1.4 日志 API 规范的要求。此外，日志级别 ALERT 和 FATAL 专用于 Sun ONE Application Server，而在 JDK1.4 日志 API 中不受支持。

关于客户端日志

应用程序客户机容器 (ACC) 具有自己的日志服务并只能记录到本地文件中。

ACC 通常在不同于应用程序服务器的其它主机上以自己的进程运行，因此它具有自己的日志基础结构和自己的日志文件。 ACC 配置保存在 sun-acc.xml 文件中。

ACC 的客户机子系统日志元素为 log-service。下表列出了元素和属性，并标明了默认值和值的范围。

ACC 日志元素

元素	属性	说明
log-service	file	ACC 日志文件；当为空或缺少时，记录到 stdout。
log-service	level	ACC 日志级别。

*《Sun ONE Configuration File Reference》*中提供了 sun-acc.xml 文件的一个示例。

重定向应用程序和服务器日志输出

对于开发者来说，在进行应用程序组件和 J2EE 应用程序的部件测试时，可以随时访问应用程序日志和服务器日志非常重要。在 Windows 平台上，开发者更希望看到服务器日志消息显示在桌面命令窗口中。在 UNIX 平台上，简单地将日志消息分到终端窗口 （从中启动了服务器实例）中的 stderr，或在命令结尾处使用 -f 来查看写入日志文件的日志消息会令许多开发者感觉方便。

server.xml 文件中包含了可以为 stdout 和 stderr 设置的属性，以将日志消息定向到日志文件或终端窗口等位置。有关使用 stdout 和 stderr 的详细信息，请参见 *《Sun ONE Application Server Configuration File Reference》*。

有关日志服务的信息，请参见第 103 页上的 “配置日志服务”。

日志文件管理

可以将访问和事件日志 (server.log) 文件设置为自动归档。在某一时间或在指定的间隔后用户的日志将被旋转。 Sun ONE Application Server 将保存旧的日志文件，并用含有保存日期和时间的名称标记所保存的文件。

注意 尽管可以创建多个虚拟服务器并为每个虚拟服务器关联一个日志文件，但是系统不支持单个虚拟服务器的日志旋转设置。

例如，可以将访问日志文件设置为每小时旋转一次，而 Sun ONE Application Server 将保存该文件并将其命名为 “access.199907152400”，其中将日志文件名、年、月、日和 24 小时制时间全组合成一个字符串。根据设置的日志旋转类型不同，日志归档文件的实际格式会有所不同。

注意 这些设备主要是为非 Solaris 平台提供的。

对于 Solaris，默认情况下不会启用这些设备，必须使用本机 Solaris 操作系统日志管理设备，例如 Solaris 9 上的 logadm。在 Solaris 8 上建议使用的日志管理公用程序为 " 使用 Solaris cron 公用程序调度 logadm 的执行 " 中所述的 cron 设备。

根据操作系统的不同，有四种不同的日志旋转方法。将在以下小节中对它们进行介绍。其中包括以下主题：

对于 UNIX 和 Windows

- 内部守护程序日志旋转
- 基于调度程序的日志旋转

对于 Solaris 9

- 使用 Solaris logadm 公用程序。详细信息，请参见第 98 页上的 “使用 Solaris logadm 公用程序旋转”。

对于 Solaris （任何版本）

- 使用 Solaris cron 公用程序。详细信息，请参见第 100 页上的 “使用 Solaris “cron”公用程序旋转”。

内部守护程序日志旋转

无论是在 UNIX 还是在 Windows 操作系统中，都可以使用内部守护程序日志旋转。内部守护程序日志旋转发生在 HTTP 守护程序中并且只能在启动服务器实例时进行配置。将用以下格式保存用此方法旋转的日志：

access.<YYYY><MM><DD><HHMM>

error.<YYYY><MM><DD><HHMM>

可以指定用来旋转日志文件和开始新日志文件的基准时间。例如，如果旋转开始时间为 12:00 a.m.，并且旋转间隔为 1440 分钟 （一天），那么保存时系统将立即创建一个新的日志文件而不管当前的时间，并一直收集信息，直到旋转开始。日志文件在每天的 12:00 a.m. 进行旋转，而访问日志被标记为 12:00 a.m. 并保存为 access.199907152400。同样，如果将间隔设置为 240 分钟（四小时），开始时间为 12:00 a.m.，则访问日志文件将包含从 12:00 a.m. 到 4:00 a.m.，从 4:00 a.m. 到 8:00 a.m. 依此类推时间内收集到的信息。

如果启用了日志旋转，将在服务器启动时开始进行日志文件旋转。第一个要旋转的日志文件将收集从当前时间至下次旋转时间之间的信息。以上一个例子为例，如果将开始时间设置为 12:00 a.m.，并将旋转间隔设置为 240 分钟，而当前的时间为 6:00 a.m.，则第一个要旋转的日志文件将包含从 6:00 a.m. 至 8:00 a.m 之间收集到的信息，下一个日志文件将包含 8:00 a.m. 至 12:00 p.m. （中午）的信息，并依此类推。

基于调度程序的日志旋转

调度程序日志旋转可以用于将日志文件立即归档，或使服务器在特定日期中的特定时间将日志文件归档。要将日志文件立即归档，请从管理界面左侧窗格中选择 “Admin Server”。然后，单击右侧页面顶部的 “Logging”链接。下一步，单击 “Log Rotation”。最后，单击 “Archive”。

使用调度程序方法旋转的日志将以在原文件名的后面加上文件旋转日期和时间的形式保存。例如，当在 4:30 p.m 旋转文件时， access 将变成 access.24Apr-0430PM。

日志旋转在服务器启动时进行初始化。如果开启了旋转， Sun ONE Application Server 将创建一个用时间标记的访问日志文件并在服务器启动时开始进行旋转。

旋转开始以后，如果发生需要记录到访问日志文件或错误日志文件的请求或错误，Sun ONE Application Server 将创建用新时间标记的日志文件，这将在预先调度的 “下次旋转时间”之后进行。

注意	对于 Windows 平台以及定向到 Solaris 中 syslog 以外的某个文件的服务器日志，您必须将服务器日志归档。

要将日志文件归档并指定 schedulerd 控制方法的用法，请从管理界面左侧窗格中选择 “Admin Server”。然后，单击右侧页面顶部的 “Logging”链接。下一步，单击 “Scheduler based Log Rotation”框。最后，单击 “OK”。将显示 scheduler 的当前状态。

使用 Solaris logadm 公用程序旋转

Solaris 9 操作系统中包含了公用程序 logadm，它可以使用记录的消息来执行一组功能。

特别是对于 Sun ONE Application Server，当从 Solaris （时钟）守护程序中运行此公用程序来执行日志旋转任务时，此公用程序非常有用，如第 101 页上的 “使用 Solaris cron 公用程序调度 logadm 的执行”中所述。

可以指定以下有关日志文件的日志旋转的详细资料：

- 系统中必须旋转的所有日志文件名
- 旋转间隔
- 将触发旋转的条件
- 要保存的备份日志文件的数目
- 要保存的备份日志文件的命名惯例

以上详细资料在 logadm.conf 文件中指定，该文件位于：

n /etc/logadm.conf

以下是 logadm.conf 的一个样例文件：

```
# Copyright 2001-2002 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)logadm.conf 1.2 02/02/13 SMI"
#
# logadm.conf
#
# Default settings for system log file management.
# The -w option to logadm(1M) is the preferred way to write to this
# file,
# but if you do edit it by hand, use "logadm -V" to check it for
```

```
# errors.
#
# The format of lines in this file is:
# <logname> <;options>
# For each logname listed here, the default options to logadm
# are given.Options given on the logadm command line override
# the defaults contained in this file.
# # logadm typically runs early every morning via an entry in
# root's crontab (see crontab(1)).
#
/var/log/syslog -C 8 -P 'Tue Jul 9 10:10:00 2002' -a 'kill -HUP `cat
/var/run/syslog.pid`' /var/adm/messages -C 4 -P 'Tue Jul 30 10:10:00 2002' -a
'kill -HUP `cat /var/run/syslog.pid`' /var/cron/log -c -s 512k -t /var/cron/olog
/var/lp/logs/lpsched -C 2 -N -t '$file.$N'
#
# The entry below is used by turnacct(1M)
#
/var/adm/pacct -C 0 -N -a '/usr/lib/acct/accton pacct' -g adm -m 664 -o adm -p never
#
# The entry below will rotate SUN One application server's default logfile
# every day provided the current logfile size is >= 512k.It will compress
# the old log file before archiving it and also delete the old files after 30
# days.The compression is done with gzip(1) and the resulting log file has
# the suffix of .gz.
/var/appserver/domains/domain1/server1/logs/server.log -A 30d -s 512k -p 1d -z
```

也可以用互动的方式调用 logadm 命令在特定文件中开始日志旋转。

以下示例将旋转 syslog 并保留八个日志文件。旧日志文件将放在目录 /var/oldlogs 中，而不是 /var/log 中：

```
% logadm -C8 -t'/var/oldlogs/syslog.$n' /var/log/syslog
```

还可以使用交互式命令行选项对在 /etc/logadm.conf 中指定的文件调用旋转，但是要使用不同的或已修改的选项。

如果在 /etc/logadm.conf 和命令行中都指定了选项，则首先应用在 /etc/logadm.conf 文件中指定的选项。因此，命令行选项将覆盖那些在 /etc/logadm.conf 中指定的选项。有关此操作的示例如下所示：

% logadm /var/appserver/domains/domain1/server1/logs/server.log -p now

以上命令将使用在 /etc/logadm.conf 中为给定文件提供的所有选项来旋转该文件。

注意 当同时指定多个选项时，各选项之间隐含着 AND 的关系。这意味着在旋转日志之前必须满足所有的条件。

有关 logadm 公用程序及其选项的详细信息，请参考以下手册页：

% man logadm

或

% logadm -h

使用 Solaris “cron”公用程序旋转

在 Solaris 8 上， cron 公用程序可以用来执行应用程序服务器的日志旋转。可以使用以下命令完成此操作：

% crontab -e

此命令将启动用户常用的编辑器 （由 env. 变量 $EDITOR 定义），因此用户可以提供 （时钟）守护程序条目的列表。

注意 此命令同样可以在用户一退出编辑器就调用 /etc/cron.d/logchecker 脚本。此脚本将 changed/new crontab 条目送至 （时钟）守护程序。因此，（时钟）守护程序将立即拾取用此方法添加的条目，并且日志旋转也会立即开始。

用户不需要重新启动 （时钟）守护程序来启用日志旋转。

本节包括以下主题：

- 关于 crontab 条目格式
- 使用 Solaris cron 公用程序调度 logadm 的执行

关于 crontab 条目格式

crontab 文件由多行构成，每行包含六个字段。各字段用空格或制表位隔开。前五个字段为整数模式，分别指定以下内容：

- 分钟 (0-59)
- 小时 (0-23)，
- 该月的日期 (1-31)，
- 该年的月份 (1-12)，
- 星期几 （0-6，其中 0= 星期日）。

使用此格式，可以指定在特定的日期/星期/月份要旋转的访问文件和事件日志文件，以及重复进行旋转的时间调度。例如，

0 0 * * 1-5

/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs

0 12 * * 1-5

/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs

0 * * * 1-5

/opt/SUNWappserver7/appserver/domains/domain1/mainserver/bin/rotatelogs

server1 的访问文件和日志文件将在星期一至星期五的每天午夜和中午旋转；而 mainserver 的访问文件和日志文件将在星期一至星期五的每天每小时旋转一次。

crontab 文件将存储在 /var/spool/cron/crontabs/ 下。可以以最终用户或超级用户的身份创建 crontab 文件。根据用户的权限不同，可以使用以下命令查看 crontab 条目：

% crontab -l *username*

使用 Solaris cron 公用程序调度 logadm 的执行

cron 命令将启动进程，在指定的日期和时间执行命令。定期调度的命令可以根据 crontab 文件 （位于 /var/spool/cron/crontabs 目录中）中找到的指令来指定。

作为与 cron 一起使用的定期调度的命令示例，以下 crontab 中的条目将在每天午夜启动 logadm。

0 0 * * 0-6 logadm

请注意，用户可以使用 crontab(1) 命令提交自己的 crontab 文件。

要保留 cron 进行的所有操作的日志，必须在 /etc/default/cron 文件中指定 CRONLOG=YES （默认情况下）。 /etc/cron.d/logchecker 是用来检查日志文件是否超出了系统 ulimit 的脚本。如果超出了该限制，日志文件将移至 /var/cron/olog。

通过命令行界面配置日志

可以从命令行为服务器实例或虚拟服务器实例配置日志服务的各个方面。

注意 本节的所有命令示例都假设已经设置了环境变量。

获取服务器实例的所有 log-service 属性的步骤：

asadmin> get *instance_name*.log-service.*

log-service 属性还在第 105 页上的 “日志服务属性”表中进行了定义。

以下是使用指定的服务器实例名运行此命令的一个示例：

asadmin> get *server1*.log-service.*

将返回 server1 实例的日志服务属性列表。使用 set 命令可以配置每一个列出的属性。

要启用虚拟服务器实例的虚拟服务器 ID 的日志，请在出现终端提示后输入以下命令：

asadmin> get *instance_name*.LogVirtualServerId

将返回 LogVirtualServerId 的当前状态。如果状态为 False，可以使用以下 set 命令来启用它：

asadmin> set *instance_name*.LogVirtualServerId=true

要设置虚拟服务器实例的日志文件名，请使用以下 set 命令：

asadmin> set *instance_name*.virtual-server.<virtual server id>.logFile=<log file>

例如，将发出以下设置日志文件命令：

asadmin> set
instance2.virtual-server.*instance2*.logFile=/space/IAs7se/appserver7/appserv/domains/domain1/instance2/logs/log

有关命令语法的详细信息，请参见命令行界面帮助。

有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

通过管理界面配置日志

本节介绍了通过 Sun ONE Application Server 管理界面可以执行的任务，以便为服务器范围 （全局）元素、指令和应用程序组件配置可用的日志服务选项。

本节包括以下主题：

- 配置日志服务
- 为应用程序服务器组件和子系统配置日志
- 为错误日志配置指令

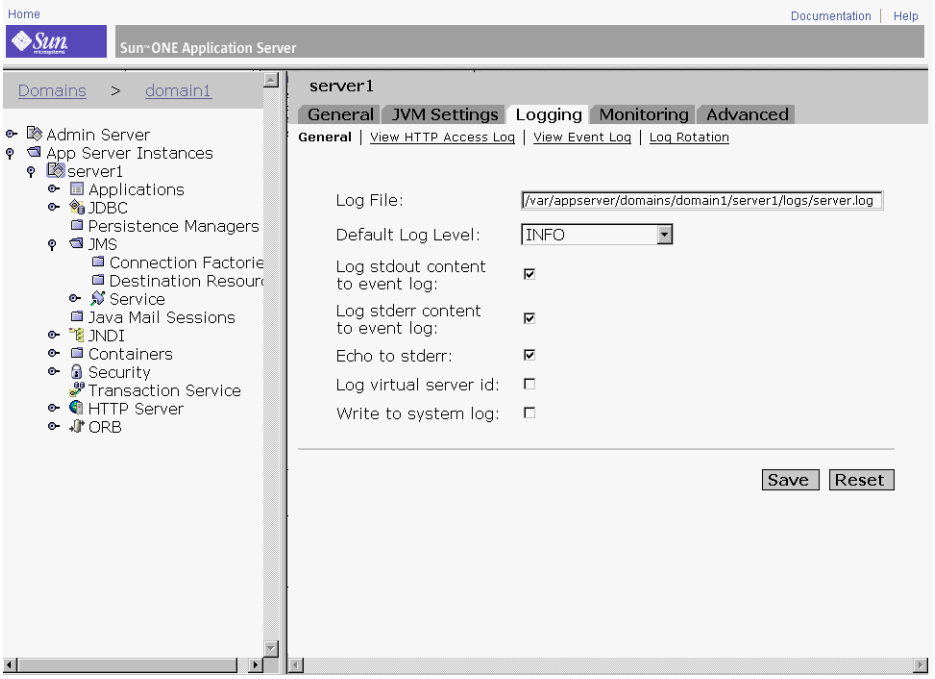
配置日志服务

日志服务是 server.xml 文件中 J2EE 服务元素类别中的一种元素，如 *《Sun ONE Application Server Configuration File Reference》* 中所述。日志服务可以用来配置系统日志服务，其中包括以下日志文件：

- 服务器日志
- 访问日志
- 事务日志
- 虚拟服务器日志

系统日志服务配置包括指定 log service 元素的属性值。

服务实例的日志服务管理



可以通过管理界面为 log service 元素配置以下属性，如第 104 页上的 “服务实例的日志服务管理”图中所示。

- 日志文件
- 默认日志级别
- 将 “标准输出”内容记录到事件日志
- 将 “标准错误”内容记录到事件日志
- 反射到 “标准错误”中
- 创建控制台
- 记录虚拟服务器 ID
- 写入系统日志

可以从管理界面左侧窗格展开的服务器实例树状结构中访问 “Log Service”链接。下表介绍了每个可以配置的属性以及默认值和允许值的范围。

日志服务属性

属性	默认值	说明
file	server.log [1]	（可选）覆盖服务器日志的名称或位置。无论使用哪个用户账户运行服务器，保存服务器日志的文件和目录都必须是可写入的。
level	INFO	（可选）控制由其它元素记录到服务器日志中的消息的默认类型。允许的值如下所示 （从高到低排列）：FINEST、 FINER、 FINE、 CONFIG、 INFO、 WARNING、 SEVERE、 ALERT 和 FATAL。 每个值都记录所有较低值的消息；例如 FINEST 记录所有消息，而 FATAL 仅记录 FATAL 消息。默认值为 INFO，它将记录所有 INFO、 WARNING、 SEVERE、 ALERT 和 FATAL 消息。
log-stdout	True	（可选）如果为 True，请将 stdout 输出重定向到服务器日志。有效值为 On、Off、Yes、No、1、0、True 和 False。
log-stderr	True	（可选）如果为 True，请将 stderr 输出重定向到服务器日志。有效值为 On、Off、Yes、No、1、0、True 和 False。
echo-log-messages-to-stderr	True	（可选）如果为 True，除了将日志消息发送到服务器日志以外，还将它发送到 stderr。有效值为 On、Off、 Yes、 No、 1、 0、 True 和 False。
create-console	False	（可选）如果为 True，请在 Windows 操作系统中为 stderr 输出创建一个控制台窗口。有效值为 On、Off、 Yes、 No、 1、 0、 True 和 False。
log-virtual-server-id	False	（可选）如果为 True，虚拟服务器 ID 将显示在虚拟服务器日志中。如果有多个 virtual-server 元素共享同一个日志文件，这些属性很有用。
use-system-log	False	如果为 True，请使用 UNIX syslog 服务或 Windows 事件日志来生成和管理日志。

1. 位于由 server 元素的 log-root 属性指定的目录中。

为应用程序服务器组件和子系统配置日志

本节介绍了如何启用日志以及如何为 Sun ONE Application Server 组件及子系统选择日志级别。请注意， Java 事务服务组件包含多个日志文件。因为大多数组件和子系统的处理方法和配置日志级别的有关方法相同，所以对于组件和子系统的特定编组，选择日志级别的操作仅记录一次。

以下组件和子系统可以选择性记录服务器消息。通过参考本指南中介绍的其它主题，用户可以逐渐熟悉这些组件和子系统。

- ORB - 为基于 Corba 的客户机配置支持
- Web 容器 - 配置 J2EE 服务
- EJB 容器 - 配置 J2EE 服务
- MDB 容器 - 配置 J2EE 服务 （在 EJB 容器内）
- Java 事务服务 - 配置 J2EE 服务
- JMS 服务 - Java 信息服务
- 虚拟服务器 - 使用虚拟服务器

指定日志级别的步骤

要为 ORB、Web 容器、EJB 容器、MDB 容器（在 EJB 容器内）、Java 事务服务和 JMS 服务指定日志级别，请执行以下步骤：

1. 在管理界面的左侧窗格中，展开 Sun ONE Application Server 实例，将显示要编辑的组件和子系统。
2. 单击所需组件或子系统的链接。
3. 在管理界面的右侧页面中，从 “Log Level”下拉列表中选择以下日志级别参数之一。这些日志级别在第 88 页上的 “关于日志级别”中进行介绍。

指定日志文件的步骤：（虚拟服务器）

要指定日志文件，请执行以下步骤：

1. 在管理界面的左侧窗格中，展开 Sun ONE Application Server 实例，将显示 HTTP 服务器子系统。
2. 单击 “HTTP Server”链接。
3. 单击 “Virtual Server”链接。
4. 再单击所需的服务器实例链接。

5. 在管理界面的右侧页面中，在 “General”选项卡下的 “Log File”字段中输入所需的目录路径和文件名。

指定事务日志位置的步骤：（Java 事务服务）

要指定事务日志的位置，请执行以下步骤：

1. 在管理界面的左侧窗格中，展开 Sun ONE Application Server 实例，将显示事务服务子系统。
2. 单击 “Transaction Service”链接。
3. 在管理界面的右侧页面中，在 “Advanced”字段组下的 “Transaction Log Location”字段中输入所需的目录路径和文件名。

为错误日志配置指令

Sun ONE Application Server 中包含 init.conf 文件的错误日志指令。其中包括以下指令：

- **错误日志日期格式。** ErrorLogDateFormat 指令指定了服务器日志使用的日期格式。
- **日志刷新间隔。** LogFlushInterval 确定了最大的刷新时间间隔 （以秒为单位），在此之前访问日志会从内存刷新到 access.log 文件中。
- **进程 ID 日志。**PidLog 指定了用来记录基本服务器进程的进程 ID (pid) 文件。有些服务器支持程序假设此日志位于 logs/pid 服务器根目录中。

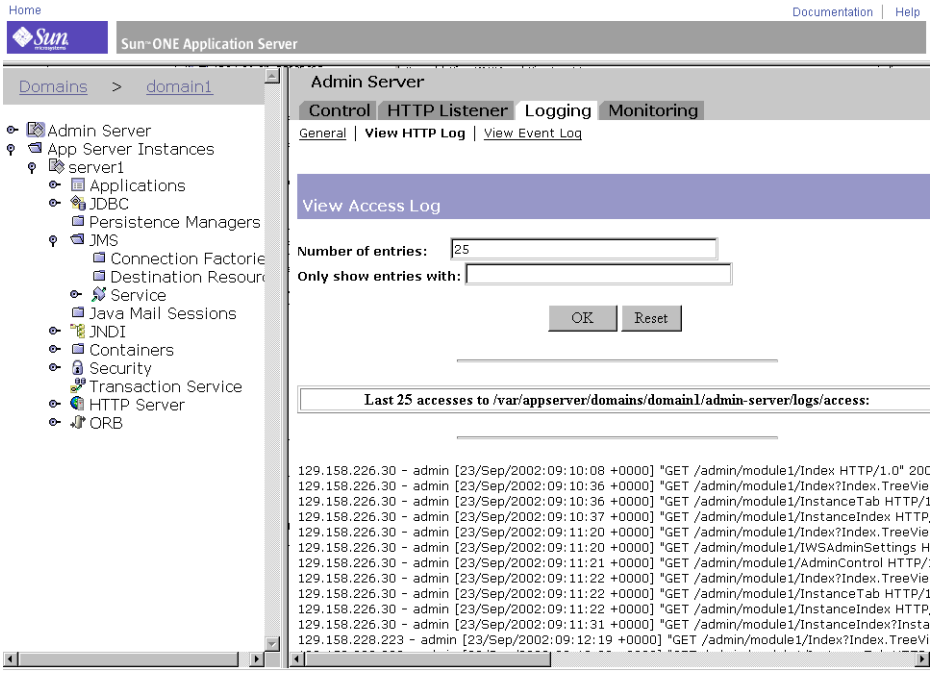
所有 init.conf 的指令都在 *《Sun ONE Application Server Configuration File Reference》* 中进行了详细的介绍。

查看访问日志文件

既可以查看管理员服务器的 http 日志文件，也可以查看 Sun ONE Application Server 实例的 http 日志文件。

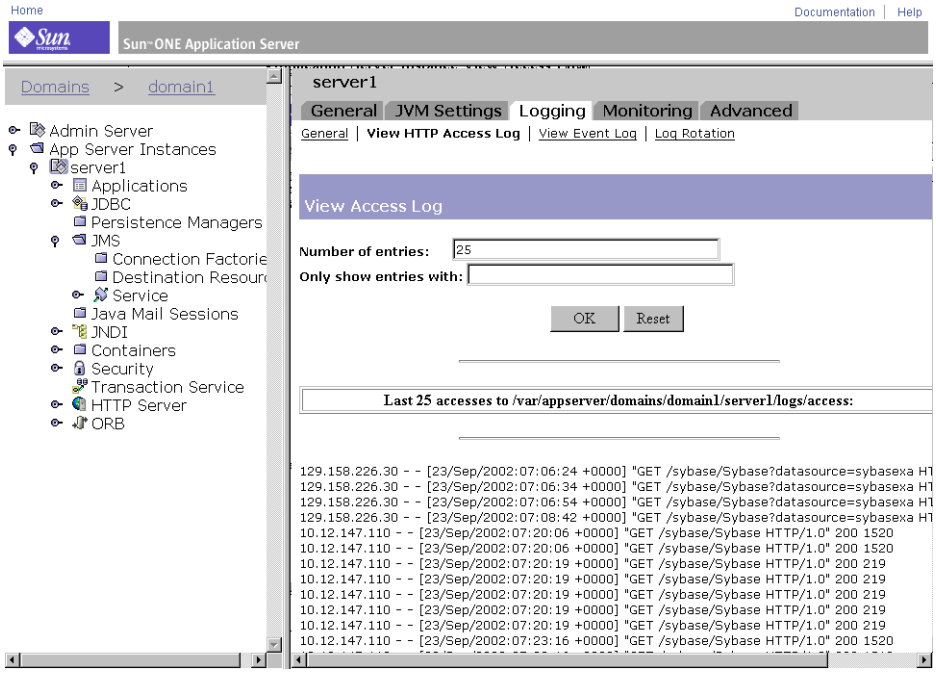
要查看管理员服务器的 http 日志，请先从管理界面的左侧窗格中选择 “Admin Server”，再从右侧页面中选择 “Logging”选项卡。将显示 “View HTTP Access Log”链接。选择此链接以查看已配置的访问日志。第 108 页上的 “管理服务器的查看 HTTP 访问日志”图中列出了所显示日志的示例。

管理服务器的查看 HTTP 访问日志



要查看应用程序服务器实例的访问日志，请在管理界面的左侧窗格中单击所需的服务器实例。在右侧窗格中单击 “Logging”选项卡。单击 “View Access Log”链接，将显示该服务器实例已配置的活动访问日志。第 109 页上的 “应用程序服务器实例的查看访问日志”图中显示了一个示例。

应用程序服务器实例的查看访问日志

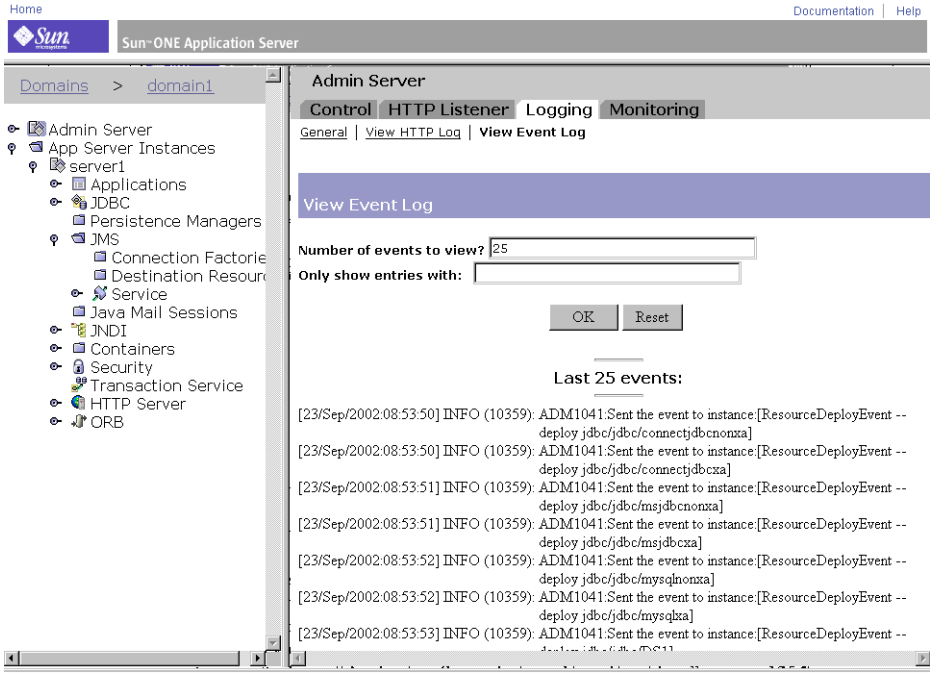


查看事件日志文件

既可以查看管理员服务器的活动事件日志文件，也可以查看 Sun ONE Application Server 实例的活动事件日志文件。

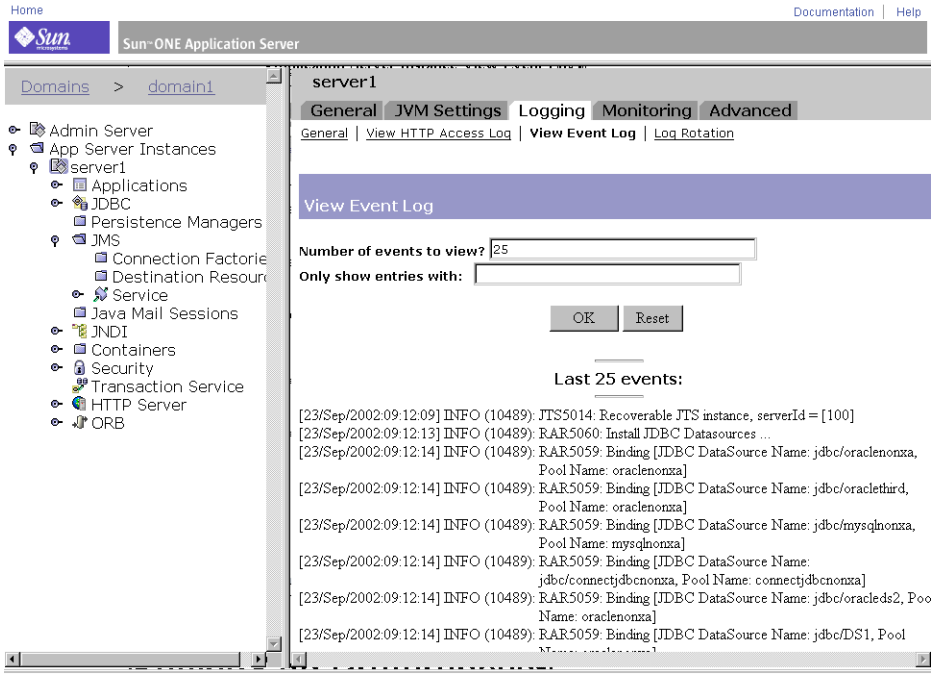
要查看管理员服务器的事件日志，请从左侧窗格中选择 “Admin Server”，再从右侧页面中选择 “Logging”选项卡。将显示 “View Event Log”链接。选择此链接以查看已配置的事件日志。第 110 页上的 “管理服务器的查看事件日志”图中列出了所显示日志的示例。

管理服务器的查看事件日志



要查看应用程序服务器实例的事件日志，请先在管理界面的左侧窗格中单击所需的服务器实例，再从右侧窗格中选择 “Logging” 选项卡。将显示 “View Event Log” 链接。选择此链接以查看已配置的事件日志。第 111 页上的 “应用程序服务器实例的查看事件日志” 图中列出了所显示日志的示例。

应用程序服务器实例的查看事件日志



设置日志首选项

在安装过程中，将为服务器创建名为 access 的访问日志文件。通过指定是否记录访问、日志使用什么格式，以及当客户机访问资源时服务器是否要查找客户机的域名，用户可以自定义任意资源的访问日志。

要对多个虚拟服务器使用一个日志文件，请在 server.xml 文件中为事件日志打开 LogVsId。详细信息，请参见 *《Sun ONE Application Server Configuration File Reference》*。也可以在管理界面的 “Admin Server Logging” 选项卡上打开 LogVsID。

请按以下步骤从管理界面中启用记录虚拟服务器 ID。重新启动管理服务器后更改才能生效。

1. 请在管理界面的左侧窗格中单击 “Admin Server”。
2. 在右侧页面中单击 “Logging” 选项卡。

3. 单击 “Log virtual server ID” 复选框。
4. 单击 “Save” 按钮以应用对 Sun ONE Application Server 所做的更改。

要使更改生效，需要重新启动 Sun ONE Application Server。

运行日志分析程序

flexanlg 是用来进行日志文件报告的日志分析程序工具。仅在将日志定向到 syslog 以外的文件时，才可以使用日志分析程序。

使用日志分析程序可以生成有关默认服务器的统计数据，例如活动摘要、最常访问的 URL、一天中访问服务器的高峰时段等等。除了可以生成默认服务器的统计数据以外，日志分析程序不能生成任何虚拟服务器的统计数据。但是，可以查看每个虚拟服务器的统计数据，如第 107 页上的 “查看访问日志文件” 中所述。

注意 在运行日志分析程序之前，必须旋转服务器日志。详细信息，请参见第 96 页上的 “日志文件管理”。

通过使用 flexanlg 工具可以从命令行运行日志分析程序命令，该工具位于 *install_dir*/bin/flexanlg 目录中。

要运行 flexanlg，请在出现命令提示后键入以下命令和选项：

flexanlg [-P] [-n name] [-x] [-r] [-p order] [-i file]* [-m metafile]* [o file][c opts] [-t opts] [-l opts] [-h help]

命令选项 （可以重复标有 * 的选项）。

-i *filename*

输入日志文件

-P

代理日志格式

-n *servername*

服务器的名称

-x

以 HTML 形式输出

-r

解析主机名的 IP 地址

-p [c, t, l]

输出顺序；默认的顺序为计数、时间统计和列表

-m *filename*

元文件

-o *filename*

输出日志文件；默认为 stdout

-c [h, n, r, f, e, u, o, k, c, z]

对这些项目进行计数；默认为：h、 n、 r、 e、 u、 o、 k、 c

- h：找到的总数
- n：304 未修改的状态代码 （使用本地副本）
- r：302 找到的状态代码 （重定向）
- f：404 未找到的状态代码 （未找到文档）
- e：500 服务器错误状态代码 （配置错误）
- u：唯一 URL 的总数
- o：唯一主机的总数
- k：传送的千字节总数
- c：高速缓存中保存的千字节总数
- z：不对任何项目进行计数

-t [sx, mx, hx, xx, z]

查找常规统计数据；默认为：s5m5h24x10

- s （数目）：查找日志的最大 （数目）秒钟数
- m （数目）：查找日志的最大 （数目）分钟数
- h （数目）：查找日志的最大 （数目）小时数
- u （数目）：查找日志的最大 （数目）用户数
- a （数目）：查找日志的最大 （数目）用户代理数
- r （数目）：查找日志的最大 （数目）参考数
- x （数目）：查找最大 （数目）的杂项关键字
- z：不查找任何常规统计数据

-l [cx, hx]

创建指定子选项的列表；默认为：c+3h5

c (x, +x)：最常访问的 URL

x：仅列出 x 项

+x：仅当访问超出 x 次时列出

h (x, +x)：最常访问用户服务器的主机或 IP 地址

x：仅列出 x 项

+x：仅当访问超出 x 次时列出

z：不创建任何列表

EXAMPLE：使用 flexanlg 命令

flexanlg -i /var/opt/SUNQappserver7/domains/domain1/server1/logs/access

---

注意 在运行日志分析程序之前，应将服务器日志归档。

查看事件 (Windows 2000 Pro)

除了将错误记录到 server.log 文件以外， Sun ONE Application Server 还将严重的系统错误记录到事件查看器中。事件查看器可用于监视系统中发生的事件。在打开错误日志之前，可以使用事件查看器查看因基础配置问题而出现的错误。

要使用事件查看器，请执行以下步骤：

1. 从 “开始” 菜单中，依次选择 “程序” 和 “管理工具”。在 “管理工具” 程序组中选择 “事件查看器”。
2. 从 “日志” 菜单中选择 “应用程序”。

   应用程序日志将显示在事件查看器中。来自 Sun ONE Application Server 的错误具有 https-*serverid* 的源标签。
3. 从 “查看” 菜单中选择 “查找” 以在日志中搜索这类标签之一。从 “查看” 菜单中选择 “刷新” 以查看更新后的日志项。

有关事件查看器的详细信息，请参见相关系统文档。

第 6 章

监视 Sun ONE Application Server

本章提供了有关监视的信息，同时介绍了在 Sun ONE Application Server 中使用的简单网络管理协议 (SNMP) 的特征和功能。

本章包括以下主题：

- 关于监视 Sun ONE Application Server
- 使用 CLI 提取监视数据
- 使用 CLI 管理事务服务
- 使用 HTTP 服务质量
- 关于 SNMP
- 设置 SNMP
- 启用和启动 SNMP 主代理

关于监视 Sun ONE Application Server

您可以通过收集系统上关键数据点的活动统计数据来监视 Sun ONE Application Server。统计数据显示了服务器处理的请求的数目以及这些请求的处理状况。您可以查看单个虚拟服务器的某些统计数据，也可以查看整个应用程序服务器实例的统计数据。您可以使用 asadmin 公用程序或 SNMP 来监视 Sun ONE Application Server。

本节包括以下主题：

- 统计数据
- SNMP
- 监视 HTTP 服务器

- 监视应用程序组件和子系统
- 服务质量 (QOS)

统计数据

大多数 Sun ONE Application Server 应用程序组件和子系统 （包括 HTTP 服务器）都会始终启用统计数据的收集，因而不需要任何启用功能。但是，某些统计数据仅当在子系统上明确启用了监视，或仅当启用了相关功能时才会进行收集。这些统计数据包括以下数据点：

- EJB 方法的统计数据
- 活动事务
- 连接 （仅当启用了服务质量）
- DNS （仅当启用了 DNS 高速缓存）

可以通过管理界面启用对应用程序子系统或组件的监视功能可以通过管理界面启用对应用程序子系统或组件的监视功能，如第 117 页上的 “监视应用程序组件和子系统”中所述。

如果服务器监视器报告该服务器处理的请求过多，您可能需要调整服务器配置或系统的网络内核。有关调整服务器配置的详细信息，请参见 *《Sun ONE Application Server Performance Tuning and Sizing Guide》*。

SNMP

Sun ONE Application Server 通过它的信息收集工具，使用简单网络管理协议 (SNMP) 提供网络管理信息，该协议用于在网络范围内交换管理和监视信息。称为 *代理* 的程序使用 SNMP 监视网络上的各种设备 （集线器、路由器、网桥等等）。另一个程序负责从代理处收集数据。由监视操作创建的数据库称为 *管理信息库* (MIB)。这些数据用于检查网络上的所有设备是否正常运行。

使用 SNMP 只能监视 HTTP 服务器；而使用命令行界面 (CLI) 可以监视所有组件和系统。

有关 SNMP 的详细信息，请参见第 143 页上的 “关于 SNMP”和第 150 页上的 “设置 SNMP”。

监视 HTTP 服务器

默认情况下， HTTP 服务器的监视被启用，这意味着不必专门启用它。 HTTP 服务器的监视基于一个 XML 文件，可以作为一个包含三个可监视属性的属性集使用 asadmin 命令进行访问。第 130 页上的 “可监视的 HTTP 服务器元素”和第 131 页上的 “可监视的 HTTP 服务器属性”描述了该 XML 文件的元素、子元素和属性。

注意 使用 SNMP 只能监视 HTTP 服务器的统计数据。而使用命令行界面可以获得 Sun ONE Application Server 所有子系统 （包括 HTTP 服务器）的统计数据。

有关使用 asadmin 的详细信息，请参见第 375 页上的 “使用命令行界面”。

监视应用程序组件和子系统

Sun ONE Application Server 中的某些子系统或组件不需要启用监视，因为系统始终会收集某些相关的统计数据。例如，可以启用或禁用应用程序组件 （例如容器）的监视。启用监视时，除了始终收集的统计数据外，还会收集有关所有 EJB 方法的其它统计数据。对 JDBC 连接池的监视始终被启用。连接池在第一次访问时被初始化，您可以在这之后的任何时刻监视相关统计数据。

有关可监视数据点的完整列表，请参见第 126 页上的 “可监视的属性名”。

您可以通过管理界面或命令行界面 (CLI) 为选定的应用程序组件和子系统启用监视。例如，要从 CLI 为 EJB 容器启用监视，请在终端窗口中键入以下命令：

```
set server1.ejb-container.monitoringEnabled=true
reconfig server1
```

其中 server1 是实例名。

在管理界面上的 “Containers”节点下可以访问相同的功能。

本节包括以下主题：

- 监视容器子系统
- 监视 ORB 服务
- 监视事务服务

监视容器子系统

启用 EJB 容器的监视时，将收集与所有实体 Bean、状态会话 Bean 及无状态会话 Bean 的方法相关的统计数据。这些统计数据包括：

- 错误总数
- 调用总数
- 成功总数
- 以毫秒计的执行时间 （对于方法的最后一次调用）

容器子系统的所有其它统计数据始终会被收集。某些监视的数据点包括与以下内容相关的统计数据：

- 池中无状态 Bean 的初始、最小及最大数目
- 高速缓存中状态会话 Bean 和实体 Bean 的最小及首选数目
- 高速缓存中无状态会话 Bean 的最小及首选数目
- 已创建和已破坏的 Bean 的数目
- 其它相关统计数据

监视 ORB 服务

对于 ORB 服务，监视的数据点包括为 ORB 连接和 ORB 线程池收集的统计数据。ORB 统计数据始终会被收集，因此不必为 ORB 服务启用监视。

监视事务服务

对于 Java 事务服务 (JTS) 服务，监视的数据点包括：

- 完成的事务总数
- 回滚的事务总数
- 正在执行的事务总数
- 正在执行的事务列表

有关详细信息，请参见第 138 页上的 “使用 CLI 管理事务服务”。

服务质量 (QOS)

*服务质量*是指为服务器实例虚拟服务器类或虚拟服务器设置的性能限制。例如，如果您是 Internet 服务提供商 (ISP)，则可能希望根据所提供的带宽收取不同的虚拟服务器费用。您可以在两方面进行限制：带宽大小和连接数目。

Sun ONE Application Server 提供的服务质量信息用于根据以下各项确定服务器在运行时的效率：

- 启动时间
- 服务器通信量和通信量对带宽的影响
- 实时数据与静态数据分析
- 其它数据元素

有关详细信息，请参见第 138 页上的 “使用 CLI 管理事务服务”。

使用 CLI 提取监视数据

使用 asadmin 命令，您可以通过命令行界面 (CLI) 使用 list 和 get 命令提取监视数据。

注意 set 命令仅用于设置事务服务的监视，如第 138 页上的 “使用 CLI 管理事务服务”中所述。

本节包括以下主题：

- list --monitor 命令
- get --monitor 命令
- CLI 名称映射
- HTTP 服务器的可监视对象

list --monitor 命令

list 命令提供有关当前监视的指定服务器实例名称的应用程序组件和子系统的信息。使用此命令，您可以查看某个服务器实例的可监视组件和子组件。

示例：

asadmin> list --monitor server1

返回以下启用了监视的应用程序组件和子系统列表：

iiop-service
transaction-service
application.converter
application.myApp
http-server

您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数据时，这会很有用。

示例：

asadmin> list --monitor server1.application

返回：

converter
myApp

有关更复杂的示例，请参见第 122 页上的 “Petstore 示例”。

get --monitor 命令

此命令检索以下监视信息：

- 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 命令的详细信息，请参见第 121 页上的 “CLI 名称映射”。

示例 1

尝试获取某个子系统的某个特定属性的所有属性：

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.*

total-inbound-connections=1
total-outbound-connections=1
```

示例 2

尝试获取某个 J2EE 应用程序的所有属性：

```
asadmin> get --monitor server1.application.converter.*

Attribute name(s) not found
```

该 J2EE 应用程序级别上没有暴露可监视的属性，因此该命令失败。

示例 3

尝试获取某个子系统的某个特定属性：

```
asadmin> get --monitor server1.transaction-service.inflight-tx

Attribute name = inflight-tx Value = No active transaction found.
```

示例 4

尝试获取某个子系统属性中的某个未知属性：

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.bad-name

        Could not get the attribute

Execution failed for the command:get --monitor server1.iiop-service.orb-connection.bad-name
```

CLI 名称映射

Sun ONE Application Server 使用树结构来跟踪可监视的对象。树中的每个节点都具有名称和类型。如果是单元素类型，则任意父节点下只能有一个该类型的节点。有关树中节点类型的详细信息，请参见第 124 页上的 “可监视的对象类型”。

树中的根对象由 Sun ONE Application Server 实例名表示。例如，名为 server1 的实例的根监视对象为：

```
server1
```

然后，所有子对象将使用点字符 (.) 作为分隔符进行指定。如果子节点是单元素类型，则只需要使用监视对象类型来指定对象；否则，需要使用 type.name 名称格式来指定对象。

例如， http-server 是一种有效的可监视对象类型，并且是单元素类型。要指定表示实例 server1 的 http-server 的单元素类型子节点，该名称为：

server1.http-server

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。要指定表示应用程序 Petstore 的非单元素类型子节点，该名称为：

server1.application.petstore

CLI 名称还可以指定可监视对象中的特定属性。例如， http-server 具有一个名为 summary 的可监视属性。以下名称指定了该 summary 属性：

server1.http-server.summary

监视对象所暴露的属性名没有固定的命名规范。

通常情况下您不需要知道要在 CLI 中使用的有效名称，您可以通过使用 list 命令检查可用的可监视对象，也可以使用带有通配符参数的 get 命令则检查任意可监视对象的所有可用属性。

以下示例显示了某些客户机名称映射情况：

Petstore 示例

用户要检查对某方法的调用次数，该方法位于部署在名为 server1 的 Sun ONE Application Server 实例上的 Petstore 应用程序中。这里同时使用了 list 和 get 命令，以访问该方法的所希望的统计数据。

1. 以多模式调用 CLI。
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=4848
3. 列出实例 server1 的可监视组件：

   asadmin>list --monitor server1

   输出为：
   iiop-service
   transaction-service
   application.CometEJB

application.ConverterApp
application.petstore
http-server
resources

可监视组件列表包括 iiop-service、 http-server, transaction-service、 resources 以及所有部署 （及启用）的应用程序。

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 server1.application.petstore

输出为：
ejb-module.signon-ejb_jar
ejb-module.catalog-ejb_jar
ejb-module.uidgen-ejb_jar
ejb-module.customer-ejb_jar
ejb-module.petstore-ejb_jar
ejb-module.AsyncSenderJAR_jar
ejb-module.cart-ejb_jar

5. 列出 Petstore 应用程序的 EJB 模块 sigon-ejb_jar 中的可监视子组件：

asadmin>list -m server1.application.petstore.ejb-module.signon-ejb_jar

输出为：
entity-bean.UserEJB
stateless-session-bean.SignOnEJB

6. 列出 Petstore 应用程序的 EJB 模块 sigon-ejb_jar 的实体 Bean UserEJB 中的可监视子组件：

asadmin>list -m server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB

输出为：
bean-method.create0
bean-method.findByPrimaryKey1
bean-method.remove2
bean-method.getUserName3
bean-method.setPassword4
bean-method.getPassword5
bean-method.matchPassword6
bean-method.remove7
bean-method.isIdentical8
bean-method.getEJBLocalHome9
bean-method.getPrimaryKey10
bean-pool
bean-cache

7. 列出实体 Bean UserEJB（位于 Petstore 应用程序的 EJB 模块 sigon-ejb_jar 中）的方法 getUserName3 的可监视子组件：

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3
```

输出为：

```
No monitorable entities for element
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3
```

8. 该方法没有可监视的子组件。获取方法 getUserName3 的所有可监视的统计数据。

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.*
method-name = public abstract java.lang.String
com.sun.j2ee.blueprints.signon.user.ejb.UserLocal.getUserName()
total-num-errors = 0
total-num-success = 2
execution-time-millis = 1
total-num-calls = 2
```

9. 您也可以获取特定的统计数据，例如执行时间。

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.execution-time-millis
execution-time-millis = 1
```

可监视的对象类型

用于监视的对象树包含多个节点。节点是对象树中的特定条目，由它的类型、名称及父节点唯一标识。某些节点类型是单元素类型，这意味着在父节点下只能存在一个该类型的节点。单元素类型的节点并不需要名称。

非单元素类型的节点需要名称。“实例名”列描述了可能的名称空间。

下表按照各种节点类型之间的父子关系描述了树结构并列出了某些节点类型的名称空间。

监视对象类型

节点类型	单元素？	叶？	子节点类型	实例名
root	是	否	http-server iiop-service resources transaction-service application standalone-ejb-module	

监视对象类型 *（续）*

节点类型	单元素？	叶？	子节点类型	实例名
http-server	是	否	virtual-server process	
virtual-server	是	是		
process	是	是		
iiop-service	是	是	orb	
orb	否	否	orb-connection orb-thread-pool	为系统 ORB 保留了 system。所有用户 ORB 都获取一个源自 TCP 端点的名称。
orb-connection	是	是		
orb-thread-pool	是	是		
resources	是	否	jdbc-connection-pool	
jdbc-connection-pool	否	是		这些名称与用户创建连接池时指定的名称相同。
transaction-service	是	是		
application	否	否	ejb-module	在 server.xml 中注册的应用程序的名称。
ejb-module	否	否	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	EJB 模块的名称。它源自 EJB JAR 名称。
standalone-ejb-module	否	否	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	在 server.xml 中注册的独立 EJB 模块的名称。
stateless-session-bean	否	否	bean-pool bean-method	部署描述符中的 Bean 名称。
stateful-session-bean	否	否	bean-cache bean-method	部署描述符中的 Bean 名称。
entity-bean	否	否	bean-cache bean-pool bean-method	部署描述符中的 Bean 名称。
message-driven-bean	否	否	bean-pool bean-method	部署描述符中的 Bean 名称。

监视对象类型 *（续）*

节点类型	单元素？	叶？	子节点类型	实例名
bean-pool	是	是		
bean-cache	是	是		
bean-method	否	是		对于消息驱动的 Bean，其名称是 onMessage，它是方法名后跟一个表示其它企业 Bean 的数字后缀 （该后缀用来唯一标识过载的方法。）

可监视的属性名

并非每个可监视对象都需要暴露其可监视属性。某些对象只是用来编组其它对象。对于 Sun ONE Application Server，除了一个 http-server 节点外，只有树的叶节点具有属性。http-server 节点类型既具有子节点，也具有属性。下表列出了各种节点的可能的可监视属性名。

http-server

属性名	数据类型	说明
summary	字符串 （具有格式）	HTTP 服务器摘要。包括虚拟服务器和进程。 注意：有关具有格式的字符串中暴露了哪些数据的详细信息，请参见第 130 页上的 “HTTP 服务器的可监视对象”一节。

virtual-server

属性名	数据类型	说明
<vs-id>	字符串 （具有格式）	虚拟服务器信息。每个应用程序服务器实例都可能有一个或多个虚拟服务器。可以从 http-server 的摘要 (summary) 属性获取虚拟服务器 ID 的列表。可以使用 server1.http-server.virtual-server.<vs-id> 格式的 get 命令参数查找特定虚拟服务器的统计数据。可以使用 server1.http-server.virtual-server.* 格式的 get 命令参数查找所有虚拟服务器的统计数据。 注意：有关具有格式的字符串中暴露了哪些数据的详细信息，请参见第 130 页上的 “HTTP 服务器的可监视对象”一节。

process

属性名	数据类型	说明
<pid>	字符串 （具有格式）	进程信息。每个应用程序服务器实例都有一个进程。可以从 http-server 的摘要 (summary) 属性获取进程 ID。可以使用 server1.http-server.process.<pid> 格式的 get 命令参数获取进程的统计数据。 注意：有关具有格式的字符串中暴露了哪些数据的详细信息，请参见第 130 页上的 “HTTP 服务器的可监视对象”一节。

orb-connection

属性名	数据类型	说明
total-inbound-connections	整数	至 ORB 的入站连接总数。
total-outbound-connections	整数	自 ORB 的出站连接总数。

orb-thread-pool

属性名	数据类型	说明
thread-pool-size	整数	ORB 线程池中的线程总数。
waiting-thread-count	整数	等待工作到来的线程池线程数。

jdbc-connection-pool

属性名	数据类型	说明
total-threads-waiting	整数	等待 JDBC 连接的线程总数。
total-outbound-connections	整数	JDBC 连接验证失败总数。
total-connections-timed-out	整数	超时的连接请求总数。

transaction-service

属性名	数据类型	说明
total-tx-completed	整数	完成的事务总数。
total-tx-rolled-back	整数	回滚的事务总数。
total-tx-inflight	整数	正在执行 （活动）的事务总数。
isFrozen	字符串	事务系统是否被冻结 （True 或 False）？
inflight-tx	字符串 （具有格式）	正在执行的事务列表。

bean-pool

属性名	数据类型	说明
max-pool-size	整数	池中 Bean 实例的最大数目。
steady-pool-size	整数	池中通常保留的 Bean 实例数。首次创建池时，池的总装大小等于 steady-pool-size。从池中删除实例时，池将被异步补充，所以池的大小将等于或大于 steady-pool-size。
pool-resize-quantity	整数	池增长到 max-pool-size 或缩小到 steady-pool-size 的增量。
idle-timeout-in-seconds	整数	定义池清除线程的执行频率。检查池的当前大小是否大于 steady-pool-size。如果大于，则删除 pool-resize-quantity 元素；如果小于，则增加 pool-resize-quantity，但不会超过 min (current-pool-size+pool - resize-quantity, max-pool-size)。只有当对象在 pool-idle-timeout-in-seconds 时间内未被访问时，才能删除该对象。
num-beans-in-pool	整数	池中可用 Bean 的数目。
num-threads-waiting	整数	等待空闲 Bean 的线程数。
total-beans-created	整数	到目前为止创建的 Bean 的数目。
total-beans-destroyed	整数	到目前为止破坏的 Bean 的数目。
jms-max-messages-load	整数	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。

bean-cache

属性名	数据类型	说明
cache-resize-quantity (resize-quantity)	整数	当高速缓存中 Bean 的数目等于 max-cache-size 时（即发生高速缓存溢出时），高速缓存的大小所减少的数量。
cache-misses	整数	用户请求未在高速缓存中找到 Bean 的次数。
idle-timeout-in-seconds	整数	所安排的高速缓存清除器线程的执行频率。此清除器线程将检查高速缓存中的所有 Bean 并挂起那些在 cache-idle-timeout-in-seconds 时间内没有被访问的 Bean。
cache-hits	整数	用户请求在高速缓存中找到某条目的次数。
total-beans-in-cache	整数	高速缓存中 Bean 的数目。这是高速缓存的当前大小。
max-beans-in-cache	整数	高速缓存中可保留的 Bean 的最大数目，超过此值将发生高速缓存溢出。
num-passivations	整数	被挂起的数目。仅适用于状态会话 Bean。
num-passivation-errors	整数	挂起时的错误数。仅适用于状态会话 Bean。
num-expired-sessions-removed	整数	清除线程删除的过期会话数。仅适用于状态会话 Bean。
num-passivation-success	整数	成功完成挂起的次数。仅适用于状态会话 Bean。

bean-method

属性名	数据类型	说明
method-name	字符串	方法的全限定名称。
total-num-calls	整数	方法被调用的次数。如果启用了 EJB 容器的监视，则会为无状态会话 Bean、状态会话 Bean 以及实体 Bean 收集此数据；如果启用了消息驱动的 Bean 容器的监视，则会为消息驱动的 Bean 收集此数据。
total-num-errors	整数	方法执行时出现异常的次数。如果在 EJB 设置下启用了监视，则会为无状态会话 Bean、状态会话 Bean 以及实体 Bean 收集此数据；如果在 MDB 设置下启用了监视，则会为消息驱动的 Bean 收集此数据。
total-num-success	整数	方法成功执行的次数。如果启用了 EJB 容器的监视，则会为无状态会话 Bean、状态会话 Bean 以及实体 Bean 收集此数据；如果启用了容器的监视，则会为消息驱动的 Bean 收集此数据。

bean-method *（续）*

属性名	数据类型	说明
execution-time-millis	长型	最后一次成功运行此方法时此方法的执行时间。如果启用了 EJB 容器的监视，则会为无状态会话 Bean、状态会话 Bean 以及实体 Bean 收集此数据；如果启用了容器的监视，则会为消息驱动的 Bean 收集此数据。

HTTP 服务器的可监视对象

HTTP 服务器的可监视属性名 summary 将显示 Server 元素的属性值及其子元素的摘要，包括每个子元素的数目及每个子元素的属性值。 HTTP 服务器的 virtual-server 属性将显示 VirtualServer 元素的属性值及其每个子元素的详细资料。 process 属性将显示 Process 元素的属性值及其每个子元素的详细资料。

要启用 NSAPI 性能探查并获取 Profile 和 ProfileBucket 元素的统计数据，请参见《*Sun ONE Application Server Developer's Guide to NSAPI*》。

有关如何使用监视统计数据进行性能优化的信息，请参见 《*Sun ONE Application Server Performance and Tuning Guide*》。

可监视的 HTTP 服务器元素

下表列出了 HTTP 服务器的可监视元素。

可监视的 HTTP 服务器元素

元素名	子元素	说明
Server	ConnectionQueue ThreadPool Profile Process VirtualServer	服务器实例。
ConnectionQueue	无	该队列中的请求优先于正在被服务的请求。Sun ONE Application Server 7 中只有一个连接队列。
ThreadPool	无	在 init.conf 文件中定义的线程池。
Profile	无	在 init.conf 文件中定义的 NSAPI 性能概要桶。

可监视的 HTTP 服务器元素 *（续）*

元素名	子元素	说明
Process	ConnectionQueueBucketThr eadPoolBucket DnsBucket DnsBucket KeepaliveBucket CacheBucket Thread	服务器实例中的单个服务器进程。
ConnectionQueueBucket	无	跟踪与特定 ConnectionQueue 有关的统计数据。
ThreadPoolBucket	ThreadPoolBucket	跟踪与特定 ThreadPool 有关的统计数据。
DnsBucket	无	跟踪 DNS 统计数据。
KeepaliveBucket	无	跟踪保持活动 （永久连接）的统计数据。
CacheBucket	无	跟踪文件高速缓存 (NSFC) 统计数据。
Thread	RequestBucket ProfileBucket	描述了某个请求处理线程。
VirtualServer	RequestBucket ProfileBucket	描述了某个虚拟服务器。
RequestBucket	无	跟踪与请求相关的统计数据。
ProfileBucket		跟踪与 Profile 元素有关的统计数据。

可监视的 HTTP 服务器属性

下表列出了 HTTP 服务器的可监视属性。

Server

属性名	值	说明
Id		服务器实例 ID （例如， server1）。
VersionServer		包含 Sun ONE Application Server 版本的字符串。
TimeStarted	GMT	此服务器实例的启动时间。
SecondsRunning		此服务器实例启动后的运行时间 （秒数）。
TicksPerSecond		每秒的周期数。此值由系统决定。
MaxProcs		最大进程数。
MaxThreads		最大处理线程数。

Server *（续）*

属性名	值	说明
MaxVirtualServers		所跟踪的虚拟服务器的最大数目。
FlagProfilingEnabled	0（关），1（开）	指示是否启用了 NSAPI 性能探查（启用时为 1）。
FlagVirtualServerOverflow	0（否），1（是）	指示是否配置了多于 MaxVirtualServers 的虚拟服务器（多于时为 1）。如果此属性设置为 1，则并非所有虚拟服务器的统计数据都被跟踪。
LoadMinuteAverage		1 分钟内的负载平均值。
Load5MinuteAverage		5 分钟内的负载平均值。
Load15MinuteAverage		15 分钟内的负载平均值。
RateBytesTransmitted	每秒字节数	某个服务器定义的时间间隔内数据的传输速率。如果此信息不可用，则值为 0。
RateBytesReceived	每秒字节数	某个服务器定义的时间间隔内数据的接收速率。如果此信息不可用，则值为 0。

ConnectionQueue

属性名	值	说明
Id		连接队列 ID。

ThreadPool

属性名	值	说明
Id		线程池 ID。
名称		线程池的符号名称。

Profile

属性名	值	说明
Id		NSAPI 性能概要桶 ID。
名称		NSAPI 性能概要桶的符号名称。

Profile *（续）*

属性名	值	说明
说明		NSAPI 性能概要桶的说明。

Process

属性名	值	说明
Pid		唯一标识此进程的操作系统进程标识符。
Mode	unknown active	此进程处于活动状态时显示 “active”。
TimeStarted	GMT	此进程的启动时间。
CountConfigurations		某配置被装入的次数。如果此信息不可用，则值为 0。
SizeVirtual	千字节	此进程使用的虚拟内存大小。
SizeResident	千字节	此进程使用的驻留内存大小。
FractionSystemMemoryUsage		此进程使用的系统内存部分。

ConnectionQueueBucket

属性名	值	说明
ConnectionQueue		ConnectionQueue 元素的 ID。
CountTotalConnection		已接受的新连接总数。
CountQueued		当前排队的连接数。
PeakQueued		队列中同时存在的最大连接数。
MaxQueued		队列可容纳的最大连接数。
CountOverflow		队列太满而无法容纳连接的次数。
CountTotalQueued		排队的连接总数。某个给定连接可能被多次排队，因此 CountTotalQueued 可能大于或等于 CountTotalConnections。
TicksTotalQueued		连接在队列中花费的周期总数。周期是系统决定的时间单位，请参见 TicksPerSecond。

ThreadPoolBucket

属性名	值	说明
Thread-pool		ThreadPool 元素的 ID。
CountThreadsIdle		当前处于空闲状态的请求处理线程数。
CountThreads		请求处理线程的数目。
MaxThreads		可同时存在的最大请求处理线程数。
CountQueued		排队等候此线程池处理的请求数。
PeakQueued		队列中同时存在的最大请求数。
MaxQueued		队列可容纳的最大请求数。

DnsBucket

属性名	值	说明
FlagCacheEnabled	0 （关）， 1 （开）	指示是否启用了 DNS 高速缓存 （启用时为 1）。
CountCacheEntries		当前高速缓存中的 DNS 条目数。
MaxCacheEntries		高速缓存可容纳的最大 DNS 条目数。
CountCacheHits		DNS 高速缓存查找成功的次数。
CountCacheMisses		DNS 高速缓存查找失败的次数。
FlagAsyncEnabled	0 （关）， 1 （开）	指示是否启用了异步 DNS 查找 （启用时为 1）。
CountAsyncNameLookups		已执行的异步 DNS 名称查找总数。
CountAsyncAddrLookups		已执行的异步 DNS 地址查找总数。
CountAsyncLookupsInProgress		当前正在执行的异步 DNS 查找总数。

KeepaliveBucket

属性名	值	说明
CountConnections		当前处于保持活动模式的连接数。
MaxConnections		同时保持活动的最大连接数

KeepaliveBucket *（续）*

属性名	值	说明
CountHits		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		服务器关闭保持活动的连接的次数。
CountTimeouts		保持活动的连接超时的次数。
SecondsTimeouts		超时的秒数，之后服务器将关闭空闲的保持活动的连接。
CountRefusals		服务器拒绝保持活动的连接的次数。

CacheBucket

属性名	值	说明
FlagEnabled	0（关），1（开）	指示是否启用了文件高速缓存 （启用时为 1）。
SecondsMaxAge	秒数	文件高速缓存条目的最长寿命。
CountEntries		文件高速缓存中的当前条目数。
MaxEntries		文件高速缓存可以同时容纳的最大高速缓存条目数。
CountOpenEntries		与打开的文件关联的条目数。
MaxOpenEntries		文件高速缓存可以同时容纳的与打开的文件关联的高速缓存条目的最大数目。
SizeHeapCache	字节数	高速缓存的文件内容使用的堆的大小。
MaxHeapCacheSize	字节数	文件高速缓存用于高速缓存的文件内容的最大堆大小。
SizeMmapCache	字节数	内存映射的文件内容使用的地址空间大小。
MaxMmapCacheSize	字节数	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		高速缓存条目查找成功的次数。
CountMisses		高速缓存条目查找失败的次数。
CountInfoHits		文件信息查找成功的次数。
CountInfoMisses		文件信息查找失败的次数。
CountContentHits		内容查找成功的次数。

CacheBucket *（续）*

属性名	值	说明
CountContentMisses		内容查找失败的次数。

Thread

属性名	值	说明
Mode	unknown、 idle、 DNS、 request、 processing、 response、 updating	线程最后的已知状态。
TimeStarted	GMT	此线程的启动时间。
ConnectionQueue		线程正在服务的 ConnectionQueue 的 ID。

VirtualServer

属性名	值	说明
Id		虚拟服务器 ID。
Mode	unknown、 active	此虚拟服务器处于活动状态时显示 “active”。
Hosts		此虚拟服务器所服务的软件虚拟服务器主机名 （例如， www.foo.com foo.com foo.isp.com）。
Interfaces		为其配置了该虚拟服务器的接口 （监听器）（例如， 192.168.1.2:80 192.168.1.2:443）。

RequestBucket

属性名	值	说明
CountRequests		已服务的请求数。
CountBytesReceived		收到的字节数。如果此信息不可用，则值为 0。
CountBytesTransmitted		传输的字节数。如果此信息不可用，则值为 0。
RateBytesTransmitted	每秒字节数	某个服务器定义的时间间隔内数据的传输速率。如果此信息不可用，则值为 0。

RequestBucket *（续）*

属性名	值	说明
MaxByteTransmissionRate		某个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		打开的连接的数目。如果此信息不可用，则值为 0。
MaxOpenConnections		打开的连接的最大数目。如果此信息不可用，则值为 0。
Count2xx		已发送的 200 级响应的数目。
Count3xx		已发送的 300 级响应的数目。
Count4xx		已发送的 400 级响应的数目。
Count5xx		已发送的 500 级响应的数目。
CountOther		已发送的非 200、 300、 400 或 500 级响应的数目。
Count200		已发送的 200 级响应的数目。
Count302		已发送的 302 级响应的数目。
Count304		已发送的 304 级响应的数目。
Count400		已发送的 400 级响应的数目。
Count401		已发送的 401 级响应的数目。
Count403		已发送的 403 级响应的数目。
Count404		已发送的 404 级响应的数目。
Count503		已发送的 503 级响应的数目。

ProfileBucket

属性名	值	说明
Profile		Profile 元素的 ID。
Countcalls		NSAPI SAF 的调用次数。
CountRequests		已处理的请求数。
TicksDispatch		分发请求花费的周期数。周期是系统决定的时间单位，请参见 TicksPerSecond。
TicksFunction		在 NSAPI SAF 中花费的周期数。周期是系统决定的时间单位，请参见 TicksPerSecond。

使用 CLI 管理事务服务

您可以使用 set 命令管理要为 JTS 监视的统计数据。

示例 1

要将事务添加到回滚列表中 （这将导致指定事务的回滚），请按以下格式发出 set 命令：

```
set --monitor server1.transaction-service.rollback-list=txnid1
```

示例 2

要冻结事务服务，请按以下格式发出 set 命令：

```
set --monitor server1.transaction-service.freeze=true
```

下表描述了可以监视以便为 JTS 收集统计数据的属性。这些属性可以按照第 121 页上的 “CLI 名称映射”中描述的规则通过命令行进行设置。

有关 Java 事物服务的详细信息，请参见第 9 章 “使用事务服务”。

使用 HTTP 服务质量

以下设置控制了通信量的计算方式以及带宽的重新计算频率：

- 重新计算时间间隔 - 指示带宽的计算频率 （毫秒）。
- 公制时间间隔 - 数据可以用于通信量计算的时间段。

在管理界面中，您可以为服务器实例或虚拟服务器的某个类启用这些服务器或类级别的设置。也可以为单独的虚拟服务器覆盖这些设置。

本节包括以下主题：

- 服务质量示例
- 配置服务质量 (QOS)
- 必须对 obj.conf 文件进行的更改
- 服务质量的已知限制

服务质量示例

以下示例显示了如何收集并计算服务质量信息。

- 服务器的公制时间间隔为 30 秒。
- 服务器在 0 秒启动。
- 在 1 秒时，至/自该服务器的 HTTP 连接产生了 5000 字节的通信量。
- 之后没有进行更多连接。在 30 秒时，最后 30 秒内的总通信量为 5000 字节。
- 在 32 秒时，在 1 秒时获得的通信量采样数据被放弃，因为已经超过了 30 秒的公制时间间隔。现在，最后 30 秒的总通信量为 0。

重新计算时间间隔的工作方式与此类似。服务器的重新计算时间间隔为 100 毫秒。

我们继续前面的示例，每隔 100 毫秒重新计算一次带宽。该计算以通信量和公制时间间隔为基础。

- 在 0 秒时，将首次计算带宽。总通信量为 0，除以 30 秒的公制时间间隔，得出带宽为 0。
- 在 1 秒时，将第 10 次计算带宽 （1000 毫秒/100 毫秒）。总通信量为 5000 字节，除以 30 秒，得出带宽为 5000/30 = 166 字节/秒。
- 在 30 秒时，将第 300 次计算带宽。总通信量为 5000 字节，除以 30 秒，得出带宽为 5000/30 = 166 字节/秒。
- 在 32 秒时，将第 320 次计算带宽。现在通信量为 0 （因为产生通信量的那次连接发生的时间太早，所以不能计算在内），除以 30 秒，得出带宽为 0 字节/秒。

配置服务质量 (QOS)

通过管理界面可以配置服务器实例或虚拟服务器的类的服务质量。

注意 要强制服务质量设置，您还必须在 obj.conf 文件中设置服务器应用程序功能 (SAF)，如第 142 页上的 “必须对 obj.conf 文件进行的更改” 中所述。

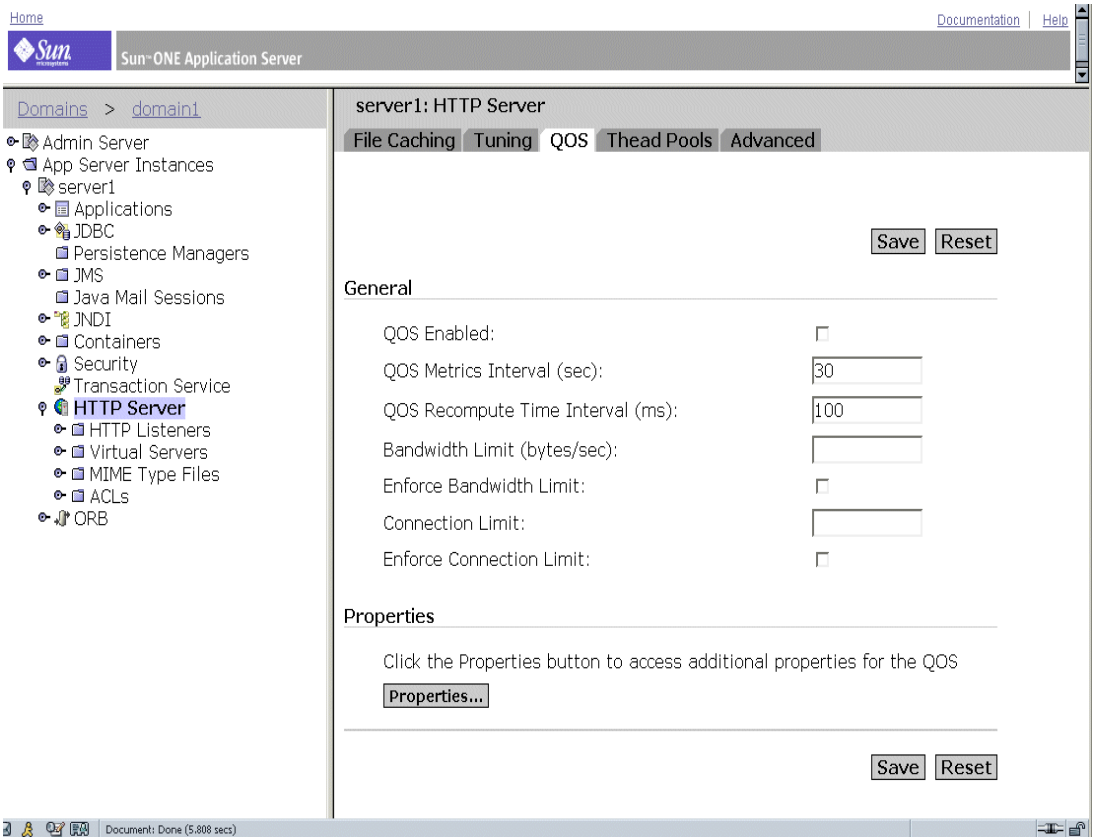
要配置服务质量，请执行以下步骤：

1. 在左侧窗格中选择 “App Server Instances”节点。
2. 展开服务器实例节点以显示 “HTTP Server”节点。

3. 单击 “HTTP Server”节点以显示 “QOS”选项卡。
4. 单击 “QOS”选项卡。

   以下页面显示了服务质量的常规设置，下面还有一个 “Properties”按钮。

虚拟服务器实例 “QOS”选项卡



5. 要启用此 HTTP 服务器的服务质量，请单击 “QOS Enabled”。

   注意：默认情况下，服务质量被禁用。启用服务质量会稍微增加服务器的开销。
6. 指定 “QOS Metrics Interval”。

公制时间间隔是在服务器通信量计算过程中采样数据的时间间隔（秒）。默认值为 30 秒。

如果您的站点通常要传输大量文件，请为此字段设置一个较大的值（几分钟或更长）。如果公制时间间隔较短，大量文件的传输可能会占用所允许的所有带宽。如果您强制了最大带宽设置，这将导致连接被拒绝。由于带宽是按公制时间间隔均分的，因此较长的时间间隔可以缓和由大量文件引起的高峰通信量。

如果带宽限制比可用带宽低很多（例如，带宽限制为 1 Mbps，但是与主干的连接为 1 Gbps），则应缩短公制时间间隔。

注意：如果要传输大量静态文件，而带宽限制比可用带宽低很多，则必须决定调整哪种状况，因为这两个问题所需的解决方案正好是相反的。

7. 指定 “QOS Recompute Time Interval”。

   重新计算时间间隔是所有服务器、类及虚拟服务器的每次带宽计算之间的毫秒数。默认值为 100 毫秒。

8. 指定 “Bandwidth Limit”。

   这是服务器实例的最大带宽限制（字节/秒）。它在某种程度上与 QOS 公制时间间隔相互影响。

9. 选择是否强制最大带宽设置。

   如果选择强制最大带宽，当服务器达到其带宽限制时，额外的连接将被拒绝。

   如果没有强制最大带宽，当超过最大值时，服务器将在事件日志中记录一条消息。

10. 指定连接限制。

    这是所处理的并行请求数。

11. 选择是否强制连接限制设置。

    如果选择强制最大连接数，当到服务器达到其最大连接数目时，额外的连接将被拒绝。如果没有强制最大连接数，当超过最大值时，服务器将在事件日志中记录一条消息。

12. （可选）。如果要为服务质量指定附加的 “名称 - 值”对属性，请单击 “Properties”按钮。

    有关服务质量属性的所允许的 “名称 - 值”对列表，请参见联机帮助。

13. 单击 “Save”，提交对服务器实例所做的更改。

14. 访问左侧窗格中 “App Server Instances”下您的服务器实例，然后单击 “Apply Changes”。

必须对 obj.conf 文件进行的更改

要强制服务质量，您必须在 obj.conf 文件中包含指令以调用以下服务器应用程序功能 (SAF)：

- AuthTrans qos-handler
- Error qos-error

要正常工作， qos-handler AuthTrans 指令必须是在默认对象中配置的第一个 AuthTrans。服务质量处理程序的作用是检查虚拟服务器、虚拟服务器类以及全局服务器的当前统计数据并通过返回错误来强制限制。 Sun ONE Application Server 包含一个称为 qos-handler 的内置样例服务质量处理程序 SAF。当达到限制时，此 SAF 将进行记录并向服务器返回一个 503 Server busy 错误，以便由 NSAPI 处理。

Sun ONE Application Server 还包含一个称为 qos-error 的内置样例错误 SAF，它将返回一个错误页面，说明引起 503 错误的限制以及触发该限制的统计数据的值。

有关这些 SAF 以及如何使用它们的详细信息，请参见 *《Sun ONE Application Server Developer's Guide to NSAPI》*。

服务质量的已知限制

使用服务质量功能时，请记住以下限制：

- 服务质量功能仅测量应用程序级别的 HTTP 带宽。由于以下多种原因， HTTP 带宽可能与实际的 TCP 网络带宽不同：
  - 如果启用了 SSL，握手和客户机证书交换将增大通信量，但是不会被测量。
  - 如果单向或双向启用了块编码，块层将删除块消息头，并且块消息头不会计入到通信量中。其它消息头或协议项将被计入到通信量中。
- 服务质量功能不能精确测量 PR_TransmitFile 调用的通信量。对于诸如 PR_Send()/net_write 或 PR_Recv()/net_read 等基本 I/O 操作，带宽管理器可以快速计算出传输的数据，因为在一个系统调用中传输的字节数通常即是缓冲区的大小，并且 I/O 调用会快速返回。这非常适用于测量动态内容应用程序的即时带宽。但是，因为 PR_TransmitFile 传输的数据量仅在传输结束时才会知道，所以在传输完成前无法进行测量。

  如果 PR_TransmitFile 很短，则服务质量功能将能够充分执行。但是，如果 PR_TransmitFile 很长，例如在拨号用户下载长文件的情况下，传输的全部数据量将在完成时计算。当带宽管理器在下一个重新计算时间间隔开始后重新计算带宽时，由于最近发生的这一较大的 PR_TransmitFile，计算的带宽将显著增加。这种

状况可能导致服务器拒绝所有请求，直至下一个公制时间间隔。那时带宽管理器将使该传输文件操作过期 （因为它发生的时间已经太早），从而使带宽值回落。如果您的站点通常要下载很长的静态文件，您应当增加原来默认为 30 秒的公制时间间隔。

- 计算的带宽始终是一个近似值，因为它不是即时测量的，而是在一段时间内以固定的时间间隔计算的。例如，如果公制时间间隔是默认的 30 秒，而服务器空闲了 29 秒，然后在下一秒中，客户机有可能在 1 秒内使用了 30 倍的带宽限制。
- 无论何时动态重新配置了服务器，都将丢失服务质量带宽统计数据。此外，服务质量限制在具有基于较早的、非活动配置的连接的线程中不会被强制，因为带宽管理器线程仅计算活动配置的带宽统计数据。还有一种潜在可能，即长时间未关闭其套接字并保持活动（从而服务器没有使其超时）的客户机在动态重新配置服务器后不会再受服务质量限制的约束。
- 虚拟服务器的并行连接计算间隔与虚拟服务器类和全局服务器实例的计算间隔不同。单个虚拟服务器的连接计数器在请求被分析并路由到虚拟服务器后会立即自动递增，而在该请求的响应处理结束时会自动减少。这意味着虚拟服务器的连接统计数据在任意时刻都是准确的。

  但是，虚拟服务器类和全局服务器实例的连接统计数据不能即时更新。它们按照每个重新计算时间间隔由带宽管理器线程更新。虚拟服务器类的连接数是该类的所有虚拟服务器上的连接总数；全局服务器实例的连接数是所有虚拟服务器类上的连接总数。

  由于这些值的计算方式，虚拟服务器的连接数始终是正确的，如果为该连接数设置一个强制限制，则永远也不能超过该限制。虚拟服务器类和服务器实例的值则不会如此准确，因为它们是按照一定的时间间隔计算的。

关于 SNMP

简单网络管理协议 (SNMP) 是用于在网络范围内交换管理信息和监视信息的协议。使用 SNMP，数据可以在被管理的设备和网络管理站 (NMS) 之间传输。*被管理的设备*可以是运行 SNMP 的任何设备：主机、路由器、 HTTP 服务器以及网络上的其它服务器。

本节包括以下主题：

- 网络管理站 (NMS)
- 管理信息库 (MIB) 对象
- SNMP 消息
- SNMP 陷阱目标
- SNMP 代理社区

网络管理站 (NMS)

网络管理站 (NMS) 是用于远程管理某特定网络的计算机。通常，NMS 软件将提供一个图形界面以显示收集的数据，或者使用该数据确保服务器在指定的公差范围内运行。

NMS 通常是安装有一个或多个网络管理应用程序的功能强大的工作站。网络管理应用程序 （例如 HP OpenView）以图形方式显示有关被管理设备 （例如 HTTP 服务器）的信息。例如，它可能显示了您的企业中已打开或关闭的服务器，或者收到的错误信息的数量和类型。将 SNMP 与 Sun ONE Application Server 一起使用时，这些信息将通过使用以下两种代理在 NMS 和服务器之间传输：子代理和主代理。

子代理收集在各个域中运行的服务器实例的信息并将信息传递给主代理。每个 Sun ONE Application Server 安装都有一个主代理和一个子代理。

注意	对 SNMP 配置进行任何更改后，必须单击 “Apply” 按钮，然后重新启动 SNMP 子代理。

主代理 在各个子代理和 NMS 之间交换信息。主代理随 Sun ONE Application Server 一起安装。

您可以在一台主机上安装多个子代理，但是只能安装一个主代理。例如，如果在同一台主机上安装了 Sun ONE Directory Server、 Sun ONE Application Server 及 Sun ONE Messaging Server，则每个服务器的子代理将与同一个主代理通信。

NMS 将从服务器请求信息，或者更改存储在服务器 MIB 中的变量的值。例如：

1. NMS 将消息发送给管理服务器主代理。消息可能是数据请求 （一条 GET 消息），也可能是一条指示在 MIB 中设置变量的指令 （一条 SET 消息）。
2. 主代理将消息转发给相应的子代理。
3. 子代理将检索数据或更改 MIB 中的变量。
4. 子代理将数据或状态报告给主代理，然后主代理将消息 （一条 GET 消息）转发回 NMS。
5. NMS 通过它的网络管理应用程序以文字或图形方式显示该数据。

管理信息库 (MIB) 对象

Sun ONE Application Server 存储了与管理和监视网络范围内的信息有关的变量。主代理可以访问的变量称为被管理对象。这些对象在称为 *管理信息库* (MIB) 的树状结构中定义。 MIB 提供了对 HTTP 服务器的网络配置、状态及统计数据的访问。使用 SNMP，您可以通过网络管理工作站 (NMS) 查看这些信息。

MIB 树的顶层显示出 Internet 对象标识符具有以下子树：

- directory (1)
- mgmt (2)
- experimental (3)
- private (4)

private (4) 子树包含 enterprises (1) 节点。 enterprises (1) 节点中的每个子树都被分配给一个单独的企业，企业是注册有自己特定 MIB 扩展的组织。然后企业可以在其子树下创建特定产品的子树。由企业创建的 MIB 位于 enterprises (1) 节点下。

每个 Sun ONE Application Server 子代理都提供了一个用于 SNMP 通信的 MIB。服务器通过发送包含这些变量的消息 （或陷阱）将重要事件报告给 NMS。 NMS 可以查询服务器的 MIB 以获取数据。

每个 Sun ONE Application Server 都有其自己的 MIB，该 MIB 位于 *install_dir*/lib 下。

Sun ONE Application Server 的 MIB 是一个称为 appserv.mib 的文件。此 MIB 包含与 Sun ONE Application Server 的网络管理有关的各种变量的定义。

Sun ONE Application Server MIB 具有一个

appserver 1 (as appserver7 OBJECT IDENTIFIER ::= {appserver 1 }) 对象标识符，它位于 *install_dir*/lib 目录下。

您可以使用 Sun ONE Application Server MIB 查看有关 Sun ONE Application Server 的管理信息并实时监视服务器。下表列出并描述了存储在 appserv.mib 文件中的被管理对象。

appserv.mib 被管理对象和说明

被管理对象	说明
iwsCpuID	CPU 标识符。
iwsCpuIdleTime	CPU 空闲时间。
iwsCpuKernelTime	CPU 内核时间。
iwsCpuTable	Sun ONE Application Server CPU。

appserv.mib 被管理对象和说明 *（续）*

被管理对象	说明
iwsCpuUserTime	CPU 用户时间。
iwsInstanceTable	Sun ONE Application Server 实例。
iwsInstanceId	服务器实例标识符。
iwsInstanceVersion	字符串，例如 SunONE-ApplicationServer-Enterprise/7 BB1-01/24/2001 17:15 (SunOS DOMESTIC)。
iwsInstanceDescription	服务器实例的说明。
iwsInstanceOrganization	负责服务器实例的组织。
iwsInstanceContact	负责服务器实例的人员的联系信息。
iwsInstanceLocation	服务器的位置。
iwsInstanceStatus	服务器实例的状态。
iwsInstanceUptime	服务器已经运行的时间。
iwsInstanceDeathCount	服务器实例进程关闭的次数。
iwsInstanceRequests	服务器实例处理的请求数。
iwsInstanceInOctets	服务器实例接收的八位位组数。如果此信息不可用，将显示 0。
iwsInstanceOutOctets	服务器实例传输的八位位组数。如果此信息不可用，将显示 0。
iwsInstanceCount2xx	服务器实例发出的 200 级 （成功）响应的数目。
iwsInstanceCount3xx	服务器实例发出的 300 级 （重定向）响应的数目。
iwsInstanceCount4xx	服务器实例发出的 400 级 （客户机错误）响应的数目。
iwsInstanceCount5xx	服务器实例发出的 500 级 （服务器错误）响应的数目。
iwsInstanceCountOther	服务器实例发出的其它（非 2xx、3xx、4xx 或 5xx）响应的数目。
iwsInstanceCount200	服务器实例发出的 200 （成功）响应的数目。
iwsInstanceCount302	服务器实例发出的 302 （暂时移动）响应的数目。
iwsInstanceCount304	服务器实例发出的 304 （未修改）响应的数目。
iwsInstanceCount400	服务器实例发出的 400 （错误请求）响应的数目。
iwsInstanceCount401	服务器实例发出的 401 （未授权）响应的数目。
iwsInstanceCount403	服务器实例发出的 403 （禁止）响应的数目。
iwsInstanceCount404	服务器实例发出的 404 （未找到）响应的数目。

appserv.mib 被管理对象和说明 *（续）*

被管理对象	说明
iwsInstanceLoad1MinuteAverage	运行服务器实例的系统的 1 分钟负载平均值。
iwsInstanceLoad5MinuteAverage	运行服务器实例的系统的 5 分钟负载平均值。
iwsInstanceLoad15MinuteAverage	运行服务器实例的系统的 15 分钟负载平均值。
iwsInstanceNetworkInOctets	网络上每秒传输的八位位组数。
iwsInstanceNetworkOutOctets	网络上每秒接收的八位位组数。
iwsVsTable	服务器的虚拟服务器。
iwsVsId	虚拟服务器标识符。
iwsVsRequests	虚拟服务器处理的请求数。
iwsVsInOctets	虚拟服务器接收的八位位组数。
iwsVsOutOctets	虚拟服务器传输的八位位组数。
iwsVsCount2xx	虚拟服务器发出的 200 级 （成功）响应的数目。
iwsVsCount3xx	虚拟服务器发出的 300 级 （重定向）响应的数目。
iwsVsCount4xx	虚拟服务器发出的 400 级 （客户机错误）响应的数目。
iwsVsCount5xx	虚拟服务器发出的 500 级 （服务器错误）响应的数目。
iwsVsCountOther	虚拟服务器发出的其它（非 2xx、3xx、4xx 或 5xx）响应的数目。
iwsVsCount200	虚拟服务器发出的 200 （成功）响应的数目。
iwsVsCount302	虚拟服务器发出的 302 （暂时移动）响应的数目。
iwsVsCount304	虚拟服务器发出的 304 （未修改）响应的数目。
iwsVsCount400	虚拟服务器发出的 400 （错误请求）响应的数目。
iwsVsCount401	虚拟服务器发出的 401 （未授权）响应的数目。
iwsVsCount403	虚拟服务器发出的 403 （禁止）响应的数目。
iwsVsCount404	虚拟服务器发出的 404 （未找到）响应的数目。
iwsProcessTable	Sun ONE Application Server 进程。
iwsProcessId	操作系统进程标识符。
iwsProcessThreadCount	请求处理线程的数目。
iwsProcessThreadIdle	当前处于空闲状态的请求处理线程数。
iwsProcessConnectionQueueCount	连接队列中的当前连接数。
iwsProcessConnectionQueuePeak	同时排队的最大连接数。

appserv.mib 被管理对象和说明 *（续）*

被管理对象	说明
iwsProcessConnectionQueueMax	连接队列中所允许的最大连接数。
iwsProcessConnectionQueueTotal	已接受的连接数。
iwsProcessConnectionQueueOverflows	由于连接队列溢出而被拒绝的连接数。
iwsProcessKeepaliveCount	保持活动的队列中的当前连接数。
iwsProcessKeepaliveMax	保持活动的队列中所允许的最大连接数。
iwsProcessSizeVirtual	进程大小 （千字节）。
iwsProcessSizeResident	进程驻留大小 （千字节）。
iwsProcessFractionSystemMemoryUsage	系统内存中的进程部分。
iwsListenTable	Sun ONE Application Server 监听套接字。
iwsListenId	监听套接字标识符。
iwsListenAddress	套接字监听的地址。
iwsListenPort	套接字监听的端口。
iwsListenSecurity	加密支持。
iwsThreadPoolCount	被拒绝的请求数。
iwsThreadPoolMax	队列中所允许的最大请求数。
iwsThreadPoolPeak	同时排队的最大请求数。
iwsThreadPoolTable	Sun ONE Application Server 线程池。
iwsVsCount503	已发出的 503 （不可用）响应的数目。
iwsInstanceCount503	已发出的 503 （不可用）响应的数目。

SNMP 消息

GET 和 SET 是 SNMP 定义的两种消息。

MIB 中的每个对象都分配有一个唯一的标识符。 SNMP 管理器通过发出 GET 和 GETNEXT 命令 （在其中指定对象的唯一标识符）来访问对象。代理的代理程序将获取指定对象的值并将其传输到 SNMP 管理器。如果添加到日志的事件符合陷阱过滤器条件，则可能生成 SNMP 陷阱。未生成陷阱的事件仅记录为维护日志表中的一个条目，由 SNMP 管理器通过标准的 GET 和 GETNEXT 命令访问。

GET 和 SET 消息将由网络管理站 (NMS) 发送给主代理。您可以通过管理界面使用其中一个或两个都使用。

SNMP 以协议数据单元 (PDU) 的格式交换网络信息。这些单元包含有关存储在被管理设备 （例如 HTTP 服务器）上的变量的信息。这些变量 （也称为被管理对象）具有值和消息头，值和消息头将在需要时报告给 NMS。由服务器发送给 NMS 的协议数据单元称为*陷阱*。以下各节将对 GET、 SET 的使用以及陷阱消息进行详细讨论。

SNMP 陷阱目标

*SNMP 陷阱*是 SNMP 代理发送给网络管理站 (NMS) 的消息。例如，当接口的状态由打开变为关闭时， SNMP 代理将发送一个陷阱。 SNMP 代理必须知道 NMS 的地址，以便知道向何处发送陷阱。

您可以通过 Sun ONE Application Server 管理界面为 SNMP 主代理配置陷阱目标。还可以查看、编辑并删除已配置的陷阱目标。使用管理界面配置陷阱目标时，实际上是在编辑 CONFIG 文件。

发生重要事件时，服务器子代理将向 NMS 发送一条消息 （或陷阱）。例如：

1. 子代理通知主代理服务器已停止。
2. 主代理发送一条消息 （或陷阱），将该事件报告给 NMS。
3. NMS 通过它的网络管理应用程序以文字或图形方式显示该信息。

有关设置 SNMP 陷阱端口的说明，请参见第 153 页上的 “安装 SNMP 主代理”。

SNMP 代理社区

*SNMP 代理社区*由一个社区字符串及分配给该指定社区的操作组成。社区字符串是网络管理站 (NMS) 名称的文本字符串， SNMP 将用它进行授权。这意味着 NMS 在发送给代理的每条消息中都带有一个社区字符串。

所分配的操作是 get 和/或 set。然后， SNMP 代理可以验证 NMS 是否被授权执行 get 和/或 set 操作以便进行数据交换。社区字符串在 SNMP 包中发送时不被隐藏；字符串以 ASCII 文本格式发送。

您可以通过管理界面为每个指定的社区配置并管理社区字符串及允许的操作。有关设置 SNMP 代理社区的说明，请参见第 153 页上的 “安装 SNMP 主代理”。

设置 SNMP

一般来讲，要使用 SNMP，系统上必须安装并运行有一个主代理和至少一个子代理。要启用子代理，必须先安装主代理。请参见第 153 页上的 “安装 SNMP 主代理”。

设置 SNMP 的过程根据不同的系统而不同。下表概述了在不同情况下所要执行的过程。本章后面将对实际过程进行详细介绍。

如果服务器满足以下条件……	……请执行以下过程。（后续各节将详细讨论）。
当前没有运行本地代理	1. 启动主代理。 2. 为系统上安装的每个服务器启用子代理。
• 本地代理当前正在运行。 • 无 SMUX • 不需要继续使用本地代理	1. 为管理服务器安装主代理时，停止本地代理。 2. 启动主代理。 3. 为每个服务器实例配置 SNMP 子代理。
• 本地代理当前正在运行。 • 无 SMUX • 需要继续使用本地代理	1. 安装 SNMP 代理的代理程序。 2. 启动 SNMP 代理的代理程序。 3. 使用主代理端口号以外的端口号重新启动本地代理。 4. 启动主代理。 5. 为系统上安装的每个服务器启用子代理。

开始前，应当验证两件事情：

- 您的系统是否已经运行了 SNMP 代理 （操作系统的本地代理）。
- 如果是，该本地 SNMP 代理是否支持 SMUX 通信。

有关如何验证这些信息的说明，请参见您的系统文档。

注意 在更改了管理服务器中的 SNMP 设置、安装了新的服务器或删除了现有服务器后，您必须执行以下步骤：

- (Windows 2000) 重新启动 Windows SNMP 服务或重新引导计算机。
- (UNIX) 使用管理服务器重新启动 SNMP 主代理和 SNMP 子代理。

本节包括以下主题：

- 使用 SNMP 代理的代理程序 (UNIX/Linux)
- 安装 SNMP 主代理

使用 SNMP 代理的代理程序 (UNIX/Linux)

如果已经运行有一个本地代理，并且希望将它与一个 Sun ONE Application Server 主代理一起并行使用，则需要使用一个 SNMP 代理的代理程序。在启动之前，请确保停止本地主代理。（有关详细信息，请参见您的系统文档。）

注意 要使用代理程序，需要安装并启动它还必须使用 Sun ONE Application Server 主代理运行端口以外的端口重新启动本地 SNMP 主代理。

本节包括以下主题：

- 安装 SNMP 代理的代理程序
- 启动 SNMP 代理的代理程序
- 重新启动本地 SNMP 守护程序

安装 SNMP 代理的代理程序

如果某个 SNMP 代理正在系统上运行，并且希望继续使用本地 SNMP 守护程序，请执行以下各节中的步骤：

1. 安装 SNMP 主代理。请参见第 153 页上的 “安装 SNMP 主代理”。
2. 安装并启动 SNMP 代理的代理程序，然后重新启动本地 SNMP 守护程序。请参见第 151 页上的 “使用 SNMP 代理的代理程序 (UNIX/Linux)”。

3. 启动 SNMP 主代理。请参见第 156 页上的 “启用和启动 SNMP 主代理”。

4. 启用子代理。请参见第 162 页上的 “启用子代理”。

要安装 SNMP 代理的代理程序，请编辑位于服务器根目录下的 *install_dir*/lib/snmp/sagt 中的 CONFIG 文件 （您可以为此文件指定其它名称），以使其包含 SNMP 守护程序的监听端口。SNMP 代理的代理程序还需要包括它要转发的 MIB 树和陷阱。

下面是一个 CONFIG 文件示例：

```
AGENT AT PORT 1161 WITH COMMUNITY public
SUBTREES 1.3.6.1.2.1.1,
        1.3.6.1.2.1.2,
        1.3.6.1.2.1.3,
        1.3.6.1.2.1.4,
        1.3.6.1.2.1.5,
        1.3.6.1.2.1.6,
        1.3.6.1.2.1.7,
        1.3.6.1.2.1.8
FORWARD ALL TRAPS;
```

启动 SNMP 代理的代理程序

要启动 SNMP 代理的代理程序，请在命令提示符下输入以下内容：

```
# sagt -c CONFIG&
```

重新启动本地 SNMP 守护程序

启动 SNMP 代理的代理程序后，在 CONFIG 文件中指定的端口重新启动本地 SNMP 守护程序。

要重新启动本地 SNMP 守护程序，请在命令提示符下输入以下内容：

```
# snmpd -P port_number
```

其中 *port_number* 是在 CONFIG 文件中指定的端口号。例如，在 Solaris 平台上，使用前面提到的 CONFIG 文件示例中的端口，您需要输入：

```
# snmpd -P 1161
```

安装 SNMP 主代理

注意 您不能使用管理界面安装和启动 SNMP 主代理，除非服务器是作为 root 运行的。

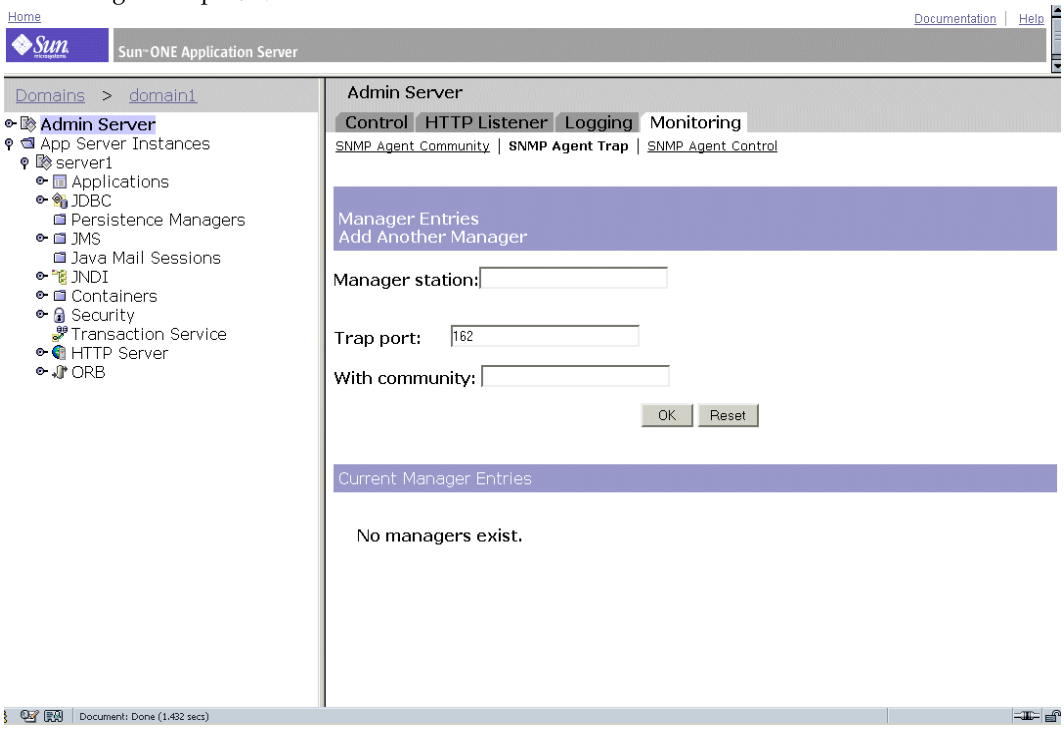
安装 SNMP 主代理的步骤：

1. 以超级用户身份登录。
2. 检查端口 161 上是否运行有 SNMP 守护程序 (snmpd)。

   如果没有运行 SNMP 守护程序，请转到第 4 步。

   如果运行有 SNMP 守护程序，请确保知道如何重新启动它以及它支持哪些 MIB 树。
3. 如果运行有 SNMP 守护程序，请结束它的进程。
4. 在管理界面的左侧窗格中，选择 “Admin Server”节点。
5. 选择 “Monitoring”选项卡以显示 “SNMP Agent Trap”页面，如下图所示。

“SNMP Agent Trap”页面

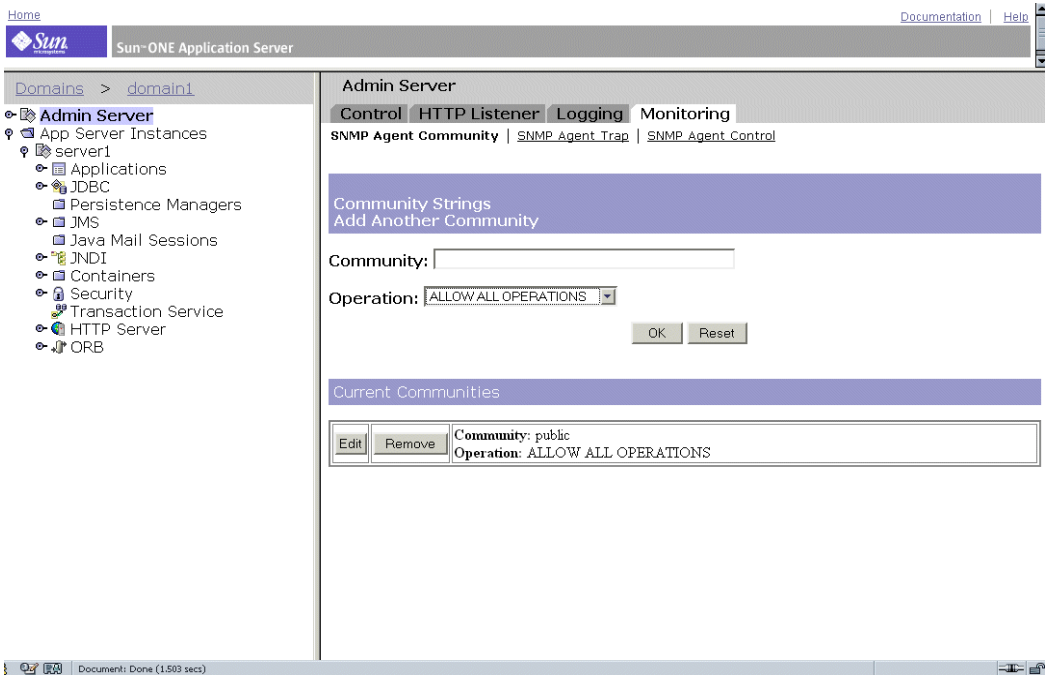


此页面显示了管理器条目信息。

6. 键入正在运行网络管理软件的系统的名称。
7. 键入网络管理系统用来监听陷阱的陷阱端口号。（常用的端口是 162。）有关陷阱的详细信息，请参见第 149 页上的 “SNMP 陷阱目标”。
8. 键入要在陷阱中使用的社区字符串。有关社区字符串的详细信息，请参见第 149 页上的 “SNMP 代理社区”。
9. 单击 “OK”。
10. 在 “Monitoring”选项卡上，单击 “SNMP Agent Community”链接。

    将显示社区字符串信息，如下图所示。

“SNMP Agent Community”页面



11. 键入主代理的社区字符串。
12. 选择社区的操作级别。

    建立社区后，可以使用此页面上“Current Communities”标题下的按钮编辑其设置或删除它。
13. 单击“OK”。
14. 访问左侧窗格中“App Server Instances”下您的服务器实例，然后单击“Apply Changes”。

启用和启动 SNMP 主代理

主代理操作是在名为 CONFIG 的代理配置文件中定义的，您可以手动编辑此文件。要启用 SNMP 子代理，必须先安装 SNMP 主代理。

注意 如果在重新启动主代理时出现类似于 System Error:Could not bind to port 的绑定错误，请使用 ps -ef | grep snmp 检查 magt 是否在运行。如果正在运行，请使用 kill -9 *pid* 命令结束该进程。然后， SNMP 的 CGI 将重新开始工作。

本节包括以下主题：

- 在其它端口上启动主代理
- 手动配置 SNMP 主代理
- 编辑主代理的 CONFIG 文件
- 定义 sysContact 和 sysLocation 变量
- 配置 SNMP 主代理
- 启动 SNMP 主代理
- 启用子代理

在其它端口上启动主代理

管理界面只能在 161 端口上启动 SNMP 主代理。但是，您可以使用以下步骤在其它端口上手动启动主代理：

1. 编辑 *install_dir*/lib/snmp/magt/CONFIG 以指定所希望的端口。
2. 运行以下启动脚本：

   cd *instance_root*/admin-server ./start -shell install_dir/lib/snmp/magt/magt
   install_dir/lib/snmp/magt/CONFIG
   install_dir/lib/snmp/magt/INIT

然后，主代理将在所希望的端口上启动。但是，管理界面能够检测出主代理正在运行。

手动配置 SNMP 主代理

手动配置 SNMP 主代理的步骤：

1. 以超级用户身份登录。
2. 检查端口 161 上是否运行有 SNMP 守护程序 (snmpd)。

   如果运行有 SNMP 守护程序，请确保知道如何重新启动它以及它支持哪些 MIB 树。然后结束它的进程。
3. 编辑位于服务器根目录下 lib/snmp/magt 中的 CONFIG 文件。
4. （可选）在 CONFIG 文件中定义 sysContact 和 sysLocation 变量，如第 158 页上的“定义 sysContact 和 sysLocation 变量”中所述。

编辑主代理的 CONFIG 文件

CONFIG 文件定义了将与主代理一起工作的社区和管理器。管理器的值应当是有效的系统名或 IP 地址。

下面是一个基本 CONFIG 文件的示例：

```
COMMUNITY        public
            ALLOW ALL OPERATIONS

MANAGER          manager_station_name
            SEND ALL TRAPS TO PORT 162
             WITH COMMUNITY public
```

定义 sysContact 和 sysLocation 变量

您可以编辑 CONFIG 文件，为指定了 sysContact 和 sysLocation MIB-II 变量的 sysContact 和 sysLocation 添加初始值 。此示例中 sysContact 和 sysLocation 的字符串放在了引号内。任何包含空格、换行符、制表符等的字符串都必须放在引号内。您也可以用十六进制记数法来指定值。

下面是一个 CONFIG 文件示例，其中定义了 sysContact 和 sysLocation 变量：

```
COMMUNITY        public
                 ALLOW ALL OPERATIONS

MANAGER          nms2
                 SEND ALL TRAPS TO PORT 162
                 WITH COMMUNITY public

INITIAL          sysLocation "Server room
901 San Antonio Road
Palo Alto CA 94303
USA"

INITIAL          sysContact "John Doe
email:jdoe@sun.com"
```

配置 SNMP 子代理

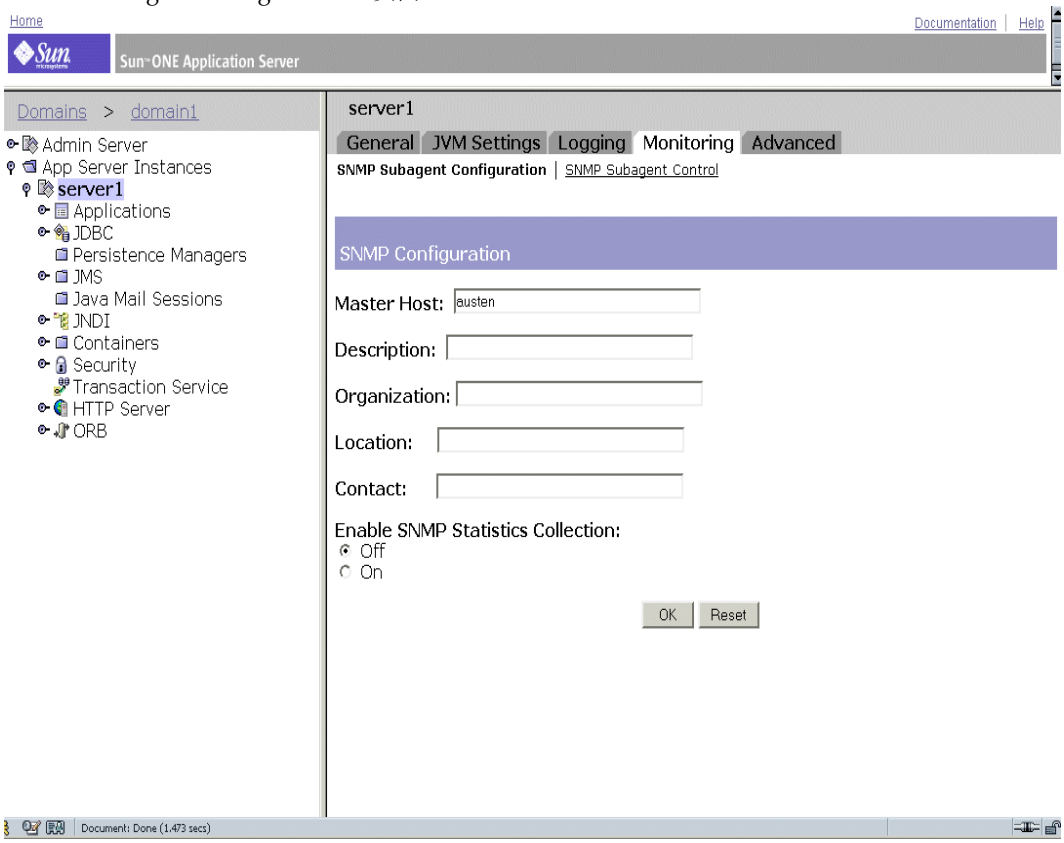
要配置 SNMP 子代理，请执行以下步骤：

1. 在左侧窗格中，选择 “Admin Server”下的服务器实例节点。
2. 在右侧窗格中，选择 “Monitoring”选项卡。

3. 选择 “SNMP Subagent Configuration”链接。

   将显示以下页面：

“SNMP Subagent Configuration”页面



4. （只适用于 UNIX）在 “Master Host”字段中，输入服务器的名称和域。
5. 输入服务器的说明，包括操作系统信息。
6. 输入负责该服务器的组织。
7. 输入服务器实例的位置。
8. 在 “Contact”字段中，输入负责该服务器的人员的姓名和联系信息。
9. 选择 “On”，启用 SNMP 统计数据收集。
10. 单击 “OK”。

11. 访问左侧窗格中 “App Server Instances”下您的服务器实例，然后单击 “Apply Changes”。

启动 SNMP 主代理

安装 SNMP 主代理后，您可以手动启动它或通过管理界面使用管理服务器进行启动。

手动启动 SNMP 主代理

要手动启动主代理，请在命令提示符下输入以下内容：

```
# magt CONFIG INIT&
```

INIT 文件是包含 MIB-II 系统组信息 （包括系统位置和联系信息）的非易失性文件。如果 INIT 不存在，首次启动主代理时将创建它。

注意 如果 CONFIG 文件中的管理器名称无效，将导致主代理启动失败。

要在非标准端口上启动主代理，请使用以下两种方法之一：

方法 1：在 CONFIG 文件中，为主代理用来监听来自管理器的 SNMP 请求的每个接口指定传输映射。传输映射允许主代理接受标准端口和非标准端口上的连接。主代理还可以在非标准端口上接受 SNMP 通信。并行 SNMP 的最大数目受限于目标系统对每个进程的打开的套接字或文件描述符数目的限制。下面是一个传输映射条目示例：

```
TRANSPORT       extraordinary   SNMP
                OVER UDP SOCKET
                AT PORT 11161
```

手动编辑 CONFIG 文件后，您应当在命令提示符下键入以下内容以便手动启动主代理：

```
# magt CONFIG INIT&
```

方法 2：编辑 /etc/services 文件，以允许主代理接受标准端口和非标准端口上的连接。

使用管理服务器启动 SNMP 主代理

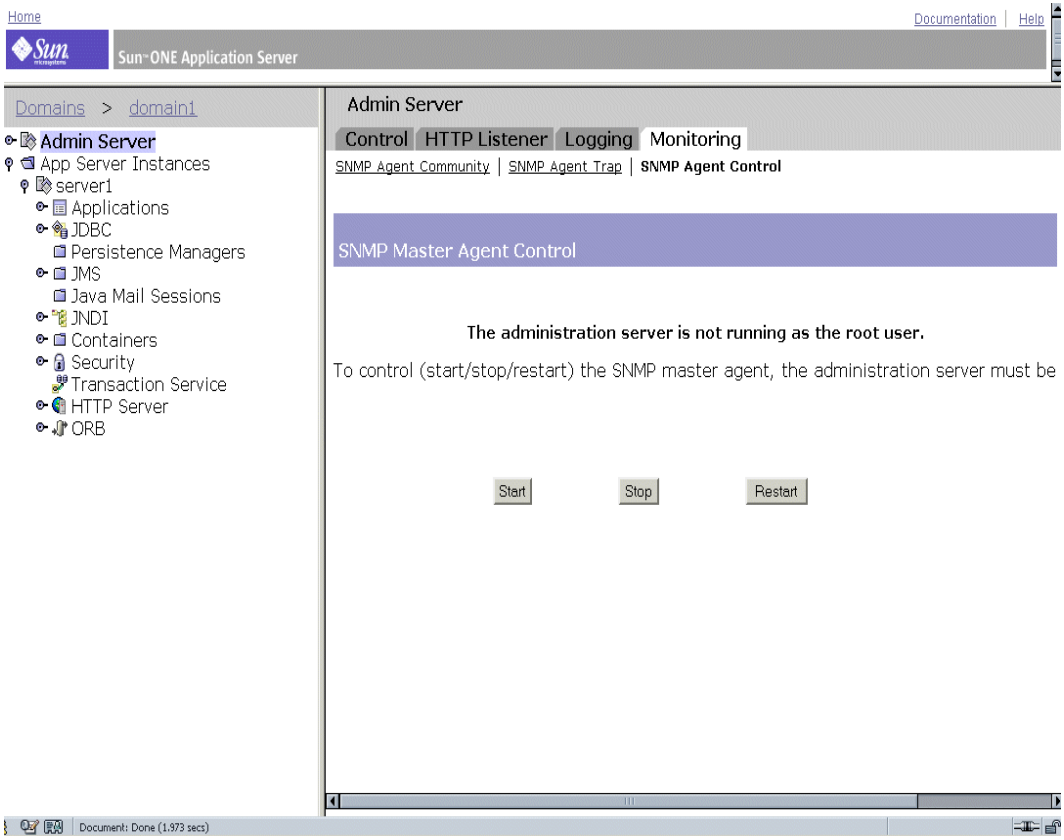
要使用管理服务器启动 SNMP 主代理，请执行以下步骤：

注意 您必须以超级用户身份登录 Sun ONE Application Server 才能启动 SNMP 主代理。

1. 登录到管理服务器。
2. 在左侧窗格中选择 “Admin Server”节点，然后选择 “Monitoring”选项卡。
3. 在右侧窗格中靠近顶部的位置，选择 “SNMP Agent Control”链接。

   将显示以下页面。

“SNMP Agent Control”页面



4. 单击 “Start”。

   您还可以在 “SNMP Agent Control”页面上停止和重新启动 SNMP 主代理。

启用子代理

安装了管理服务器附带的主代理后，您必须在尝试启动它之前为您的服务器实例启用子代理。有关安装主代理的信息，请参见第 153 页上的 “安装 SNMP 主代理”。

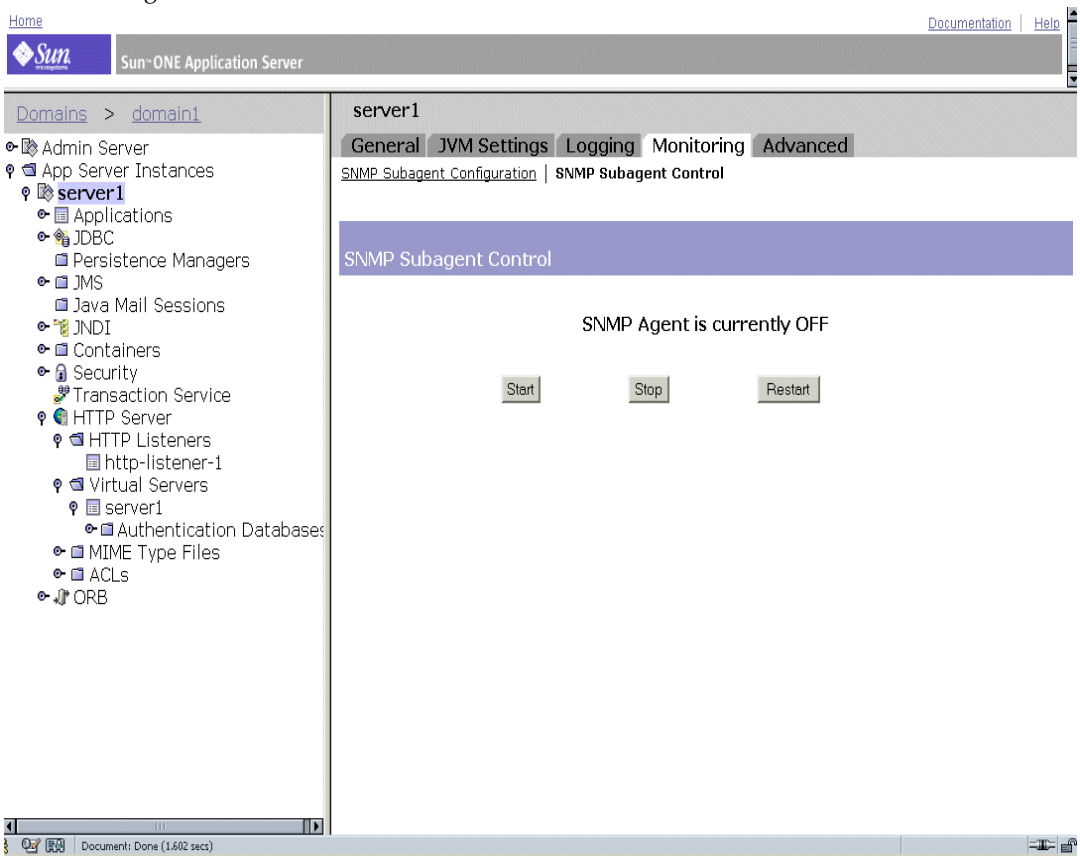
在 UNIX/Linux 平台上，您可以使用子代理停止 SNMP 功能。您必须先停止子代理，然后再停止主代理。如果先停止主代理，可能无法停止子代理。如果发生这种情况，请重新启动主代理，停止子代理，然后停止主代理。

启用 SNMP 子代理的步骤：

1. 在左侧窗格中，展开 “App Server Instances”节点。
2. 选择服务器实例，然后单击 “Monitoring”选项卡。

3. 选择 “SNMP Subagent Control”选项以显示如下图所示的页面。

“SNMP Subagent Control”页面



在此页面中，您可以启动、停止或重新启动 SNMP 子代理。子代理的状态将显示在控制按钮的上面。

在 Windows 平台上，Windows SNMP 服务用于监视 Sun ONE Application Server；可以使用 “控制面板” - “管理工具”中的 “服务”选项中对其进行控制。

注意 对 SNMP 配置进行任何更改后，必须单击 “OK”，然后从 “SNMP Subagent Control”页面中重新启动 SNMP 子代理。

第 7 章

配置 Web 服务器插件

本章介绍 Sun ONE Application Server 如何处理超文本传输协议 (HTTP) 请求，以及如何配置并与 Sun ONE Application Server 一起使用 Web 服务器插件。此外，还介绍了如何配置并与 Microsoft IIS 和 Apache Web 服务器一起使用 Web 服务器插件。

本章包括以下主题：

- 关于 Web 服务器插件
- 处理客户机请求
- Web 服务器插件配置
- Web 服务器插件 SAF 参考
- 使用 Web 服务器插件
- 配置 Microsoft IIS 以使用 Web 服务器插件
- 配置 Web 服务器插件以用于 Apache 服务器

关于 Web 服务器插件

Web 服务器插件是 HTTP 反向代理插件，用于指示 Sun ONE Web Server 或 Sun ONE Application Server 将某些 HTTP 请求转发至其它服务器。例如，您可以对连接至因特网的 Web 服务器进行配置，以便将特定 Web 应用程序的请求转发至位于公司防火墙之后的应用程序服务器。

在 Sun ONE Application Server 中，Web 服务器插件允许在不同的服务器实例之间转发 HTTP (Web) 请求。

Web 服务器插件执行以下功能：

- 重新使用代理服务器连接（如果可能的话）。这样一来，不必打开新连接，便可处理传入的请求。
- Web 服务器插件在开始接收请求和响应时便以数据流形式对请求和响应进行处理。也就是说，此插件不会等到完全收集请求或响应后，才将其转发至远程服务器。
- Web 服务器插件与同一远程服务器保持多个出站 HTTP 连接（如果合适的话）。为 Web 服务器插件转发的请求建立的连接称作出站 HTTP 连接。

要了解 Web 服务器插件的工作原理，必须了解 HTTP 请求的基本原理，尤其是 Sun ONE Application Server 用来处理 HTTP 请求的方法。

处理客户机请求

Sun ONE Application Server 是可以直接接受并响应 HTTP 请求的应用程序服务器。本节将介绍 HTTP 的基本原理以及 Sun ONE Application Server 处理请求的方式。本节包括以下主题：

- HTTP 基本原理
- 请求处理进程的执行步骤

HTTP 基本原理

简而言之， HTTP/1.1 协议的工作原理如下：

- 客户机 （通常为浏览器）打开服务器连接并发送请求。
- 服务器处理请求、生成响应并关闭连接 （如果找到 Connection:Close 消息头）。

请求由表示方法 （例如 GET 或 POST）的行、表示所请求的资源的通用资源标识符 (URI) 以及 HTTP 协议版本组成，各部分以空格分隔。

请求后面通常包含一些消息头、表示消息头结尾的空白行，有时还会包含正文数据。消息头可提供各种有关请求或客户机正文数据的信息。消息头通常只针对 POST 和 PUT 方法发送。

以下显示的示例请求由浏览器发送，用于请求服务器 foo.com 返回 */index.html* 中的资源。此示例使用了方法 GET，因此未发送任何正文数据 （此请求用于获取而不是发送某些数据）。

```
GET /index.html HTTP/1.0
User-agent:Mozilla
Accept:text/html, text/plain, image/jpeg, image/gif, */*
Host:foo.com
```

此服务器接收请求并对其进行处理。尽管它可以同步处理多个请求，但仍单独处理每个请求。每个请求被分为一系列步骤，这些步骤合起来构成请求处理进程。

此服务器生成响应，其中包含 HTTP 协议版本、 HTTP 状态代码和原因词组，三者之间以空格分隔。响应后面通常包含一些消息头。消息头结尾以空白行表示。之后是响应的正文数据。典型的 HTTP 响应可能如下所示：

```
HTTP/1.0 200 OK
Server:Standard/7.0
Content-type:text/html
Content-length: 83

<HTML>
<HEAD><TITLE>Hello World</Title></HEAD>
<BODY>Hello World</BODY>
</HTML>
```

状态代码和原因词组通知客户机服务器处理请求的方式。通常情况下，返回状态代码 200，表示已成功处理请求，且正文数据包含请求的项目。其它结果代码表示重定向到其它服务器或浏览器的高速缓存，或者返回各种类型的 HTTP 错误，例如，“未找到 404”。

请求处理进程的执行步骤

Sun ONE Application Server 在首次启动时将执行某些初始化任务，然后等待客户机（例如浏览器）发出的 HTTP 请求。它在接收到请求时，将首先选择虚拟服务器。

选择虚拟服务器后，该虚拟服务器的 obj.conf 文件将指定如何通过以下步骤处理请求：

1. **AuthTrans**（授权转换）

   验证请求中发送的任何授权信息（例如，名称和密码）。

2. **NameTrans**（名称转换）

   将逻辑 URI 转换为本地文件系统路径。

3. **PathCheck**（路径检查）

   检查本地文件系统路径是否有效，并检查请求者对此文件系统上的请求资源是否具有访问权限。

4. **ObjectType**（对象键入）

   确定请求资源的 MIME 类型（多用途 Internet 邮件编码），例如， text/html、image/gif 等。

5. **Service**（生成响应）

   生成响应并将其返回客户机。

6. **AddLog**（添加日志条目）

   向日志文件中添加条目。

7. **Error**（服务）

   只要前面的步骤中发生错误，就会执行此步骤。如果发生错误，服务器将记录错误信息并中止此进程。

Web 服务器插件配置

Web 服务器插件的配置和行为由一组配置文件确定。 Sun ONE Application Server 在每次处理客户机请求时将查看这些文件中定义的配置。这些配置文件名为 obj.conf 和 init.conf。 obj.conf 文件以虚拟服务器的名称作为前缀，例如 server1-obj.conf。有关详细信息，请参见第 337 页上的 “obj.conf 文件”。

Sun ONE Application Server 的每个实例都有各自的 init.conf 文件，服务器在启动时将引用此文件。

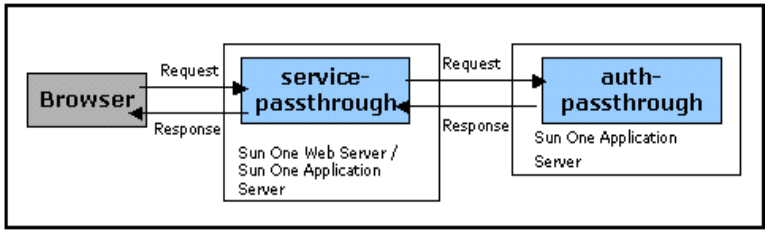
正如前面的主题所介绍的， obj.conf 配置文件包含一系列指令，用于通知 Sun ONE Application Server 在客户机请求和响应进程的每个阶段要执行的操作。每个指令都调用服务器应用程序函数 (SAF)。

obj.conf 文件对 Sun ONE Application Server 的运行至关重要。通过管理界面对服务器进行更改时，系统将自动更新 obj.conf。

init.conf 配置文件用于设置在初始化过程中配置服务器的变量的值。服务器将在启动过程中执行此文件中指定的配置参数。有关详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

下图显示 Web 浏览器、前端 Web 服务器、后端应用程序服务器以及 Web 服务器插件的 service-passthrough 和 auth-passthrough SAF 之间的关系：

Web 浏览器、 Web 服务器、应用程序服务器以及 Web 服务器插件 SAF 之间的关系



Web 服务器插件 SAF 参考

本节介绍以下服务器应用程序函数 (SAF) 的功能和行为：

- init-passthrough
- auth-passthrough
- service-passthrough
- check-passthrough

init-passthrough

init-passthrough 函数用于初始化 Web 服务器插件。必须调用此函数，才能使用 Web 服务器插件。

示例：

```
Init fn="load-modules" shlib="c:/plugins/passthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough"
NativeThread="no"

Init fn="init-passthrough"
```

auth-passthrough

auth-passthrough SAF 可在 AuthTrans-class 指令中使用。

auth-passthrough 函数检查传入的 HTTP (web) 请求中是否存在由中间服务器上运行的 service-passthrough 函数编码的客户机信息。客户机信息包括：

- 请求源自的 IP 地址
- 初始客户机使用的 SSL 密钥大小
- 初始客户机提供的 SSL 客户证书

当 auth-passthrough 检测到编码的客户机信息时，它将把请求作为初始客户机的直接请求，而不是作为由运行 service-passthrough 的中间服务器转发的请求处理。

这在两层部署方案中很有用；

- Sun ONE Application Server 实例由公司防火墙之后的第二个防火墙隐藏。
- 不允许与 S1AS 实例直接建立客户机连接。

在此类网络体系结构中，客户机始终与运行代理插件的前端 Web 服务器连接。此 Web 服务器用于将请求转发至 Sun ONE Application Server。这表示 Sun ONE Application Server 只能从代理主机 （本例中为 Web 服务器）接收请求，而不能直接从客户机主机接收请求。这意味着，如果应用程序 （部署在两个防火墙之后的 Sun ONE Application Server 实例中）查询客户机信息 （例如，客户机的 IP 地址），则将获得代理主机 IP （因为它是中继请求的实际初始主机）。可以使用 auth-passthrough SAF 修改此行为，以显示远程 （代理）客户机信息。

由于 auth-passthrough 可能会覆盖用于验证的信息 （例如，请求源自的 IP 地址），因此必须只允许将信任的客户机或服务器连接至运行 auth-passthrough 的服务器。作为一种预防措施，建议您只使用公司防火墙之后的服务器来运行 auth-passthrough。可通过 Internet 访问的服务器不应运行 auth-passthrough SAF。只有在需要有关初始客户机的相关信息时，才能使用 auth-passthrough SAF。

请注意，在以上所述的方案中， SSL 客户机验证只能在 Web 服务器上打开，并且始终要在应用程序服务器上关闭，这样配置才能正常工作。

命令示例：

AuthTrans fn="auth-passthrough"

service-passthrough

service-passthrough SAF 可在 Service-class 指令中使用。

service-passthrough SAF 将请求从一个服务器转发至另一个服务器进行处理。可以将 service-passthrough SAF 配置为使用 SSL 连接或非 SSL （HTTPS 或 HTTP）连接与远程服务器建立连接，并且不受用于接收初始请求的连接类型的影响。
service-passthrough SAF 用于对有关初始客户机的信息进行编码，此信息可以由远程服务器上运行的 auth-passthrough 函数进行解码。

service-passthrough 指令通常与 obj.conf 配置文件中的其它指令组合使用，如下所示：

<Object name="*passthrough*">

ObjectType fn="*force-type*" type="*magnus-internal/passthrough*"

Error reason="*Bad Gateway*" fn="send-error" uri="$docroot/*badgateway*.html"

</Object>

<Object name="default">

....

NameTrans fn="*assign-name*" from="(/webapp1|/*webapp1*/*)" name="*passthrough*"

...

</Object>

如果后端应用程序服务器已关闭，则将向用户显示本地 HTML 文件 badgateway.html。如果运行 service-passthrough SAF 的服务器需要提供它可以访问的文件，并只将被拒绝的请求转发到后端应用程序服务器，则 ObjectType 行将改为：

ObjectType fn="check-passthrough" type="magnus-internal/passthrough"

check-passthrough

check-passthrough SAF 可在 ObjectType-class 指令中使用。

check-passthrough 函数检查本地服务器中是否存在请求的资源 （例如， HTML 文档或 GIF 图像）。如果不存在， check-passthrough SAF 将设置类型，指示应将请求传递到其它服务器以便由 service-passthrough SAF 进行处理。

参数：

type - （可选）请求资源不存在时设置的类型。默认类型为 "magnus-internal/passthrough"。

示例

ObjectType fn="check-passthrough"

使用 Web 服务器插件

要向 Sun ONE Web Server 或 Sun ONE Application Server 实例中添加插件，请执行以下步骤：

1. 确保 libpassthrough.so (UNIX) 位于 *install_dir*/plugins/passthrough/bin 目录中，且 passthrough.dll 文件 (Windows) 位于 *install_dir*/bin 目录中。向 Sun ONE Web Server 6.0 中添加插件时，必须从 Sun ONE Application Server 7 安装中复制插件。
2. 在 *install_dir*/config/init.conf （对于 Sun ONE Application Server 7）或 *install_dir*/config/magnus.conf （对于 Sun ONE Web Server 6.0）配置文件中添加以下行。每行均以 Init 开头，且所有内容均在一行上。

Windows：

```
Init fn="load-modules" shlib="c:/install_dir/bin/passthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough, service-passthrough"
NativeThread="no"

Init fn="init-passthrough"
```

UNIX：

```
Init fn="load-modules"
shlib="install_dir/plugins/passthrough/bin/libpassthrough.so"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-passthrough"
NativeThread="no"

Init fn="init-passthrough"
```

3. 通过在 config/*appserver-server-instance*-obj.conf 配置文件的顶部添加以下行，将 passthrough <Object> 添加到 obj.conf 中。在 Sun ONE Application Server 上，必须在 config/<*appserver-server-instance*>-obj.conf 文件中执行此步骤。

```
<Object name="passthrough">
ObjectType fn="force-type" type="magnus-internal/passthrough"
Service type="magnus-internal/passthrough" fn="service-passthrough"
servers="server"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</Object>
```

其中， server 是以下格式的 URL：

http://*servername*:port

4. 通过在 *install_dir*/config/obj.conf 配置文件中的默认对象的顶部添加以下行，来配置要转发的 URI：

```
NameTrans fn="assign-name" from="(/uri|/uri/*)" name="passthrough"
```

uri 是远程服务器上部署的 Web 应用程序的上下文根， passthrough 与 obj.conf 中的 <Object> 的名称 （第 3 步 中给出）相对应。

例如：

```
<Object name="default">
...
NameTrans fn="assign-name" from="(/webapp1|/webapp1/*)" name="passthrough"
...
</Object>
```

5. 重新启动此服务器。

注意 对于 Solaris 和 Linux，插件库的名称为 libpassthrough.so。对于 Windows，插件库的名称为 passthrough.dll。

配置 Microsoft IIS 以使用 Web 服务器插件

配置 Microsoft Internet Information 服务以使用 Web 服务器插件涉及到对 Web 服务器插件和 Microsoft IIS 进行配置，使两者可以相互使用。

此外，您还可以配置服务器池，以处理在不同服务器上运行的多个应用程序。

本节包括以下主题：

- 为 IIS 配置 Web 服务器插件
- 配置 IIS 以使用 Web 服务器插件
- 配置多个服务器池
- 样例 sun-passthrough.properties 文件

为 IIS 配置 Web 服务器插件

要为 IIS 配置 Web 服务器插件，请执行以下任务：

1. 在 C:\ 命令行提示处键入以下命令，在 IIS wwwroot 目录下为 Web 服务器插件创建一个目录：

   md \Inetpub\wwwroot\sun-passthrough

2. 将插件文件复制到 C:\Inetpub\wwwroot\sun-passthrough 目录。

3. 使用文本编辑器将安装了 Sun ONE Application Server 的计算机的 URL 添加到 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 文件中。

   您需要通过文本编辑器添加以下信息：

   server=http://*appservername*:*port*

   其中，*appservername* 是安装了 Sun ONE Application Server 的计算机的主机名或 IP 地址，port 是 Sun ONE Application Server 侦听的端口号（此值通常设置为 80）。

4. 在 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 文件中列出 Sun ONE Application Server 要处理的上下文根。

   这些上下文根应该与 Sun ONE Application Server 中部署的应用程序的上下文根相对应。对这些上下文根的请求将由 Sun ONE Application Server 处理，而其它请求则由 IIS Web 服务器处理。用于将请求传递给 Web 应用程序的命令行如下：

   passthrough=/*webapplication*

   其中，*/webapplication* 是 Web 服务器的上下文根。要将所有请求传递给 Sun ONE Application Server，请添加以下行：

   passthrough=/

至此，您已完成了对 Microsoft IIS 根目录下的 Web 服务器插件的配置。要完成整个过程，还需要配置 Microsoft IIS 以使用此 Web 服务器插件。

配置 IIS 以使用 Web 服务器插件

要配置 IIS 以使用 Web 服务器插件，需要打开 Windows Internet 服务管理器。Internet 服务管理器位于 “控制面板”文件夹的 “管理工具”文件夹中。

打开 Internet 服务管理器，并执行以下任务：

1. 选择要启用此插件的 Web 站点。此 Web 站点通常命名为 “默认的 Web 站点”。
2. 在此 Web 站点上单击鼠标右键，并选择 “属性”以打开 “属性”记事本。
3. 打开 “ISAPI 过滤器”选项卡，单击 “添加”按钮，然后执行以下步骤添加新的 ISAPI 过滤器：
   a. 在 “过滤器名”字段中，输入 Sun ONE Application Server
   b. 在 “可执行文件”字段中，键入 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll
   c. 单击 “确定”关闭 “属性”记事本。
4. 现在，您需要创建并配置一个新的虚拟目录。执行以下步骤，创建并配置一个新的虚拟目录：
   a. 在默认的 Web 站点上单击鼠标右键，选择 “新建”，然后选择 “虚拟目录”。将打开 “虚拟目录创建向导”。
   b. 在 “别名”字段中，键入 sun-passthrough。
   c. 在 “目录”字段中，键入 C:\Inetpub\wwwroot\sun-passthrough

d. 确保选中 “执行权限”复选框，并使所有其它权限复选框保持未选中状态。

e. 单击 “完成”。

5. 您需要停止并启动 Web 服务器，才能使新的设置生效。要停止 Web 服务器，请在此 Web 站点上单击鼠标右键，然后选择 “停止”。要启动 Web 服务器，请在此 Web 站点上单击鼠标右键，然后选择 “启动”。

接下来，在 Web 浏览器中键入以下地址，以访问此 Web 应用程序的上下文根：

http://*webservername*/*webapplication*

其中， *webservername* 是 Web 服务器的主机名或 IP 地址， */webapplication* 是在 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 文件中列出的上下文根，用于验证 Web 服务器、 Web 服务器插件和 Sun ONE Application Server 是否正常运行。

配置多个服务器池

通过在 sun-passthrough.properties 文件中配置服务器池，可以将 Web 应用程序分布到多个应用程序服务器中 （即，在一组服务器上运行一些应用程序，而在另一组服务器上运行其它应用程序）。为每个服务器池选择一个唯一的名称 （由字母和数字组成）。按照第 174 页上的 “配置 Microsoft IIS 以使用 Web 服务器插件”一节中的介绍完成为 Microsoft IIS 安装和配置 Web 服务器插件的步骤后，应编辑 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 文件并在相关的服务器和 passthrough 属性行之前加上为服务器池选择的唯一名称作为前缀。在服务器池名称后加上句点 (.)。

例如， sun-passthrough.properties 文件中的以下行定义了两个服务器池。第一个服务器池包含 server-a 以及上下文根 /app1 的服务请求。第二个服务器池包含 server-b 以及 /app2 和 /app3 上下文根的服务请求。

server=http://server-a

passthrough=/app1

serverpool2.server=http://server-b

serverpool2.passthrough=/app2

serverpool2.passthrough=/app

样例 sun-passthrough.properties 文件

```
# Sun ONE Application Server web server plugin for IIS
#
# This file is used to configure the Sun ONE Application Server web server
# plugin for IIS.Lines beginning with a '#' are ignored.
# server
#
# The server property specifies the URL of an application server.If multiple
# server properties are given, the plugin will distribute load across the
# specified application servers.
#
server=http://localhost:8080
# passthrough
#
# The passthrough property specifies the context root (virtual directory) of a
# web application.Requests for the given context root will be passed to the
# application server for processing.If 'passthrough=/' is specified, all
# requests will be passed to the application server for processing.
#
# passthrough properties should be ordered from most to least specific.For
# example, 'passthrough=/apps/app1' should appear before 'passthrough=/apps'.
#
# Multiple passthrough properties are allowed.
#
#passthrough=/webapp
#passthrough=/servlets
#passthrough=*.jsp
passthrough=/
# prefix
#
```

```
# The prefix property specifies the IIS virtual directory that contains the
# plugin DLL, sun-passthrough.dll.
#
prefix=/sun-passthrough
# error-url
#
# The error-url property specifies the URL of a page to redirect the client to
# when the application server is unavailable.
#
#error-url=/badgateway.htm
# It is possible to configure multiple server pools by prefixing the server
# and passthrough property names with a pool name followed by a period ('.').
# Pool names can be any sequence of letters and numbers.
#
# For example, the following properties define two server pools.One server
# pool will service the web applications at '/app1' and the other will service
# the web applications at '/app2' and '/app3':
#
#serverpool1.server=http://server-a
#serverpool1.passthrough=/app1
#
#serverpool2.server=http://server-b
#serverpool2.passthrough=/app2
#serverpool2.passthrough=/app3
```

配置 Web 服务器插件以用于 Apache 服务器

本节介绍如何编译 Apache 源代码，以及如何配置 Apache Web 服务器的安装。

1. 从 www.apache.org 中下载最新的 Apache 源代码分发

   将源代码分发解压缩。源代码分发以压缩归档文件的形式提供。如果要安装 Apache 1.3.22，则源代码分发归档文件将为 apache_1.3.22.tar.gz。

2. 使用以下命令解压缩归档文件：

   $ tar -zxvf *apache_1.3.26*.tar.gz

   此命令将在当前的工作目录中创建名为 apache_1.3.26 的目录。

3. 现在，您需要对环境进行配置以便对 Apache 源代码进行编译。源代码分发附带了一个名为 configure 的脚本，此脚本用于检查环境中是否存在成功编译 Apache 所需的必要支持文件 （例如，消息头、共享库和公用程序）。

   要配置环境，请转至 Apache 源目录并继续执行以下步骤：

   a. 确保在 Solaris 或 Linux 上安装 Apache 时存在以下路径：

      - CC=/usr/dist/share/devpro/5.x-sparc/bin/cc
      - PATH （您将在路径 /usr/ccs/bin 中找到此命令）中应存在 ar。
      - Make file （您将在 /usr/ccs/bin/make 路径中找到此命令）

   b. 运行以下命令：

      $ ./configure –enable-module=proxy --prefix=/*usr/local/apache*

   以上命令中指定的目录为一个变量。您可以指定安装 Apache 的路径。prefix 参数指示要将 Apache 安装到的位置。此命令将在屏幕上输出多个行。此命令实际上是根据系统配置为编译创建 Makefile。如果 configure 中存在错误，您可能会丢失在继续操作之前必须安装的一些消息头文件或公用程序。

4. configure 脚本成功运行后，您可以使用 make 命令对 Apache 进行编译，如下所示：

   $ make

   此命令将在屏幕上输出多个行，表示进程正在编译 Apache 源代码并正在链接 Apache。此进程通常不会出现错误。但如果出现错误，请检查是否正确下载了 Apache 的所有库文件和公用程序。

5. 现在，您需要安装 Apache。 Apache 将自动安装到 /usr/local/apache 目录 （或指定的任何其它目录）中。要安装 Apache，请运行以下命令：

   $ make install

   如果此命令成功执行，则表明系统上已安装了 Apache。您应该在以下目录中看到 Apache 的安装文件：

   /usr/local/apache

   主配置文件 httpd.conf 将安装到 /usr/local/apache/ 目录中。

6. 通过运行以下命令来配置 Apache：

   $ Configure Apache

   Apache 通过文件 httpd.conf 进行配置。此文件包含一些 Apache 指令，用于确定 Apache 服务器的各种操作参数。对于 Apache 的简单安装，需要修改以下几个指令：

   - Server Root “（Server Root 是安装 Apache 的路径。例如：Server Root &/usr/local/apache 或 /space/apache）。
   - Port “5000”（用户选择） ”

现在，便可以对 Apache 进行配置，以执行默认的行为和 Web 服务。如果需要设置其它参数，请参考配置文件。

7. Apache 附带了名为 apachectl 的脚本，此脚本可简化 Apache 的启动、停止和重新启动。运行以下命令启动 Apache

   $ /usr/local/apache/bin/apachectl start

   要停止 Apache，请运行以下命令：

   use /usr/local/apache/bin/apachectl stop

   启动后便可以测试 Apache 的安装。运行 Apache 后，请在 Web 浏览器中键入以下地址：http://localhost/。如果安装成功且 Apache 正在运行，则应该看到一个显示此信息的测试页。

第 8 章

配置 J2EE 容器

Sun ONE Application Server 提供了各种符合 J2EE 1.3 规范的 J2EE 容器。容器为 J2EE 应用程序组件 （例如，企业 Java Bean [EJB] 和消息驱动的 Bean [MDB]）提供运行时支持。 MDB 和 EJB 与其它 J2EE 应用程序从不直接进行交互，它们使用 EJB 容器的协议和方法彼此进行交互或和其它平台服务 （例如， Java 事务服务）进行交互。该容器位于应用程序组件和 J2EE 服务之间，因而，容器可以透明地引入组件部署描述符定义的服务，例如，声明的事务管理、安全性检查、资源加入和状态管理。

Sun One Application Server 中集成了 Web 容器和 EJB 容器。

本章包括以下主题：

- 关于 Web 容器
- 关于 EJB 容器

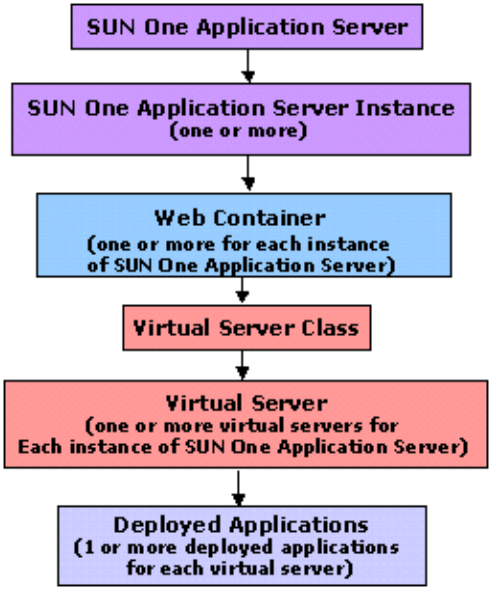
关于 Web 容器

Web 容器是一个用来存储 Web 应用程序的 J2EE 容器。Web 容器通过为开发者提供运行 servlet 和 Java Server Page (JSP) 的环境，扩展了 Web 服务器的功能。Servlet 提供了基于组件且独立于平台的方法，使用该方法可以创建基于 Web 的应用程序而不受到 CGI (Common Gateway Interface) 程序的性能限制。 JSP 技术对 servlet 技术进行了扩展，它用于支持 HTML 和 XML 页面的创建。 Web 容器中包括的 Servlet 或 JSP 具有调用企业 Java Bean (EJB) 容器中的 Bean 方法的能力。Bean 方法可以从本地调用，也可以使用 Object Request Broker (ORB) 进行远程调用。

Web 容器还可以使 Web 应用程序访问使用 JNDI (Java Naming Directory Interface) 定位的本地 EJB。

下图 Sun ONE Application Server 体系结构中的 Web 容器解释了 Web 容器在 Sun ONE Application Server 体系结构中的角色和位置：

Sun ONE Application Server 体系结构中的 Web 容器



本节包括以下主题：

- 了解 Web 容器的角色
- Web 应用程序配置
- Web 应用程序部署
- 单一登录功能
- 记录 Web 容器

了解 Web 容器的角色

Web 容器的主要角色如下：为 Web 应用程序提供运行时环境并为存储在容器中的 Web 应用程序提供服务 （数据库访问、安全性、多线程等等）。一个 Web 应用程序是 servlet、 HTML 页面、类和其它资源的集合，这些资源构成了 Sun ONE Application Server 上的完整应用程序。

下面列出了 Web 应用程序的组成元素：

- Servlet
- JSP 页面
- 工具类
- 静态文档 （html、图形、声音文件等）
- 客户端 Java applet、 Bean 和类
- 将以上所有的元素连接到一起的说明性元信息。

Web 应用程序可以部署到 Sun ONE Application Server 中运行的 Web 容器。

有关如何在 Sun ONE Application Server 上配置和使用 Web 服务器插件的详细信息，请参见第 165 页上的 “配置 Web 服务器插件”。

Web 应用程序配置

您还可以将 Web 容器配置为在虚拟服务器中部署 Web 应用程序。 Web 容器可以配置为包含多个虚拟服务器。每个虚拟服务器可以设置为存储任何数目的 Web 应用程序。 Web 应用程序位于虚拟服务器的上下文之内。有关虚拟服务器的详细信息，请参见第 15 章 “使用虚拟服务器”。

本节包括以下主题：

- 虚拟服务器的属性
- Web 模块属性

虚拟服务器的属性

您可以为虚拟服务器的某些可配置属性指定值。一个虚拟服务器可以有多个 Web 应用程序与之相关联。用户需要*登录* Web 应用程序。

如果单一登录的属性 sso-enabled 在 server.xml 文件中设置为默认值 True，则用户可以登录任何与指定虚拟服务器相关联的 Web 应用程序。同时，该用户的身份将被运行在同一虚拟服务器上所有其它 Web 应用程序认可。如果 sso-enabled 的值设为 False，则该虚拟服务器上的所有 Web 应用程序的单一登录将被禁用。

sso-enabled 属性可以动态配置，并且无需重新启动服务器即可使更改生效。

第 185 页上的 “单一登录功能”一节中介绍了有关单一登录的详细信息。

Web 模块属性

Sun ONE Application Server 特定的部署描述符在名为 sun-web.xml 的文件中进行了指定，该文件可以在给定 Web 应用程序的 WEB-INF 目录中找到。

通常，每个 Web 应用程序都配置有一个 sun-web.xml 文件。但是， Web 容器不需要每个 Web 应用程序都有一个 sun-web.xml 文件。在没有 sun-web.xml 文件的情况下，Web 容器将采用所有的 Sun ONE Application Server 特定属性的默认值。

context-root 属性

该属性定义了安装 Web 应用程序安装的上下文根。如果该属性为空字符串，则这个 Web 应用程序将被指定为虚拟服务器的默认 Web 应用程序。虚拟服务器的默认 Web 应用程序负责处理虚拟服务器中的其它 Web 应用程序不能响应的所有请求。每个虚拟服务器都有一个默认 Web 应用程序。

对于默认的 Web 应用程序，这个字段的值应该为空字符串 ""。

location 属性

该属性的输入值应该是一个有效目录路径，它指示了默认的 Web 应用程序的位置。在安装过程中，默认 Web 应用程序的位置被设置到 modules/*default-web-app*/ 目录。

location 属性为必需的属性，其值为提取 WAR (Web ARchive) 文件内容的目录的全限定路径或相对路径。如果指定的路径是相对路径，则该路径应该是相对于在虚拟服务器级别上定义的*应用程序根目录*的路径。

例如：

location="*applications/<ear name>/<war-module name>/*"

location="*modules/<war-module name>*"

location="*/u/myapps/<war-module name>*"

location="*/u/myapps/<ear-name>/<war-module name>*"

enabled 属性

该属性的默认值是 True，表示为服务请求启用了 Web 应用程序。通过将 enabled 属性设置为 False，可以暂时使请求无法获取 Web 应用程序。但是， Web 应用程序的内容 （例如，存储在硬盘上的内容）不会被删除。

Web 应用程序部署

Web 容器部署来自 Web ARchive (WAR) 文件的 Web 应用程序，或者部署来自包含 WAR 文件分解视图的目录 （WEB-INF/lib、WEB-INF/classes，等等）的 Web 应用程序。您不需要重新启动计算机即可部署应用程序。

Web 容器在每个虚拟服务器上部署一个 “默认” Web 应用程序。默认的位置 （目录）为虚拟服务器上 app root 目录下的 modules/*default-web-app*/ 子目录。这个默认 Web 应用程序负责处理虚拟服务器中的其它 Web 应用程序不能响应的所有请求。该 Web 应用程序包含一个用于处理对 /servlet/* 的请求的调用程序 servlet，以及一个用于处理 JSP 页面的 JSP servlet。只要用户在 web.xml 和 sun-web.xml 文件中指示了 EJB 参考，默认 Web 应用程序即可访问 EJB。

在虚拟服务器的 server.xml 中定义的默认 Web 应用程序应如下所示：

<web-module context-root="" location="modules/*default-web-app*/">

动态重新部署和热部署

动态重新部署这项功能可以重新部署现有应用程序，而无需重新启动服务器。动态重新部署会在应用程序的配置 （其 xml 文件的内容）和某些类发生改变时进行。动态重新部署过程的结果和动态重新装入整个应用程序的类结果相同。另外，动态重新部署创建了新应用程序上下文 （Web 和 ejb）并删除旧的上下文。这样，进行动态重新部署之后，将产生全新的应用程序实例 （除了现有会话数据）。该功能只在开发模式下得到支持，并可能会导致出现类似于动态重新装入中的异常情况。而且，需要服务器重启才能生效的配置修改在重新启动之前不能生效。只有那些中心配置中指定了动态重新装入的应用程序和未共享的独立模块才能激活动态重新装入。

当 Web 应用程序重新装入后，无论会话管理器是否配置了持久性机制，所有现有会话信息都将自动进行保存和恢复。

热部署是指在无需重新启动服务器即可在服务器运行时部署应用程序的功能。该功能使用的体系结构与动态重新部署相同。但是，因为上一阶段未留下任何状态，此功能在生产时可用。

单一登录功能

只要用户只访问位于指定虚拟服务器上的 Web 应用程序中的未保护资源，则不需要进行身份验证。

当用户访问任何与指定虚拟服务器关联的 Web 应用程序中的受保护资源时，需要使用当前要访问的 Web 应用程序指定的登录方法对自己进行身份验证。

身份验证通过后，用户的相关角色将获取对所有相关 Web 应用程序的访问控制决策。此时用户不再需要对每个 Web 应用程序单独进行身份验证。

在用户登录退出 Web 应用程序之后，其在所有 Web 应用程序中的会话将失效。在此之后，如果想再次对应用程序中的受保护资源进行访问，则需要用户再次进行身份验证。

单一登录功能使用 HTTP Cookie 传输一个将每个请求和保存的用户身份相关联的标记，因此该功能只能在支持 Cookie 的客户机环境中使用。

记录 Web 容器

通过设置不同的*日志级别*，您可以控制存储在虚拟服务器上的 Web 容器和应用程序的默认记录行为。注意，该记录行为不影响应用程序本身的记录。

指定控制要记录的消息类型的日志级别。例如，如果您指定仅日志级别为 FATAL 的消息被记录，则日志级别比这个值 “更高”的消息会自动被忽略。只有以明确日志级别记录的消息才会与该值进行比较。

如果记录的消息的日志级别不明确，则该消息将无条件地被记录。该默认的行为是记录所有警告、错误和严重错误的消息。

为 Web 容器设置日志级别的步骤：

1. 在管理界面的左侧窗格中，展开 Sun ONE Application Server 实例树，找到要修改的 Web 容器配置。
2. 展开 “Containers”标签，从显示的 J2EE 容器列表中选择 “Web Container”。您会在管理界面的右侧窗格中看到以下页面，如图管理界面所示。

记录 Web 容器

3. 从 “Log Level”下拉列表中，选择所需的日志级别。有关所有日志级别及其说明的一览表，请参见第 5 章 “使用日志”。
4. 单击 “Save”对设置进行保存。

要为 Web 容器创建其它属性，单击 “Properties”按钮。

关于 EJB 容器

企业 Java Bean 容器是一个运行时环境，它用于控制企业 Bean 并为企业 Bean 提供重要的系统级服务。 EJB 是在 EJB 容器内执行的组件，而 EJB 容器在 EJB 服务器内执行。为 Bean 提供的系统级服务有：

- 事务管理
- 安全性
- 生命周期管理
- 远程连接
- 数据库连接加入
- 命名服务

企业 Bean 是用 Java 编写的服务器组件，它包含了商业逻辑。使用 EJB 容器可以对 Bean 进行远程访问。 EJB 总是在容器的上下文中工作，而容器充当 EJB 及其所在的服务器之间的链接。EJB 容器使用您自己的组件和其他供应商的组件，使分布式应用程序生成。

通过 EJB 容器， Sun ONE Application Server 提供了高级别的事务、状态管理、多线程和资源加入包装，这样您就无需了解低级别的 API 细节。该容器提供了所有 2.0 EJB 规范中说明的标准容器服务，还提供了 Sun ONE Application Server 特定的其它服务。

该容器使用钝化和激活进程来管理 Bean 的活动以确保可扩展性。

本节包括以下主题：

- 了解 EJB 容器的角色
- 配置 EJB 容器

了解 EJB 容器的角色

EJB 容器提供了以下标准服务：

- 钝化

  将 EJB 从内存转移到第二个存储器的进程。钝化使一个 Bean 的资源得到释放，同时又不会损坏该 Bean。这样，一个 Bean 就可以持久存在，并且可以重新调用，而无需进行实例化。

- 激活

  将 EJB 从第二个存储器转移到内存的进程。EJB 容器合同在 EJB 及其容器之间建立联系，而且对于客户机而言是完全透明的。这种联系包括：

  - 生命周期

    对于会话 Bean，这种联系包括 javax.ejb.SessionBean 和 javax.ejb.SessionSynchronization 接口实现。对于实体 Bean，这种联系包括 javax.ejb.EntityBean 接口实现。对于消息驱动的 Bean，这种联系包括 javax.ejb.MessageDriven 接口实现。

  - 会话上下文

    容器实现 javax.ejb.SessionContext 接口，以便在 Bean 实例创建后将服务和信息传送到会话 Bean 实例。

  - 实体上下文

    容器实现 javax.ejb.EntityContext 接口，以便在 Bean 实例创建后将服务和信息传送到实体 Bean。

  - 消息上下文

    容器实现 javax.ejb.MDBContext 接口，以便在 Bean 实例创建后将服务和信息传送到消息驱动的 Bean。

  - 环境

    容器实现 java.util.Properties 并使这些属性可供它的 EJB 使用。

  - 服务信息

    容器使其服务可供 EJB 使用。

Sun ONE Application Server 服务包括远程访问、命名、安全性、并行性、事务控制和数据库访问。

使用 EJB 容器可以：

- 创建允许远程连接的实现对象 (EJBObject)。
- 创建允许创建 EJBObject 的主实现对象。
- 将主实现对象绑定到命名服务，以备客户机查找。
- 确保只有授权的客户机才能调用该 Bean 方法 （通过 EJBObject）。
- 确保该商业方法在适当的事务中进行调用。
- 管理 Bean 的生命周期。管理 Bean 的生命周期包括：
  - 将 Bean 聚集成池
  - 调用相应的回叫方法 （例如 ejbActivate/ejbPassivate）
  - 管理数据库连接的池，这样应用程序可以更有效地使用和重复使用连接。

实际实现细节是容器的一部分 （该容器基于某个容器及其 EJB 之间的符合标准的接口）。您不需要了解如何处理平台特定的实现细节。您可以创建通用且针对于任务的 EJB，这些 EJB 可用于支持 EJB 标准的任何供应商的产品。

如果我们了解 Sun ONE Application Server 所用的 EJB 的类型，这将非常有用。

企业 Java Bean 的类型

EJB 是代表以下项之一的对象：

- 一个与某个客户机的会话，它将自动保持多客户机调用的方法的状态。
- 一个持久实体对象，它可能由多个客户机共享。
- 一个无状态服务，例如，消息处理。

实体 Bean 主要用于使用 Java Database Connectivity (JDBC) API 来处理数据访问，而会话 Bean 提供了暂时应用程序对象并执行互不关联的商业任务。 EJB 有三种类型，请参见以下主题：

- 关于会话 Bean
- 关于实体 Bean
- 关于消息驱动的 Bean

关于会话 Bean

会话 Bean 为某个客户机请求实现商业规则或逻辑。

会话 Bean 用于表示暂时对象和进程，例如，单数据库记录、用于编辑的文档或专门用于单独客户机的商业对象。这就是说，会话 Bean 是一种专用资源，仅供创建它的客户机使用。因为这些对象仅供一个客户机使用，因而会话 Bean 可以保持客户机特定的会话信息，称为*会话*状态。

例如，您可能创建 EJB 来模仿电子购物车。每次用户登录到应用程序时，该应用程序创建购物车的会话 Bean 来保存该用户的购物信息。用户注销或结束购物之后，该会话 Bean 将被释放。

会话 Bean 具有以下特性：

- 会话 Bean 为一个客户机执行。
- 会话 Bean 的生命周期相对较短。
- 会话 Bean 并不总是能够在服务器崩溃后仍然存在。
- 如果 EJB 容器崩溃，会话 Bean 将被删除。
- 会话 Bean 还根据属性设置来处理事务管理。该特性是可选的。
- 会话 Bean 升级基础数据库中的共享数据。该特性是可选的。
- 会话 Bean 即可以是无状态会话，也可以是状态会话。

无状态会话 Bean。无状态会话 Bean 将特定用户所需的一个临时的商业逻辑段封装了一段时间。无状态会话 Bean 不保持会话状态。

状态会话 Bean。状态会话 Bean 是暂时的，但该会话 Bean 保持会话状态，以便保留客户机呼叫之间的内容和值。会话状态使得 Bean 的容器可以保持该会话 Bean 状态的信息，并可以在以后执行程序时根据所需重新创建该状态。

关于实体 Bean

实体 Bean 通常表示持久数据，这些数据直接保留在数据库中或作为对象通过 Enterprise Information System (EIS) 应用程序进行访问。存储 EJB 和 EJB 容器的服务器提供了可扩展的运行环境，可用于同时处于激活状态的实体 EJB。

实体 Bean 的一个简单示例：一个实体 Bean 被定义为表示数据库表中的一行，其中每个 Bean 实例表示特定的一行。实体 Bean 的一个复杂示例：一个实体 Bean 被设计为表示数据库中联合表的复杂视图，其中每个 Bean 实例表示一个购物车的内容。

实体 Bean 具有以下特性：

- 实体 Bean 提供了 EIS 资源 （通常是一个数据库）中数据的对象视图。
- 所有用户都可以访问实体 Bean。

- 实体 Bean 在服务器崩溃时可以透明地保存下来。
- 实体 Bean 使用的事务由容器管理或由 Bean 管理。

实体 Bean 将持久数据表示为容器管理的持久性或 Bean 管理的持久性。实体 Bean 的持久性可以由 Bean 管理或由容器管理。

Bean 管理的持久性。实体 Bean 管理其自身的持久性。Bean 开发者直接在 EJB 类方法中实现持久性代码（例如 JDBC 呼叫）。这其中不利的一面是可能会丢失可移植性（如果使用专用接口），并可能会导致将 Bean 连接到特定数据库。

容器管理的持久性。实体 Bean 持久性由容器管理。因为容器透明地管理持久性状态，因而您不需要在 Bean 方法中实现任何数据访问代码。这种方法不仅实现起来更简单，而且使 Bean 可以实现完全的可移植性，而不与任何特定数据库相关联。

使用容器管理的持久性的实体 Bean 从本质上讲是一个使用 Bean 管理持久性的实体 Bean 的自动生成（由该容器生成）的版本。

有关创建和使用实体 Bean 的详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Enterprise JavaBeans Technology》*。

关于消息驱动的 Bean

消息驱动的 Bean 是一个 EJB，它允许 J2EE 应用程序异步处理消息。消息驱动的 Bean 由返回的 Java Message Service 消息所驱动。

消息驱动的 Bean 实例从其创建到删除都位于消息驱动的 Bean 的容器中。该容器提供了消息驱动的 Bean 实例的安全性、事务、消息并行处理、生命周期管理，以及消息驱动的 Bean 的其它服务。存储 EJB 和 EJB 容器的服务器提供了可扩展的运行环境，可用于同时处于激活状态的消息驱动的 Bean。

J2EE 1.3 平台中的 Java Message Service API 指定了以下项：

- 应用程序客户机、 EJB 组件和 Web 组件可以发送或同步接收 Java Message Service 消息。另外，应用程序客户机可以异步使用 Java Message Service 消息。
- 消息驱动的 Bean 启用了消息异步处理功能。 Java Message Service 提供商可以选择使消息驱动的 Bean 实现对消息的同步处理。

消息驱动的 Bean 表示一个无状态服务；它本质上是一个完全匿名且身份对用户不可见的异步消息接收客户机。消息驱动的 Bean 即没有主接口，也没有组件接口。客户机通过向消息驱动的 Bean 类是 MessageListener 的 Java Message Service 目标（队列或主题）发送消息来访问消息驱动的 Bean。

只有消息驱动的 Bean 可以异步接收消息。 Java Message Service MessageListener 不允许是会话或者实体 Bean。

消息驱动的 Bean 具有以下特性：

- 在接收到单用户消息后执行。
- 为异步调用。
- 生命周期相对较短。
- 不直接表示数据库中的共享数据，但是可以访问和升级该数据。
- 如果 EJB 服务器崩溃，即被删除。
- 为无状态。
- （可选）具有事务意识。

配置 EJB 容器

您可以为 EJB 容器配置日志级别，也可以启用监视。 EJB 容器既处理 EJB，也处理 MDB。使用管理界面，您可以为该容器管理的 EJB 和 MDB 配置设置。本节包括以下主题：

- 进行通用配置
- 配置 EJB 设置
- 配置 MDB 池设置

进行通用配置

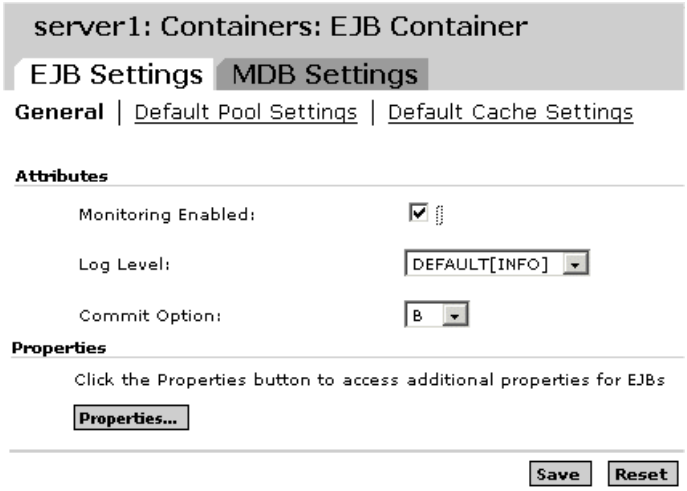
您可以对 EJB 容器的以下方面进行配置：

- 记录
- 监视
- 事务属性

设置 EJB 容器的日志级别、启用监视器和设置事务属性的步骤：

1. 在管理界面的左侧窗格中，打开 Sun ONE Application Server 实例树，找到要修改的 EJB 容器配置。
2. 展开 “Containers”标签，从显示的 J2EE 容器列表中选择 “EJB Container”。您会在管理界面右侧窗格中看到以下窗口，如图 EJB 容器 - 通用配置所示。

EJB 容器 - 通用配置



3. 选中 “Monitoring Enabled”旁边的复选框，将启用对 EJB 容器的监视。您现在已为 Sun ONE Application Server 的该特定实例的 EJB 容器启用了监视。有关 EJB 容器的可监视方面的信息列表，请参见表 "EJB 容器的监视统计。

4. 从 “Log Level”下拉列表中，选择所需的日志级别。有关所有日志级别及其说明的一览表，请参见第 5 章 “使用日志”。指定控制要记录的消息类型的日志级别。例如，如果您指定仅日志级别为 FATAL 的消息被记录，则日志级别比这个值 “更高”的消息会自动被忽略。只有以明确日志级别记录的消息才会与该值进行比较。

   如果记录的消息的日志级别不明确，则该消息将无条件地被记录。该默认的行为是记录所有警告、错误和严重错误的消息。

5. 从 “Commit Option”下拉列表中，选择要用于 EJB 容器的 “Commit Option”。

   事务可以以两种方式结束：使用提交或回滚。当事务提交时，其语句对数据所做的修改将被保存。当您设计企业 Bean 时，要决定该提交是容器管理的事务还是 Bean 管理的事务。因而用户界面中选项为：B 表示 Bean 管理的提交， C 表示容器管理的提交。

6. 单击 “Properties”按钮为 EJB 容器创建新的属性。

7. 单击 “OK”对设置进行保存。

下表显示了可以监视的 EJB 容器的属性：

EJB 容器的监视统计

统计名称	数据类型和单位	值的范围	说明
minBeansInPool	整数型	0-MAXINT	池中 Bean 的推荐的最小数目 （适用于无状态会话 Bean）。
initialBeansInPool	整数型	0-MAXINT	池中 Bean 的初始数目 （适用于无状态会话 Bean）。
maxBeansInPool	整数型	0-MAXINT	池中 Bean 的最大数目。（适用于无状态会话 Bean）。
beanIdleTimeoutInSeconds	整数型	0-MAXLONG	空闲超时 （以秒为单位），空闲值大于此值的 Bean 将被删除。
numBeansCreated	整数型	0-MAXINT	到目前为止创建的 Bean 的数目。
numBeansDestroyed	整数型	0-MAXINT	到目前为止删除的 Bean 的数目。
numThreadsWaitaing	整数型	0-MAXINT	等待空闲的 Bean 的线程数目。
numBeansInPool	整数型	0-MAXINT	池中可用的 Bean 的数目。（如果此数目大于零，则 numThreadsWaitaing 必须为 0）
maxBeansInCache	整数型	0-MAXINT	高速缓存中 Bean 的最大数目 （适用于实体 Bean 和有状态 Bean）。
minBeansInCache	整数型	0-MAXINT	高速缓存中 Bean 的推荐的最大数目 （适用于实体 Bean 和有状态 Bean）。
cacheFaultsPercentage	双精度型		引起从备份存储中进行激活的高速缓存丢失的数目。

配置 EJB 设置

使用管理界面，您可以为 EJB 容器管理的 EJB 配置默认池和 Bean 高速缓存设置，请参见以下主题：

- 配置 EJB 池设置
- 配置 EJB 高速缓存设置

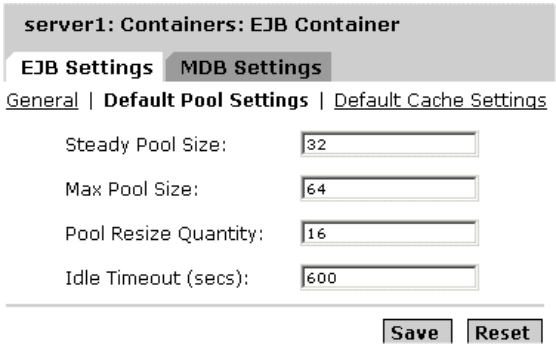
配置 EJB 池设置

配置 EJB 池设置的步骤：

1. 在管理界面的左侧窗格中，打开 Sun ONE Application Server 实例树，从中修改要修改的 EJB 设置。

2. 展开 “Containers”标签，从显示的 J2EE 容器列表中选择 “EJB Container”。您会在管理界面右侧窗格中看到以下窗口，如图编辑 EJB 池设置所示。

编辑 EJB 池设置



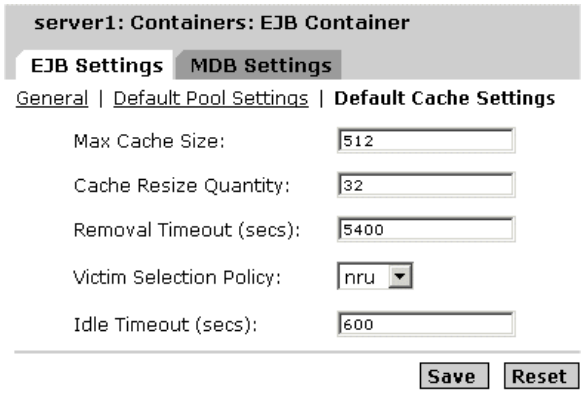
3. 在 “Steady Pool Size”字段中，指定池中 Bean 数目的最小值。此设置适用于无状态会话 Bean。
4. 在 “Max Pool Size”下拉列表中，指定该池在任何指定时间所拥有的 Bean 的最大数目。此设置适用于无状态会话 Bean。
5. 在 “Pool Resize Quantity”字段中，指定当 Bean 的空闲时间超过 idle-timeout-in-seconds 标记中指定的时间时，要从池中删除的 Bean 的数目。
6. 在 “Idle Timeout (secs)”字段中，指定 Bean 可以保持空闲状态的时间 （以秒为单位）。如果超过该时间段而 Bean 仍处于空闲状态，则它会被删除。
7. 单击 “Save”保存所做的更改。

配置 EJB 高速缓存设置

配置 EJB 高速缓存设置的步骤：

1. 在管理界面的左侧窗格中，打开 Sun ONE Application Server 实例树，从中修改要修改的 EJB 设置。
2. 展开 “Containers”标签，从显示的 J2EE 容器列表中选择 “EJB Container”。您会在管理界面右侧窗格中看到以下窗口，如图编辑 EJB 池设置所示。

编辑 EJB 高速缓存设置



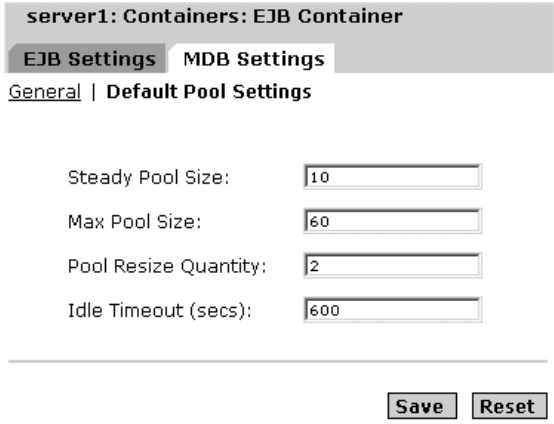
3. 在 “Max Cache Size”字段中，指定高速缓存中保留的 Bean 的最大数目。这个属性的默认值如在 idle-timeout-in-seconds 属性中指定一样。
4. 在 “Cache Resize Quantity”字段中，指定如果池中的 Bean 数目超过了 “Max Cache Size”属性中指定的数目时，要删除的 Bean 的数目。
5. 在 “Removal Timeout (secs)”字段中，指定在备份存储中处于空闲状态的 Bean 可以保持钝化的时间值。如果 Bean 在 removal-timeout-in-seconds 属性中指定的时间值内未得到客户机的访问，则该 Bean 将从备份存储中删除，因而客户机也将无法对其进行访问。
6. 从 “Victim Selection Policy”下拉列表中，选择用于选择牺牲 Bean 以将其从池中删除所需的牺牲选择算法。
7. 在 “Idle Timeout (secs)”字段中，指定允许 Bean 在高速缓存中保持空闲的时间。如果 Bean 处于空闲状态的时间超过该值，则将被钝化。 Bean 保持钝化状态 （在空闲备份存储中）的时间段由 removal-timeout-in-seconds 参数控制。
8. 单击 “Save”保存所做的更改。

配置 MDB 池设置

使用管理界面，您可以配置由 EJB 容器管理的 MDB 的默认池设置。配置 MDB 的默认池设置的步骤：

1. 在管理界面左侧窗格中，打开 Sun ONE Application Server 实例树，从中修改要修改的 MDB 容器配置。
2. 展开 “Containers”标签，从显示的 J2EE 容器列表中选择 “EJB Container”。您会在管理界面右侧窗格中看到以下窗口，如图配置 MDB 池设置所示。

配置 MDB 池设置



3. 单击 “MDB Settings”。在 “Steady Pool Size”文本字段中，指定池中 Bean 的最小数目。此设置适用于无状态会话 Bean。
4. 在 “Max Pool Size”字段中，指定该池在任何指定时间所拥有的 Bean 的最大数目。
5. 在 “Pool Resize Quantity”字段中，指定当 Bean 的空闲时间超过 idle-timeout-in-seconds 标记中指定的时间时，要从池中删除的 Bean 的数目。
6. 在 “Idle Timeout (secs)”字段中，指定 Bean 可以保持空闲状态的时间 （以秒为单位）。如果超过该时间段而 Bean 仍处于空闲状态，则它会被删除。
7. 单击 “Save”对设置进行保存。

第 9 章

使用事务服务

事务是商业运作的重要组成部分。典型的商业事务为两方或多方之间资产的转移。准确的记录通常存储在一个或多个数据库中。由于此信息对于商业运作十分关键，因此它必须有效、实时、可靠。对于编程新手来说，事务处理是比较困难的。 J2EE 平台提供了多种抽象概念，用于简化可靠事务处理应用程序的开发。本章将讨论 J2EE 事务及 Sun ONE Application Server 中的事务支持。

本章概述了 Java 事务，并详述了 Sun ONE Application Server 中的事务支持。

本章包括以下主题：

- 什么叫事务？
- J2EE 中的事务
- 事务资源管理器
- 本地和分布式事务
- 容器管理的事务
- Bean 管理的事务
- 事务服务管理

什么叫事务？

要模拟商业事务，程序需要执行几个步骤。例如，通过执行以下伪代码中列出的步骤，财务程序可以将资金从支票帐户转到储蓄帐户中：

```
begin transaction
debit checking account
credit savings account
update history log
commit transaction
```

在以上伪代码中， begin 语句和 commit 语句标记出事务的界限。要完成此事务，必须完成以上三个步骤。否则，数据的完整性将被破坏。

该保证称为原子性。事务可以通过两种方式结束：使用 commit 或 rollback。提交事务时，在事务边界内通过语句进行的修改将被永久保存。更改是*可持续的*，即可以在系统出现故障之前一直存在。如果事务内的语句出现故障，事务将回滚，撤销事务内已执行的所有语句的效果。例如，在伪代码中，如果磁盘驱动器在贷出过程中崩溃，事务将回滚并撤销借入语句所做的数据更改。

即使是事务失败，数据完整性也不会受到破坏，因为事务帐户仍然是平衡的。事务操作的这个特点称为事务*一致性*。

事务服务还具有*隔离*的特点，也就是说提交或回滚事务之前其它应用程序和线程无法看到事务中进行的阶段。事务被提交后，应用程序和线程即可安全查看提交的事务。

J2EE 中的事务

J2EE 中的事务处理涉及以下五个实体：事务管理器、应用程序服务器、资源管理器、资源适配器和用户应用程序。通过实现不同的 API 和功能，每个实体都有助于实现事务处理过程的可靠性，如下所述：

- 事务管理器提供支持事务划分、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。
- 应用程序服务器提供了支持应用程序运行时环境 （包括事务状态管理）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所使用的事务资源接口来参与分布式事务，从而与事务关联、事务完成及恢复工作进行通信。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，应用程序服务器或客户机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 J2EE 应用程序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）。可以使用 EJB 的声明事务属性设置或明确的程序事务划分。

术语 “资源管理器”通常可以和 “资源适配器”互用，因为这两个实体之间有着紧密的联系。

事务资源管理器

J2EE 事务支持下列事务资源管理器。

- 数据库
- JMS 提供商
- J2EE 连接器

数据库

数据库是 J2EE 应用程序中最常见的事务资源管理器。 JDBC 是 J2EE 组件用来访问数据库的 API。数据库资源被配置为 JDBC 资源。 JDBC 资源由资源管理器或 JDBC 驱动程序来管理。 JDBC 驱动程序支持本地事务或全局事务，在某些情况下可以同时支持这两种事务。

Sun ONE Application Server 支持在各种 J2EE 组件中使用 JDBC 和事务。有关注册和配置 JDBC 资源的详细信息，请参见第 239 页上的 “关于 JDBC 资源”。应用程序服务器负责提供事务的连续性 （即启动事务和从多个应用程序组件访问数据库）。例如， servlet 可以启动事务、访问数据库、调用访问同一事务中同一数据库的企业 Bean，最后提交事务。

JMS 提供商

JMS 表示 Java 消息传送服务。在 J2EE 术语中， JMS 提供商就是指消息代理服务。JMS API 能够在应用程序之间提供可靠的事务消息交换。对 JMS 数据源的支持是 J2EE 中必需的功能。 JMS 资源和 JDBC 资源可以参与同一事务。

Sun ONE Application Server 与 Sun ONE Message Queue 集成在一起，该软件是一个全功能 JMS 提供商和对应的事务资源管理器。 Sun ONE Application Server 以这种方式从 servlet、 JSP 页面和企业 Bean 中启用事务 JMS 访问。 Sun ONE Application Server 也可以与第三方 JMS 提供商结合使用。有关详细信息，请参见第 11 章 “使用 JMS 服务”。

J2EE 连接器

Sun ONE Application Server 支持将 XATransaction 模式用作事务资源管理器的资源适配器。平台必须启用从 servlet、 JSP 页面和企业 Bean 对资源适配器的事务访问。也可以从一个事务的多个应用程序组件访问资源适配器。例如， servlet 可能希望启动事务、访问资源适配器、调用访问同一事务中的资源适配器的企业 Bean，最后提交事务。

本地和分布式事务

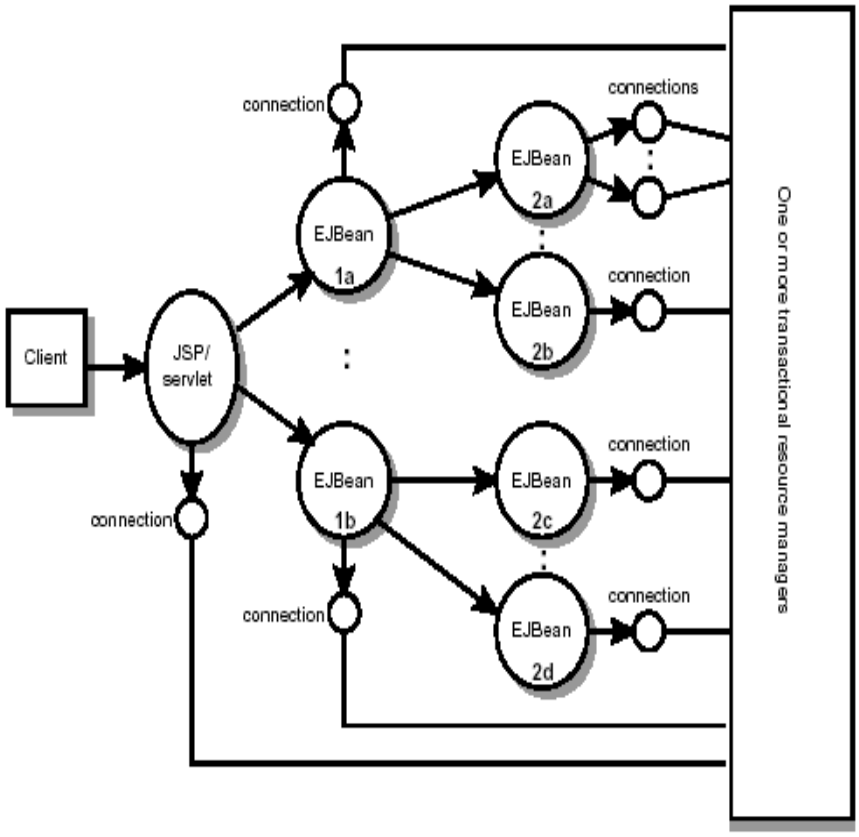
只涉及一种资源的事务可以使用本地事务来完成。本地事务还要求所有参与的应用程序组件都在一个进程中执行。涉及多种资源或多个参与者进程的事务则是分布式或全局事务。本地事务优化使用资源管理器的特定优化，这种优化对于 J2EE 应用程序是透明的。

事务类型主要由相关资源管理器实现的接口来确定。例如，实现 javax.sql.DataSource 接口的 JDBC 数据源可以参与本地事务。实现 javax.sql.XADataSource 的数据源可以参与全局事务。有些 JDBC 资源可以实现两个接口，并且当这类 JDBC 资源使用 Sun ONE Application Server 注册时，可能需要提供 Sun ONE Application Server 中的其它配置信息来表示该资源的首选功能。

本地事务较为简单，自然要比全局事务的效率高。但是，如果需要转换的数据分布在多个数据源，则本地事务不能满足需要。有时，无法预测一个事务中需要多少个数据源。因此，在现实世界中最常见的是全局事务。某些性能优化可能在全局事务中更有效。

J2EE 支持由用于访问事务中多个企业 Bean 的 servlet 或 JSP 的任意组合组成的事务应用程序。每个组件都需要一个或多个连接来访问一个或多个事务资源管理器。在下图中，调用树从访问多个企业 Bean 的 servlet 或 JSP 页面启动，然后再访问其它企业 Bean。这些组件通过连接访问资源管理器。

事务中访问资源的 J2EE 组件



例如，应用程序可能要求上图中的所有组件都访问一个事务中的资源。应用程序服务器提供商必须提供支持这种方案的事务功能。

J2EE 事务管理支持平面事务。平面事务不能包含子 （嵌套）事务。

事务恢复是分布式事务的重要方面。在重要的阶段无法访问资源或出现其它不可恢复的错误时，则需要考虑分布式事务的状态。自动或手动恢复出现问题或未完成的事务时是 Sun ONE Application Server 的重要功能。可以使用管理界面启用自动事务恢复。有关如何控制事务恢复的详细信息，请参见第 211 页上的 “事务服务管理”。

连接 （此处用作资源的同意词）可以被标记为可共享或不可共享。准备以不可共享方式使用连接的 J2EE 应用程序组件必须提供有关连接不可共享的部署信息，以防容器共享连接。例如，改变安全属性、隔离级别、字符设置和本地化配置时可能需要此部署信息。

容器不能尝试共享标记为不可共享的连接。如果连接未标记为不可共享，则不管连接是否共享，该连接对于应用程序来说必须是透明的。

J2EE 应用程序组件可以使用可选的部署描述符元素 res-sharing-scope 来表示与资源管理器的连接是否可共享。如果没有部署提示，则容器应假设连接是可共享的。J2EE 应用程序组件可以高速缓存连接对象，并在多重事务中重复使用这些对象。提供连接共享的容器应该能够透明地交换高速缓存连接对象 （分发时），以便在正确的事务范围内指向适当的共享连接。

当设计企业 Bean 应用程序时，开发者必须确定如何指定边界。

容器管理的事务

在包含容器管理事务的企业 Bean 中， EJB 容器可以设置事务的边界。可以通过任意类型的企业 Bean 使用容器管理事务：会话、实体或消息驱动。容器管理的事务可以简化开发过程，因为企业 Bean 代码不能明确地标记事务的边界。代码不包括开始和结束事务的语句。

通常，企业 Bean 方法开始之前，容器会立即启动一个事务，并在方法即将退出之前提交事务。每种方法都可以与单个事务相关联。一个方法内不允许包含嵌套事务或多个事务。

容器管理事务不要求所有方法都与事务相关联。部署 Bean 时，可以通过设置事务属性来指定与事务相关联的 Bean 方法。

本节包括以下主题：

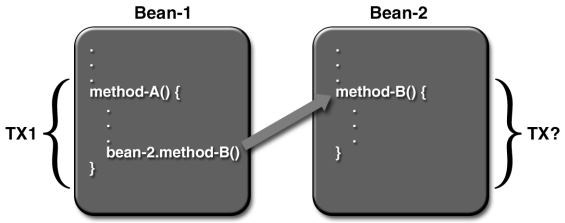
- 事务属性
- 设置事务属性
- 回滚容器管理的事务
- 同步会话 Bean 的实例变量

- 容器管理事务中不允许的方法

事务属性

事务属性用于控制事务的范围。下图说明了控制事务范围的重要性。在以下图表中，method-A 启动了一个事务，然后调用 Bean-2 的 method-B。执行 method-B 时，它是在由 method-A 启动的事务范围内运行还是执行新的事务？答案取决于 method-B 的事务属性。

事务属性



事务的属性值可以为以下值之一：

- Required
- RequiresNew
- Mandatory
- NotSupported
- Supports
- Never

Required

如果客户机在事务内运行并调用企业 Bean 的方法，则方法在客户机的事务内执行。如果客户机与事务不关联，容器将在运行方法之前启动新事务。

Required 属性适用于大多数事务。因此，您可能需要将其用作默认属性，至少在开发初期希望如此。由于事务属性是声明性的，因此以后可以很方便地进行更改。

RequiresNew

如果客户机在事务内运行并调用企业 Bean 的方法，容器将执行以下步骤：

- 挂起客户机的事务
- 启动新事务
- 将调用委托给方法
- 方法完成后恢复客户机事务

如果客户机与事务不关联，容器将在运行方法之前启动新事务。

如果希望确保方法始终在新事务中运行，则应该使用 RequiresNew 属性。

Mandatory

如果客户机在事务内运行并调用企业 Bean 的方法，则方法在客户机的事务内执行。如果客户机与事务不关联，容器将抛出 TransactionRequiredException。

如果企业 Bean 的方法必须使用客户机的事务，则可以使用 Mandatory 属性。

NotSupported

如果客户机在事务内运行并调用企业 Bean 的方法，则容器将在调用方法之前挂起客户机的事务。方法完成后，容器将恢复客户机的事务。

如果客户机与事务不关联，运行方法之前，容器不会启动新事务。

为不需要事务的方法使用 NotSupported 属性。由于事务会占用系统资源，因此该属性可以改善性能。

Supports

如果客户机在事务内运行并调用企业 Bean 的方法，则方法在客户机的事务内执行。如果客户机与事务不关联，运行方法之前，容器不会启动新事务。

由于方法的事务操作可能各不相同，因此应该谨慎使用 Supports 属性。

Never

如果客户机在事务内运行并调用企业 Bean 方法，容器将抛出 RemoteException。如果客户机与事务不关联，运行方法之前，容器不会启动新事务。

属性小结

下表总结了事务属性的效果。 T1 和 T2 事务都由容器来控制。 T1 事务与企业 Bean 中调用方法的客户机关联。大多数情况下，客户机是另一种企业 Bean。而 T2 事务仅在方法执行前由容器来启动。

在最后一列，“None”表示在由容器控制的事务中不执行商业方法。但是，这种商业方法中的数据库调用可以由 DBMS 的事务管理器来控制。

事务属性

事务属性	客户机事务	商业方法的事务
Required	无	T2
	T1	T1
RequiresNew	无	T2
	T1	T2
Mandatory	无	错误
	T1	T1
NotSupported	无	无
	T1	无
Supports	无	无
	T1	T1

设置事务属性

由于事务属性存储在部署描述符中，因此，在 J2EE 应用程序部署的很多阶段中都可以对其进行更改：企业 Bean 的创建阶段、应用程序汇编阶段以及部署阶段。但是，开发者需要在创建 Bean 时指定属性。只有将组件组合为大型应用程序的应用程序开发者才能修改这些属性，部署 J2EE 应用程序的用户不负责指定应用程序属性。

可以为整个企业 Bean 或单个方法指定事务属性。如果为某一方法指定了一个属性，而为 Bean 指定了另一个属性，则方法的属性优先。为单个方法指定属性时，要求会因 Bean 的类型不同而异。会话 Bean 需要定义的属性用于商业方法，而不允许将这些属性用于创建方法。实体 Bean 要求事务属性用于商业、创建、删除和查找器方法。消息驱动的 Bean 要求事务属性 （Required 属性或 NotSupported 属性）用于 onMessage 方法。

回滚容器管理的事务

回滚容器管理的事务有两种方法。第一种方法是：如果抛出系统异常，容器将自动回滚事务。第二种方法是：通过调用 EJBContext 接口的 setRollbackOnly 方法， Bean 方法将指示容器回滚事务。如果 Bean 抛出一个应用程序异常，回滚则不是自动的，但可以通过调用 setRollbackOnly 启动。

在以下示例中， BankEJB 示例的 transferToSaving 方法说明了 setRollbackOnly 方法。如果发生负检查平衡， transferToSaving 将调用 setRollBackOnly，并抛出应用程序异常 (InsufficientBalanceException)。 updateChecking 和 updateSaving 方法将更新数据库表。如果更新失败，这些方法将抛出 SQLException， transferToSaving 方法将抛出 EJBException。由于 EJBException 是一个系统异常，因此会引起容器自动回滚事务。以下是 transferToSaving 方法的代码：

```
public void transferToSaving(double amount) throws
    InsufficientBalanceException {

    checkingBalance -= amount;
    savingBalance += amount;

if (checkingBalance < 0.00) {
context.setRollbackOnly();

throw new InsufficientBalanceException();
}
    try {
        updateChecking(checkingBalance);

    updateSaving(savingBalance);
    } catch (SQLException ex) {
        throw new EJBException
            ("Transaction failed due to SQLException: "
            + ex.getMessage());
  }
}
```

当容器回滚事务时，通常会撤销事务内由 SQL 调用对数据所做的更改。然而，只有在实体 Bean 中，容器才撤销对实例变量所做的更改。（通过自动调用实体 Bean 的 ejbLoad 方法可以完成此操作，即从数据库加载实例变量。）发生回滚时，会话 Bean 必须明确地重置事务中已更改的实例变量。要重置会话 Bean 的实例变量，最简单的方法是实现 SessionSynchronization 接口。

还可以通过将事务 ID 传递给命令行界面来回滚事务。有关详细信息，请参见第 213 页上的 “使用命令行界面管理事务”。

同步会话 Bean 的实例变量

通过 SessionSynchronization 接口 （可选）可以同步数据库中的实例变量及其相应的值。在事务的各主要阶段，容器将调用 SessionSynchronization 方法 （afterBegin、beforeCompletion 和 afterCompletion）。

afterBegin 方法通知实例新的事务已开始。容器首先调用 afterBegin，然后调用事务内的第一个商业方法。 afterBegin 方法适用于从数据库加载实例变量。例如，BankBean 类可以在 afterBegin 方法中加载 checkingBalance 和 savingBalance 变量。

```
public void afterBegin() {

    System.out.println("afterBegin()");
    try {
        checkingBalance = selectChecking();
        savingBalance = selectSaving();
    } catch (SQLException ex) {
        throw new EJBException("afterBegin Exception: " +
            ex.getMessage());
    }
}
```

商业方法完成后，容器将调用 beforeCompletion 方法，但仅在提交前调用。beforeCompletion 方法是会话 Bean 回滚事务的最后一种方法 （通过调用 setRollbackOnly）。如果尚未使用实例变量的值更新数据库，会话 Bean 则可以在 beforeCompletion 方法中完成该操作。

afterCompletion 方法表示事务已完成。它是一个单布尔参数。如果事务被提交，其值为 True ；如果事务被回滚，其值则为 False。如果发生回滚，会话 Bean 可以在 afterCompletion 方法中刷新数据库的实例变量：

```
public void afterCompletion(boolean committed) {

    System.out.println("afterCompletion:" + committed);
    if (committed == false) {
        try {
            checkingBalance = selectChecking();
            savingBalance = selectSaving();
} catch (SQLException ex) {
            throw new EJBException("afterCompletion SQLException:
            " + ex.getMessage());
}
}
}
```

容器管理事务中不允许的方法

不要调用可能影响容器设定的事务边界的任何方法。以下是禁用的方法的列表：

- java.sql.Connection 的提交方法和 setAutoCommit 和回滚方法
- javax.ejb.EJBContext 的 getUserTransaction 方法
- javax.transaction.UserTransaction 的方法

但是，可以使用这些方法设置 Bean 管理的事务中的边界。

Bean 管理的事务

在 Bean 管理的事务中，会话 Bean 或消息驱动的 Bean 中的代码明确地标出了事务的边界。实体 Bean 不能包含 Bean 管理的事务，而必须使用容器管理的事务。尽管带有容器管理事务的 Bean 需要较少的编码，但它也有一个不足之处：执行方法时，它可以仅与一个事务相关联，也可以与事务根本不相关联。如果此限制增加了编写代码的难度，则应该考虑使用 Bean 管理的事务。

以下伪代码说明了可以通过 Bean 管理的事务获得的精密控制的类型。通过检查各种条件，伪代码可以确定启动或停止商业方法中的各种事务。

```
begin transaction
...
update table-a
...
if (condition-x)
    commit transaction
else if (condition-y)
    update table-b
    commit transaction
else
    rollback transaction
    begin transaction
    update table-c
    commit transaction
```

事务服务管理

可以通过使用管理界面或命令行界面来管理事务。

本节包括以下主题：

- 使用管理界面管理事务
- 使用命令行界面管理事务

使用管理界面管理事务

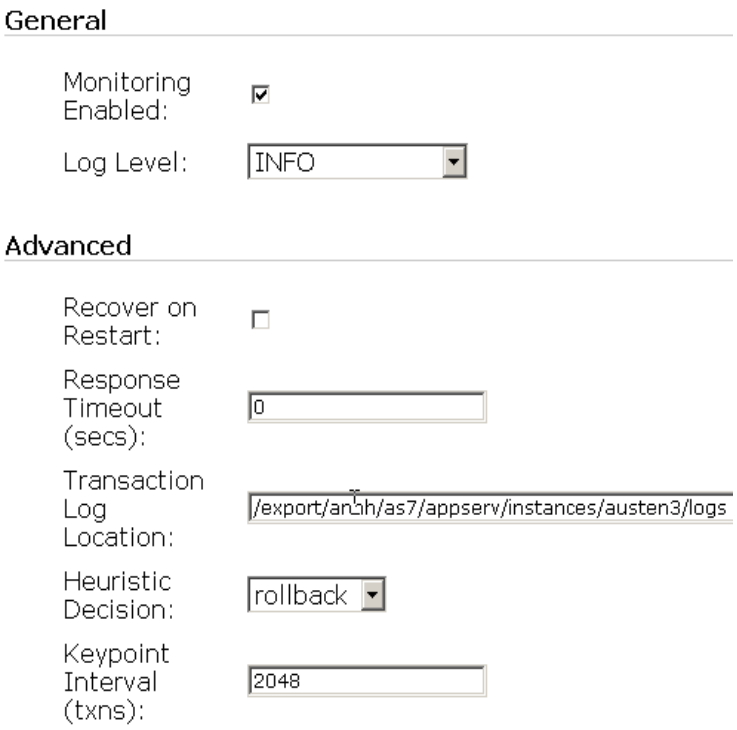
可以使用管理界面启用监视、设置日志级别并指定事务的高级选项。

可以控制实例范围的事务服务属性，例如恢复方针和超时。此处指定的属性和配置存储在 server.xml 文件中。

要配置事务服务选项，请执行以下任务：

1. 在管理界面的左侧窗格中，打开要修改其事务配置的 Sun ONE Application Server 实例树。
2. 从显示的 J2EE 服务列表中选择事务服务。将看到以下窗口，如管理界面的右侧窗格中 " 配置事务服务选项 " 图中所示：

配置事务服务选项



3. 要启用对事务的监视，请选中 “Monitoring Enabled”复选框。下表列出了可以被监视的 Java 事务服务的功能：

Java 事务服务的可监视属性

属性	类型	说明
transactionsCompleted	int	启用监视后完成的事务数量
transactionsRolledBack	int	启用监视后回滚的事务数量
transactionsRecovered	int	启用监视后恢复的事务数量
transactionsInFlight	int	当前正在处理的事务数量
timeStamp	long	在生成统计的位置记录的时间 （以毫秒为单位）。它记录 System.getCurrentTimeInMillis() 报告的所有内容。

4. 从 “Log Level”下拉列表中选择要为事务设置的日志级别。有关日志级别以及如何使用它们的详细信息，请参见第 5 章 “使用日志”。
5. 选中 “Recover on Restart”旁边的复选框，以便在服务器重新启动时自动恢复失败的事务。在重要阶段无法访问资源或出现其它不可恢复的错误时，事务将无法完成，仍然保留在事务日志文件中。如果选中了此复选框，服务器将在重新启动服务器时尝试恢复出现问题的事务。如果相关资源仍然不可用，则会延迟服务器的重新启动。默认情况下不选中此复选框。
6. 对于使用容器管理事务的企业 Bean，则可以通过设置 “Transaction Timeout (secs)”属性来控制事务超时间隔。

   如果该属性的值设置为 0，事务将不会超时。

   在 “Transaction Timeout (secs)”字段中，指定事务超时间隔。如果在指定时间内未完成，事务将回滚。如果为此属性设置的值为 0，事务则不会超时。
7. 在 “Transaction Log Location”字段中，指定要存储日志文件的目录的绝对路径。要使新的事务日志目录生效，需要重新启动服务器。
8. 从 “Heuristic Decision”下拉框中，选择要应用到事务中的试探性决定。如果不能明确确定结果，请从指定的选项中选择 “Commit”或 “Rollback”，指定应用程序服务器应该在恢复期间确定的事务的结果。如果将 “Heuristic Decision”设置为 “Rollback”，将回滚事务。在某些情况下，提交这些事务更为合适。
9. 在 “Keypoint Interval (transactions)”字段中指定日志中密钥点操作之间的事务数量。通过删除已完成的事务和正在压缩文件的项，密钥点操作可以减小事务日志文件的大小。属性值越大，事务日志文件也越大，但同时密钥点操作更少，性能可能更佳。属性值越小 （例如， 100），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。

使用命令行界面管理事务

可以使用命令行界面 (CLI) 管理并监视数据库事务，如以下各节中所述：

- 列出进行中的事务
- 管理事务
- 冻结事务服务
- 监视事务

以下各节讲述了如何使用命令行界面管理和监视事务。

列出进行中的事务

应使用以下命令获取正在处理的事务数据 （假设处于多模式，并且已经设置了用户名和密码）：

- asadmin> get --monitor <instanceName>.transaction-service.inflight-tx

多行输出结果为：

Transaction Id State Elapsed Time (ms)

txnid1 Prepared 20

txnid2 Active 100

txnid3 Active 120

... ...

管理事务

在列出进行中的事务中给出的示例中，假设您要使用以下事务 ID 回滚事务：txn-ids、 txnid2 和 txnid3。回滚选定事务的样例命令如下：

asadmin> set --monitor <instanceName>.transaction-service.rollback-list=txnid2,txnid3

冻结事务服务

要冻结事务服务，请运行以下命令：

asadmin> set --monitor <instanceName>.transaction-service.freeze=true

当冻结事务服务后，应用程序服务器中的事务管理器将挂起所有进行中的事务。建议不要在生产部署系统中冻结事务。

要解冻事务服务，请运行以下命令：

asadmin> set --monitor <instanceName>.transaction-service.freeze=false

当重新设置事务服务时，系统将从停止点继续运行。如果活动系统处于冻结状态的时间太长，某些数据库连接则可能会超时，而导致事务回滚。

监视事务

要监视事务的数据 （包括进行中的事务数据），请运行以下命令：

asadmin> get --monitor <instanceName>.transaction-service.*

如果运行此命令时没有处于活动状态的事务，将获得以下输出结果：

total-tx-completed = 5

total-tx-rolledback = 2

total-tx-inflight = 0

isFrozen = false

tx-inflight = No active transactions found.

如果运行此命令时有处于活动状态的事务，将获得以下输出结果：

total-tx-completed = 5

total-tx-rolledback = 2

total-tx-inflight = 2

isFrozen = false

tx-inflight =

Transaction Id State Elapsed Time(ms)

txnid1 Prepared 500

txnid2 Active 360

第 10 章

配置命名和资源

本章介绍了 Sun ONE Application Server 使用的 J2EE 资源，并讨论了用于创建和管理这些资源的方法：

本章包括以下主题：

- 关于 J2EE 命名服务和资源
- 关于 Java 命名和目录接口 (JNDI)
- 关于 Persistence Manager 资源
- 关于 JDBC 资源
- 关于 Java 邮件资源

关于 J2EE 命名服务和资源

J2EE 应用程序（包括 EJB、Web 应用程序组件和应用程序客户机）可以访问许多种类的资源，例如，资源管理器、数据源 （如 SQL 数据源）、连接工厂、邮件会话、Java Message Service 目标对象以及 URL 连接工厂。 J2EE 平台通过 Java 命名和目录接口 (JNDI) 命名服务将这些资源提供给应用程序。

Sun ONE Application Server 使您可以创建和管理以下 J2EE 资源：

- JDBC 数据源
- Java 邮件会话
- JMS 目标

JDBC 数据源

JDBC 数据源是一种 J2EE 资源，可使用 Sun ONE Application Server 进行创建和管理。

JDBC API 是用于和关系数据库系统进行连接的 API。 JDBC API 有两个部分：

- 一个应用程序级接口，应用程序组件使用它来访问数据库。
- 一个服务提供商接口，用于将 JDBC 驱动程序连接到 J2EE 平台。

JDBC DataSource 对象代表以 Java 编程语言编写的数据源。从本质上来讲，数据源是一种存储数据的工具。数据源可能会像大型公司的综合数据库一样复杂，也可能会像包含行和列的文件一样简单。JDBC 数据源是一种 J2EE 资源，可通过 Sun ONE Application Server 进行创建和管理。

有关 JDBC 数据源的详细信息，请参见第 239 页上的 “关于 JDBC 资源”。

Java 邮件会话

JMS 目标是 J2EE 资源，可通过 Sun ONE Application Server 进行创建和管理。

许多 Internet 应用程序都需要具有发送电子邮件通知的能力，因此， J2EE 平台提供了 JavaMail API 以及一个使应用程序组件可以发送 Internet 邮件的 JavaMail 服务提供商。 JavaMail API 有两个部分：

- 一个应用程序级接口，应用程序组件使用它来发送邮件。
- 一个服务提供商接口，以 J2EE SPI 级别使用。

Java 邮件会话是 J2EE 资源，可通过 Sun ONE Application Server 进行创建和管理。

有关 Java 邮件会话的详细信息，请参见第 257 页上的 “关于 Java 邮件资源”。

JMS 目标

Java 消息传送服务 (JMS) 是一种用于消息传送的标准 API，支持可靠的点对点消息传送以及“发布 - 订阅”模式。此服务要求 JMS 提供商既可以实现点对点消息传送，同时也可以实现 “发布 - 订阅”消息传送。

JMS 提供了两种通用的管理对象类型：连接工厂和目标。尽管这两种类型都包含提供商特定的信息，但它们在 JMS 客户机中的用途极为不同。连接工厂用于创建与消息服务器的连接，而目标对象则用于标识 JMS 消息传送服务的物理目标。

关于 Java 命名和目录接口 (JNDI)

本节将讨论 Java 命名和目录接口 (JNDI)。 Java 命名和目录接口 (JNDI) 是一种应用程序编程接口 (API)，用于访问不同类型的命名和目录服务。 J2EE 组件通过调用 JNDI 查找方法来定位对象。

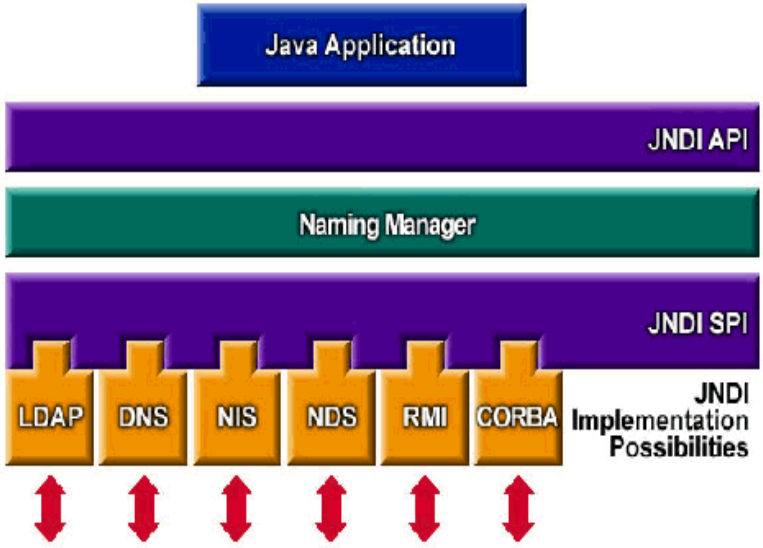
本节包括以下主题：

- JNDI 体系结构
- J2EE 命名服务
- 命名参考和绑定信息
- J2EE 标准部署描述符中的命名参考
- JNDI 连接工厂

JNDI 体系结构

JNDI 体系结构包含一个应用程序编程接口 (API) 以及一个服务提供商接口 (SPI)。 Java 应用程序使用 JNDI API 访问各种命名和目录服务。SPI 可以使各种命名和目录服务透明地插入，这样 Java 应用程序则可以使用 JNDI API 访问它们的服务。下图“JNDI 体系结构概况”说明了可通过 JNDI API 访问的服务：

JNDI 体系结构概况



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，因而我们通常将一个对象的便于使用的名称称作该对象的 JNDI 名称。 Pointbase 数据库的 JNDI 名称为 jdbc/Pointbase。 Pointbase 启动时， Sun ONE Application Server 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、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环境在运行时提供的所有的环境项。
- 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项的值。
- 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。有关 J2EE 容器 （如 Web 容器和 EJB 容器）如何使用 JNDI 命名服务查找对象的详细信息，请参见第 181 页上的 “配置 J2EE 容器”。

命名参考和绑定信息

资源参考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，编码名称参考资源的连接工厂。在下节给出的示例中，资源参考名称是 jdbc/SavingsAccountDB。

资源的 JNDI 名称与资源参考的名称是不同的。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。由于此命令方法可用于分离，因而如果组件以后需要访问其它资源，则不必在代码中更改名称。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 “JNDI 查找及其关联参考”列出了 Sun ONE Application Server 使用的 J2EE 资源的 JNDI 查找及其关联参考。

JNDI 查找及其关联参考

JNDI 查找名称	**关联参考**
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 参考
java:comp/UserTransaction	UserTransaction 参考
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 连接工厂和目标
java:comp/ORB	应用程序组件之间共享的 ORB 实例

J2EE 标准部署描述符中的命名参考

命名参考是一种字符串，应用程序使用这种字符串在指定的命名上下文中查找对象。每个 J2EE 应用程序都有一个命名上下文，而且其参考在标准组件部署描述符中进行了配置。本节介绍了 Sun ONE Application Server 中使用的标准部署描述符功能。本节包括以下主题：

- 应用程序环境项
- EJB 参考
- 资源管理器连接工厂的参考
- 资源环境参考
- UserTransaction 参考
- COSNaming 服务

应用程序环境项

环境项是使用 <env-entry> 定义的，它提供了一种为 J2EE 应用程序指定部署时间参数的方法。注意，如果是 Web 应用程序，则可以使用 <context-param> 定义 servlet 上下文初始化参数，但 <env-entry> 是首选方法，因为应用程序部署者通过明确指定此类应用程序参数的名称、类型和值来对其进行配置。

下面的样例介绍了 J2EE 标准部署描述符中指定的 <env-entry> 的语法：

```
<env-entry>
<description> Send pincode by mail </description>
<env-entry-name> mailPincode </env-entry-name>
<env-entry-value> false </env-entry-value>
<env-entry-type> java.lang.Boolean </env-entry-type>
</env-entry>
```

<env-entry-type> 标记为项指定了一个全限定的类名。下面是一个代码段，它使用 JNDI 从应用程序组件 （该术语指 servlet/JSP 或实体 Bean 或 IIOP 应用程序客户机）中查找 <env-entry>：

```
Context initContext = new InitialContext();
Boolean mailPincode = (Boolean)
initContext.lookup("java:comp/env/mailPincode");

// 用户可以在子上下文中使用相对名
Context envContext = initContext.lookup("java:comp/env");
Boolean mailPincode = (Boolean)
envContext.lookup("mailPincode");
```

EJB 参考

除了部署描述符支持之外， JNDI 命名服务使应用程序能够使用 “逻辑”名称 （称为 EJB 参考）映射到企业 Bean 的主接口，如下面的示例所述：

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
<ejb-link> EmployeeEJB </ejb-link>
</ejb-ref>
```

像 JSP 这样的应用程序组件可以使用 JNDI 访问 EJB 主对象，如下面的示例所述：

```
Context initContext = new InitialContext();
Context envContext = initContext.lookup("java:comp/env");
Object result = envContext.lookup("ejb/EmplRecord");
EmployeeRecordHome emplRecordHome = (EmployeeRecordHome)
javax.rmi.PortableRemoteObject.narrow(result, EmployeeRecordHome.class);
```

ejb-ref-name 元素定义了应用程序代码中使用的字符串 （如上面给出的示例所示）。ejb-link 元素将此参考链接到使用 ejb-name 元素定义的目标企业 Bean，而 ejb-name 元素属于 ejb-jar.xml 中定义的实体 Bean。还可以提供链接功能，而无需修改应用程序部署描述符或企业 Bean 描述符。

资源管理器连接工厂的参考

工厂是一种根据需要创建其它对象的对象。资源工厂创建资源对象，例如，数据库连接或消息服务连接。它们是使用标准部署描述符中的 <resource-ref> 元素进行配置的。

以下示例介绍了工厂的使用：

示例 A：

以下示例声明了对返回对象类型为 javax.sql.DataSource 的 JDBC 连接工厂的参考：

```
<resource-ref>
<description> Primary database </description>
<res-ref-name> jdbc/primaryDB </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

示例 B：

下面是对 JavaMail 会话资源工厂的参考的示例：

```
<resource-ref>
<description> mail Session </description>
<res-ref-name> mail/Session </res-ref-name>
<res-type> javax.mail.Session </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

<res-type> 是该资源工厂的全限定类名。可以将 Container 或 Application 作为值指定给 <res-auth> 变量。有关配置 JavaMail 会话资源工厂的详细信息，请参见第 257 页上的 “关于 Java 邮件资源”。

如果指定了 Container，Web 容器会在将资源工厂绑定到 JNDI 查找注册表之前进行验证。如果指定了 Application，servlet 必须以编程方式进行验证。不同类型的资源工厂将在描述了资源类型的单独的子上下文中进行查找，如下所示：

- jdbc/，用于 JDBC javax.sql.DataSource 工厂
- jms/，用于 JMS javax.jms.QueueConnectionFactory 或 javax.jms.TopicConnectionFactory
- mail/，用于 JavaMail javax.mail.Session 工厂
- url/，用于 java.net.URL 工厂

下面是一个代码段，它通过用容器处理验证，从应用程序组件获取 JDBC 连接：

```
InitialContext initContext = new InitialContext();
DataSource source =
(DataSource) initContext.lookup("java:comp/env/jdbc/primaryDB");
Connection conn = source.getConnection();
```

注意，为了确保这些资源参考正常工作， res-ref-name 必须在运行时映射到有效的资源工厂。

资源环境参考

资源环境参考提供了一种可以通过 JNDI 查找访问与资源关联的管理对象的方法。例如，某个应用程序可能需要访问 JMS 目标对象。标准部署描述符中定义的 <resource-env-ref> 元素使应用程序可以声明资源要求。

<resource-env-ref> 与 <resource-ref> 元素之间的主要区别是 <resource-ref> 没有特定的资源验证要求，但两种元素都必须通过资源工厂描述符进行备份。

示例：

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

使用下面代码段，您可以访问 JMS 主题对象：

```
InitialContext initContext = new InitialContext();
javax.jms.Topic myTopic =
(javax.jms.Topic) initContext.lookup("java:comp/env/jms/MyTopic");
```

注意，为了使这些 resource-env-ref 变量正常工作，管理员必须在运行时使目标资源工厂可用。有关访问 JMS 主题目标和队列目标的详细信息，请参见第 11 章 “使用 JMS 服务”。

UserTransaction 参考

J2EE 要求容器在 JNDI 名称 java:comp/UserTransaction 处提供 UserTransaction 对象实现。 UserTransaction 对象使应用程序可以启动、提交和中止事务。

要以编程方式启动和执行事务，组件通过针对 java:comp/UserTransaction 执行 JNDI 参考，获取容器的默认事务协调器的参考。返回的对象实现 javax.transaction.UserTransaction 接口，并可用于在程序中启动、提交、回滚和查询事务的状态。 Sun ONE Application Server 中的 JNDI 实现支持诸如对事务协调器的查找。有关 javax.transaction.UserTransaction 接口的详细信息，请参见第 199 页上的 “使用事务服务”。

初始命名上下文

Sun ONE Application Server 中的命名支持主要基于 J2EE 1.3，同时添加了一些增强功能。当应用程序组件通过 InitialContext() 创建初始上下文时， Sun ONE Application Server 返回一个充当应用程序命名环境的句柄的对象。此对象又为 java:comp/env namespace 提供了子上下文。每个应用程序获取其自己的名称空间，这就是说， java:comp/env namespace 是针对每个应用程序的，而且一个应用程序的名称空间中绑定的对象不与其它应用程序中绑定的对象发生冲突。

COSNaming 服务

EJB 互操作性协议要求使用 COSNaming 协议，以便使用 JNDI API 查找 EJB 对象。

要在 CosNaming 服务中发布 EJBHome 对象参考，则需要使用 EJB 容器。CosNaming 必须在定义的 CosNaming 模块中实现 IDL 接口，并使客户机可以通过 IIOP 调用解析和列表操作。

CosNaming 服务在提供根 NamingContext 对象的主机、端口和对象关键字时，必须遵守 CORBA Interoperable Name Service 规范中的要求。CosNaming 必须能够服务于使用提供的主机、端口和对象关键字的根 NamingContext 的 IIOP 调用。

需要使用客户机容器 （即 EJB、 Web 或应用程序客户机容器）以包含 JNDI CosNaming 服务提供商，该服务提供商使用 Interoperable Name Service 规范中定义的机制与服务器的 CosNaming 服务进行联系，并通过标准 CosNaming API 解析 EJBHome 对象。 JNDI CosNaming 服务提供商可能使用 JNDI SPI 体系结构，也可能不使用该体系结构。 JNDI CosNaming 服务提供商必须从以下 URL 创建对象参考，才能访问服务器的 CosNaming 服务的根 NamingContext：

corbaloc:iiop:1.2@<host>:<port>/<objectkey>（其中， <host>、 <port> 和 <objectkey> 是由服务器的 CosNaming 服务提供的或通过一种等效机制提供的根 NamingContext 的值）。

部署时，客户机容器的开发者应该为客户机组件的部署描述符中每个 ejb-ref 元素获取服务器的 CosNaming 服务的主机、端口和对象关键字，以及服务器 EJBHome 对象的 CosNaming 名称（例如，通过浏览服务器的名称空间）。这样， ejb-ref-name（由 JNDI 查找调用中的客户机代码使用）将会链接到 EJBHome 对象的 CosNaming 名称。运行时，客户机组件的 JNDI 查找调用使用 CosNaming 服务提供商，该服务提供商联系服务器的 CosNaming 服务、解析 CosNaming 名称并将 EJBHome 对象参考返回到客户机组件。

由于 EJBHome 对象的名称在 CosNaming 服务 （通过提供的主机和端口可以访问）的命名空间范围之内，因此没有必要结合客户机容器和服务器容器的名称空间。

使用 CosNaming 的好处是，不仅可以获得与非 J2EE CORBA 客户机和服务器更好的内部互操作性，还可以更好地与内部互操作性所需的 IIOP 基础架构进行集成。由于 CosNaming 仅存储 CORBA 对象，因此，供应商很可能会使用其它企业目录服务，以便存储其它资源。

Sun ONE Application Server 按照 J2EE 1.3 规范集成了 JNDI 的所有命名资源。

CosNaming 提供商。 为了支持全局 JNDI 名称空间 （使其接受 IIOP 应用程序客户机的访问）， Sun ONE Application Server 包括了支持对 CORBA 参考 （远程 EJB 参考）进行绑定的、基于 J2EE 的 CosNaming 提供商。返回到 IIOP 客户机的 InitialContext 是 CosNaming 提供商。一个 Sun ONE Application Server 服务器的实例注册了 IIOP 客户机要查找并绑定到的实体 Bean。

注意， Sun ONE Application Server 处理存储在 CosNaming 中的对象，而本地 JNDI 命名环境是暂时的：也就是说，在每次服务器启动以及应用程序重新加载时，都会再次把所有相关对象重新绑定到名称空间。有关配置对 CORBA/IIOP 客户机的支持的详细信息，请参见第 295 页上的 “为 Corba/IIOP 客户机配置服务器”。

JNDI 连接工厂

对于 J2EE Web 应用程序， web.xml 文件中的部署描述符是定义应用程序环境项、资源管理器 （如 SQL 数据源）连接工厂或 EJB 的参考的占位符。应用程序使用 J2EE 容器提供的 JNDI InitialNamingContext 查找这些参考。这样，仅通过对部署描述符进行更改 （即不需要访问或修改应用程序的源代码），应用程序即可移植到不同的应用程序服务器环境中。同样地， J2EE 要求将实体 Bean 的部署描述符 (ejb-jar.xml) 以及 IIOP 应用程序客户机 (application-client.xml) 作为这些 JNDI 命名参考的主要工具。

连接工厂是一种对象，它生成使 J2EE 组件能够访问资源的连接对象。数据库的连接工厂是一种 javax.sql.DataSource 对象，该对象创建 java.sql.Connection 对象。

在 Sun ONE Application Server 中，您可以配置访问以下资源和资源工厂的方式：

- JDBC 连接工厂
- 基于 MQ 的 JMS 连接工厂
- JavaMail 会话连接工厂
- JCA 连接器工厂
- 用户编写的普通自定义资源对象工厂。
- 支持诸如 LDAP 这样的外部资源系统信息库

所有 Sun ONE Application Server 资源工厂是在 server.xml 中的 <resources> </resources> 标记内指定的，并且具有一个用 jndi-name 属性指定的 JNDI 名称。此属性用于在服务器范围内的名称空间中注册工厂。部署者可以使用 resource-ref-mapping 元素，将用户指定的、应用程序特定的资源参考名称 （在 resource-ref 或 resource-env-ref 元素内声明）映射到这些服务器范围内的资源工厂。这样有助于在部署时进行决定给定的应用程序使用哪些 JDBC 驱动程序 （和其它资源工厂）。

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。

本节包括以下主题：

- 创建自定义资源

- 创建外部 JNDI 资源
- 访问外部 JNDI 系统信息库
- 映射应用程序资源参考
- 关于 URL 连接工厂资源
- 映射应用程序资源环境参考
- 映射 EJB 参考

创建自定义资源

server.xml 中定义的 custom-resource 元素提供了一种指定自定义的服务器范围的资源对象工厂的方法。这种对象工厂实现 javax.naming.spi.ObjectFactory 接口。此元素将一个要在服务器范围的命名空间中使用的 JNDI 名称 （通过 jndi-name 子元素进行指定，就像其它 Sun ONE Application Server 资源一样）、类型、其资源工厂类的名称以及一组用于实例化相同属性的标准属性关联起来。

以下示例说明了如何实现 javax.naming.spi.ObjectFactory 接口：

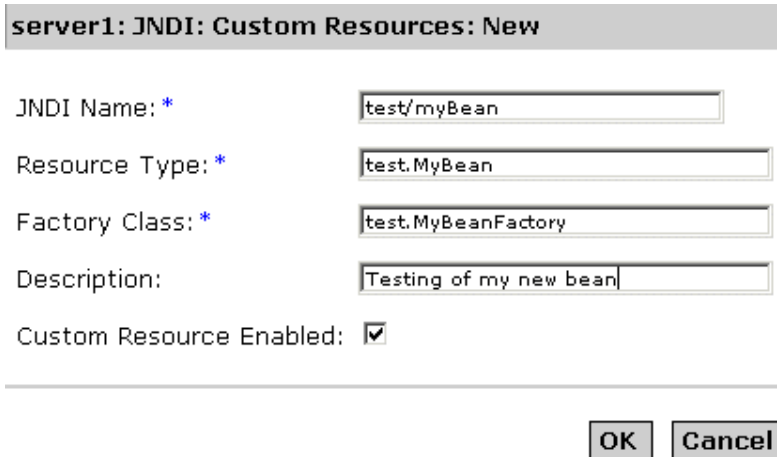
```
<resources> <custom-resource jndi-name="test/myBean"
res-type="test.MyBean"factory-class="test.MyBeanFactory"
enabled="true">

<property name="foo" value="test custom bean prop" />
</custom-resource>
</resources>
```

您需要确保该资源参考的环境参考和 EJB 参考已链接到配置的服务器范围的资源；而这些资源是使用 server.xml 中的 custom-resource 和 external-jndi-resource 标记定义的。动态地重新部署应用程序组件是 JNDI 命名环境中的一个问题。 Sun ONE Application Server 将发布所有应用程序特定的参考，并将所有的新参考重新绑定到新安装的应用程序的命名上下文中。

使用管理界面创建自定义资源的步骤：

1. 在管理界面的左侧窗格中，打开要修改其 JNDI 配置的 Sun ONE Application Server 实例。
2. 打开 “JNDI”标签，并单击 “Custom Resources”。任何已经创建的自定义资源将会列表显示在右侧窗格中。要创建新的自定义资源，单击 “New”。您会在管理界面的右侧窗格中看到下图 ““JNDI Custom Resources”页”：

“JNDI Custom Resources”页



3. 在 “JNDI Name”字段中，输入要访问资源的名称。此名称将注册到 JNDI 命名服务中。
4. 在 “Resource Type”字段中，输入全限定的类型定义，如上面示例中所示。应遵照以下格式在 “Resource Type”中输入定义：xxx.xxx。
5. 在 “Factory Class”字段中，为要创建的自定义资源输入工厂类名。该字段中的值是用户指定的工厂类的名称。此类实现了 javax.naming.spi.ObjectFactory 接口。
6. 在 “Description”字段中，为所创建的资源输入说明。此说明是字符串值，最多可以包含 250 个字符。
7. 选中 “Custom Resource Enabled”复选框，将启用自定义资源。
8. 单击 “OK”保存自定义资源。

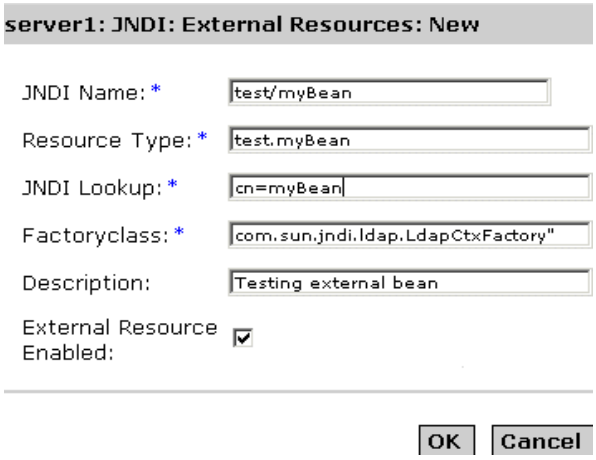
创建外部 JNDI 资源

使用管理界面创建外部资源的步骤：

1. 在管理界面的左侧窗格中，打开要修改其 JNDI 配置的 Sun ONE Application Server 实例。
2. 打开 “JNDI”并选择 “External Resources”。任何已经创建的外部资源将会以列表形式显示在右侧窗格中。要创建新的外部资源，单击 “New”。

您会在管理界面的右侧窗格中看到以下窗口，如 “JNDI External Resources” 页”中所示：

“JNDI External Resources”页



3. 在 “JNDI Name”字段中，输入要访问资源的名称。此名称将注册到 JNDI 命名服务中。
4. 在 “Resource Type”字段中，输入全限定的类型定义，如上面示例中所示。应遵照以下格式在 “Resource Type”中输入定义：xxx.xxx。
5. 在 “JNDI Lookup”字段中，输入要在外部系统信息库中查找的 JNDI 值。例如，当您创建一个与外部系统信息库连接的外部资源时，为了测试某个 Bean 类，“JNDI Lookup”可能会读取 cn=*testmybean*。
6. 在 “Factory Class”字段中，输入 JNDI 工厂类外部系统信息库 （例如，com.sun.jndi.ldap）。此类实现了 javax.naming.spi. ObjectFactory 接口。
7. 在 “Description”字段中，为所创建的资源输入说明。此说明是字符串值，最多可以包含 250 个字符。
8. 选中 “External Resource Enabled”复选框，将启用外部资源。
9. 单击 “OK”保存自定义资源。

访问外部 JNDI 系统信息库

通常， Sun ONE Application Server 上运行的应用程序需要访问外部 JNDI 系统信息库中存储的资源。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。

示例：

```
< 资源 >
<!-- external-jndi-resource 元素指定如何访问存储在外部
-- JNDI 系统信息库中的 J2EE 资源。下面的示例
-- 说明如何访问 LDAP 中存储的 Java 对象。
-- factory-class 元素指定了访问资源工厂时所需的
-- JNDI InitialContext 工厂。属性元素
-- 与适用于外部 JNDI 上下文的环境相匹配，
-- jndi-lookup-name 引用 JNDI 名称，查找并获取
-- 指定的 （在此示例中为 java）对象。
-->
<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>
```

映射应用程序资源参考

应用程序特定的资源参考必须被映射到预定义的服务器范围的资源工厂。 Sun ONE Application Server 特有的资源参考映射元素就用于此种用途。

下面的示例中说明了一个 Web 应用程序的部署描述符 web.xml，它将资源参考指定为 JDBC DataSource。

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

所需的 res-ref-name 也可以映射到容器范围的 Orcale JDBC 连接资源工厂，如下面所示：

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </resource-ref-name>
<jndi-name> jdbc/estore/InventoryDB </jndi-name>
</resource-ref>
```

关于 URL 连接工厂资源

URL 连接工厂不需要在 server.xml 中定义任何资源。相应的 Sun ONE Application Server 应用程序 （Web 或 ejb）部署描述符的 jndi-name 元素指定了目标 URL。

例如，我们假定某个 Web 应用程序的部署描述符 web.xml 指定了一个 java.net.URL 资源参考，而且将该资源参考映射到 sun-web.xml 中的 URL http://www.*sun.com/index.html*：

映射情况如下所示：

```
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<res-type>java.net.URL</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

```
<sun-web-app>
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<jndi-name> http://www.sun.com/index.html </jndi-name>
</resource-ref>
</sun-web-app>
```

映射应用程序资源环境参考

应用程序特定的资源环境参考声明必须被映射到该应用程序服务器的运行时环境中可用的目标资源对象。 Sun ONE Application Server 特定的配置文件中定义的资源环境映射元素使部署者可按如下方式进行映射：

示例：

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

此参考被映射到 server.xml 中定义的 jms/iMQ/Topics/Stocks/SUNW 主题。有关详细信息，请参见 《*Sun ONE Application ServerAdministrator's Configuration File Reference*》。

```
<resource-env-ref-mapping>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<jndi-name> jms/iMQ/Topics/Stocks/SUNW </jndi-name>
</resource-env-ref-mapping>
```

映射 EJB 参考

也可以将真正用于应用程序代码中的 ejb-name 和用于目标企业 Bean 的 ejb-name 分离开。当您不想修改 Web 应用程序部署描述符 web.xml 以及使用企业 Bean 部署描述符的 ejb-name 时，这尤其有用。 Sun ONE Application Server 特定的配置使您可以将 ejb-ref-name 元素映射到目标 Bean 的 ejb-name，而无需使用 Sun ONE Application Server 特定的部署描述符中的 ejb-ref-mapping 元素。

示例：

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
</ejb-ref>
```

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<jndi-name> AccountEJB </jndi-name>
</ejb-ref-mapping>
```

关于 Persistence Manager 资源

此模块介绍了持久性，并且其中建立的框架可用于 Sun ONE Application Server 支持的可插接的 Persistence Manager。

此模块包括以下主题：

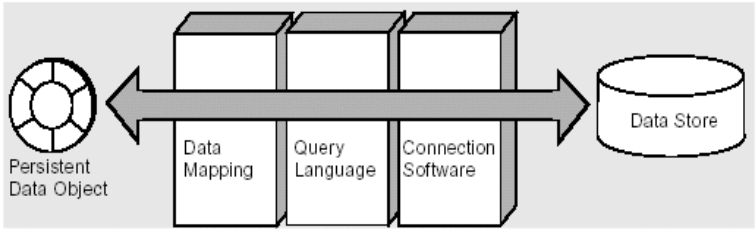
- 什么是持久性？
- Persistence Manager 的角色
- 预部署 Bean 配置
- 创建新的 Persistence Manager

什么是持久性？

多数商业应用程序的一个关键方面是对持久性数据进行程序操作；持久性数据即存储在应用程序之外的长期存在的数据。尽管将持久性数据读入临时内存中仅为了使用或修改目的，但该数据会写出到关系数据库或平面文件系统进行长期存储。

在面向对象的编程系统中，持久性数据在内存中表示为应用程序代码操作的一个或多个数据对象。一般说来，数据存储库中的持久性数据与其在内存中表示为的持久性数据对象之间的通信是通过许多软件层实现的，如下图 “基本持久性模式”所示：

基本持久性模式



每个数据存储库都有一个通过驱动程序软件与外部世界的接口，驱动程序软件用于建立和维护数据存储库和应用程序之间的连接。建立连接后，将使用查询语言从数据存储库检索信息并将数据读入应用程序，或者将数据从应用程序写到数据存储库之中。另一软件层提供了内存中的数据对象与数据存储库中的信息之间的映射关系。

通过此普通模式，程序员可以将持久性数据表示为运行时对象，使其由应用程序使用和操作。此模式支持所有的基本持久性操作，经常缩写为 CRUD：

- C — 创建持久性数据 （插入到数据存储库中）
- R — 检索持久性数据 （从数据存储库中选择）
- U — 更新持久性数据
- D — 删除持久性数据

Persistence Manager 的角色

Persistence Manager (PM) 负责 EJB 容器中具有容器管理的持久性的实体 Bean 的持久性。实体 Bean 提供商负责将实体 Bean 类提供为抽象类。Persistence Manager 提供商的工具负责提供具体实现。可以通过以下方式达到以上目标：对抽象实体 Bean 及相关类进一步分类并提供具体实现，或者通过利用封装和委托。

Persistence Manager 的工具所提供的类负责管理实体 Bean 之间的关系，并负责管理对其持久性状态的访问。 PM 工具还负责提供 java.util.Collection 类的实现，这些类用于维护容器管理的关系 (CMR)。

预部署 Bean 配置

企业 Java Bean 标准为实体 Bean 提供了两种持久性类型。一种是容器管理的持久性 (CMP)，另一种是 Bean 管理的持久性 (BMP)。 EJB 2.0 规范没有定义 EJB 服务器和持久性管理器之间的标准 API。

本节讲述了部署和代码生成时的集成要求。部署可用于指多种事情。一般认为部署过程包括三个截然不同的步骤：配置、代码生成和安装。

必须为一个 Bean 指定多个属性，包括所用的持久性机制、持久性供应商、所用版本，以及持久性机制所需的其它信息。多数的持久性供应商都具有项目的概念，项目代表可以作为一个单位部署的所有相关的 Bean 及其从属类。每个项目可以有一个供应商特定的 xml 文件。

支持部署用途的三种标准文件：ejb-jar.xml、 sun-ejb-jar.xml 以及 sun-cmp-mappings.xml。 sun-ejb-jar.xml 中每个带有 CMP Bean 的 EJB 模块必须具有一个 <pm-descriptors>， <pm-descriptors> 中至少带有一个 <pm-descriptor> 元素（该元素指定了另外五个属性）。这五个属性是 pm-identifier、 pm-version、 pm-config、 pm-class-generator 以及 pm-mapping-factory。

Sun ONE Application Server 特定的描述符 （正如在 sunEjb_jar_2_0.DTD 中）定义持久性管理器相关的标记。以下是一个用 Sun ONE Application Server DTD 定义的 CMP 描述符样例：

```
PM 描述符包含一个或多个 pm 描述符，但其中只有一个描述符必需在任何指定时间都使用
-->
<!ELEMENT pm-descriptors ( pm-descriptor+, pm-inuse)>
<!--
pm-descriptor 描述了与实体 Bean 关联的持久性
管理器的属性
-->
<!ELEMENT pm-descriptor ( pm-identifier, pm-version, pm-config?, pm-class-generator?,
pm-mapping-factory?)>
<!--
此元素描述了提供 PM 实现的供应商，例如，这可能是 Sun ONE Application Server Transparent Persistence、 TopLink、 Versant 或 CocoBase：
-->
<!ELEMENT pm-identifier (#PCDATA)>
<!--
pm-version 进一步指定了要使用的 PM 供应商产品的版本
-->
<!ELEMENT pm-version (#PCDATA)>
<!--
pm-config 指定了要使用的供应商特定的配置文件
-->
<!ELEMENT pm-config (#PCDATA)>
<!--
pm-class-generator 指定了供应商特定的具体类产生器
这是供应商特定的类的名称：
-->
<!ELEMENT pm-class-generator (#PCDATA)>
<!--
```

pm-mapping-factory 指定了供应商特定的映射工厂

这是供应商特定的类的名称：

-->

<!ELEMENT pm-mapping-factory (#PCDATA)>

创建新的 Persistence Manager

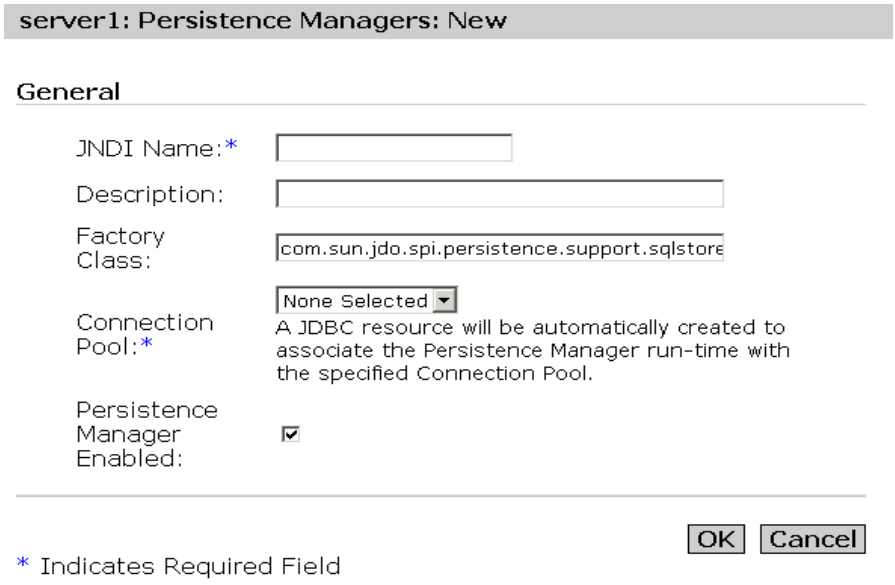
使用 管理界面，可以创建新的 Persistence Manager 实例。创建新的 Persistence Manager 实例的步骤：

1. 从管理界面的左侧窗格中，打开要为其创建新的 Persistence Manager 的 Sun ONE Application Server 实例。从显示的服务器组件列表中单击 “Persistence Manager”。

   任何已经为 Sun ONE Application Server 的该特定实例创建的 Persistence Manager 将列表显示在管理界面的右侧窗格中。

2. 要创建新的 Persistence Manager，单击 “New”。您会在看到以下窗口，如图 “创建新的 Persistence Manager”所示：

创建新的 Persistence Manager



3. 这是应用程序服务器运行时代表应用程序查找特定 Persistence Manager 时所用的 JNDI 名称。此名称必须与在 Sun 特定部署描述符的实体 Bean 的 CMP 资源元素中定义的名称相同。

4. 在 “Description”字段中，为新的 Persistence Manager 输入说明。此字段的值是字符串，最多可以包含 250 个字符。

5. 在 “Factory Class”字段中，为 Persistence Manager 输入工厂类连接。setEntityContext 通过 JNDI 名称查找功能查找此连接工厂。该工厂类名称是创建 Persistence Manager 实例的 Persistence Manager Factory 的类名。默认情况下，将此设置为 Sun ONE Application Server 的内部 Persistence Manager Factory 类；如果使用替代实现，则必须确保此类在服务器类路径中可用。

6. 从 “Connection Pool”下拉列表中，选择要将新的 Persistence Manager 加入到其中的数据库连接池。使用连接加入，实体 Bean 可以请求单个连接，并使用该连接执行多个客户机线程的并发语句。与任何其它数据库访问一样，Persistence Manager 可以使用连接加入提高性能和可缩放性。选择现有连接池，或者选择 “None Selected”（如果尚未创建连接池）。

   注意：系统将自动创建 JDBC 资源，以允许 PM 运行时使用 JNDI 绑定到连接池 — 如果 JDBC 资源的 JNDI 名称前加上一个 “PM”前缀，则它与 PM JNDI 名称完全相同。删除 Persistence Manager 同时将删除关联的 JDBC 资源。

7. 要启用 Persistence Manager，选中 “Persistence Manager Enabled”复选框。Persistence Manager 现在已为指定的连接工厂启用。

8. 单击 “OK”保存所做的更改。

关于 JDBC 资源

此模块概述了 JDBC API，然后具体介绍了 JDBC 资源及其在 Sun ONE Application Server 中的实现和应用。

此模块包括以下主题：

- 关于 JDBC API
- 关于数据库访问模式
- 关于 JDBC 数据源
- 关于 JDBC 连接
- 关于 JDBC 事务

关于 JDBC API

JDBC API 是一种 Java API，它可用于访问几乎所有类型的表格式数据。（有意思的是， JDBC 是商标名称而不是首字母缩写；然而， JDBC 通常被认作是 “Java Database Connectivity”的缩写。） JDBC API 包含一组用 Java 编程语言编写的类和接口，为工具/数据库开发者提供了标准的 API；而且使用 JDBC API 可以利用一种全 Java API 编写数据库应用程序。

JDBC API 使得将 SQL 语句发送到关系数据库系统更加容易，并且支持 SQL 的所有语言。但是 JDBC 3.0 API 超出了 SQL 的范围，而且使与其它种类数据源 （如数据库之外的文件）进行交互成为可能。

JDBC API 的优势在于应用程序可访问几乎所有的数据源，并且可以使用 Java 虚拟机在任何平台上运行。换言之，有了 JDBC API，就不必编写一个程序来访问 Sybase 数据库，编写另一个程序来访问 Oracle 数据库，再编写一个程序来访问 IBM DB2 数据库，等等。您可以使用 JDBC API 编写一个能够将 SQL 或其它语句发送到相应的数据源的程序。同时，有了用 Java 编程语言编写的应用程序，您就不必为编写在不同平台上运行的不同应用程序而感到苦恼。 Java 平台与 JDBC API 组合在一起，使程序员只需进行一次编写即可从任何地方运行代码。

JDBC API 有什么功能？

使用基于 JDBC 技术的驱动程序 （JDBC 驱动程序），可以完成以下三个步骤：

- 建立与数据源的连接
- 将查询和更新语句发送到数据源
- 处理结果

以下代码段是这三个步骤的一个简单示例：

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/AcmeDB");
Connection con = ds.getConnection("myLogin", "myPassword");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
while (rs.next()) {
        int x = rs.getInt("a");
        String s = rs.getString("b");
        float f = rs.getFloat("c");
    }
```

关于数据库访问模式

JDBC API 支持双层式和三层式数据库访问模式。Sun ONE Application Server 中集成了更常用的双层式数据库访问模式。

本节包括以下主题：

- 双层式数据库访问模式
- 三层式数据库访问模式

双层式数据库访问模式

在双层式数据库访问模式中，Java 小程序或应用程序使用 DBMS 专用协议直接与数据源进行通信。此访问模式需要 JDBC 驱动程序，该驱动程序可以与所访问的特定数据源进行通信。用户的命令传输到数据库或其它数据源，并且这些语句的结果被返回给用户。数据源可能位于用户通过网络与其连接的另一台计算机上。此配置称为客户机/服务器配置，其中，用户的计算机被当作客户机，而数据源所在的计算机被当作服务器。网络可以是一个 Intranet（例如，可用于连接某个公司内部的员工），也可以是 Internet。

三层式数据库访问模式

在三层式数据库访问模式中， Java 小程序或应用程序将命令发送到服务的 “中间层”，该中间层再将命令发送到数据源。客户机应用程序通过 HTTP、 RM、 CORBA 或其它调用与中间层进行通信。中间层通过 DBMS 专用协议与数据存储库进行通信。数据源处理命令，并将结果发送回中间层，中间层再将结果发送给用户。 MIS 主管们对三层式模型非常感兴趣，因为使用中间层可以控制其他人对公司数据的访问以及控制可对公司数据进行的更新的类型。三层式模式的另一个优点是可以简化应用程序的部署。最后一个优点是，三层式体系结构在许多情况下可以提供性能优势。

关于 JDBC 数据源

DataSource 对象在 Java 编程语言中表示数据源。从本质上来讲，数据源是一种存储数据的工具。数据源可能会像大型公司的综合数据库一样复杂，也可能会像包含行和列的文件一样简单。数据源可以位于远程服务器上，也可以位于本地桌面计算机上。应用程序使用连接访问数据源，并且可以将 DataSource 对象视为一个用于与该 DataSource 实例代表的特定数据源连接的工厂。 DataSource 接口提供了两种建立与数据源的连接的方法。

DataSource 对象中包含的属性标识并描述了该对象所代表的数据源。另外， DataSource 对象使用 JNDI 命名服务，并且该对象单独地从使用其的应用程序中创建、部署和管理。驱动程序供应商将提供一个类，该类是 DataSource 接口 （该供应商的 JDBC 2.0 or 3.0 驱动程序产品的组成部分）的基本实现。

本节包括以下主题：

- DataSource 对象的属性
- 注册 JDBC 资源

DataSource 对象的属性

DataSource 对象有一组属性，它们标识并描述了该对象所代表的现实世界的数据源。这些属性包括的信息有：数据库服务器的位置、数据库的名称、用来与服务器通信的网络协议，等等。 DataSource 属性遵照 JavaBeans 设计模式，并且通常是在 DataSource 对象部署时进行设置的。

为促使不同供应商的 DataSource 实现保持一致， JDBC 2.0 API 指定了一组标准属性，并为每个属性指定了标准名称。

一个实现 DataSource 接口的类的实例代表一个特定数据源。该实例生成的每个连接都将参考同一个数据源。在 DataSource 的基本实现中，对方法 DataSource.getConnection 的调用返回一个连接对象，该连接对象和 DriverManager 工具返回的连接对象一样，是一个与数据源的物理连接。

JNDI 为应用程序提供了一种在网络上查找和访问远程服务的统一方法。远程服务可以是任何企业服务 （包括消息传送服务或应用程序特定的服务），当然， JDBC 应用程序的主要着眼点是数据库服务。通过 JNDI 命名服务创建并注册了 DataSource 对象之后，应用程序则可以使用 JNDI API 访问该 DataSource 对象，然后可以使用该对象连接其所代表的数据源。

实现连接加入的 DataSource 对象同样会生成一个与 DataSource 类所代表的特定数据源的连接。但是，方法 DataSource.getConnection 返回的连接对象是 PooledConnection 对象的句柄，而不是物理连接。应用程序以其通常方式使用连接对象，如果发生不同的情况，一般也不会意识到。连接加入不会对应用程序代码产生任何影响，只是合并的连接就像所有连接一样，应始终明确关闭。当应用程序关闭一个加入的连接时，该连接会加入到一个可重复使用的连接池中。下次调用 DataSource.getConnection 时，就会返回池中的一个连接的句柄 （如果存在连接）。由于连接加入避免了每次请求连接时都创建一个新的物理连接，因而有助于显著提高应用程序的运行速度。

同样可以实现一个 DataSource 类，以便使用分布式事务环境。例如， EJB 服务器支持分布式事务，并需要一个 DataSource 类，该类实现后可以与其进行交互。在此情况下， DataSource.getConnection 方法返回一个可在分布式事务中使用的 Connection 对象。一般而言， EJB 服务器支持连接加入以及分布式事务。和连接加入一样，事务管理是在内部处理的，因此分布式事务非常便于容易。唯一的要求是，当事务为分布式时 （这需要两个或更多数据源），应用程序不能调用事务方法，无论 commit 还是 rollback。该应用程序也不能将连接置于自动提交模式。存在这些限制的原因是，事务管理器以隐蔽的方式开始和结束分布式事务，因此应用程序不能执行任何可能会在事务开始或结束时造成影响的操作。有关 Java 事务的详细信息，请参见第 9 章 “使用事务服务”。

注册 JDBC 资源

您可以使用管理界面或命令行界面将 JDBC 资源注册到 Sun ONE Application Server。

本节包括以下主题：

- 使用命令行注册资源
- 使用管理界面注册资源

使用命令行注册资源

要使用命令行界面接口注册 JDBC 资源，运行以下命令：

./asadmin create-jdbc-resource

用于注册 JDBC 资源的 XML 片段应指定以下的几个属性 （摘自 sun-server_7_0.dtd）。

<!-- JDBC javax.sql.DataSource resource definition -->

<!ELEMENT jdbc-resource (description?, property*)>

<!ATTLIST jdbc-resource jndi-name CDATA #REQUIRED

pool-name CDATA #REQUIRED

enabled %boolean; 'true'>

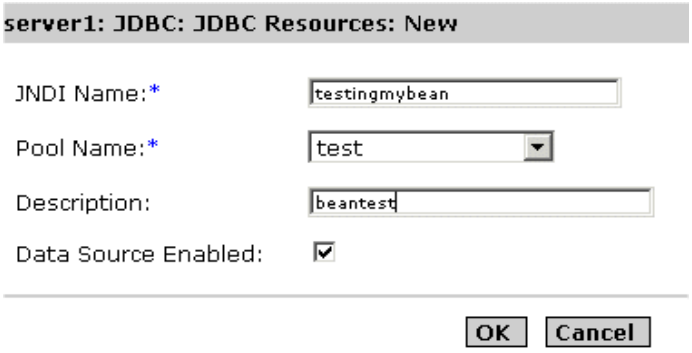
注意，此片段所指定的不过是应用程序用来引用其所代表的数据源 （从 J2EE 内部）的符号名。 pool-name 属性指向命名池定义，该定义将指定数据库连接的各个方面。管理员可以使用已启用属性关闭一些资源。

使用管理界面注册资源

使用管理界面注册资源的步骤：

1. 在管理界面的左侧窗格中，打开要为其注册 JDBC 资源的 Sun ONE Application Server 实例。
2. 打开 “JDBC”。
3. 在 “JDBC” 下，单击 “JDBC Resource”。
4. 在右侧窗格中，单击 “New”。右侧窗格中出现用于创建新的 JDBC 资源的页面，如图 “创建新的 JDBC 资源” 所示。

创建新的 JDBC 资源



5. 为创建的资源输入 JNDI 名称。

   JDBC 资源存储在 JNDI 系统信息库中，且可以通过 JNDI 名称进行访问。JNDI 名称包含明确的根 （位于 Java:comp:env/），因此不需要指定该部分名称。建议将 JDBC 资源 （数据源）存储在 “jdbc” 子上下文之下，因而您的 JNDI 名称将类似于 jdbc/EmployeeDB_DS。

6. 从 “Pool Name” 下拉列表的池名列表中，为新的数据源选择池名。所有已注册的连接池都将显示在此下拉列表中。选择的池名将指向命名池定义，该定义将指定数据库连接的各个方面。多个 JDBC 资源可以使用单个池定义。有关如何配置 JDBC 连接池的详细信息，请参见第 247 页上的 “使用管理界面创建新的 JDBC 连接池”。

7. 在 “Description” 字段中，输入描述数据源用途的简短说明。说明不能超过 250 个字符。

8. 选中 “Enabled” 复选框，将启用或禁用数据源。数据源不能用于连接数据库，除非已被启用。

9. 单击 “OK” 注册新数据源，或者单击 “Cancel” 取消新数据源。单击取消后，将返回到 JDBC 资源主页面，从中可以再次创建新数据源。

关于 JDBC 连接

Connection 对象表示与数据库的连接。连接会话包括执行的 SQL 语句以及通过该连接返回的结果。一个应用程序可以与一个数据库具有一个或多个连接，该应用程序也可以与多个不同的数据库具有连接。

用户可以通过调用 Connection.getMetaData 方法，获得关于 Connection 对象的数据库的信息。此方法返回一个 DatabaseMetaData 对象，该对象包含数据库的表、数据库支持的 SQL 语法、数据库的存储过程、此连接的功能等方面的信息。

应用程序使用 DataSource 对象生成的 Connection 对象。应用程序通常应包括“finally”块，以确保即使抛出异常也能关闭连接。如果 Connection 对象是加入的连接，这更为重要，因为它可确保始终将有效连接放回到可用连接池之中。下面的代码片段 （其中 con 为 Connection 对象）是关闭有效连接的 finally 块的示例。

```
finally{
    if (con != null) con.close();
}
```

注意，如下面的示例所示， try/catch 块后出现一个 finally 块，其中 ds 是 DataSource 对象。

```
try {
    Connection con = ds.getConnection("user", "secret");
    // . . . 执行应用程序的工作的代码
} catch {
    // . . . 处理 SQLException 的代码
} finally {
    if (con != null) con.close();
}
```

本节包括以下主题：

- 关于 JDBC URL
- 配置 JDBC 连接池
- 关于连接加入
- 监视 JDBC 连接加入
- 关于连接共享

关于 JDBC URL

URL （统一资源地址）提供了在 Internet 上定位资源的信息。它可以被视为一种地址。

JDBC URL 提供了一种标识数据源的方法，以便相应的驱动程序识别该数据源并与其建立连接。驱动程序编写者是确定标识特定驱动程序的 JDBC URL 将会是什么的人员。用户不需要关心如何构成 JDBC URL ；他们只需使用随所用驱动程序一起提供的 URL。 JDBC 旨在推荐一些惯例，以便驱动程序编写者在构造 JDBC URL 时进行遵循。

由于 JDBC URL 用于各种驱动程序，因而这些惯例是 （也必须是）非常灵活的。第一，它们允许不同的驱动程序在命名数据库时可以使用不同的模式。例如， ODBC 子协议允许 URL 包含属性值 （但并不需要属性值）。

第二， JDBC URL 允许驱动程序编写者对其中包含的所需的所有连接信息进行编码。这样，一个要与指定数据库进行通信的小程序就能够打开数据库连接，而无须用户执行任何系统管理事务。

第三， JDBC URL 允许有间接级。也就是说， JDBC URL 可能会指向由网络命名系统动态转换为实际名称的逻辑主机或数据库名称。这样，系统管理员则无需指定 JDBC 名称中的特定主机部分。有多种不同的网络名称服务，而且没有任何关于使用者的限制。

下面是 JDBC URL 的标准语法。此语法包含三个部分，它们用冒号隔开。

jdbc:<*subprotocol*>:<*subname*>

JDBC URL 的三个部分分别说明如下：

- jdbc - 协议：

  JDBC URL 中的协议始终是 jdbc。

- <subprotocol>

  驱动程序名称或数据库连接机制名称，可由一个或多个驱动程序支持。子协议名称的突出示例是 ODBC， ODBC 已经被保留用于指定 ODBC 格式的数据源名称的 URL。例如，要通过 JDBC-ODBC 桥访问数据库，用户可能使用一个像 jdbc:odbc:fred 这样的 URL。

  在此示例中，子协议为 ODBC，而子名称 fred 是本地 ODBC 数据源。

  如果想要使用网络名称服务 （这样， JDBC URL 中的数据库名称就不必是其实际名称），则该命名服务就可以是子协议。例如，某个用户可能有一个如下所示的 URL：

  jdbc:dcenaming:accounts-payable

在此示例中， URL 指定本地 DCE 命名服务应将数据库名称应付帐户解析为一个可用于连接真实的数据库的、更为具体的名称。

- <subname>:

  一种标识数据源的方法。根据子协议，子名称可以变化，而且它可以具有驱动程序编写者选择的任何内部语法，包括 sub-subname。 subname 的核心是给出用于定位数据源的足够信息。在前一个示例中， fred 就已足够，因为 ODBC 提供了其余的信息。但是，远程服务器上的数据源需要更多信息。例如，如果通过 Internet 访问数据源，就应把网络地址作为 subname 的组成部分包括在 JDBC URL 之中，并应遵守以下标准 URL 命名惯例：

  //hostname:port/subsubname

  假如 dbnet 是一个连接到 Internet 上某个主机的协议，则 JDBC URL 可能看起来像下面的样子：

  jdbc:*dbnet://wombat:356/fred*

配置 JDBC 连接池

Sun ONE Application Server 允许用户创建命名 JDBC 连接池。JDBC 连接池定义用于创建连接池的属性。池定义是命名的，并且一个定义可以重复使用来配置多个 JDBC 资源。每个命名的池定义都会导致服务器启动时实例化一个物理池。如果两个或更多的 JDBC 资源指向同一池定义，这些资源就会在运行时使用同样的连接池。

您可以使用管理界面和命令行界面创建和配置连接池，如以下各节所述：

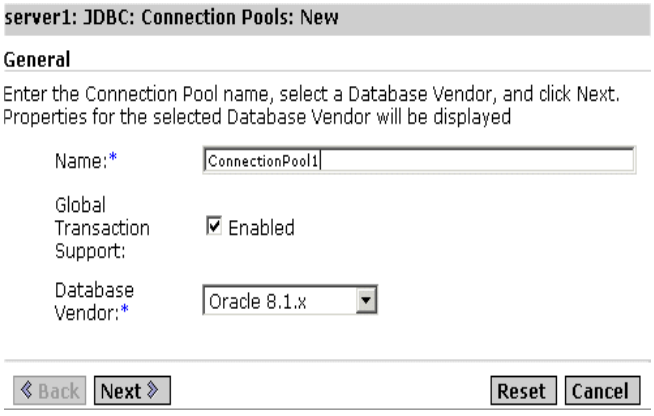
- 使用管理界面创建新的 JDBC 连接池
- 使用命令行界面创建新的 JDBC 连接池
- 使用命令行界面管理 JDBC 连接池

使用管理界面创建新的 JDBC 连接池

使用管理界面创建新的 JDBC 连接池的步骤：

1. 在管理界面的左侧窗格中，打开要为其创建 JDBC 连接池的 Sun ONE Application Server 实例。
2. 从 Sun ONE Application Server 下列出的 J2EE 服务列表中选择 “JDBC”，并打开它下面的 “ConnectionPools”标签。您会在管理界面的右侧窗格中看到图 “创建新的 JDBC 连接池”。

创建新的 JDBC 连接池



3. 在 “Name”字段中，为创建的连接池输入 JNDI 名称。
4. 选中 “Global Transaction Support Enabled”复选框，将对新的连接池启用全局事务支持。能够参加全局事务的连接池称为具有 XA 能力的连接池。
5. 从 “Database Vendor”下拉列表中选择一个数据库供应商，并单击 “Next”。您需要在随后显示的屏幕中配置连接池设置。

配置连接池设置

要配置连接池设置，请执行第 247 页上的 “使用管理界面创建新的 JDBC 连接池”中的第 1 步到第 5 步。单击 “Next”（如第 5 步中所述）后，管理界面的右侧窗格中会出现一个新的页面。它包含以下部分：

- 常规
- 属性
- 池设置
- 连接验证
- 事务隔离

在此页的 “General”部分中，根据下表中的说明为提供的参数指定值：

常规设置

参数	说明
Name	连接池的名称。
DataSource ClassName	实现 DataSource 和/或 XADataSource API 的供应商特定类名。
Description	连接池的说明。

在此页的 “Properties”部分中，您指定标准和专用 JDBC 连接池属性；其中许多属性为可选。默认情况下，提供所有标准属性的名称。您需要参考数据库供应商的文档，以确定必需的标准特定的属性和供应商特定的属性。

在此窗口的 “Pool Settings”部分中，根据下表中的说明为提供的参数指定值：

连接池设置

参数	说明
Steady Pool Size	指定池中必须维持的最小连接数目。连接提供给请求的线程之后，该连接将从池中删除，从而减小了当前的池大小。稳定池大小还与服务器启动时添加到池中的条目数目有关。
Max Pool Size	指定该池在任何指定时间中允许的最大连接数目。
Pool Resize Quantity	当池向稳定池大小方向收缩时，将成批调整大小。此值确定批处理的大小。将此值设置过大会延迟连接回收，而将值设置过小则会导致效率太低。注意，池容量每次只增加一个连接，因此该字段不影响池容量的增加。
Idle Timeout (secs)	连接在池中保持空闲的最长时间 （以秒为单位）。超过此时间后，池实现可以关闭此连接。
Max Wait tim	达到连接超时前呼叫者等待的时间。默认的等待时间很长，即呼叫者可以等待很长时间。

在此窗口的 “Connection Validation”和 “Transaction Isolation”部分中，根据下表中的说明为连接池选择验证方法和事务隔离方法：

连接验证和事务隔离

参数	说明
Connection Validation Required	如果选中此字段，则连接在传递到应用程序前将被验证。这样，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，应用程序服务器将自动重新建立数据库连接。连接验证将引起额外负担，并会导致性能稍有下降。
Validation Method	有三种验证数据库连接的方法可供应用程序服务器使用；您需要了解您的数据库的能力，才能确定适当的方法。这三种验证方法为： • 自动提交、元数据 — con.getAutoCommit() 和 con.getMetaData() 方法常用来验证连接，遗憾的是，许多 JDBC 驱动程序将这些调用的结果高速缓存起来，因而并不总是可以提供可靠的验证。您应与供应商进行核实，以确定这些调用是否被高速缓存。 • table：此方法要求应用程序服务器基于用户指定的表格执行查询。实际的查询是 “select (count *) from <table-name>”。表必须存在并且可以访问，尽管该表不要求输入任何行的内容。不应使用包含许多行的现有表或经常访问的表。
Table Name	如果从 “Validation Method”下拉列表中选择了最后一种验证选项 “table”，则在此处指定表名。
Fail All Connections	选中此复选框，将使池中所有连接都失败，并在确定某个连接已失败时重新创建这些连接。如果未选中此复选框，则连接仅在被使用时才单独重新创建。
Transaction Isolation	允许您为此连接选择事务隔离级别。如果未选中该项，池将使用 JDBC 驱动程序提供的默认隔离级别进行操作。
Guarantee Isolation Level	该项仅在指定了隔离级别的情况下才适用。这可确保从池中获取的任何连接都将具有同样的隔离级别。例如，如果连接在上次使用时，它的隔离级别通过编程方式被更改 （例如，con.setTransactionIsolation），则此机制就会将该连接的隔离级别更改回指定的隔离级别。

使用命令行界面创建新的 JDBC 连接池

本节通过示例讲述如何使用命令行界面创建 JDBC 连接池。

下表列出创建连接池时需要的所有选项，例如服务器名、密码。下表中使用了样例值。建议在运行本节中介绍的命令之前，设置好 Sun ONE Application Server 的安装所特有的参数。

使用命令行界面创建 JDBC 连接池时所必需的选项

必需选项说明	**样例值**
Application server admin user name	*admin*
Application server admin password	*adminadmin*
Application server admin port	*8888*
Application server machine name	*sas.sun.com*
Application server instance name	*server1*
Datasource classname for the connection pool	*oracle.jdbc.xa.client.OracleXADataSource* 注意：使用为其创建连接池的数据库的数据源类名。此实例中使用的数据库为 Oracle。
Jdbc resource description Sample	*Jdbc Resource*
Connection pool description Sample	*Jdbc Connection Pool*
Jdbc resource name	*jdbc/SampleJdbcResource*
Connection pool name	*SampleJdbcConnectionPool*
Database user name	*oracle*
Database password	*oracle*
Jdbc connection URL	*jdbc:oracle:thin:@oracleserver.sun.com:1521:ORA*

以下示例使用了表 “使用命令行界面创建 JDBC 连接池时所必需的选项”中列出的变量。

示例 1：

此示例创建了一个称作 SampleJdbcConnectionPool 的 JDBC 连接池。此示例中使用了一个两步进程创建 JDBC 连接池，如下所示：

- 步骤 1 — 创建连接池
- 步骤 2 — 将更改应用到实例

步骤 1 — 创建连接池

下面是用于创建 JDBC 连接池的命令行界面语法：

asadmin create-jdbc-connection-pool --user *admin_user* [--password *admin_password*] [--host *localhost*] [--port *4848*] [--secure | -s] [--instance *instancename*] --datasourceclassname *classname* [--restype *res_type*] [--steadypoolsize *8*] [--maxpoolsize *32*] [--maxwait *60000*] [--poolresize *2*] [--idletimeout *300*] [--isolationlevel *isolation_level*] [--isisolationguaranteed] [--isconnectvalidatereq=false] [--validationmethod *auto-commit*] [--validationtable *tablename*] [--failconnection=false] [--description *text*] [--property (name=value)[:name=value]*] *connectionpool_id*

例如，以下命令将创建一个称作 SampleJdbcConnectionPool 的连接池。

asadmin create-jdbc-connection-pool --user admin --password adminadmin --host sas.sun.com --port 8888 --instance server1 --restype javax.sql.XADataSource --datasourceclassname oracle.jdbc.xa.client.OracleXADataSource --description "Sample Jdbc Connection Pool" --property User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\:ORA" SampleJdbcConnectionPool

注意 如果要为新的连接池启用 “Global Transaction Support”，设置 --restype javax.sql.XADataSource。在 URL 属性中，用 (\:) 替换冒号 (:)

成功创建 JDBC 连接池后，您会看到以下消息：

Created the JDBC connection pool resource with id = SampleJdbcConnectionPool

步骤 2 — 将更改应用到实例

既然已成功创建了 JDBC 连接池，就需要将更改应用到 Sun ONE Application Server 的当前实例。

下面是将更改应用到 Sun ONE Application Server 的实例的语法。

asadmin reconfig --user *admin_user* [--password *admin_password*] [--host *localhost*] [--port *adminport*] [--secure | -s] [--discardmanualchanges=false|--keepmanualchanges=false] *instancename*

例如，以下命令将更改应用到 *server1*，即 Sun ONE Application Server 的实例。

asadmin reconfig --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888 server1*

将更改应用到 Sun ONE Application Server 的实例之后，您就会看到以下消息。

Successfully reconfigured

使用命令行界面管理 JDBC 连接池

可以使用命令行界面来管理 JDBC 连接池及其属性，如本节中所述：

列出连接池。以下命令列出为 *server1* （即，步骤 2 中使用的 Sun ONE Application Server 的实例）创建的所有连接池。

asadmin list-jdbc-connection-pools --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888 server1*

更改 JDBC 连接池属性。您可以更改 JDBC 连接池的属性 （例如， maxPoolSize 属性），步骤如下：

1. 运行以下命令，获取为 JDBC 连接池属性 maxPoolSize 指定的值。

   asadmin get -u *admin* -w adminadmin -H *sas.sun.com* -p *8888 server1.jdbc-connection-pool.SampleJdbcConnectionPool*.maxPoolSize

   运行此命令时，您会看到以下结果：

   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize = 32

   通过运行以下命令将 MaxPoolSize 的值更改为 80：

   asadmin set -u *admin* -w *adminadmin* -H *sas.sun.com* -p *8888 server1.jdbc-connection-pool.SampleJdbcConnectionPool*.maxPoolSize="*80*"

   按照说明的方法指定了值之后，您就会看到以下消息：

   Attribute maxPoolSize set to 80

2. 使用以下命令将更改应用到 Sun ONE Application Server 的实例：

   asadmin reconfig --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888 server1*

更改用户属性：在下面的代码样例片段中，您可以将属性 “User”从 oracle 更改为 System。

asadmin create-jdbc-connection-pool --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888* --instance *server1* --restype *javax.sql.XADataSource* --datasourceclassname *oracle.jdbc.xa.client.OracleXADataSource* --description "*Sample Jdbc Connection Pool*" --property User="*oracle*":Password="*oracle*":URL="*jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\:ORA*" *SampleJdbcConnectionPool*

1. 运行以下命令更改 User 属性。

   asadmin set -u *admin* -w *adminadmin* -H *sas.sun.com* -p *8888 server1.jdbc-connection-pool.SampleJdbcConnectionPool*.property.User="*System*"

   用户名称已从 *Oracle* 更改为 *System*。

2. 更改用户名之后，运行以下命令来应用您的更改：

   asadmin reconfig --user *admin* --password *adminadmin* --host sas.sun.com --port *8888* *server1*

创建一个称作 SampleJdbcResource 的 JDBC 资源。您可以创建 JDBC 资源，详细说明如下。下面是创建 JDBC 资源的语法：

asadmin create-jdbc-resource --user *admin_user* [--password *admin_password*] [--host *localhost*] [--port *4848*] [--secure | -s] [--instance *instancename*] --connectionpoolid *id* [--enabled=*true*] [--*description text*] [--property (name=value)[:name=value]*] *jndiname*

1. 运行以下命令，将创建一个称作 SampleJdbcResource 的 JDBC 资源。

   asadmin create-jdbc-resource --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888* --instance *server1* --description "*Sample Jdbc Resource*" --connectionpoolid SampleJdbcConnectionPool jdbc/SampleJdbcResource

   运行此命令时，将创建 JDBC 资源，而且您会看到以下消息：

   Created the external JDBC resource with jndiname = jdbc/SampleJdbcResource

2. 下面，您需要运行以下命令将更改应用到 Sun ONE Application Server 的实例。

   asadmin reconfig --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888* *server1*

3. 运行以下命令，将在实例 *server1* 中列出所有的 JDBC 资源。

   asadmin list-jdbc-resources --user *admin* --password *adminadmin* --host *sas.sun.com* --port *8888 server1*

关于连接加入

应用程序可以通过使用 JNDI 查找数据源来获取连接。以下的代码样例片段可用于完成该任务：

```
InitialContext ctx  = new InitialContext();

DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/employee_ds");
```

获得 DataSource 之后，应用程序组件可根据 J2EE 部署描述符的 <res-auth> 元素中设置的值，用两种方法获得连接。如果此元素的值为 Container，则应用程序可以使用 ds.getConnection() 方法获得连接 （即无需指定任何登录信息）。否则，应用程序必须提供登录信息，才能从像 ds.getConnecion.(*userName, password*) 这样的资源管理器中获得连接。

getConnection() 的所有请求都将从一个池中获得满足。 JDBC 连接池将根据 server.xml 中描述的一组参数进行创建。该连接池创建时，池中包含了最初可用的连接数目。因此，池中的当前可用的连接可以满足 ds.getConnection() 请求的需要。下一个请求 （如果任何以前的连接都没有被返回到池）将会发现池是空的，并会导致创建新的连接，但不会超出该池指定的最大连接数目的限制。池实现将跟踪创建的连接的数目。如果 getConnection() 请求发现池是空的，或者已创建的连接的数目等于池中最大连接数目，则当前请求将不予满足。此类情况仅会在不可能进行连接共享时发生，而且这种状况会一直保持下去，直到一个连接返回到池中。

当服务器仍然运行时，即使数据库崩溃并又恢复，连接池也将继续正常工作。只有在启用连接验证的情况下，才能实现这一点，如第 248 页上的 “配置连接池设置” 所述。

根据您从 “Validation Type” 下拉列表中选择的值，池实现程序执行以下参数：

- 如果选择自动提交作为连接验证类型，系统将通过执行 conn.getAutoCommit() 方法检查连接是否有效。如果该方法不抛出 SQLException，则该连接被视为处于有效状态。自动提交是此参数的默认选项。
- 如果选择了元数据作为连接验证类型，系统将执行 conn.getMetaData() 方法，检查连接上是否存在元数据，而且如果该方法没有抛出 SQLException，则系统将认为该连接有效。
- 如果选择表作为连接验证类型，系统将执行查询 “Select * From <table-name>”。如果此调用不抛出 SQLException，系统将认为该连接有效。

如果启用 fail-all-connections 属性，则池中的任何连接都将被发现处于无效状态，从而所有连接都将关闭并重新建立。否则，失效和重新建立将不会轻易发生，而仅在使用单个连接时发生。

池实现还可以回收池中所有可用的连接。因此，如果连接处于空闲状态的时间超出了指定的空闲时间，它们将会关闭，从而使池大小达到稳定池大小。如果池极为空闲，则可能导致容器不得不重新建立无效连接，并始终使池中具有稳定的可用连接数目。您在确定是否相对于最大池大小设置稳定池大小时，必须将这一点牢记在心。

监视 JDBC 连接加入

您可能需要定期监视池操作，以确定池大小配置是否有效地工作。下表列出了所有可监视的 JDBC 连接加入参数。

注意，可能会在未来的版本中改进启用监视的工作方式以及可以监视的属性。

用于监视的 JDBC 连接池参数

属性名称	数据类型	说明
total-threads-waiting	整数	等待 JDBC 连接的线程的总数。

用于监视的 JDBC 连接池参数

属性名称	数据类型	说明
total-outbound-connections	整数	JDBC 连接验证失败的总数
total-connections-timed-out	整数	请求超时的连接的总数。

关于连接共享

当某个 J2EE 应用程序获得的多个连接使用同一资源管理器时，池实现将在同一事务范围内提供连接共享。可通过以下示例理解术语 “事务范围”：

Bean_A 启动事务 (Tx1) 并获得连接。然后， Bean_A 调用同一事务 (Tx1) 中的 Bean_B 中的一个方法。现在，如果 Bean_B 使用同样的登录信息从同一 DataSource 获取连接，则很明显，同一连接可以被共享，因为只有 Bean_A 将完成事务。另外，请注意只有在 J2EE 部署描述符中把资源共享范围设置为 Shareable 的情况下，该连接才能共享。如果不希望连接共享，则必须在部署描述符中将资源共享范围设置为 Unshareable。 Sun ONE Application Server 提供了连接共享，因为这样可以提高性能。

关于 JDBC 事务

一个事务包含一个或多个已执行、完成然后提交或回滚的语句。方法 commit 或 rollback 被调用时，当前事务将结束且另一个事务开始启动。

默认情况下，新的 Connection 对象通常处于自动提交模式，即语句完成时，将自动对该语句调用方法 commit。在此情况下，由于每个语句分别进行提交，因而一个事务仅包含一个语句。如果已禁用自动提交模式，则在明确调用 commit 或 rollback 方法之前，事务不会终止，因此，该事务将包括上次调用 commit 或 rollback 以来已经执行的所有语句。在第二个示例中，该事务中的所有语句将作为一个组进行提交或回滚。

方法 commit 使 SQL 语句对数据库所做的任何更改都成为永久更改，而且该方法同时释放该事务保持的任何锁定。方法 rollback 将放弃这些更改。

如果一个事务中有两个更新，有时您可能不希望一个更改在某个更新中生效，除非另一个更新也受到影响。这可通过禁用自动提交并将两个更新组合到一个事务中来实现。如果两个更新均成功，则 commit 方法将被调用，这样两个更新的效果将长久保持；如果一个更新失败或两个更新都失败，则 rollback 方法将被调用，从而还原为执行更新前的值。多数 JDBC 驱动程序支持事务。

javax.sql 软件包中的类和接口可以使 Connection 对象成为分布式事务 （即，一种涉及与多个 DBMS 服务器的连接的事务）的组成部分。必须由已经实现的 DataSource 对象生成 Connection 对象，才能使用中间层服务器的分布式事务基础结构。与 DriverManager 生成的 Connection 对象不同的是，默认情况下，由这样一个 DataSource 对象生成的 Connection 对象的自动提交模式将被禁用。另一方面，DataSource 对象的标准实现将生成与 DriverManager 类生成的 Connection 对象完全相同的对象。

当 Connection 对象是分布式事务的组成部分时，事务管理器确定何时对 Connection 对象调用 commit 或 rollback 方法。因此，Connection 对象加入分布式事务时，应用程序不应进行任何影响连接开始或结束的时间的操作，例如，调用方法 Connection.commit 或 Connection.rollback，或者关闭该连接的自动提交模式。这些操作会干扰事务管理器对分布式事务的处理。

关于 Java 邮件资源

JavaMail API 允许访问消息存储库中包含的电子邮件消息，并允许使用消息传输创建和发送电子邮件消息。JavaMail API 中包含了对 Internet 标准 MIME 消息的特定支持。访问消息存储库和传输是通过支持特定存储库和传输协议的协议提供商进行的。JavaMail API 规范不要求任何特定的协议提供商，但 JavaMail 包括一个 IMAP 消息存储库提供商和一个 SMTP 消息传输提供商。

JavaMail API 提供了一组定义构成邮件系统的对象的抽象类。 API 定义像 Message、 Store 和 Transport 这样的类。可以扩展 API，并可将其进一步分类，以提供新的协议并在必要时添加功能。另外， API 提供抽象类的具体子类。这些子类（包括 MimeMessage 和 MimeBodyPart）实现了广泛使用的 Internet 邮件协议。

JavaMail API 很大程度上汲取了 IMAP、MAPI、CMC、c-client 和其它邮件消息传送系统 API 的功能。 JavaMail API 支持许多不同的消息传送系统实现，例如，不同的消息存储库、不同的消息格式以及不同的消息传输。 JavaMail API 提供了一组定义客户机应用程序的 API 的基类和接口。开发者可以将 JavaMail 类进一步分类，以便提供特定消息传送系统的实现，例如 IMAP、 POP3 和 SMTP。

本节包括以下主题：

- 关于 JavaMail 消息处理进程
- 关于 JavaMail 体系结构组件
- 关于 JavaBeans Activation Framework (JAF)
- 关于 JavaMail 配置参数
- JavaMail 会话参考的 J2EE 部署描述符
- Sun ONE Application Server 部署描述符中的项

- 创建新的 JavaMail 会话
- 配置高级资源属性

关于 JavaMail 消息处理进程

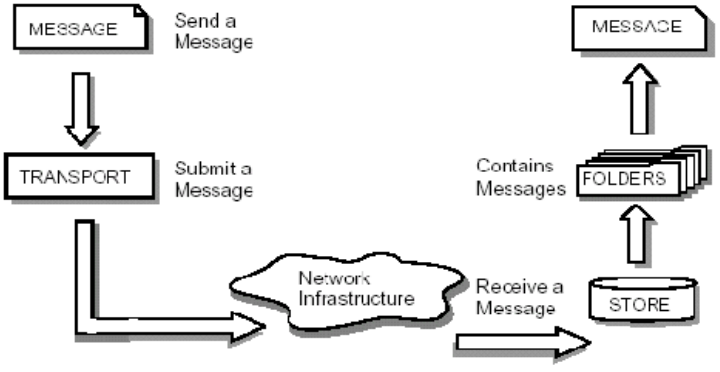
JavaMail API 执行以下功能，这些功能构成了典型客户机应用程序的标准邮件处理进程：

- 创建一个邮件消息，其中包含消息头属性集和具有 Content-Type 消息头字段中指定的某种已知数据类型的数据块。JavaMail 使用 Part 接口和 Message 类定义邮件消息。它使用 JAF 定义的 DataHandler 对象以包含该消息中的数据。
- 创建 Session 对象，它验证用户并控制对消息存储库和传输的访问。
- 将消息发送到其收件人列表。
- 从消息存储库检索消息。
- 对检索到的消息执行高级命令。像查看和打印这样的高级命令可通过 JAF-Aware JavaBean 实现。

注意 目前， JavaMail 框架没有定义支持消息传送、安全性、断开连接操作、目录服务或过滤器功能的机制。

下图说明了 JavaMail API 如何处理消息处理进程：

Java Mail API 的消息处理进程



JavaMail API 是通过使用静态工厂方法创建 javax.mail.Session 进行配置的。 Sun ONE Application Server 使用 JNDI 请求 Session 对象，并使用 resource-ref 元素在其部署描述符中列出对 Session 对象的需要。 JavaMail API Session 对象被视为资源工厂。

JavaMail API 中提供了消息传输，它可以处理类型为 javax.mail.internet.InternetAddress 的地址以及类型为 javax.mail.internet.MimeMessage 的消息。必须正确配置默认消息传输，才能使用 javax.mail.Transport 类的 send 方法发送这样的消息。

JavaMail API 的抽象层声明了旨在支持所有邮件系统支持的邮件处理功能的类、接口和抽象方法。包含抽象层的 API 元素旨在根据需要进行进一步分类以及扩展，以便支持标准数据类型，并且旨在根据需要与消息访问和消息传输协议进行连接。

Internet 实现层使用 Internet 标准 （RFC822 和 MIME）实现部分抽象层。

关于 JavaMail 体系结构组件

本节介绍了包含 JavaMail 体系结构的主要组件，其中包括以下主题：

- Message 类
- 消息存储和检索
- 消息撰写和传输

Message 类

Message 类是一种定义一组属性和邮件消息内容的抽象类。 Message 类的属性指定寻址信息，并定义内容的结构，其中包括内容类型。内容被表示为封装实际数据的 DataHandler 对象。

Message 类实现 Part 接口。 Part 接口定义了要定义并格式化 Message 对象带有的数据内容时，以及要成功地与邮件系统进行连接时所需的属性。 Message 类通过消息传输系统添加消息路由所必需的 “From”、“To”、“Subject”、“Reply-To” 以及其它属性。 Message 对象包含在一个文件夹时具有一组与其关联的标志。 JavaMail 提供 Message 子类，它支持特定消息传送实现。

消息内容是一个字节集合，或一个字节集合的引用，它们封装在 Message 对象内。 JavaMail 不知道消息内容的数据类型和格式。 Message 对象通过中间层 JavaBeans Activation Framework (JAF) 与其内容进行交互。此分隔层使 Message 对象可以通过调用相同的 API 方法，处理任何任意内容并使用任何适当的传输协议传输这些内容。消息收件人通常知道内容数据类型和格式，并且知道如何处理该内容。

JavaMail API 还支持多部分的 Message 对象，其中，每个 Bodypart 定义其自己的属性集和内容。

消息存储和检索

消息存储在 Folder 对象中。 Folder 对象可包含子对象以及消息，因而提供了一种树状文件夹分层结构。 Folder 类声明获取、附加、复制和删除消息的方法。 Folder 对象还可以将事件发送到注册为事件监听器的组件。

Store 类

Store 类定义一个数据库它将文件夹分层结构与其消息结合在一起。 Store 类还指定访问文件夹和检索文件夹中存储的消息的访问协议。Store 类还提供建立与数据库的连接、获取文件夹以及关闭连接的方法。实现消息访问协议 （IMAP、 POP3 等）的服务提供商从对 Store 类进一步分类开始。用户通常通过连接特定的 Store 实现启动和邮件系统的会话。

消息撰写和传输

客户机通过实例化适当的 Message 子类创建新消息。它设置诸如收件人地址和主题这样的属性，并将内容插入 Message 对象。最后，它通过调用 Transport.send 方法发送 Message。 Transport 类模仿将消息路由到其目标地址的传输代理。此类提供了将消息发送到收件人列表的方法。对 Message 对象调用 Transport.send 方法会根据其目标地址识别相应的传输。

Session 类

Session 类定义全局属性和每用户邮件相关属性，这些属性定义了支持邮件的客户机与网络之间的接口。

JavaMail 系统组件使用 Session 对象设置或获取特定属性。Session 类还提供一种桌面应用程序可以共享的默认的验证会话对象。Session 类是一个最终具体类。无法对 Session 类进一步分类。 Session 类还充当实现特定访问和传输协议的 Store 和 Transport 对象的一个工厂。通过对 Session 对象调用适当的工厂方法，客户机可以获取支持特定协议的 Store 和 Transport 对象。

关于 JavaBeans Activation Framework (JAF)

JavaMail 使用 JavaBeans Activation Framework (JAF)，以便封装消息数据并处理用于与这些数据进行交互的命令。与消息数据交互应通过 JAF-Aware JavaBeans 进行，而 JavaMail API 没有提供与消息数据交互功能。

通过使用 JavaBeans Activation Framework 标准扩展，使用 Java 技术的开发者可以利用标准服务确定任意数据段的类型、封装对于该数据段的访问、发现该数据段上可用的操作以及实例化执行上述操作的适当 Bean。例如，如果浏览器获得一个 JPEG 图像，则此框架使浏览器可以将数据流识别为 JPEG 图像，而且根据该数据类型，浏览器可以查找和实例化一个可以操作此 JPEG 图像的对象，或可以查看该图像。

JavaBeans Activation Framework API 支持各种 MIME 数据类型。JavaMail API 必须为和下表所示的 Java 编程语言类型对应的下列 MIME 数据类型，将 javax.activation.DataContentHandlers 包括在内。

JavaMail API MIME 数据类型到 Java 类型的映射

MIME 类型	Java 类型
Text/Plain	java.lang.String
Multipart/	javax.mail.internet.MIME.Multipart
Message/rfc822	javax.mail.internet.MIME.Message

JavaBeans Activation Framework 将对 MIME 数据类型的支持集成到 Java 平台之中。可以使用 avax.activation.DataContentHandlerobjects 将 MIME 字节流和 Java 编程语言对象相互转换。可以指定 JavaBeans 组件在 MIME 数据上进行操作，例如查看或编辑数据。 JavaBeans Activation Framework 还提供一种将文件扩展名映射到 MIME 类型的机制。JavaMail API 使用 JavaBeans Activation Framework 处理电子邮件消息中包括的数据。通常情况下， J2EE 应用程序将不需要直接使用 JavaBeans Activation Framework，然而复杂地使用电子邮件的应用程序可能需要使用此框架。

关于 JavaMail 配置参数

以下配置参数由 Sun ONE Application Server 中的 JavaMail 资源使用。这些配置参数为名称、值对，它们将从 server.xml 文件的 mail-resource 元素中读取。

- JNDI Name

  JNDI Name 指定用来从 J2EE 应用程序中参考此邮件资源的名称。

- Enabled

  enabled 配置参数指定此邮件资源是否将在 JNDI 树中发布，以及是否可以被参考。如果 J2EE 应用程序参考一项已禁用的资源，该应用程序将会收到 NameNotFoundException 异常。

- store-protocol

  指定默认的消息访问协议。 Session.getStore() 方法返回一个实现此协议的 Store 对象。客户机可以覆盖此属性，并用 Session.getStore(String protocol) 方法明确地指定该协议。

- store-protocol class

  指定实现上面指定的存储协议的类名。此类的默认值为 com.sun.mail.imap.IMAPStore。

- transport-protocol

  指定默认的传输协议。 Session.getTransport() 方法返回一个实现此协议的 Tranport 对象。客户机可以覆盖此属性，并用 Session.getTransport(String protocol) 方法明确地指定该协议。

- transport-protocol class

  指定实现上面指定的传输协议的类名。此类的默认值为 com.sun.mail.smtp.SMTPTransport。

- host

  指定默认的邮件服务器。 Store 和 Transport 对象的连接方法使用此属性来定位目标主机 （如果协议专用的主机属性不存在）。

- user

  指定连接到邮件服务器时要提供的用户名。 Store 和 Transport 对象的连接方法使用此属性来获取用户名 （如果协议专用的用户名不存在）。

- from

  指定当前用户的返回地址。由 InternetAddress.getLocalAddress 方法用来指定当前用户的电子邮件地址。

- debug

  指定初始调试模式。将此属性设置为 True 将会打开调试模式，而将此其设置为 False 则会关闭调试模式。

- mail-<protocol>-host

  指定协议专用的默认邮件服务器。这会覆盖 mail.host 属性。此属性可以根据 store-protocol 属性的值进行设置。如果 store-protocol 的值为 imap 或 POP，您需要分别添加名称为 mail.imap.host 或 mail.pop3.host 的属性。特定属性的值应设置为适合邮件系统配置的值。例如，如果将 store-protocol 设置为 IMAP，property name:mail-imap-host 会带有以下值：spaceduck.acme.com。

- mail-<protocol>-user

  指定用来连接到邮件服务器的、协议专用的默认用户名。这会覆盖 mail.user 属性。因此，此属性根据 store-protocol 属性的值，可能是 mail.imap.user 或 mail.pop3.user。例如，如果将 store-protocol 设置为 IMAP， property name:mail-imap-user 会带有值 fredbloggs。

JavaMail 会话参考的 J2EE 部署描述符

JavaMail 资源使用服务器注册后，任何 J2EE 应用程序组件都可以使用 JNDI 查找对其进行参考。为了部署一个参考资源管理器连接工厂的应用程序，组件提供商必须在标准 J2EE 1.3 部署描述符中声明所有的资源管理器连接工厂参考。

JavaMail 参考的完整 J2EE1.3 描述符元素如下所示：

```
<resource-ref>
  <description>
    JavaMail resource used for sending my mail
  </description>
  <res-ref-name>mail/MyMailSession</res-ref-name>
  <res-type>javax.mail.Session</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Sun ONE Application Server 部署描述符中的项

对于每个参考邮件资源的已部署的组件，部署者必须将组件中使用的资源名称映射为 DataSource 使用命名服务注册的实际 jndi 名称。部署工具有助于部署者容易地进行此映射。此映射注册到 Sun ONE Application Server 特定的 xml 中。下面的示例片段显示了包含此映射的 Sun ONE Application Server 特定的 XML：

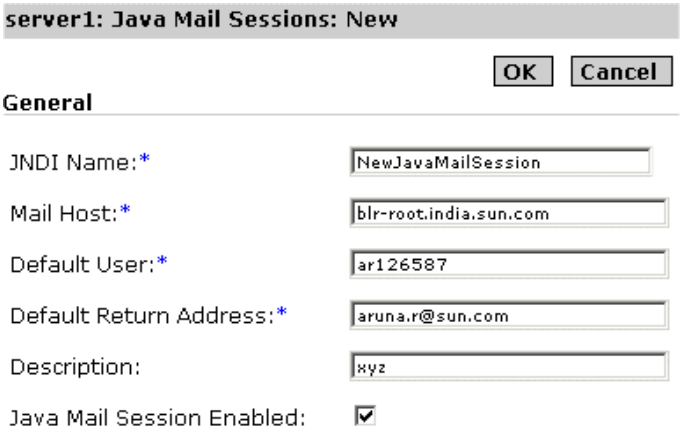
```
<resource-ref>
    <res-ref-name>mail/MyMailSession</res-ref-name>
    <jndi-name>mail/Session</jndi-name>
</resource-ref>
```

创建新的 JavaMail 会话

您可以使用管理界面配置 JavaMail 会话。创建和配置新的 JavaMail 会话的步骤：

1. 在管理界面的左侧窗格中，展开要为其创建新的 JavaMail 会话的 Sun ONE Application Server 实例。
2. 单击 “JavaMail Sessions”。您会在管理界面的右侧窗格中看到以下窗口，如图 “配置 JavaMail 会话”所示：

配置 JavaMail 会话



3. 在 “JNDI Name”文本字段中，为创建的 JavaMail 会话输入 JNDI 名称。Java 邮件资源使用服务器注册后，任何 J2EE 应用程序组件都可以使用 JNDI 查找对其进行参考。
4. 在 “Mail Host”文本字段中，指定默认邮件服务器的 DNS 名称。 Store 和 Transport 对象的连接方法使用此属性来定位目标主机 （如果特定协议的主机特性不存在）。
5. 在 “Default User”文本字段中，指定连接邮件服务器时要提供的用户名。 Store 和 Transport 对象的连接方法使用此属性获取用户名 （如果协议专用的用户名不存在）。
6. 在 “Default Return Address”字段中，指定当前用户的默认返回地址。默认地址应为以下格式：username@host。
7. 在 “Description”字段中，为此 Java 邮件会话输入说明。
8. 选中 “Java Mail Session Enabled”复选框，将启用所创建的 Java 邮件会话。

9. 单击 “OK”保存已配置的新的 JavaMail 会话。

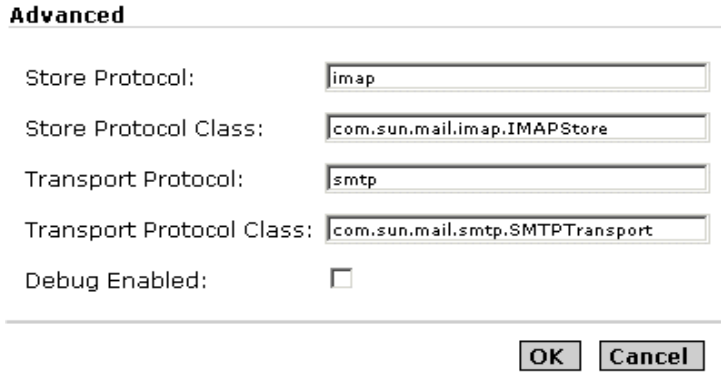
配置高级资源属性

您可以使用管理界面为新的 JavaMail 会话配置多个其它属性。属性名和值对取决于所用的邮件协议。您也可以在 server.xml 文件中直接指定这些属性。

配置其它属性的步骤：

1. 在管理界面的左侧窗格中，展开要修改其 JavaMail 会话配置的 Sun ONE Application Server 实例。
2. 单击 “JavaMail Sessions”。您会在管理界面的右侧窗格 （在 “创建新的 JavaMail 会话”中解释的主配置部分下面）中看到以下窗口，如图 “为 JavaMail 会话配置其它资源”所示：

为 JavaMail 会话配置其它资源



3. 在 “Store Protocol”文本字段中，指定要用于此 JavaMail 会话的存储协议，例如， POP3 或 IMAP。
4. 在 “Store Protocol Class”文本字段中，指定已指定的存储协议的类名，如上例所示。
5. 在 “Transport Protocol”文本字段中，指定要用于此 JavaMail 会话的传输协议，例如， SMTP。
6. 在 “Transport Protocol Class”文本字段中，指定已为此会话指定的传输协议的类名，如上例所示。

7. 选中 “Debug Enabled”复选框，将启用此 JavaMail 会话的调试。启用此复选框将打开调试模式。
8. 单击 “OK”保存其它属性配置。

下面代码段是一个完整的邮件资源配置示例：

```
<mail-resource
    jndi-name = "mail/Session"
    enabled = "true"
    store-protocol = "imap"
    store-protocol-class = "com.sun.mail.imap.IMAPStore"
    transport-protocol = "smtp"
    transport-protocol-class = "com.sun.mail.smtp.SMTPTransport"
    host = "gopostal.acme.com"
    user = "kingkong"
    from = "kingkong@acme.com"
    debug = "false">
    <property name = "mail-imap-host" value = "spaceduck.acme.com"/>
    <property name = "mail-imap-user" value = "fredbloggs"/>
</mail-resource>
```

第 11 章

使用 JMS 服务

Sun ONE Application Server 支持使用 Java 消息传送服务 (JMS) 应用程序编程接口 (API) 进行消息传送操作的应用程序。 JMS 是一组编程接口，为 Java 应用程序提供了在分布式环境中创建、发送、接收和阅读消息的常用方法。

JMS 尤其是基于标准的方法，通过这种方法 Java 2 Enterprise Edition (J2EE) 应用程序可以执行异步消息传送。因此， J2EE 组件 （Web 组件或企业 JavaBeans [EJB] 组件）可以使用 JMS API 发送由专用 EJB 异步使用的消息，称为消息驱动的 Bean (MDB)。

Sun ONE Application Server 通常都支持 JMS 消息传送，尤其支持 MDB，它需要消息传送中间件来实现 JMS 规范（即 JMS 提供商）。Sun ONE Application Server 将 Sun ONE Message Queue (MQ) 3.01 版用作本地 JMS 提供商。

MQ 紧密地集成到 Sun ONE Application Server 中，支持透明的 JMS 消息传送。该支持 （在 Sun ONE Application Server 中称为 *JMS 服务*）仅要求最小管理。

本章提供理解和管理内置 JMS 服务 （通过 Sun ONE Message Queue 提供）所需的信息，其中包括以下主题：

- 关于 JMS
- 内置 JMS 服务
- 管理内置 JMS 服务

关于 JMS

JMS 规范说明了一组支持分布式企业消息传送的编程接口。企业消息传送系统允许独立分布式组件或应用程序通过消息进行交互。不管这些组件位于同一系统、同一网络还是通过因特网松散连接，它们都使用消息来传递数据和协调各自的功能。

为了支持企业规模的消息传送， JMS 提供了可靠的异步消息传送。

可靠传送。从一个组件到另一组件的消息不会由于网络或系统出现故障而丢失。这意味着系统能够保证消息成功传送。

异步传送。对于可以同时交换信息、支持高密度吞吐量的许多组件，消息的发送不能取决于用户能否立即接收。如果用户处于繁忙或脱机状态，系统允许用户发送消息，并在其做好准备后进行接收。这就叫做异步消息传送，通常称为存储和转发消息。

本主题将简要介绍 JMS 的概念和术语：

- 消息传送系统的基本概念
- JMS 规范
- 消息驱动的 Bean

有关 JMS 的详细说明，请参见以下位置的 JMS 1.0.2 规范：

http://java.sun.com/products/jms/docs.html

消息传送系统的基本概念

企业消息系统 （尤其是 JMS）通常以其中的几个基本概念为基础。本主题中将讨论这些概念。

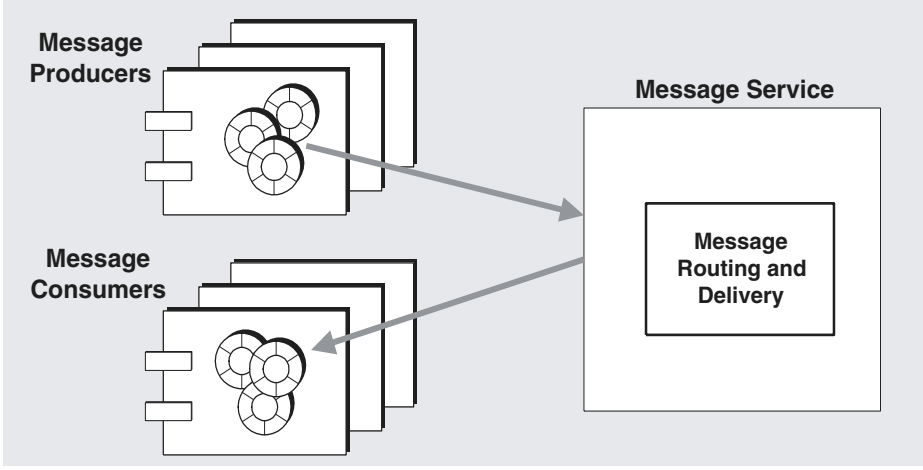
消息

消息由具有一定格式 （消息正文）的数据和说明消息特性或属性的元数据 （消息头）组成，例如由消息传送系统确定的目标、生命周期或其它特性。

消息服务体系结构

“消息服务体系结构”图中说明了消息传送系统的基本体系结构。消息传送系统包括消息生成方和消息用户，他们通过常见的消息服务交换信息。一般来说，同一消息传送组件中可以驻留任意数量的消息生成方和用户。消息生成方可以将消息传送给消息服务。然后，消息服务使用消息路由和消息传送组件将消息传送到一个或多个已注册表示对该消息感兴趣的消息用户。消息路由和消息传送组件负责保证将消息传送给所有感兴趣的用户。

消息服务体系结构



消息传送模式

生成方和用户之间存在一对一、一对多和多对多等许多关系。例如，发送消息的方式可能为：

- 一个生成方到一个用户
- 一个生成方到多个用户
- 多个生成方到一个用户
- 多个生成方到多个用户

这些关系通常可以简化为两种消息传送模式：*点对点*和*发布/订阅*模式的消息传送。点对点传送模式的关键是由特定生成方发送、并由特定用户接收的消息。发布/订阅传送模式的关键是由任意数量的生成方发送、并由任意数量的用户接收的消息。这些消息传送模式可以重复。

消息传送系统以前一直支持这两种模式的各种组合。 JMS API 旨在创建支持点对点传送模式和发布/订阅传送模型的通用编程方法。

JMS 规范

JMS 可以指定消息结构、编程模式以及一组管理消息传送操作的规则和语义。

JMS 消息结构

根据 JMS 规范，消息由三部分组成：消息头、属性 （可以视为消息头的扩展）和正文。

消息头。消息头可以指定消息的 JMS 特性：目标、是否具有永久性、生命周期和优先级。这些特性能够管理消息传送系统传送消息的方法。

属性。属性是可选的，提供应用程序可以根据各种选择条件来过滤消息的值。属性是可选的。

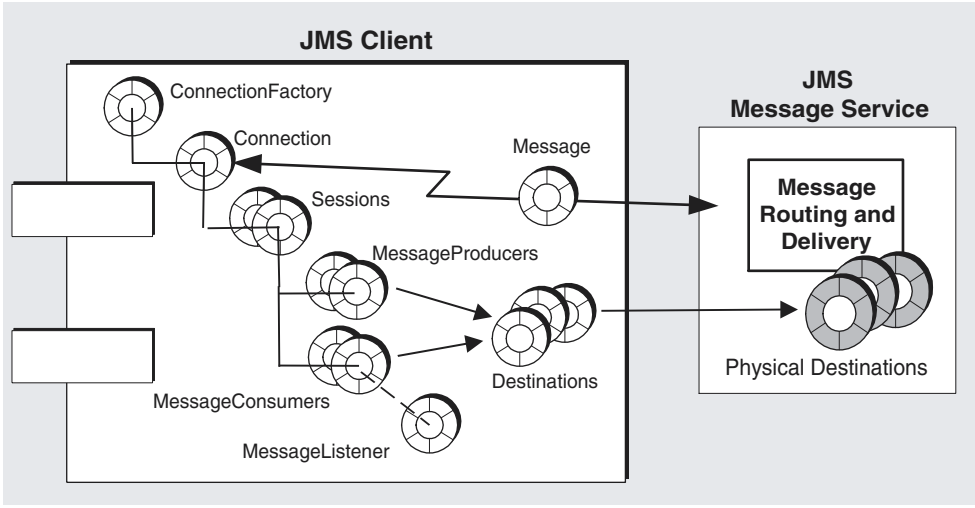
消息正文。消息正文包括要交换的实际数据。 JMS 支持六种正文类型。

JMS 编程模式

在 JMS 编程模式中， JMS 客户机 （组件或应用程序）通过 JMS 消息服务来交换消息。消息生成方将消息传送到消息服务，消息用户从消息服务接收消息。这些消息传送操作通过使用一组实现 LMS API 的对象（由 JMS 提供商提供）来执行。“JMS 编程对象”图显示了用于对消息传送进行编程的 JMS 对象。

在 JMS 编程模式中， JMS 客户机使用 ConnectionFactory 对象创建连接，可以通过该连接将消息传送到 JMS 消息服务或从 JMS 消息服务接收消息。 Connection 是一种 JMS 客户机与消息服务的活动连接。创建连接时会分配通信资源和验证客户机。

JMS 编程对象



连接可以用于创建会话。会话是生成和使用消息的单线程上下文，用于创建发送消息的生成方和接收消息的用户。会话支持通过很多确认选项或事务 （可由分布式事务管理器进行管理）实现的可靠传送。

JMS 客户机使用 MessageProducer 向指定的物理目标发送消息，该目标在 API 中表示为目标对象。消息生成方可以指定默认的发送模式 （永久性和非永久性消息）、优先级和生命周期，用于管理生成方发送到物理目标的所有信息。

同样， JMS 客户机使用 MessageConsumer 从指定物理目标接收消息，该目标在 API 中表示为目标对象。消息用户支持同步或异步使用消息。异步使用通过向用户注册一个 MessageListener 来实现。当会话线程调用 MessageListener 对象的 onMessage() 方法时，客户机将使用一条消息。

管理的对象：提供商独立性

第 270 页上的 “JMS 编程模式”图中说明的两个对象取决于 JMS 提供商实现 JMS 消息服务的具体方法。连接工厂对象取决于要发送消息的提供商所使用的基础协议和机制，目标对象则取决于提供商所使用物理目标的特定命名惯例和功能。

正常情况下，这些特定于提供商的特性会使 JMS 客户机代码依赖于 JMS API 实现的详细信息。但是，要使 JMS 客户机代码独立于提供商， JMS 规范要求以标准化的方法访问特定于提供商的对象 （称为管理的对象），而不是直接在客户机代码中实例化。

管理的对象封装了特定于提供商的实现和配置信息。这些对象由管理员创建和配置、存储在名称服务中、并由客户机应用程序通过标准 JNDI 查找代码来访问。通过这种方法使用管理的对象使 JMS 客户机代码独立于提供商。

JMS 提供了两种通用类型的管理的对象：连接工厂和目标。两类对象都封装了特定于提供商的信息，但是它们在 JMS 客户机中的用途不同。连接工厂用于创建到消息服务器的连接，而目标对象用于识别 JMS 消息服务所使用的物理目标。

注意 在 Sun ONE Application Server 上下文中， JMS 管理的对象被视为 JMS 资源，与其它应用程序服务器资源类似。

消息驱动的 Bean

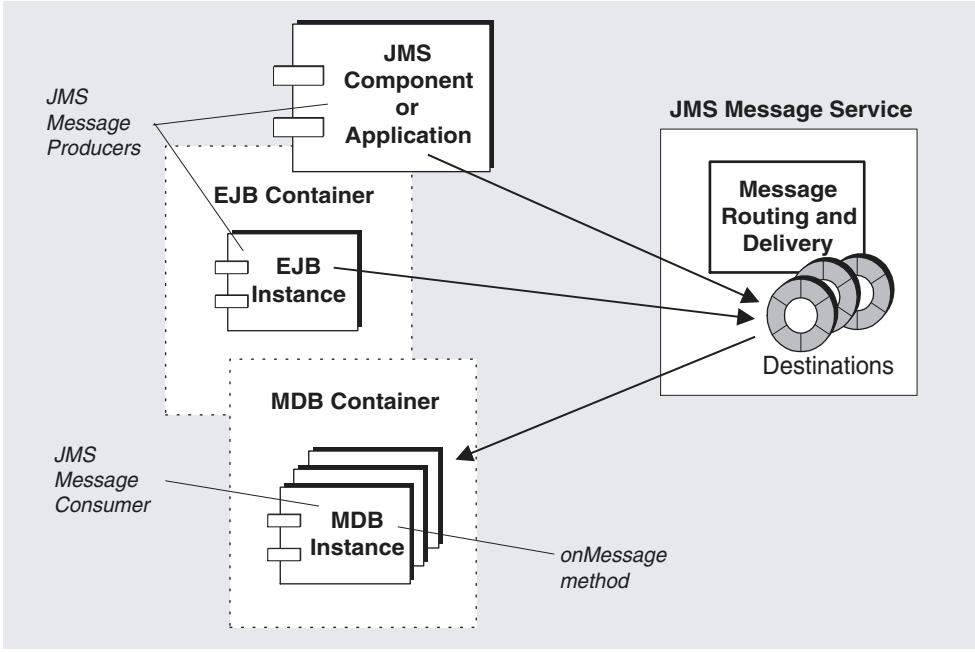
除了第 270 页上的“JMS 编程模式”图中介绍的通用 JMS 客户机编程模式外，J2EE 应用程序的上下文中还采用了更专业的 JMS API。专用 JMS 客户机称为 *消息驱动的 Bean*，它是在 EJB 2.0 规范 (http://java.sun.com/products/ejb/docs.html) 中指定的 EJB 组件系列中的一员。

由于其它 EJB 组件 （会话 Bean 和实体 Bean）只能同步调用，因此出现了消息驱动的 Bean。因此，当您调用这些 Bean 中的方法时，在方法完成之前可以一直中断资源。这些 EJB 组件没有异步接收消息的机制，因为只能通过标准 EJB 接口进行访问。

但是，许多企业应用程序需要异步消息传送。因此，需要 EJB 组件能够接收消息并使用这些消息，而无需紧密耦合到消息的生成方。

MDB 是专用 EJB 容器 （为所支持的组件提供分布式服务的软件环境）支持的专用 EJB 组件。

MDB 消息用户



消息驱动的 Bean。 MDB 是实现 JMS MessageListener 接口的 JMS 消息用户。 MDB 容器收到消息时将调用它的 onMessage 方法 （由 MDB 开发者编写）。与任何 JMS MessageListener 对象中的 onMessage 方法一样，onMessage 方法也使用消息。MDB 可以使用单个目标中的消息。消息可以由独立的 JMS 客户机应用程序、Web 组件或其它 EJB 组件生成，如图 “MDB 消息用户”所示。

MDB 容器 MDB 由专用 EJB 容器支持，它负责创建 MDB 实例，并为异步使用消息对这些实例进行设置。其中包括设置与消息服务的连接 （包括验证）、创建与给定目标相关联的会话池、以及在会话池与关联的 MDB 实例之间收到消息时管理消息的发布。由于容器控制 MDB 实例的生命周期，因此它可以管理 MDB 的实例池以便容纳传入的消息负载。

与 MDB 相关联的是部署描述符，它为设置消息使用时容器使用的管理的对象指定 JNDI 查找名称：连接工厂和目标。部署描述符也可以包括配置容器时部署工具使用的其它信息。每一个这类的容器都支持仅包含一个 MDB 的实例。

有关为 Sun ONE Application Server 配置 EJB 容器的 MDB 设置的信息，请参见第 191 页上的 “关于消息驱动的 Bean”。

内置 JMS 服务

一般情况下支持内置于 Sun ONE Application Server 中的 JMS 消息传送 （尤其是 MDB）。这种支持是通过紧密结合 Sun ONE Message Queue 和 Sun ONE Application Server 实现的，它提供本地、内置 JMS 服务。

本主题讲述了以下主题，以便于理解内置 JMS 服务：

- 关于 Sun ONE Message Queue (MQ)
- MQ 的集成 Sun ONE Application Server

有关管理内置 JMS 服务的信息，请参见第 281 页上的 “管理内置 JMS 服务”。

关于 Sun ONE Message Queue (MQ)

Sun ONE Message Queue (MQ) 是一种实现 JMS 开放标准的企业消息传送系统：它是 JMS 提供商。

MQ 产品所具有的功能超过了 JMS 规范中对于可靠、异步消息传送的最低要求。某些功能 （集中管理、可优化的性能、支持多种消息传输、用户验证和授权）可以从集成到 Sun ONE Application Server 的 MQ Platform Edition 中获得。其它功能（可伸缩的消息服务器和安全消息传送）可以通过升级到 MQ Enterprise Edition 获得。

MQ 消息传送系统包括许多部件 （如图第 275 页上的 “MQ 系统结构”所示），这些部件通过协同工作来提供可靠的消息传送。

MQ 消息传送系统的主要部件为：

- MQ 消息服务器
- MQ 客户机运行时
- MQ 管理的对象
- MQ 管理工具

以下主题中对这些部件进行了简要的介绍。有关 MQ 消息传送系统的完整说明，请参见以下站点中的 MQ *Administrator's Guide*：

http://docs.sun.com/

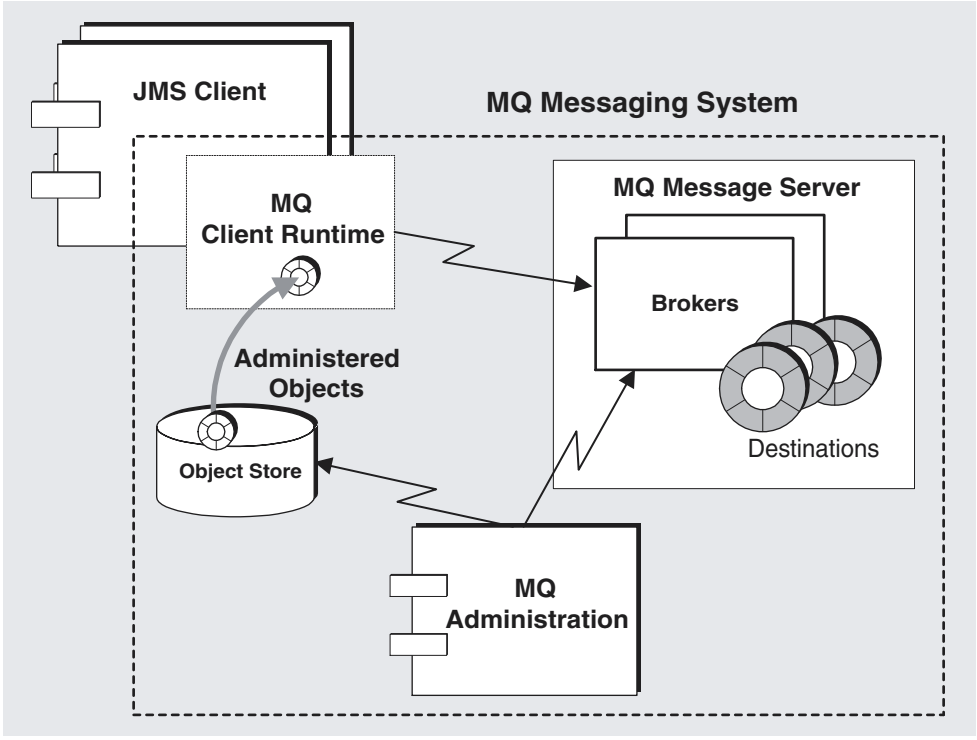
MQ 消息服务器

MQ 消息服务器的主要部件为代理和物理目标，如图第 275 页上的 “MQ 系统结构”所示。

代理。代理可以为 MQ 消息传送系统提供传送服务。消息传送通过各种支持组件来完成，这些组件用于处理连接服务、消息路由和传送、持久性、安全性及记录。消息服务可以部署一个或多个代理来实现可伸缩性。

物理目标。消息的传送分为两个阶段进行：首先，消息从生产方客户机传送到由代理维护的物理目标；然后，消息从目标传送到一个或多个用户客户机。物理目标表示代理的物理内存或永久存储器中的位置 （有关详细信息，请参见第 276 页上的 “物理目标”）。

MQ 系统结构



代理

MQ 消息传送系统中的消息传送 （从生成方客户机到目标，然后从目标到一个或多个用户客户机）是通过代理 （在 MQ 3.01 Enterprise Edition 中，则为串联工作的代理群集）执行的。要执行消息传送，代理必须配置客户机中的通信频道、执行验证和授权、正确地路由信息、保证传送的可靠性并提供用于监视系统性能的数据。

要执行这一系列的复杂功能，代理使用许多不同的组件，这些组件在传送过程中发挥各自特定的作用。可以根据负荷条件、应用程序的复杂程度等方面的因素来配置这些内部组件，以便优化代理的性能。有关详细信息，请参见 MQ *Administrator's Guide*。

物理目标

MQ 消息传送将分两个阶段进行：首先，消息从生成方客户机传送到代理上的目标；然后，消息从代理上的目标传送到一个或多个用户客户机。目标有两种类型：队列（点对点传送模式）和主题 （发布/订阅传送模式）。这些目标表示代理物理内存中的位置，传入的消息在路由到用户客户机之前在此处进行编排。

物理目标通常可以使用管理工具来创建，但也可以由代理根据收到的消息自动创建。

队列目标。队列目标用于点对点的消息传送，点对点发送表示消息只一次性地发送给已注册到该目标的用户组中的一个用户。生成方客户机的消息到达后，它们将排队并传送到用户客户机。

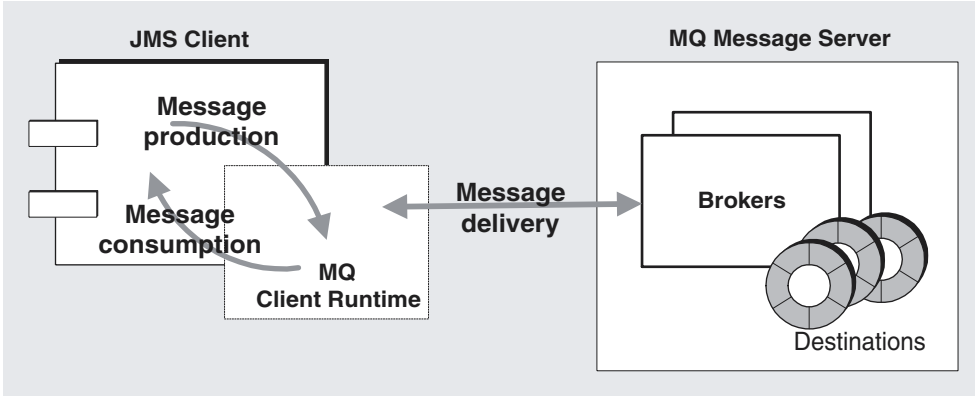
主题目标。主题目标用于发布 / 订阅消息传送，这种方式是将消息永久发送到注册到目标的所有用户。生成方的消息到达后，它们将路由到订阅主题的所有用户。如果用户注册了长期订阅主题，则不必在消息传送到主题时处于活动状态，代理将存储该消息，直到用户再次被激活，然后传送该消息。

MQ 客户机运行时

MQ 客户机运行时间为 JMS 客户机 （独立的应用程序、 Web 组件或 EJB 组件）提供与 MQ 消息服务器通信的接口，它向 JMS 客户机提供客户机将消息发送到目标和从这类目标接收消息时所需的所有编程接口实现。

第 277 页上的 “消息传送操作”图说明了消息的生成和使用如何在 JMS 客户机和 MQ 客户机运行时之间进行交互，以及消息传送如何在 MQ 客户机运行时和 MQ 消息服务器之间进行交互。

消息传送操作



MQ 管理的对象

MQ 管理的对象允许 JMS 客户机代码独立于提供商 （参见第 272 页上的 “管理的对象：提供商独立性”）。通过封装对象 （该对象可以通过独立于提供商的方式由客户机应用程序使用）中特定于提供商的实现以及配置信息，从而使 JMS 客户机代码独立于提供商。 MQ 管理对象由管理员创建和配置、存储在名称服务中、并由 JMS 客户机通过标准 JNDI 查找代码来访问。

连接工厂管理的对象。连接工厂对象用于创建 JMS 客户机 （独立应用程序、 Web 组件或 EJB 组件）与 MQ 消息服务器之间的物理连接。连接工厂对象在代理中没有物理表示，它仅用于启用 JMS 客户机，以建立与代理之间的连接。它还用于指定连接和使用该连接访问代理的客户机运行时的操作：因此， MQ 连接工厂具有大量可配置的属性，通过这些属性可以优化 MQ 系统的属性。

目标管理的对象。目标管理的对象 （队列或主题）表示代理中的物理目标，它与公开命名的目标管理的对象相对应。通过创建目标管理的对象，可以允许 JMS 客户机（消息用户和/或消息生成方）访问相应的物理目标。

MQ 管理工具

MQ 管理工具分为两类：命令行公用程序和图形用户界面 (GUI) 管理控制台。

管理控制台。可以使用管理控制台连接到代理并对其进行管理、在代理中创建物理目标、连接到对象存储、以及向对象存储添加或更新管理的对象或者从中删除管理的对象。有些任务不能通过管理控制台执行，主要包括启动代理、创建代理群集、配置更专业的代理属性、以及管理用户数据库。

命令行公用程序使用 MQ 公用程序可以执行所有通过管理控制台执行的任务，而且，还可以启动和管理代理、配置更专业的代理属性、以及管理 MQ 用户数据库。

MQ 的集成 Sun ONE Application Server

作为 Sun ONE Application Server 安装过程的一部分，将自动安装 MQ Platform Edition。有关详细信息，请参阅 《*Sun ONE Application Server Developer's Guide*》。

本安装为 Sun ONE Application Server 提供了一个 JMS 消息传送系统，支持任意数量的 Sun ONE Application Server 实例。默认情况下，每个服务器实例都有关联的内置 JMS 服务，这些服务支持所有 JMS 客户机在实例中运行。

本主题包括以下主题：

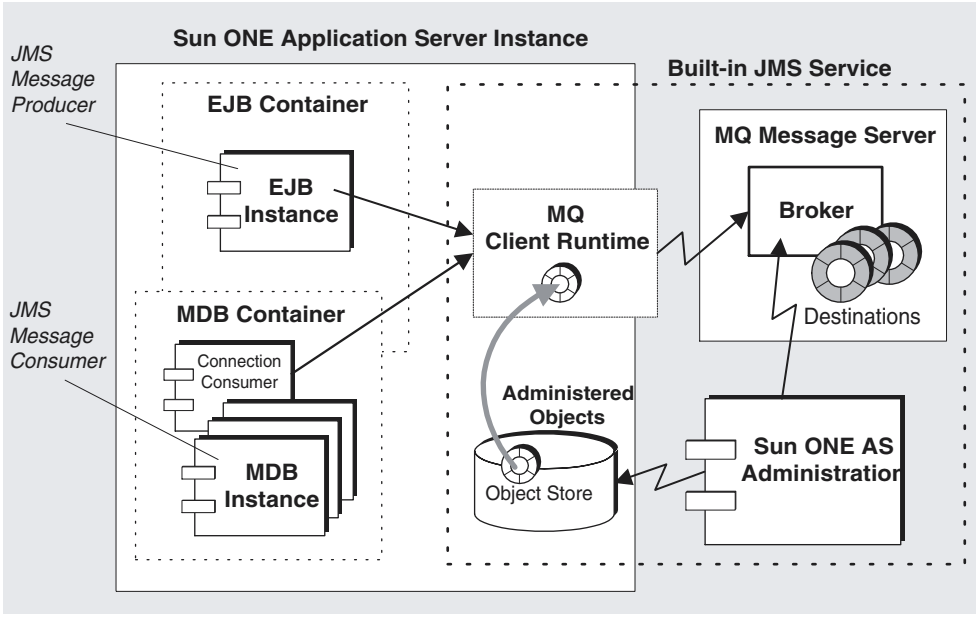
- 内置 JMS 服务的体系结构
- 禁用内置 JMS 服务

内置 JMS 服务可以使用 Sun ONE Application Server 管理工具（参见第 281 页上的“管理内置 JMS 服务”）进行管理。

内置 JMS 服务的体系结构

第 279 页上的 “内置 MQ 消息传送系统”图中说明的内置 JMS 服务与普通 MQ 消息传送系统 （如第 275 页上的 “MQ 系统结构”图中所示）类似，但下述资格除外。

内置 MQ 消息传送系统



MQ 消息服务器 . 每个 Sun ONE Application Server 实例都与自身内置的 JMS 服务相关联。内置 JMS 服务使用单个代理消息服务器。代理在 Sun ONE Application Server 实例之外的单独进程中运行，如 “内置 MQ 消息传送系统” 图中所示。默认情况下，代理实例 （内置 JMS 服务）会在与之关联的服务器实例启动时启动，并在服务器实例关闭时关闭。服务器实例内置 JMS 服务的配置信息记录在 Sun ONE Application Server 配置存储 （server.xml 文件）中，并且可以根据第 282 页上的 “配置 JMS 服务” 中的说明进行修改。

MQ 客户机运行时。 JMS 服务的客户机运行时部件是一组支持 JMS API 的库。任何在服务器实例中运行的 JMS 客户机 （JMS 客户机组件，包括 MDB）都可以访问这些库。

MQ 管理的对象。 内置 JMS 服务使用 Sun ONE Application Server 提供的对象存储。每个服务器实例都有自己的对象存储。 JMS 服务将管理的对象 （连接工厂和目标资源）存储在此对象存储中。根据第 287 页上的 “管理被管理的对象资源” 中的说明创建这些管理的对象资源，这些资源可以通过 JMS 客户机使用 JNDI 查找代码进行访问。

Sun ONE Application Server 管理。Sun ONE Application Server 管理界面和命令行公用程序可以实现有限的 MQ 管理功能子集。通过管理界面和命令行，可以配置内置 JMS 服务、创建和删除物理目标、以及创建和删除 JMS 客户机执行 JMS 消息传送操作所使用的管理对象资源。但是，这些管理工具不允许 （或不便于）执行更复杂的管理任务，例如设置代理的属性、优化 MQ 客户机运行时、修改 MQ 用户系统信息库、管理 MQ 安全性等。如果要对内置 JMS 服务执行这些管理任务，则必须使用 MQ 中安装的管理工具，有关说明可以参见 MQ *Administrator's Guide*。有关 MQ 和 Sun ONE Application Server 管理功能的比较，请参见表第 281 页上的“Sun ONE Message Queue 和 Sun ONE Application Server 管理功能比较”。

禁用内置 JMS 服务

默认情况下，关联的 Sun ONE Application Server 实例启动时，内置 JMS 服务也随之启动 （即 MQ 代理启动）。但是，由于服务器实例无需支持 JMS 消息传送，或者由于服务器实例使用*外部 JMS 服务*，因此可能需要在启动服务器实例时禁用 JMS 服务的自动启动。（要禁用内置 JMS 服务，请参见第 282 页上的 “配置 JMS 服务”。）

外部 JMS 服务是一种消息传送系统，它不受 Sun ONE Application Server 控制。对于 MQ （*本地* JMS 提供商），这意味着使用 MQ 管理工具只能独立地启动和管理 MQ 消息服务器。各种服务器实例中运行的 JMS 客户机仍然可以使用 MQ 管理对象来访问 MQ 消息服务器。这些管理对象可以存储在与应用程序服务器实例关联的对象存储中，也可以存储在由 MQ 管理工具管理的独立对象存储 （必要时多个服务器实例可以在其中共享存储）中。

服务器实例使用外部 JMS 服务的方案有很多种。最常见的方案是不同服务器中的 JMS 客户机需要访问同一物理目标时的方案。在这种情况下，所有服务器实例都必须都访问同一消息服务器。因此，需要禁用所有服务器实例的内置 JMS 服务，并将所有 JMS 客户机配置为执行相应的 JNDI 查找以访问外部 JMS 服务。此外，还需要使用外部 JMS 服务提供商的管理工具独立管理外部 JMS 服务 （管理消息服务器、创建物理目标、以及创建所有所需的管理对象）。

配置多个应用程序服务器实例来共享单个 MQ 代理实例的步骤：

1. 禁用所有服务器实例中的 JMS 服务。
2. 独立管理任何服务器实例的共享 MQ 代理 （即必须使用用于管理外部服务的管理工具启动和关闭代理），还必须独立管理 Sun ONE Application Server 的物理目标。
3. 配置每个服务器实例中的连接工厂 JMS 资源，以便指向此外部 MQ 代理 （必须正确设置 imqBrokerHostName 和 imqBrokerHostPort 属性）。
4. 在 Sun ONE Application Server 上部署 JMS 应用程序时使用此连接工厂资源。

可以同时运行外部和内置 JMS 服务。服务器实例中的 JMS 客户机可以访问所需的任何 JMS 服务。

*不*推荐多个服务器实例共享同一内置 JMS 服务 （一个被启用，其余的被禁用），因为启用的 JMS 服务仅在关联的服务器实例运行时才运行，这样管理起来非常困难。

禁用内置 JMS 服务时，同时也会禁用执行与内置 JMS 服务关联的管理任务的功能。支持外部 JMS 服务所需的所有管理工具必须使用用于管理外部服务的管理工具来执行。

管理内置 JMS 服务

本主题主要讲述内置 JMS 服务的管理。*管理的执行是逐个服务器实例进行的*。

内置 JMS 服务的管理包括以下任务：

- 配置 JMS 服务
- 管理物理目标
- 管理被管理的对象资源
- 使用命令行界面管理内置 JMS 服务

可以使用 Sun ONE Application Server 的管理界面或命令行公用程序执行管理。“Sun ONE Message Queue 和 Sun ONE Application Server 管理功能比较”表将这些管理工具与 MQ 管理工具进行了比较。

Sun ONE Message Queue 和 Sun ONE Application Server 管理功能比较

功能	**Sun ONE MQ 管理工具**	**Sun ONE AS 管理界面**	**Sun ONE AS 管理命令行**
管理 MQ 代理状态	是	启动/停止	启动/停止
配置 MQ 代理	是	否	否
管理 MQ 代理用户系统信息库	是	否	否
多代理群集	是	否	否
管理安全性	是	否	否
管理物理目标	是	创建/删除	创建/删除
管理长期订阅和事务	是	否	否
管理被管理的对象资源	是	创建/删除/配置	是

下面各节解释了如何使用 Sun ONE Application Server 管理界面执行 JMS 服务管理任务。

配置 JMS 服务

安装时已为内置 JMS 服务设置了很多 JMS 服务属性。可以通过配置 JMS 服务来更改这些属性的默认值。

“JMS 服务属性”表中说明了 JMS 服务的属性。

JMS 服务属性

属性	说明	默认值
日志级别	要写入 Sun ONE Application Server 日志文件中的记录信息的级别。有关详细信息，请参见第 5 章 “使用日志”。	DEBUG_HIGH
端口	提供内置 JMS 服务的代理实例的主端口号。默认情况下， JMS 服务使用默认的主端口号。但是，如果该端口与其它软件冲突，或者启动了多个 Sun ONE Application Server 实例，则需要为每个实例指定唯一的主端口号。 请注意，如果安装时将端口号指定给 JMS 服务，而后其它服务又使用了该端口号，则可能会出现端口冲突。在这种情况下，必须为 JMS 服务指定其它端口号。	7676
管理员用户名/密码	执行代理管理任务 （例如管理物理目标）时所需的用户名/密码 （参见第 284 页上的 “管理物理目标”）。如果希望管理员能够安全访问代理实例 （默认情况下，任何用户都具有访问权限），首先需要在代理的用户系统信息库中创建相应的项 （见 MQ Administrator's Guide），然后为管理员用户名和密码输入对应的值。	admin/admin
启动超时	指定服务器实例等待 JMS 服务启动的时间 （以秒为单位）。如果超时，则终止服务器实例的启动。	60
启动变量	指定启动 JMS 服务时使用的任何变量。imqbroker 命令的启动变量以及指定这些变量的方法可以参见 MQ *Administrator's Guide*。（如果提供了 -name 和 -port 变量，则忽略这些变量。）	

JMS 服务属性 *（续）*

属性	说明	默认值
启用启动	指定服务器实例启动时是否启动内置 JMS 服务。如果不支持 JMS 消息传送或者要使用外部 JMS 消息服务，请将该属性设置为 FALSE。	TRUE

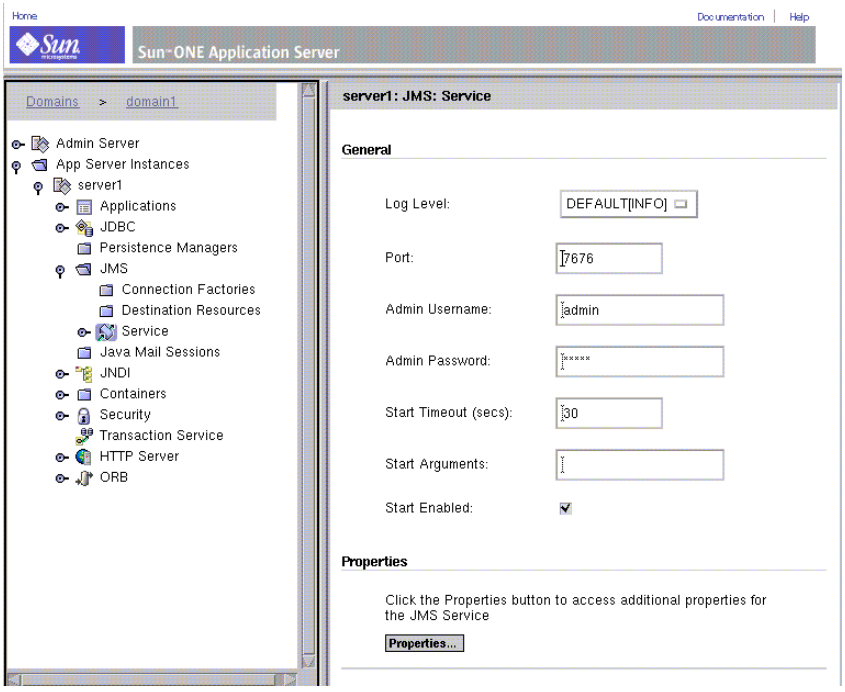
启动相应的服务器实例之前配置内置 JMS 服务。但是，如果已经运行了服务器实例，对配置的更改只有在停止服务器实例并重新启动后才能生效。

配置内置 JMS 服务的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS” 文件夹。
4. 选择 “Service” 链接。

   右侧窗格中将显示 “JMS Service configuration” 屏幕。

JMS 服务配置屏幕



5. 修改所需的任何属性 （参见表第 282 页上的 “JMS 服务属性”）。
6. 单击 “Save”按钮。

   JMS 服务屏幕将被刷新。

管理物理目标

在 JMS 消息传送中，一个 JMS 生成方向消息服务上的物理目标发送消息，然后该服务将消息分发到 JMS 用户。

对于内置 JMS 服务，可以明确地创建这些物理目标，也可以在收到消息后由 JMS 服务 （MQ 代理）自动创建。通常情况下，通过明确地创建消息传送应用程序所需的物理目标，可以更好地控制消息传送系统及其资源。当不再需要这些目标时，可以将它们删除。

为了创建或删除内置 JMS 服务的物理目标，必须运行 JMS 服务，并且管理员用户名和密码 （配置内置 JMS 服务时指定）必须与代理的用户系统信息库中的项相对应 （参见表第 282 页上的 “JMS 服务属性”）。

使用管理界面可以在内置 JMS 服务中执行以下物理目标管理任务：

- 创建队列或主题目标
- 列出物理目标
- 删除物理目标

创建队列或主题目标

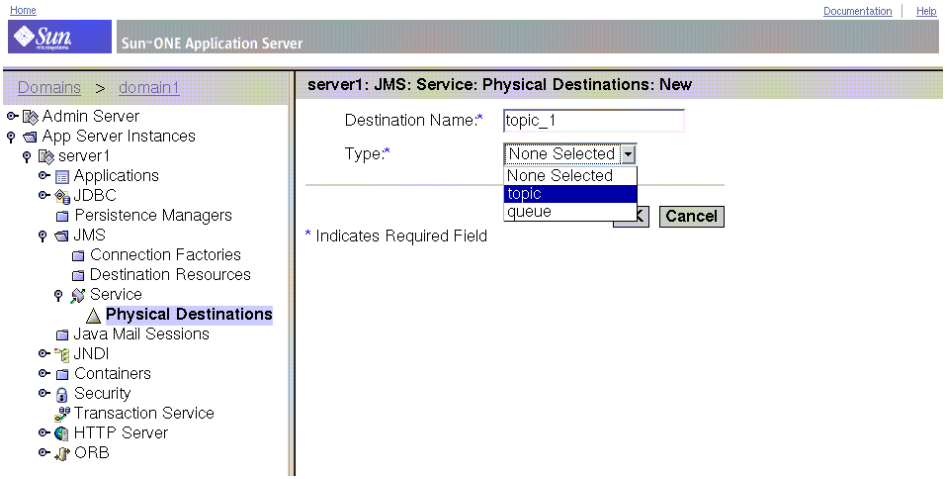
创建队列或主题目标的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS”文件夹。
4. 打开 “Service”链接。
5. 选择 “Physical Destinations”链接。

   右侧窗格中将显示 “Physical Destinations”屏幕。
6. 单击 “New”按钮。

   右侧窗格中将显示 “Physical Destinations:New”屏幕。

新的 JMS 物理目标屏幕



7. 输入物理目标的名称。
8. 从 “Type” 下拉列表中选择队列或主题。
9. 单击 “OK” 按钮。

   右侧窗格将被刷新，并在现有队列和主题目标列表中显示新的队列或主题目标。

列出物理目标

列出现有队列和主题目标的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS” 文件夹。
4. 打开 “Service” 链接。
5. 选择 “Physical Destinations” 链接。

   右侧窗格中将显示当前的物理目标。

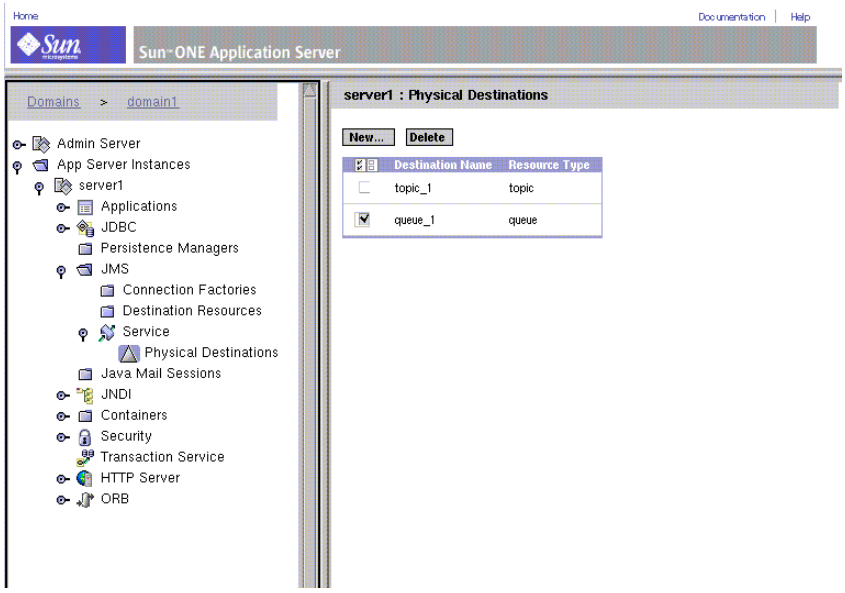
删除物理目标

可以根据需要删除队列或主题目标。

删除物理目标的步骤：

1. 列出现有目标 （参见第 285 页上的 “列出物理目标”）。
2. 单击要删除的每个目标旁边的选择框。

JMS 物理目标屏幕



3. 单击 “Delete”按钮删除选定的目标。

   列表将刷新，显示其它目标。

管理被管理的对象资源

JMS 客户机使用 MQ 管理对象 （Sun ONE Application Service 将其视为 *JMS 资源*）访问 JMS 服务 （内置或外部服务）。

J2EE 组件使用两个管理对象资源 （连接工厂和目标）连接到 JMS 服务，然后将消息传送到服务的物理目标，或从物理目标接收消息 （参见第 277 页上的 “MQ 管理的对象”）。

由于创建管理对象资源不直接包括 JMS 服务，因此，为服务器实例创建管理对象资源时无需启用 JMS 服务，也无需有效的用户名和密码（参见表第 282 页上的 “JMS 服务属性”）。

管理对象属性

要支持 JMS 消息传送，必须创建在服务器实例中运行的所有 JMS 客户机所需的管理对象资源。至少需要为每个管理对象资源指定一个 JNDI 查找名称、其类型 （连接工厂、队列或主题）、说明 （可选）以及是否启用资源。其它属性将在以下各节中讲述。

目标 （队列或主题）

对于队列或主题管理对象，还需要指定对应的物理目标的名称。

连接工厂

对于连接工厂管理对象，默认情况下，管理界面将创建指向内置 JMS 服务的连接工厂，即指向主机名为本地主机、端口号为配置 JMS 服务时设置的端口号的代理实例（参见表第 282 页上的 “JMS 服务属性”）。

但是，如果禁用了特定服务器实例的 JMS 服务，该服务器实例所支持的 JMS 客户机则必须使用外部 JMS 服务。创建连接工厂 （用于创建到外部 JMS 服务的链接）时，需要设置属性来指定相应代理实例的主机名和端口号。

连接工厂管理对象还有一些属性，用于优化服务器实例的 MQ 客户机运行时。这些属性在 MQ *Developer's Guide* 中均有说明。

使用管理界面创建的连接工厂管理对象支持分布式事务管理器。

管理对象资源管理任务

可以在管理界面中执行以下管理对象资源的管理任务：

- 创建队列或主题管理对象 （目标资源）
- 创建一个 ConnectionFactory Administered 对象
- 列出管理对象资源

- 删除管理对象资源

创建队列或主题管理对象 （目标资源）

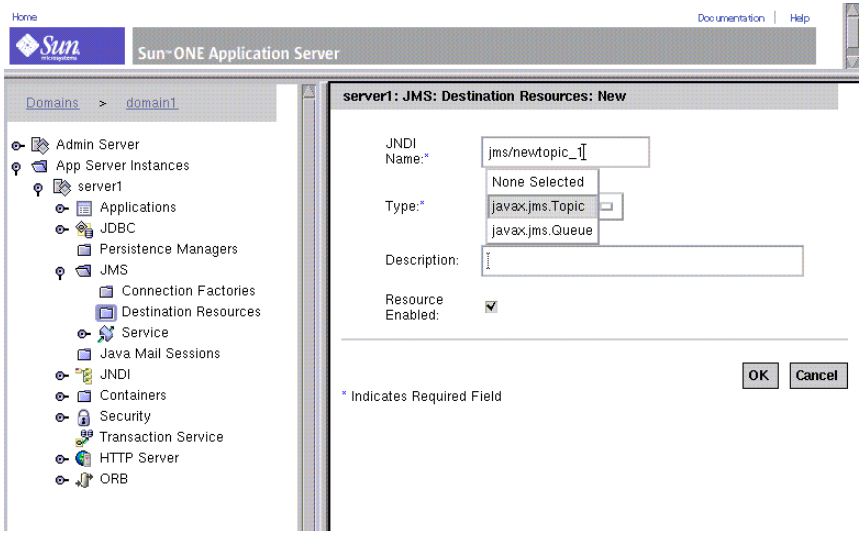
创建队列或主题管理对象的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS” 文件夹。
4. 选择 “Destination Resources” 链接。

   右侧窗格中将显示 “Destination Resources” 屏幕。
5. 单击 “New” 按钮。

   将显示 “Destination Resources:New” 屏幕。

新的目标管理对象屏幕



6. 输入与目标管理对象关联的 JNDI 查找名称。
7. 从下拉列表中选择 “队列” 或 “主题” 对象类型。
8. 单击 “OK” 按钮。

   右侧窗格中将重新显示 “Destination Resource:New” 屏幕。

此外，必须通过指定 imqDestinationName 属性来指定对象的目标名称。此属性值应该与物理目标的名称相匹配。

更改此属性值的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS”文件夹。
4. 打开 “Destination Resources”文件夹。
5. 选择要编辑的 “Destination Resource”。

   右侧窗格中将显示 “Destination Resource”屏幕。
6. 在右侧窗格中，单击 “Properties”。

   将出现 “Edit Properties”屏幕。
7. 在 “Name”字段中，输入 imqDestinationName。
8. 在 “Value”字段中，输入物理目标的名称。
9. 单击 “OK”。

   右侧窗格中将重新显示 “Destination Resources”屏幕。

创建一个 ConnectionFactory Administered 对象

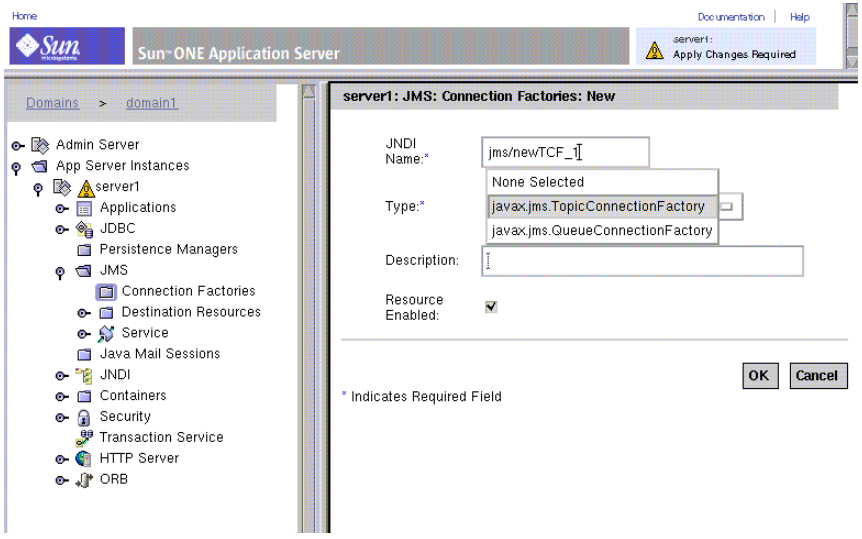
创建队列连接工厂或主题连接工厂管理对象的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS”文件夹。
4. 选择 “Connection Factories”链接。

   右侧窗格中将显示 “Connection Factory Resources”屏幕。
5. 单击 “New”按钮。

   将显示 “Connection Factory Resources:New”屏幕。

新的 ConnectionFactory 管理对象屏幕



6. 输入与该连接工厂管理对象关联的 JNDI 查找名称。
7. 从下拉列表中选择连接工厂对象类型。
8. 单击 “OK” 按钮。

   右侧窗格中将重新显示 “Connection Factory Resources” 屏幕，列表中将显示新建的连接工厂对象。

如果该连接工厂创建的到代理的连接与到内置 JMS 服务的不同，则必须设置 imqBrokerHostName 和 imqBrokerHostPort 属性的值。

更改这些属性值的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS” 文件夹。
4. 打开 “Connection Factories” 文件夹。
5. 选择要编辑的 “Connection Factory” 资源。

   右侧窗格中将显示 “Connection Factory” 屏幕。
6. 在右侧窗格中，单击 “Properties”。

   将出现 “Edit Properties” 屏幕。

7. 在 “Name” 字段中，输入 imqBrokerHostName。
8. 在 “Value” 字段中，输入属性的值。
9. 在 “Name” 字段中，输入 imqBrokerHostPort。
10. 在 “Value” 字段中，输入属性的值。
11. 单击 “OK”。

    右侧窗格中将重新显示 “Connection Factory” 屏幕。

列出管理对象资源

列出现有管理对象的步骤：

1. 打开管理界面。
2. 打开左侧窗格中的一个服务器实例。
3. 打开 “JMS” 文件夹。
4. 选择 “Destination Resources” 或 “Connection Factory Resources” 链接。

    右侧窗格中将显示当前目标或连接工厂管理对象。

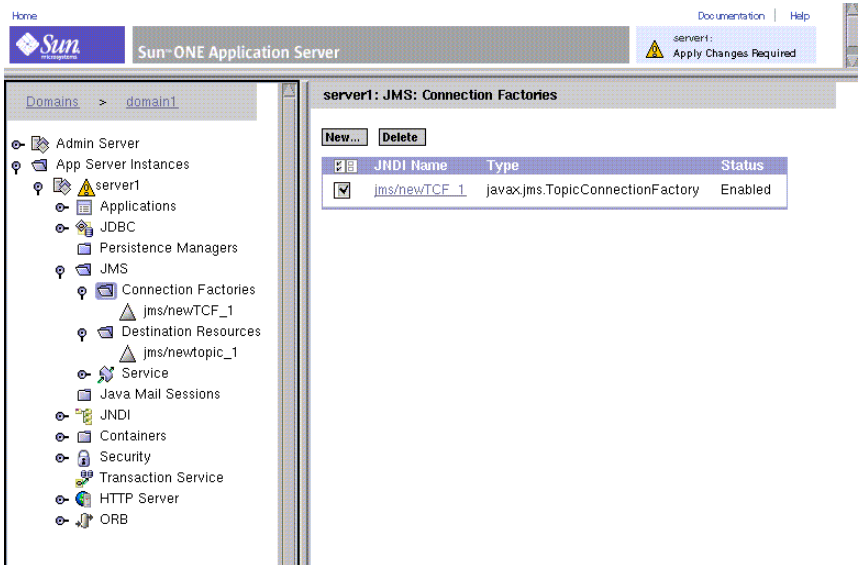
删除管理对象资源

删除管理对象资源的步骤：

1. 列出现有管理对象资源 （参见第 291 页上的 “列出管理对象资源”）。

    右侧窗格中将显示现有的管理对象资源。
2. 单击要删除的每个对象旁边的选择框。

JMS 连接工厂屏幕，已刷新



3. 单击 “Delete”按钮删除所有选定对象。

   屏幕将刷新列表，显示其它管理对象资源。

使用命令行界面管理内置 JMS 服务

Sun ONE Application Server 中包含一个命令行公用程序 (asadmin)，可以用于执行与使用管理界面执行的任务相同的任务。

使用以下 asadmin 命令可以配置和管理内置 JMS 服务。

用于管理内置 JMS 服务的 asadmin 命令

命令	用法
add-resources	为 jdbc、 jms 和 javamail 类型添加一种或多种资源。
create-jmsdest	创建一个 JMS 物理目标。
create-jms-resource	创建一个 JMS 资源。
delete-jmsdest	删除 JMS 物理目标。
delete-jms-resource	删除 JMS 资源。
jms-ping	强制回应 JMS 提供商，查看其是否正在运行。

用于管理内置 JMS 服务的 asadmin 命令

命令	用法
list-jmsdest	列出服务器实例的 JMS 物理目标。
list-jms-resources	列出服务器实例的 JMS 资源。
获取和设置 jms-service	获取和设置 JMS 服务的属性。
获取和设置 jms-resource	获取和设置 JMS 资源的属性。

有关这些命令的语法的信息，请参见 asadmin 联机帮助。有关 asadmin 的详细信息以及 jms-service 和 jms-resource 的属性列表，请参见附录 A “使用命令行界面”。

第 12 章

为 Corba/IIOP 客户机配置服务器

本章介绍了如何在 Sun ONE Application Server 环境中使用 RMI/IIOP 协议，为 CORBA/IIOP 客户机配置支持。

本章包括以下主题：

- 关于对 CORBA/IIOP 客户机的支持
- 配置 ORB

关于对 CORBA/IIOP 客户机的支持

J2EE 平台通过其互操作性要求为各种类型的客户机、不同的硬件平台和大多数软件应用程序提供了间接的支持。作为一种符合 J2EE 的产品， Sun ONE Application Server 支持标准的协议和格式集来确保互操作性。

CORBA (Common Object Request Broker Architecture) 模式为：客户机通过以远程方法请求的形式发出对分布式对象的请求，来通过一个明确的接口请求分布式对象或服务器的服务。远程方法请求带有需执行的操作的信息，包括服务提供商的对象名称（称作对象参考）和实际参数（如果有）。CORBA 自动处理大量网络程序任务，例如对象注册、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。

本节包括以下主题：

- 关于互操作性
- 关于 ORBw
- 关于 RMI/IIOP 功能
- 关于验证过程

关于互操作性

互操作性主要指企业环境将以各种语言编写的应用程序聚集在一起的能力。其中一个或多个现有的应用程序可能运行在个人计算机平台上，而其它程序可能运行于 UNIX 之上。另外，这些企业环境还可能支持 J2EE 平台不直接支持的、基于 Java 技术的独立的应用程序。

J2EE 旨在支持 CORBA IIOP (Internet Inter-Orb Protocol) 协议。 CORBA 定义了一种模式，该模式以对用户透明的方式指定了网络上分布式对象之间的互操作性。为此， CORBA 定义了用于以独立于实现的方式指定分布式对象的外部可见特性的方式。

关于 ORB

Object Request Broker （ORB 为缩写形式）是 CORBA 的核心组件。 ORB 提供了识别和定位对象、处理连接管理、传输数据和请求通信所需的基础架构。

CORBA 对象之间从不进行直接通信。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 Protocol （IIOP 为缩写形式）将该请求发送到其它计算机上的 ORB。远程 ORB 然后定位相应的对象 （程序）、处理该请求并返回结果。使用 RMI-IIOP 技术，JAVA 应用程序或对象可将 IIOP 用作 Remote Method Invocation （RMI 为缩写形式）。

关于 RMI/IIOP 功能

CORBA 指定了使处于各种位置的应用程序可以相互进行通信的 ORB。这种互操作性是通过 IIOP 提供的，通常可以在 Intranet 设置中找到。以下列出了 RMI 通过 IIOP 所提供的一部分功能：

- 与其它语言编写的对象的互操作性。
- 具有传输事务和安全性上下文的能力。
- 用于 ORB 服务的即插即用环境。
- 与 EJB 的互操作性。
- 使用 COSNaming 服务，一种基于 IIOP 的命名服务。 EJB 互操作性协议要求使用 COSNaming，以便使用 Java 命名和目录接口 （JNDI 为缩写形式） API 来查找 EJB 对象。

与 Sun ONE Application Server 一起绑定提供的 JAVA ORB 支持以下功能：

- 完全遵守 CSIv2 （Common Secure Interoperabillity 版本 2）。
- 完全符合的 COSNaming 服务实现 IDL 接口并有助于 EJB 容器发布 EJBHome 参考。
- IIOP/GIOP 版本 1.2。CORBA 指定了使处于各种位置的应用程序可以相互进行通信的 ORB。这种互操作性是通过 IIOP 提供的。

关于验证过程

验证是确认身份的过程。在网络交互环境中，验证是一方对另一方身份确认的过程。证书是支持验证的一种方法。

可以使用以下两种验证方式：

Server Authentication。服务器验证指客户机对服务器进行的信任识别；即对被认为要对位于特定网络地址的服务器负责的组织进行识别。

Client Authentication。客户机验证指服务器对客户机进行的信任识别，即对被认为使用客户端软件的人员进行识别。

客户机可以有多个证书，如同一个人可以有几个不同的身份一样。

配置 ORB

您可以为 Sun ONE Application Server 的每个实例配置多个 IIOP 监听器。默认情况下，仅配置一个 IIOP 监听器。您可以为 ORB 配置 IIOP 监听器属性并添加其它监听器。

您还可以为 ORB 启用监视，指定消息为哪一日志级别时将被记录，指定线程池设置，以及配置 IIOP 路径的 IIOP 监听器端口和 SSL 配置。在本节中，我们将讨论如何为 Sun ONE Application Server 的实例配置 ORB 支持。

本节包括以下内容：

- 进行常规 ORB 配置
- 为 ORB 配置 IIOP 监听器

进行常规 ORB 配置

使用管理界面，您可以启用监视，设置日志级别，并为线程池配置池设置。进行常规 ORB 配置的步骤：

1. 在管理界面的左侧窗格中，展开要配置 ORB 参数的 Sun ONE Application Server 实例。
2. 单击 “ORB”标签。您会在管理界面的右侧窗格中看到下图 “常规 ORB 配置”：

常规 ORB 配置

3. 在该窗口的 “General”部分中，您可以启用监视并设置 ORB 的日志级别。
   a. 要启用 ORB 监视，选中 “Monitoring Enabled”复选框。

b. 从 “Log Level”下拉列表中，选择所需的日志级别。服务器默认的日志级别通常设置为 “INFO”。 ORB 的默认级别将使用服务器的默认值。因此，在下拉菜单中，日志级别将显示为 “Default (INFO)”。

日志级别用于记录消息的严重性，范围从 FINEST 到 FATAL。设置日志级别使您可以选择在日志中显示何种消息粒度。 WARNING 的粒度将显示 WARNING、 ALERT、 SEVERE 和 FATAL 消息。通常您需要在服务器范围级设置粒度，但是您可以使用该设置来控制 Sun ONE Application Server ORB 显示的消息。

4. 在该窗口的 “Thread Pool”部分中，您可以指定 ORB 使用的请求线程的池设置。

请求线程处理对应用程序组件的用户请求。当 Sun ONE Application Server 收到请求时，它将请求分配到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。

您可以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。您所指定的最小线程池大小是 ORB 为应用程序请求预备的最小线程数。该线程数可以增加到您所指定的线程池最大值。

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。

a. 在 “Steady Pool Size”字段中，指定池中线程的最小数目。如果线程处于空闲状态的时间达到 “Idle Timeout (secs)”字段中的指定的值，池也会向这个最小数目方向进行收缩。

b. 在 “Max Pool Size”字段中，指定线程池可以增长到的最大线程数。

c. 在 “Idle Timeout (secs)”字段中，指定线程池中空闲线程的超时时间，空闲时间超过该值的线程将被清除。

5. 在该窗口的 “Advanced”部分中，您可以为 ORB 配置高级选项，步骤如下：

a. 在 “Message Fragment Size”字段中，指定 GIOP 1.2 消息大小的最大值，以便支持分段。默认的片段大小为 1024。

b. 在 “Total Connections”字段中，指定 ORB 服务器进程所允许的进入的远程 IIOP 连接的最大数目。

6. 单击 “Save”对设置进行保存。如果您希望恢复以前的设置而不保存当前的修改，单击 “Revert”。

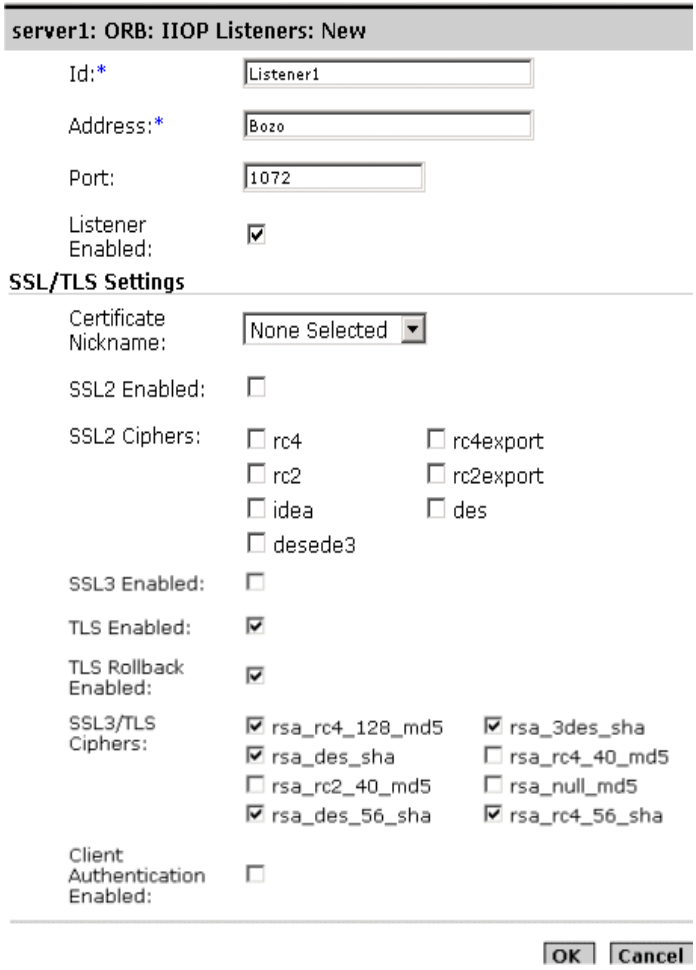
为 ORB 配置 IIOP 监听器

每个 Sun ONE Application Server 的新实例都具有默认的 ORB 配置，其中包括预先配置的 IIOP 监听器。 IIOP 监听器是一个监听套接字，它在特定端口进行监听并接受来自基于 CORBA 的客户机应用程序的进入的连接。您可以为 Sun ONE Application Server 的单个实例配置任何数目的 IIOP 监听器。

创建新的 IIOP 监听器或配置 IIOP 监听器属性的步骤：

1. 在管理界面的左侧窗格中，展开要配置 ORB 属性的 Sun ONE Application Server 实例。
2. 单击 “ORB”，然后打开其下的 “IIOP Listener”标签。您将会看到为 Sun ONE Application Server 的该特定实例配置的所有 IIOP 监听器的列表。
3. 要创建新的 IIOP 监听器，单击 “New”（如果您要编辑现有的 IIOP 监听器，打开该监听器并执行以下步骤中所列的任务即可）。单击 “New”或打开现有的 IIOP 监听器时，您将会看到图 “创建新的 IIOP 监听器”：

创建新的 IIOP 监听器



4. 您可以为 IIOP 监听器配置常规参数，步骤如下：

   a. 在 “Id”文本字段中，输入用于识别该监听器的名称。您可以使用任何标识符，例如 *ORB_Listener1*、 *ORB_Listener2* 等。

   b. 在 “Address”文本字段中，输入已安装 Sun ONE Application Server 的计算机的地址。您可以以 machinename.domainname 格式指定计算机地址（如给定的示例所示），也可以提供该计算机的 IP 地址。

c. 在 “Port”文本字段中，为 IIOP 监听器键入唯一的端口号。默认的 IIOP 监听器带有默认的端口号。您可以更改该端口号。但是，在更改端口号之前，应确保您所指定的新端口号尚未被其它应用程序或进程使用。

d. 要启用监听器，选中 “Listener Enabled”复选框。

5. 在此页面的 “SSL/TLS Settings”部分中，可以为 HTTP 监听器设置安全性。选中与 “Secure Sockets Layer (SSL)”和 “Transport Layer Security (TLS)”（其中包括了所有的加密算法）对应的复选框。可以选择 SSL2 或 SSL3/TLS 套接字。您可以为监听器配置 SSL/TLS 设置，步骤如下：

a. 在 “Certificate Nickname”字段中，输入服务器通过 SSL 握手提供给客户机的证书昵称。必须事先已安装了证书，才能在此列表中查看其昵称。

b. 选中 “SSL2 Enabled”字段，将启用该监听器路径的 SSL2 安全性选项。

c. 选择要用于此 SSL2 安全性的 SSL2 加密算法。选中所需的加密算法旁边的复选框。除非有充分的理由不使用特定的加密算法套件，否则应全部启用。

d. 选中 “SSL3 Enabled”字段，将启用该监听器路径的 SSL3 安全性选项。

e. 选中 “TLS Enabled”字段，将启用 TLS。还必须为要访问您的服务器的浏览器启用 TLS。对于 Netscape Navigator 6.0，请选中 TLS 和 SSL3。

f. 选中 “TLS Rollback Enabled”字段。为启用 TLS 回滚，需要先启用 TLS。而且，确保在启用该选项时 SSL3 和 SSL2 已被禁用。请为 Microsoft Internet Explorer 5.0 和 5.5 选中 “TLS Rollback”选项。

g. 选择要用于 SSL3 和 TLS 的 SSL3/TLS 加密算法。必须在启用 SSL3 或 TLS 之后才能选择这些选项。除非有充分的理由不使用特定的加密算法套件，否则应全部启用。

h. 选中 “Client Authentication Enabled”复选框，将指示带有客户机验证的 SSL IIOP 连接所用的 ORB 监听器端口是否已启用。客户机验证就是验证客户机证书的过程，即以密码方式验证证书签名和指向信任 CA 列表中该 CA 的证书链。

6. 单击 “OK”保存 IIOP 监听器设置。

注意

- 在安装 Sun ONE Application Server 时，将为默认的服务器实例创建 IIOP 监听器。默认 IIOP 监听器端口的默认端口号是 3700。
- 请注意，每个 IIOP 监听器的端口号必须设置为不同的值。同时还应注意，您在 “Address”文本字段中输入的计算机地址必须是已安装 Sun ONE Application Server 的计算机的地址。
- 有关监听器路径的 SSL 设置和 Sun ONE Application Server 安全性的其它详细信息，请参阅 《*Sun ONE Application Server Administrator's Guide to Security*》。

第 13 章

部署应用程序

本章介绍如何部署 Sun ONE Application Server 的各种模块和应用程序。

Sun ONE Application Server 模块和应用程序包括 J2EE 标准元素和 Sun ONE Application Server 特定元素。本章仅详细介绍 Sun ONE Application Server 特定元素。

要了解有关部署所需的封装和组建模块及应用程序，请参见 *《Sun ONE Application Server Developer's Guide》*。

本章包括以下主题：

- 关于 J2EE 模块
- 关于 J2EE 应用程序
- J2EE 标准描述符
- Sun ONE Application Server 描述符
- 命名标准
- 部署目录结构
- 运行时环境
- 关于类装入器
- 部署模块和应用程序
- 应用程序部署描述符文件

关于 J2EE 模块

J2EE 模块是一个或多个相同容器类型的 J2EE 组件以及该类型的部署描述符的集合。一个是 J2EE 标准描述符，另一个是 Sun ONE Application Server 特定描述符。J2EE 模块的类型包括：

- **Web 应用程序归档 (WAR)：**Web 应用程序是 servlet、 HTML 页面、类以及可以捆绑并部署到若干个 J2EE 应用程序服务器的其它资源的集合。 WAR 文件可以包含以下项目：servlet、 JSP、 JSP 标记库、公用程序类、静态页面、客户端小应用程序、 Bean、 Bean 类以及部署描述符 （web.xml 和可选的 sun-web.xml）。
- **EJB JAR 文件：**EJB JAR 文件是组建企业 Bean 的标准格式。这个文件包含 Bean 类（home、remote、local 和 implementation）、所有公用程序类和部署描述符 （ejb-jar.xml 和可选的 sun-ejb-jar.xml）。如果 EJB 是具有容器管理的持久性的实体 Bean，可能还会包含 CMP 部署描述符 sun-cmp-mapping.xml。
- **应用程序 (RMI/IIOP) 客户机 JAR 文件：**RMI/IIOP 客户机是一个 Sun ONE Application Server 特定类型的 J2EE 客户机。 RMI/IIOP 客户机支持标准的 J2EE 应用程序客户机规范，并且还支持对 Sun ONE Application Server 的直接访问。其部署描述符是 application-client.xml 和可选的 sun-application-client.xml。
- **资源 RAR 文件：**RAR 文件适用于 J2EE CA 连接器。连接器模块类似于一个设备驱动程序，它提供了 EJB 访问外部企业系统的可移植方式。每个 Sun ONE Application Server 连接器都拥有一个 J2EE XML 文件 ra.xml。连接器必须还拥有一个 Sun ONE Application Server 部署描述符 sun-ra.xml。

软件包定义必须在所有模块的源代码中使用，以便类装入器可以在部署模块后正确定位类。

因为部署描述符中的信息是公开的，所以无需修改源代码便可对它进行更改。在运行时， J2EE 服务器读取此信息并采取相应的行动。

EJB JAR 模块和 Web 模块还可以在任何应用程序之外组建成单独的 .jar 或 .war 文件并单独部署，如下图所示。

关于 J2EE 应用程序

J2EE 应用程序是通过应用程序部署描述符捆绑在一起的一个或多个 J2EE 模块的逻辑集合。各组件可以在模块级别上或应用程序级别上进行组建。各组件还可以在模块级别上或应用程序级别上进行部署。

各组件被组建成模块，然后又组建成准备用于部署的 Sun ONE Application Server 应用程序 .ear 文件。

每个模块都具有一个 Sun ONE Application Server 部署描述符和一个 J2EE 部署描述符。 Sun ONE Application Server 管理界面使用部署描述符来部署应用程序组件，并在 Sun ONE Application Server 上注册资源。

每个应用程序包含一个或多个模块，一个可选的 Sun ONE Application Server 部署描述符以及一个必需的 J2EE 应用程序部署描述符。所有项目都使用 Java ARchive (.jar) 文件格式组建成扩展名为 .ear 的一个文件。

J2EE 标准描述符

J2EE 平台提供了组建和部署设备。这些设备使用 JAR 文件作为组件和应用程序的标准软件包，并使用基于 XML 的部署描述符自定义参数。有关 J2EE 组建和部署进程的详细信息，请参见《*Developing Enterprise Applications with the J2EE*, v 1.0》的第 7 章。

J2EE 标准部署描述符将在 J2EE 规范 v1.3 中进行介绍。

要在部署前检查这些部署描述符的正确性，请参见 《*Sun ONE Application Server Developer's Guide*》中有关部署描述符检验器的信息。

下表 “J2EE 标准描述符”中显示了查找有关 J2EE 标准部署描述符详细信息的位置。左侧列中列出了部署描述符，右侧列中列出了查找这些描述符详细信息的位置。

J2EE 标准描述符

部署描述符	查找详细信息的位置
application.xml	《Java 2 Platform Enterprise Edition Specification, v1.3》的第 8 章 “Application Assembly and Deployment - J2EE:application XML DTD”
web.xml	《Java Servlet Specification, v2.3》的第 13 章 “Deployment Descriptor”、《JavaServer Pages Specification, v1.2》的第 7 章 “JSP Pages as XML Documents”和第 5 章 “Tag Extensions”

J2EE 标准描述符

部署描述符	查找详细信息的位置
ejb-jar.xml	《Enterprise JavaBeans Specification, v2.0》的第 16 章 “Deployment Descriptor”
application-client.xml	《Java 2 Platform Enterprise Edition Specification, v1.3》的第 9 章 “Application Clients - J2EE:application-client XML DTD”
ra.xml	《Java 2 Enterprise Edition, J2EE Connector Architecture Specification, v1.0》的第 10 章 “Packaging and Deployment”

可以从以下地址查找这些规范：

http://java.sun.com/products/

Sun ONE Application Server 描述符

Sun ONE Application Server 将使用附加的部署描述符配置专用于 Sun ONE Application Server 的功能。除连接器模块必需使用的 sun-ra.xml 文件之外，这些都是可选的。

要在部署前检查这些部署描述符的正确性，请参见 *《Sun ONE Application Server Developer's Guide》*中有关部署描述符检验器的信息。

下表 “Sun ONE Application Server 描述符”中显示了查找有关部署描述符详细信息的位置。左侧列中列出了部署描述符，右侧列中列出了查找这些描述符详细信息的位置。

Sun ONE Application Server 描述符

部署描述符	查找详细信息的位置
sun-application.xml	第 321 页上的 “应用程序部署描述符文件”。
sun-web.xml	*Sun ONE Application Server Developer's Guide to Web Applications*
sun-ejb-jar.xml 和 sun-cmp-mapping.xml	*Sun ONE Application Server Developer's Guide to Enterprise Java Beans*
sun-application-client.xml 和 sun-acc.xml	*Sun ONE Application Server Developer's Guide to Clients*

Sun ONE Application Server 描述符

部署描述符	查找详细信息的位置
sun-ra.xml	*Sun ONE J2EE CA Service Provider Implementation Administrator's Guide*

注意 Sun ONE Application Server 部署描述符必须对 UNIX 系统具有 600 级存取特权。

所有 Sun ONE Application Server 部署描述符的 DTD 架构文件都位于 *install_dir*/appserv/lib/dtds 目录中。

命名标准

应用程序和单独部署的 EJB JAR、 WAR 和连接器 RAR 模块的名称 （由 server.xml 文件中的 name 属性指定）在 Sun ONE Application Server 中必须是唯一的。如果没有明确指定名称，则默认名称为文件名的第一部分 （不包含 .war 或 .jar 扩展名）。有关 server.xml 的详细信息，请参见 《*Sun ONE Application Server Administrator's Configuration File Reference*》。

不同类型的模块在一个应用程序内可以具有相同的名称，这是因为部署应用程序时，保存各个模块的目录会以 _jar、 _war 或 _rar 后缀命名。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 Java 类似软件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仅适用于 Sun ONE Application Server，还适用于其它 J2EE 应用程序服务器。

EJB 的 JNDI 查找名称也必须是唯一的。同样，建立一致的命名惯例会有帮助。例如，将应用程序名和模块名附加到 EJB 名称上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 中 EJB 的 JNDI 名称，该模块是在应用程序 pkging.ear 中封装的。

请确保软件包和文件名中不包含空格或操作系统不认可的字符。

部署目录结构

当部署应用程序时，保存各模块的目录将以 _jar、 _war 或 _rar 后缀命名。如果使用 asadmin deploydir 命令部署目录而不是 EAR 文件，目录结构必须遵循这一惯例。

模块和应用程序目录结构将遵循 J2EE 规范中描画的结构。

下面是一个包含 Web 模块、 EJB 模块和客户机模块的简单应用程序的目录结构示例。

```
+ converter_1/
|--- converterClient.jar
|--+ META-INF/
|   |--- MANIFEST.MF
|   |--- application.xml
|   '--- sun-application.xml
|--+ war-ic_war/
|   |--- index.jsp
|   |--+ META-INF/
|   |   |--- MANIFEST.MF
|   '--+ WEB-INF/
|      |--- web.xml
|      '--- sun-web.xml
|--+ ejb-jar-ic_jar/
|   |--- Converter.class
|   |--- ConverterBean.class
|   |--- ConverterHome.class
|   '--+ META-INF/
|      |--- MANIFEST.MF
|      |--- ejb-jar.xml
|      '--- sun-ejb-jar.xml
'--+ app-client-ic_jar/
    |--- ConverterClient.class
    '--+ META-INF/
       |--- MANIFEST.MF
       |--- application-client.xml
       '--- sun-application-client.xml
```

下面是一个单独部署的连接器模块的目录结构的示例。

```
+ MyConnector/
 |--- readme.html
 |--- ra.jar
 |--- client.jar
 |--- win.dll
 |--- solaris.so
 '--+ META-INF/
    |--- MANIFEST.MF
    |--- ra.xml
    '--- sun-ra.xml
```

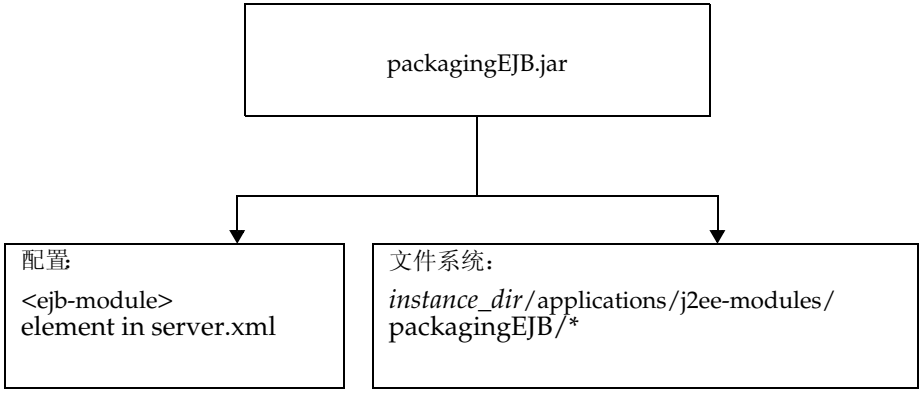
运行时环境

不论将组件部署为单独部署的模块还是部署为应用程序，部署都将影响文件系统和服务器配置。请参见图 “模块运行时环境”和 “应用程序运行时环境”。

模块运行时环境

下图 “模块运行时环境”说明了基于模块的单独部署的部署环境。

模块运行时环境



对于文件系统条目，模块将提取到以下位置：

instance_dir/applications/j2ee-modules/*module_name*
instance_dir/generated/ejb/j2ee-modules/*module_name*
instance_dir/generated/jsp/j2ee-modules/*module_name*

generated/ejb 目录中包含占位程序和捆绑程序； generated/jsp 目录中包含已编译的 JSP。

生命周期模块将提取到以下位置：

instance_dir/applications/lifecycle-modules/*module_name*

配置条目将添加到 server.xml 中，如下所示：

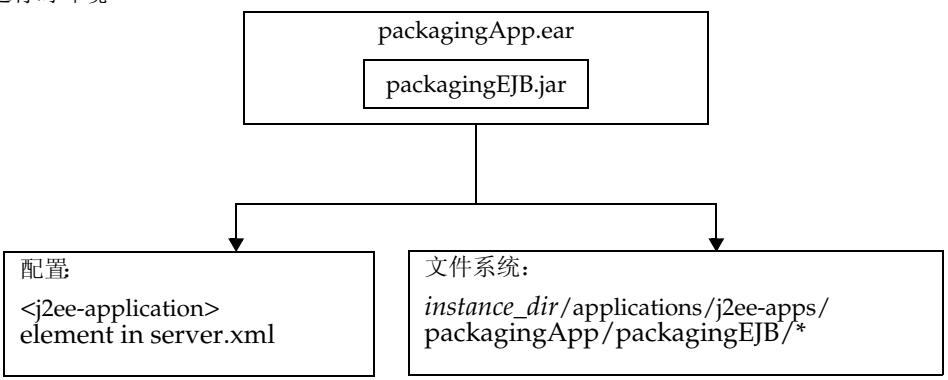
```
<server>
    <applications>
        <type-module>
            ...module configuration...
        </type-module>
    </applications>
</server>
```

server.xml 中模块的 *type* 可以是 lifecycle、 ejb、 web 或 connector。有关 server.xml 的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

应用程序运行时环境

下图 “应用程序运行时环境”说明了基于应用程序的部署环境。

应用程序运行时环境



对于文件系统条目，应用程序将提取到以下位置：

instance_dir/applications/j2ee-apps/*app_name*
instance_dir/generated/ejb/j2ee-apps/*app_name*
instance_dir/generated/jsp/j2ee-apps/*app_name*

generated/ejb 目录中包含占位程序和捆绑程序； generated/jsp 目录中包含已编译的 JSP。

配置条目将添加到 server.xml 中，如下所示：

```
<server>
    <applications>
        <j2ee-application>
            ...application configuration...
        </j2ee-application>
    </applications>
</server>
```

有关 server.xml 的详细信息，请参见 《*Sun ONE Application Server Administrator's Configuration File Reference*》。

将 server.xml 配置为使用 FastJavac 编译器

默认情况下， Sun ONE Application Server 使用内置的 JDK 编译器在部署期间编译应用程序。还可以使用 Sun One Studio 的 FastJavac 编译器，它在部署期间的编译速度更快。

在 Solaris 的捆绑安装中， FastJavac 编译器的位置不是透明的。要使用 FastJavac 编译器，需要使用编译器的路径配置管理服务器的 server.xml，如下所示：

在 server.xml 的 java-config 元素中添加以下 jvm-option：

<java-config java-home="/<*install-dir*>/jdk" server-classpath="....." >

jvm-options>-Dcom.sun.aas.deployment.java.compiler=/<*install-dir*>/studio4/bin/fastjavac/fastjavac.sun</jvm-options>

<property name="com.sun.aas.deployment.java.compiler.options"

value="-jdk /<*install-dir*>/jdk" />

</java-config>

关于类装入器

了解 Sun ONE Application Server 类装入器可以帮助您确定从何处以及如何定位模块和应用程序的支持 JAR 和资源文件。

在 Java 虚拟机 (JVM) 中，类装入器动态装入解析相关性所需的特定 Java 类文件。例如，需要创建 java.util.Enumeration 的一个实例时，其中一个类装入器将相关的类装入环境中。有关类装入器的详细信息，请参见 《*Sun ONE Application Server Developer's Guide*》。

部署模块和应用程序

本节介绍了将 J2EE 应用程序和模块部署到 Sun ONE Application Server 的几种不同方法。本节包括以下主题：

- 部署名称和错误
- 部署生命周期
- 模块或应用程序的部署
- 部署 WAR 模块
- 部署 EJB JAR 模块
- 部署生命周期模块
- 部署 RMI/IIOP 客户机
- 部署 J2EE CA 资源适配器
- 部署静态内容
- 访问共享的框架

部署名称和错误

部署应用程序或模块时， server.xml 文件中将生成唯一的名称。请不要更改此名称。部署期间，服务器将检测是否存在名称冲突，并且不装入具有非唯一名称的应用程序或模块。发生名称冲突时，相关消息将发送到服务器日志。有关命名的详细信息，请参见第 307 页上的 “命名标准”。

如果部署期间发生错误，将不部署应用程序或模块。如果应用程序内的模块包含错误，则整个应用程序都不进行部署。

有关 server.xml 的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

部署生命周期

应用程序初次部署之后，可以对其进行修改、重新装入、重新部署、禁止、重新启用和最终取消部署 （从服务器删除）操作。本节包括以下与部署生命周期相关的主题：

- 动态部署
- 禁止已部署的应用程序或模块
- 动态重新装入

动态部署

可以部署、重新部署和取消部署应用程序和模块，而不必重新启动服务器。这称为动态部署。

虽然动态部署主要用于开发者，但是它可以在操作环境中使用以联机引入新的应用程序和模块，而不必重新启动服务器。无论何时进行重新部署，该转换时间的会话都将无效。客户机必须重新启动该会话。

禁止已部署的应用程序或模块

您可以禁止已部署的应用程序或模块，而无需将其从服务器删除。每个应用程序和模块都具有 server.xml 文件中的 enabled 属性和管理界面中的相应选项，您可以对它们进行更改。有关 server.xml 的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

动态重新装入

如果已启用动态重新装入，更改代码时您不必重新部署应用程序或模块。您要做的只是将已更改的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，因为它允许快速检测代码更改。但是，不建议在生产环境中使用动态重新装入，因为它可能会降低性能。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。

要启用动态重新装入，可以执行以下操作之一：

- 使用管理界面：
  a. 打开服务器实例下的 “Applications”组件。
  b. 转至 “Applications”页面。
  c. 选中 “Reload Enabled”复选框，将启用动态重新装入。
  d. 在 “Reload Poll Interval”字段中输入秒数，将设置检查应用程序和模块的代码更改并动态重新装入的时间间隔。
  e. 单击 “Save”按钮。
  f. 转至服务器实例页面，并选择 “Apply Changes”按钮。
- 编辑 server.xml 文件的 applications 元素的以下属性：
  - dynamic-reload-enabled="true" 启用动态重新装入。
  - dynamic-reload-poll-interval-in-seconds 设置检查应用程序和模块的代码更改并动态重新装入的时间间隔。

  有关 server.xml 的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

此外，要装入新的 servlet 文件，重新装入 EJB 相关的更改，或重新装入部署描述符更改，必须执行以下操作：

1. 在已部署应用程序的根目录中创建名为 .reload 的空文件：

   instance_dir/applications/j2ee-apps/*app_name*/.reload

   对于单独部署的模块：

   instance_dir/applications/j2ee-modules/*module_name*/.reload

2. 每次进行上述更改时，都要明确更新 .reload 文件的时间标记类（在 UNIX 中是 touch .reload）。

对于 JSP，更改自动以 sun-web.xml 文件的 jsp-config 元素的 reload-interval 属性中设置的频率重新装入。要禁止 JSP 的动态重新装入，请设置 reload-interval="-1"。

部署工具

本节介绍了部署模块和应用程序可以使用的各种工具。部署工具包括：

- asadmin 公用程序
- 管理界面
- Sun ONE Studio

asadmin 公用程序

您可以使用 asadmin 公用程序在本地服务器上部署或取消部署应用程序和单独部署的模块。多台计算机上的并行部署不受支持。本节仅简要介绍了 asadmin 公用程序。

要部署生命周期模块，请参见第 318 页上的 “部署生命周期模块”。

asadmin deploy

asadmin deploy 命令可以部署 WAR、JAR、RAR、或 EAR 文件。要部署应用程序，请在命令中指定 --type application。要部署单个模块，请指定 --type ejb、 web、 connector。语法如下所示，并显示了其可选参数的默认值：

asadmin deploy --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] [--secure | -s] [--virtualservers *virtual_servers*] [--type application | ejb | web | connector] [--contextroot *contextroot*] [--force=true] [--precompilejsp=false] [--name *component_name*] [--upload=true] [--retrieve *local_dirpath*] [--instance *instance_name*] *filepath*

例如，以下命令将部署单个 EJB 模块：

asadmin deploy --user jadams --password secret --host localhost --port 4848 --type ejb --instance server1 packagingEJB.jar

asadmin deploydir

asadmin deploydir 命令将在打开的目录结构中部署应用程序或模块。该结构必须和第 308 页上的 “部署目录结构”中指定的结构相同。*instance_dir*/applications/j2ee-apps 或 *instance_dir*/applications/j2ee-modules 下的 *dirpath* 的位置将确定它是一个应用程序还是单独部署的模块。语法如下所示，并显示了其可选参数的默认值：

asadmin deploydir --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] [--secure | -s] [--virtualservers *virtual_servers*] [--type application | ejb | web | connector] [--contextroot *contextroot*] [--force=true] [--precompilejsp=false] [--name *component_name*] [--instance *instance_name*] *dirpath*

例如，以下命令将部署单个 EJB 模块：

asadmin deploydir --user jadams --password secret --host localhost --port 4848 --type ejb --instance server1 packagingEJB

asadmin undeploy

asadmin undeploy 命令将取消部署应用程序或模块。要取消部署应用程序，请在命令中指定 --type app。要取消部署模块，请指定 --type ejb、 web 或 connector。语法如下所示，并显示了其可选参数的默认值：

asadmin undeploy --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] [--secure | -s] [--type application|ejb|web|connector] [--instance *instance_name*] *component_name*

例如，以下命令将取消部署单个 EJB 模块：

asadmin undeploy --user jadams --password secret --host localhost --port 4848 --type ejb --instance server1 packagingEJB

管理界面

可以使用管理界面将模块和应用程序部署到本地和远程的 Sun ONE Application Server 站点。要使用此工具，请执行以下步骤：

1. 打开服务器实例下的 “Applications”组件。
2. 转到 “Enterprise Applications”、“Web Applications”、“Connector Modules”或 “EJB Modules”页面。
3. 单击 “Deploy”按钮。
4. 输入模块或应用程序的完整路径 （或单击 “Browse”查找），然后单击 “OK”按钮。
5. 输入模块或应用程序的名称。

   如果模块和应用程序已经存在，还可以通过选中相应的复选框重新部署模块或应用程序。这是可选的。
6. 通过选中虚拟服务器名称旁边的复选框，将应用程序或模块分配给一个或多个虚拟服务器。
7. 单击 “OK”按钮。

要部署生命周期模块，请参见第 318 页上的 “部署生命周期模块”。

Sun ONE Studio

可以使用 Sun ONE Studio 4 部署 J2EE 应用程序和模块。有关使用 Sun ONE Studio 的详细信息，请参见 *《Sun ONE Studio 4, Enterprise Edition Tutorial》*。

注意 在 Sun ONE Studio 中，部署模块或应用程序被称为*执行*。执行还包括确保服务器正在运行和显示正确的 URL 以激活模块或应用程序。

模块或应用程序的部署

可以部署应用程序或独立于应用程序的单个模块。基于应用程序的部署或基于模块的单独部署的运行时和文件系统含义在第 309 页上的 “运行时环境”中进行介绍。

如果组件需要由以下项目访问，则首选基于模块的单独部署：

- 其它模块
- J2EE 应用程序
- RMI/IIOP 客户机（基于模块的部署允许对 RMI/IIOP 客户机的 Bean、servlet 或 EJB 进行共享访问。）

模块可以组合到 EAR 文件中，然后作为单个模块部署。这与单独部署 EAR 模块类似。

部署 WAR 模块

可以用第 315 页上的 “部署工具”中介绍的方法之一来部署 WAR 模块。

通过将 -keepgenerated 属性添加到 sun-web.xml 中的 jsp-config 元素，可以保存生成的 JSP 源。如果部署 WAR 模块时包含此属性，则在应用程序中，生成的源将保存在 *instance_dir*/generated/jsp/j2ee-apps/*app_name*/*module_name* 中；而在单独部署的 Web 模块中，生成的源将保存在 *instance_dir*/generated/jsp/j2ee-modules/*module_name* 中。有关 -keepgenerated 属性的详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Web Applications》*。

部署 EJB JAR 模块

可以用第 315 页上的 “部署工具”中介绍的方法之一来部署 EJB JAR 模块。

通过将 -keepgenerated 标志添加到 server.xml 中的 java-config 元素的 rmic-options 属性，可以保存占位程序和捆绑程序生成的源。如果部署 EJB JAR 模块时包含此标志，则在应用程序中，生成的源将保存在 *instance_dir*/generated/ejb/j2ee-apps/*app_name*/*module_name* 中；而在单独部署的 EJB JAR 模块中，生成的源将保存在 *instance_dir*/generated/ejb/j2ee-modules/*module_name* 中。有关 -keepgenerated 标志的详细信息，请参见 *《Sun ONE Application Server Administrator's Configuration File Reference》*。

部署生命周期模块

有关生命周期模块的详细信息，请参见 *《Sun ONE Application Server Developer's Guide》*。

可以使用以下工具部署生命周期模块：

- asadmin 公用程序
- 管理界面

asadmin 公用程序

要部署生命周期模块，请使用 asadmin create-lifecycle-module 命令。语法如下所示，并显示了其可选参数的默认值：

asadmin create-lifecycle-module --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] [--secure | -s] [--instance *instance_name*] --classname *classname* [--classpath *classpath*] [--loadorder *load_order_number*] [--failurefatal=false] [--enabled=true] [--description *text_description*] [--property (*name=value*)[:*name=value*]*] *modulename*

例如：

asadmin create-lifecycle-module --user jadams --password secret --host localhost --port 4848 --instance server1 --classname RMIServer MyRMIServer

要取消部署生命周期模块，请使用 asadmin delete-lifecycle-module 命令。语法如下所示，并显示了其可选参数的默认值：

asadmin delete-lifecycle-module --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] [--secure | -s] [--instance *instance_name*] *module_name*

例如：

asadmin delete-lifecycle-module --user jadams --password secret --host localhost --port 4848 --instance server1 MyRMIServer

要列出在服务器实例中部署的生命周期模块，请使用 asadmin list-lifecycle-modules 命令。语法如下所示，并显示了其可选参数的默认值：

asadmin list-lifecycle-modules --user *admin_user* [--password *admin_password*] [--host localhost] [-port 4848] *instance_name*

例如：

asadmin list-lifecycle-module --user jadams --password secret --host localhost --port 4848 server1

管理界面

还可以使用管理界面来部署生命周期模块。请执行以下步骤：

1. 打开服务器实例下的 “Applications”组件。
2. 转至 “Life Cycle Modules”页面。
3. 单击 “Deploy”按钮。
4. 输入以下信息：
   - 名称 （必需） - 生命周期模块的名称。
   - 类名 （必需） - 生命周期模块的类文件的全限定名称。
   - 类路径 （可选） - 生命周期模块的类路径。指定放置模块的位置。默认位置在应用程序根目录下。
   - 装入顺序 （可选） - 确定启动时装入生命周期模块的顺序。系统将较早装入具有较小整数值的模块。值的范围可以从 101 到操作系统的 MAXINT。将保留 1 至 100 的值。
   - 致命故障 （可选） - 确定生命周期模块出现故障时是否关闭服务器。默认值为 False。
   - 启用 （可选） - 确定是否启用生命周期模块。默认值为 True。
5. 单击 “OK”按钮。

部署 RMI/IIOP 客户机

只有与 EJB 通信的客户机需要进行部署。部署 RMI/IIOP 客户机的进程分为三个步骤：

1. 部署要由 RMI/IIOP 客户机访问的 EAR 或 EJB JAR。
2. 组建必要的客户机文件并部署该客户机。
3. 部署之后，对于应用程序，将在以下位置创建客户机 JAR 文件：

   instance_dir/applications/j2ee-apps/*app_name*/*app_name*Client.jar

   而对于单独部署的模块，将在以下位置创建文件：

   instance_dir/applications/j2ee-modules/*module_name*/*module_name*Client.jar

   客户机 JAR 包含捆绑程序和 RMI/IIOP 客户机所需的类。将此文件复制到客户端计算机中，并将客户机上的 APPCPATH 环境变量设置为指向此 JAR。

现在准备运行客户机。详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Clients》*。

部署 J2EE CA 资源适配器

可以用第 315 页上的 “部署工具”中介绍的方法之一来部署连接器模块。

部署静态内容

静态内容 （HTML、图像等）能够以 Web 服务器和 Sun ONE Application Server 为主机。但是，当注册 WAR 时，静态内容将在应用程序服务器上进行部署。 Sun ONE Application Server 中附带的所有样例都将包含应用程序服务器上的静态内容。

例如，要访问应用程序服务器上的静态文件 index.html，请使用：

http://*server:port*/NASApp/&lt*context_root*/index.html

访问共享的框架

当 J2EE 应用程序和模块使用共享的框架类 （例如组件和库）时，这些类可以放入系统类装入器或公用类装入器的路径中，而不是放入应用程序或模块中。如果将一个大型的共享库组建到使用它的每个模块中，则大型文件将耗费太长的时间而不能在服务器中注册。此外，不同的类装入器中可能存在相同类的几个版本，这将造成资源浪费。

有关系统类装入器的详细信息，请参见第 312 页上的 “关于类装入器”。

应用程序部署描述符文件

Sun ONE Application Server 应用程序包括两个部署描述符文件：

- 一个 J2EE 标准文件 (application.xml)，在 《Java Servlet Specification, v2.3》的第 13 章 “Deployment Descriptors”中介绍。
- 一个可选的 Sun ONE Application Server 特定文件 (sun-application.xml)，在本节中介绍。

有关应用程序部署描述符文件的详细信息，请参见 《*Sun ONE Application Server Developer's Guide*》。

管理 HTTP 服务器功能和虚拟服务器

第 14 章

配置 HTTP 功能

本章讲述了为 Sun ONE Application Server 中的 HTTP 相关功能配置首选项的方法。有关与虚拟服务器及 HTTP 监听器相关的首选项的信息，请参见第 15 章 “使用虚拟服务器”。

本章包括以下主题：

- 关于 HTTP 功能
- 配置文件高速缓存
- 优化服务器性能
- 配置 HTTP 服务质量
- 添加和使用线程池
- 配置文件高速缓存编辑高级设置
- 配置 MIME 类型

关于 HTTP 功能

Sun ONE Application Server HTTP 功能包括设置应用程序服务器实例的性能级别、设置有关性能优化的参数、以及使用文件高速缓存改善性能。这些设置存储在以下两个配置文件中：init.conf 和 server.xml。可以在 “Advanced Settings” 页面中编辑 init.conf 设置。有关详细信息，请参见第 329 页上的 “编辑高级设置”。

用户编辑的其它属性存储在 server.xml 文件中，该文件位于 http-service 元素中。有关 init.conf 文件和 server.xml 文件的详细信息，请参见 《*Sun ONE Application Server Administrator's Configuration File Reference*》。

配置文件高速缓存

Sun ONE Application Server 使用文件高速缓存更快地提供静态服务。文件高速缓存包括有关文件和静态文件内容的信息，并且可以缓存用于加速处理服务器分析的 HTML 的信息。

默认情况下启用文件高速缓存。文件高速缓存设置包含在 nsfc.conf 文件中。只有更改了文件高速缓存默认参数，该文件才存在。有关 nsfc.conf 的详细信息，请参见《*Sun ONE Application Server Administrator's Configuration File Reference*》。

配置文件高速缓存的步骤：

1. 在左侧窗格中，单击 “HTTP Server”。
2. 单击 “File Caching”选项卡。
3. 在字段中输入所需的值。
4. 单击 “OK”。

有关使用文件高速缓存改善性能的详细信息，请参见 《*Sun ONE Application Server Performance Tuning and Sizing Guide*》。

优化服务器性能

在 “Performance Tuning”页面中，通过控制 Sun ONE Application Server 处理的请求数量、超时之前请求在没有任何活动的情况下保持打开状态的时间、以及是否要使用 DNS 反向查找客户机的 IP 地址，可以配置用于控制 Sun ONE Application Server 性能的设置。同样，如果使用的是 DNS，则可以对这些性能相关的功能（无论是否使用异步 DNS）和 DNS 高速缓存进行设置。

有关性能优化的详细信息，请参见 《*Sun ONE Application Server Performance Tuning Guide*》。

对性能优化设置进行设置的步骤：

1. 在左侧窗格中，单击 “HTTP Server”。
2. 单击 “Tuning”选项卡。
3. 在字段中输入所需的值。
4. 单击 “OK”。

有关可以通过管理界面进行优化的设置的其它信息，请参见联机帮助。

配置 HTTP 服务质量

服务质量是指用户为服务器设置的性能限制。例如， ISP 可能希望根据允许使用的带宽来对虚拟服务器收取不同的费用。

在特定虚拟服务器中使用服务质量之前，必须先为服务器实例启用该功能并设置一些值。

为服务器实例配置服务质量设置的步骤：

1. 在左侧窗格中，单击 “HTTP Server”。
2. 单击 “QOS”选项卡。
3. 要全面启用服务质量，请单击 “Enable”。

   默认情况下禁用服务质量。启用服务质量会使服务器的负担略有增加。
4. 选择 “Recompute Interval”。

   重新计算时间间隔是每次计算带宽之间间隔的毫秒数。默认值为 100 毫秒。
5. 选择 “Metric Interval”。

   公制时间间隔是测量通信量期间的时间间隔 （以秒为单位），默认值为 30 秒。计算该时间段内测量的所有带宽的平均值，即可得出每秒的字节数。

   如果站点中要传输很多大型文件，请在此字段中使用较大的值 （几分钟或更长时间）。大型文件传输可能会在很短的公制时间间隔内占用所有允许的带宽，如果强制执行最大带宽设置，还可能导致连接被拒绝。由于带宽是根据公制时间间隔的平均值进行计算的，因此增加时间间隔可以消除大型文件引起的通信量高峰。

   如果带宽限制远远低于可用带宽 （例如，带宽限制为 1 Mbps，而主干连接带宽为 1 Gbps），则应缩短公制时间间隔。

   请注意，如果传输很大的静态文件，而带宽限制又远远低于可用带宽，则需要确定要优化前者还是后者，因为两者的解决方案恰好相反。
6. 设置服务器的带宽限制 （以每秒钟的字节数为单位）。
7. 选择是否要强制带宽限制设置。

   如果选择强制带宽限制，服务器达到带宽限制后，其它连接将被拒绝。

   如果不强制带宽限制，超过限制时，服务器会将一条消息记录到错误日志中。
8. 选择服务器允许的最大连接数。

   该数值是处理的并行请求数。

9. 选择是否要强制连接限制设置。

   如果选择强制连接限制，服务器达到连接限制后，其它连接将被拒绝。

   如果不强制连接限制，超过限制时，服务器会将一条消息记录到错误日志中。

10. 要指定其它名称/值对，请单击 “Properties”按钮。

11. 单击 “OK”。

要使用命令行界面的 asadmin 公用程序配置服务质量，请使用以下命令：

- create-http-qos
- delete-http-qos

这些命令使用以下语法：

asadmin create-http-qos --user *admin_user* [--password *password*] [--host *hostname*] [--port *admin_port*] [--secure | -s] [--passwordfile *file_name*] [--virtualserver *virtual_server_id*] [--bwlimit *bandwidth_limit*] [--enforcebwlimit *enforce_bandwidth_limit*] [--connlimit *connection_limit*] [--enforceconnlimit *enforce_connection_limit*] *instancename*

asadmin delete-http-qos --user *admin_user* [--password *password*] [--host *hostname*] [--port *admin_port*] [--secure | -s] [--passwordfile *file_name*][--virtualserver *virtual_server_id*] *instancename*

如果指定了虚拟服务器，这些命令将为该虚拟服务器创建或删除服务质量信息。如果未指定虚拟服务器，命令则会影响服务器实例。

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

有关服务质量功能限制的详细信息，请参见第 138 页上的 “使用 CLI 管理事务服务”。

添加和使用线程池

可以使用线程池为特定服务分配一定数量的线程，以确保所使用的线程不超过额定值。线程池的另一用途是运行对于线程来说不安全的插件。如果将线程池的最大线程数定义为 1，一个指定的服务功能则只允许一个请求。

添加线程池时，需要指定的信息包括：最小和最大线程数、堆栈大小以及队列大小。

添加线程池的步骤：

1. 在左侧窗格中，单击 “HTTP Server”。

2. 单击 “Thread Pool”。

3. 在字段中输入所需的值。
4. 单击“OK”。

线程池将显示在页面底部。要编辑或删除线程池，请单击线程池旁边的“Edit”或“Delete”按钮。

对线程池进行设置后，可以将其指定为特定服务的线程池来使用它。

有关使用线程池改善性能的详细信息，请参见《*Performance Tuning and Sizing Guide*》。

编辑高级设置

启动 Sun ONE Application Server 时，它将在 *instance_dir*/config/ 目录中查找 init.conf 文件，以创建一组影响服务器操作和配置的全局变量设置。 Sun ONE Application Server 将执行 init.conf 中定义的所有指令。

这些设置都显示在“Advanced Settings”页面中。可以对 init.conf 文件中的某些设置进行编辑，这些设置将影响以下方面：

- DNS
- SSL
- 性能
- CGI
- 保持活动
- 日志

有关 init.conf 文件的完整说明，请参见《*Sun ONE Application Server Administrator's Configuration File Reference*》。

编辑高级设置的步骤：

1. 在左侧窗格中，单击“HTTP Server”。
2. 单击“Advanced”选项卡。
3. 单击要更改的设置的类型（DNS、SSL 等）。
4. 根据需要对设置进行更改，然后单击“OK”。

有关各种类型设置的详细信息，请参见联机帮助。

配置 MIME 类型

通过 “Mime Types”页面可以编辑服务器的 MIME 文件。 MIME （多用途网际邮件扩充协议）类型用于控制系统所支持的多媒体文件类型，还可以指定属于某些服务器文件类型的文件扩展名 （例如，指定哪些文件是 CGI 程序）。

可以根据需要创建尽可能多的 MIME 类型，并将这些类型与应用程序服务器实例或虚拟服务器关联。默认情况下，服务器中存在一个 MIME 类型文件 mime.types，并且不能删除该文件。

创建新的 MIME 类型文件的步骤：

1. 在左侧窗格的 “HTTP Server”下，单击 “MIME Type File”。
2. 在右侧窗格中，单击 “New”。
3. 输入 MIME 文件的标识符和文件名。
4. 单击 “OK”。

编辑 MIME 文件中的定义的步骤：

1. 在左侧窗格的 “HTTP Server”下，单击 “MIME Type File”旁边的图标展开视图。
2. 单击要编辑的 MIME 文件的 ID。
3. 在此页面中，编辑与 ID 关联的 MIME 文件名。
4. 要编辑 MIME 文件的文件扩展名，请单击 “Edit MIME file”。
5. 要编辑现有条目，请单击该条目旁边的 “Edit”。
6. 在出现的页面中进行更改，然后单击 “Change MIME Type”。
7. 要删除 MIME 类型，请单击该类型旁边的 “Remove”。
8. 要添加新的 MIME 类型，请在字段中输入种类、内容类型和文件后缀，然后单击 “New Type”。

要使用命令行界面的 asadmin 公用程序配置 MIME 类型，请使用以下命令：

- create-mime
- delete-mime
- list-mimes

这些命令使用以下语法：

asadmin create-mime --user *admin_user* [--password *password*] [--host *hostname*] [--port *admin_port*] [--secure | -s] [--passwordfile *file_name*] [--instance *instancename*] --mimefile *filename mime_id*

asadmin delete-mime --user *admin_user* [--password *password*] [--host *hostname*] [--port *admin_port*] [--secure | -s] [--passwordfile *file_name*] [--instance *instancename*] *mime_id*

asadmin list-mimes --user *admin_user* [--password *password*] [--host *hostname*] [--port *admin_port*] [--secure | -s] [--passwordfile *file_name*] *instancename*

有关命令语法的详细信息，请参见命令行界面帮助。有关使用 asadmin 的详细信息，请参见附录 A “使用命令行界面”。

有关在虚拟服务器中使用 MIME 类型的信息，请参见联机帮助和第 15 章 “使用虚拟服务器”。

第 15 章

使用虚拟服务器

本章介绍了如何使用 Sun ONE Application Server 设置和管理虚拟服务器。有关配置虚拟服务器内容设置的信息，请参见第 16 章 “管理虚拟服务器内容”。

本章包括以下主题：

- 虚拟服务器概述
- 在虚拟服务器中使用 Sun ONE Application Server 的功能
- 创建和配置 HTTP 监听器
- 创建和配置虚拟服务器
- 部署虚拟服务器

虚拟服务器概述

使用虚拟服务器时，可以用一个已安装的服务器提供公司或个人域名、 IP 地址和某些服务器监视功能。对用户而言，他们就像拥有了自己的 Web 服务器，但您应提供硬件并维护虚拟服务器。

安装 Sun ONE Application Server 的非捆绑版本时，系统将为应用程序服务器实例创建一个默认虚拟服务器。也就是说，对于默认的应用程序服务器实例 server1，将同时创建一个名为 server1 的虚拟服务器。如果使用的是 Solaris 9 捆绑版本，则您需要创建一个服务器实例。创建服务器实例的同时将创建一个与之同名的虚拟服务器。系统将为创建的每个附加应用程序服务器实例创建一个虚拟服务器。有关创建和配置虚拟服务器的详细信息，请参见第 343 页上的 “创建和配置虚拟服务器”。有关部署虚拟服务器的详细信息，请参见 “部署虚拟服务器”。

此虚拟服务器可以控制 Sun ONE Application Server 的 HTTP 功能，这些功能以虚拟服务器为单位提供。您可能不需要使用多个虚拟服务器，但是仍然可以通过配置与应用程序服务器实例一同创建的默认虚拟服务器来配置此应用程序服务器实例的某些属性。

虚拟服务器的设置存储在 *instance_dir*/config 目录的 server.xml 文件中的 virtual-server 元素中。有关此文件的详细信息，请参见 《*Sun ONE Application Server Administrator's Configuration File Reference*》。

与虚拟服务器相关的某些信息存储在虚拟服务器的 obj.conf 文件中。每个虚拟服务器都有一个单独的 obj.conf 文件。

本节包括以下主题：

- HTTP 监听器
- 虚拟服务器
- obj.conf 文件
- 选择用于处理请求的虚拟服务器
- 文档根目录
- 使用访问日志文件和服务器日志文件

HTTP 监听器

服务器与客户机之间的连接在 HTTP 监听器 （也称作监听套接字）上进行。您创建的每个 HTTP 监听器都有一个 IP 地址、端口号、返回服务器名和默认虚拟服务器。要使 HTTP 监听器监听计算机给定端口上所有已配置的 IP 地址，请使用 0.0.0.0、any、 ANY 或 INADDR_ANY 作为 IP 地址。有关创建和配置 HTTP 监听器的详细信息，请参见第 340 页上的 “创建和配置 HTTP 监听器”。

安装 Sun ONE Application Server 的非捆绑版本时，系统将自动创建一个 HTTP 监听器 http-listener-1。此 HTTP 监听器使用 IP 地址 0.0.0.0 和安装过程中指定为 HTTP 服务器端口号的端口号（默认值为 80，如果以超级用户以外的身份进行安装，则在 UNIX 中为 1024）。不能删除默认的 HTTP 监听器。如果使用多个虚拟服务器，则既可以对所有虚拟服务器使用默认 HTTP 监听器，也可以创建多个 HTTP 监听器。

使用 Solaris 9 捆绑的 Sun ONE Application Server 时， HTTP 监听器将在您创建服务器实例时创建。它具有 IP 地址 0.0.0.0 和创建实例时指定的端口号。

由于 HTTP 监听器是 IP 地址和端口号的组合，因此您可以拥有多个 IP 地址相同但端口号不同（或 IP 地址不同但端口号相同）的 HTTP 监听器。例如，您既可以使用 1.1.1.1:81 和 1.1.1.1:82，也可以使用 1.1.1.1:81 和 1.2.3.4:81，只要将计算机配置为响应这两个地址即可。但是，如果使用 0.0.0.0 IP 地址监听端口上的所有 IP 地址，则不能为监听特定 IP 地址的同一端口的其它 IP 地址设置 HTTP 监听器。例如，如果某个 HTTP 监听器使用 0.0.0.0:80（端口 80 上的所有 IP 地址），则不能同时创建一个使用 1.2.3.4:80 的 HTTP 监听器。

每个 HTTP 监听器还有一个默认虚拟服务器，用于在无法连接到请求中指定的虚拟服务器时，将请求路由到此默认虚拟服务器。

此外，还应在 HTTP 监听器中指定接收方线程 （有时称作接受线程）的数目。接受线程是等待连接的线程，用于接受连接并将其置于队列中以便随后由辅助线程拾取。理想情况下，您需要有足够的接受线程，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。默认线程数为 1。最好是系统上的每个 CPU 都有一个接受线程。如果发现性能问题，可以调整此值。

还应指定是否为 HTTP 监听器启用安全性以及使用哪种类型的安全性 （例如，哪种类型的 SSL 以及哪些加密算法）。

虚拟服务器

要创建虚拟服务器，首先必须确定所需的虚拟服务器类型。既可以使用基于 IP 地址的虚拟服务器，也可以使用基于 URL 主机的虚拟服务器。要创建虚拟服务器，需要指定一个虚拟服务器 ID、一个或多个 HTTP 监听器和一个或多个 URL 主机。

本节包括以下主题：

- 虚拟服务器的类型
- 基于 IP 地址的虚拟服务器
- 基于 URL 主机的虚拟服务器
- 默认虚拟服务器

虚拟服务器的类型

所有虚拟服务器都需要指定 URL 主机。然而，也可以基于 HTTP 监听器将虚拟服务器与 IP 地址关联。如果虚拟服务器的 HTTP 监听器监听特定的 IP 地址，则该虚拟服务器称作基于 IP 地址的虚拟服务器。

如果多个虚拟服务器监听同一 IP 地址，则根据 URL 主机对它们进行区分，这样的虚拟服务器称作基于 URL 主机的虚拟服务器。

当新请求传入时，服务器将根据 IP 地址或 Host 消息头中的值确定将此请求发送到哪个虚拟服务器。首先，它将提取 IP 地址。有关详细信息，请参见第 337 页上的“选择用于处理请求的虚拟服务器”。

基于 IP 地址的虚拟服务器

要使一台计算机具有多个 IP 地址，必须通过操作系统对其进行映射或提供附加卡。要通过操作系统设置多个 IP 地址，请使用 “网络控制面板” (Windows) 或 ifconfig 公用程序 (UNIX)。请注意， ifconfig 的用法因平台而异。有关详细信息，请参见操作系统文档。

要创建基于 IP 地址的虚拟服务器，可创建一个用于监听特定 IP 地址的 HTTP 监听器。然后，关联一个虚拟服务器作为此 HTTP 监听器的默认虚拟服务器。有关虚拟服务器部署方法的详细信息，请参见第 349 页上的 “部署虚拟服务器”。

基于 URL 主机的虚拟服务器

要设置基于 URL 主机的虚拟服务器，可为其提供唯一的 URL 主机。 Host 请求消息头的内容将服务器定向到正确的虚拟服务器。

例如，如果要为用户 *aaa*、 *bbb* 和 *ccc* 设置虚拟服务器以便每个用户都可以拥有单个域名，首先对 DNS 进行配置，使其能够识别每个用户的 URL （www.aaa.com、www.bbb.com 和 www.ccc.com）将解析为所使用的 HTTP 监听器的 IP 地址。然后对每个虚拟服务器的 URL 主机进行正确的设置 （例如， www.aaa.com）。注意，应将主机映射到 /etc/hosts 文件中的 IP 地址。

可以将任何数目的基于 URL 主机的虚拟服务器与 HTTP 监听器关联。

由于基于 URL 主机的虚拟服务器使用 Host 请求消息头将用户定向到正确的页面，因此并非所有客户机软件都可以与它们一起工作。不支持 HTTP Host 消息头的旧客户机软件无法正常运行。这些客户机将接收 HTTP 监听器的默认虚拟服务器。

默认虚拟服务器

系统使用 Host 请求消息头选择基于 URL 主机的虚拟服务器。如果最终用户的浏览器未发送 Host 消息头，或者如果服务器找不到指定的 Host 消息头，则 HTTP 监听器的默认虚拟服务器将处理该请求。

同样，对于基于 IP 地址的虚拟服务器，如果 Sun ONE Application Server 找不到指定的 IP 地址，则 HTTP 监听器的默认虚拟服务器将处理该请求。可以将默认虚拟服务器配置为发送错误信息或特殊文档根目录中的服务器页。

注意 请勿将 HTTP 监听器的默认虚拟服务器与安装服务器时创建的默认虚拟服务器混淆。默认虚拟服务器是默认应用程序服务器实例的虚拟服务器。 HTTP 监听器的默认虚拟服务器是指定为默认虚拟服务器的任何虚拟服务器。

可在创建 HTTP 监听器时指定默认虚拟服务器。可以随时更改此默认虚拟服务器。

obj.conf 文件

默认情况下，每个虚拟服务器都有一个单独的 obj.conf 文件，用于存储虚拟服务器设置。通过管理界面或命令行界面更改设置时，这些更改将在配置文件（包括虚拟服务器的 obj.conf 文件）中自动进行。所有 obj.conf 文件都位于 *instance_dir*/config 目录中。本指南中提到的“obj.conf 文件”是指所有 obj.conf 文件，或所介绍的虚拟服务器的 obj.conf 文件。

无前缀的 obj.conf 文件是一个 Sun ONE Application Server 用来为每个虚拟服务器创建 obj.conf 文件的模板。编辑此文件不会影响任何现有的虚拟服务器，但是可以影响随后创建的任何虚拟服务器。有关直接编辑 obj.conf 文件的详细信息，请参见《*Sun ONE Application Server Administrator's Configuration File Reference*》。

默认情况下，每个活动的 obj.conf 文件都名为 *virtual_server_name*-obj.conf。由于服务器实例的默认虚拟服务器是根据实例命名的，因此首次创建服务器实例时，它的 obj.conf 文件名为 *instance_name*-obj.conf。直接对其中的某个文件进行编辑，或通过管理界面进行编辑可更改虚拟服务器的设置。

选择用于处理请求的虚拟服务器

服务器在可以处理请求前，必须先通过 HTTP 监听器接受该请求，然后再将其定向到正确的虚拟服务器。本节介绍了确定虚拟服务器的方法。

- 如果 HTTP 监听器只配置了一个默认虚拟服务器，则此虚拟服务器将被选中。
- 如果 HTTP 监听器配置了多个虚拟服务器，则将请求 Host 消息头匹配到虚拟服务器的 hosts 属性。如果 Host 消息头不存在，或与 hosts 属性不匹配，则选择 HTTP 监听器的默认虚拟服务器。

如果向 SSL HTTP 监听器配置了虚拟服务器，则在服务器启动时将检查该虚拟服务器的 hosts 属性与证书的主题模式是否匹配，如果不匹配，则将生成警告并写入服务器日志。

确定虚拟服务器后， Sun ONE Application Server 将执行虚拟服务器的 obj.conf 文件。有关服务器如何确定在 obj.conf 文件中执行哪些指令的详细信息，请参见《*Sun ONE Application Server Administrator's Configuration File Reference*》。

文档根目录

文档根目录 （有时称作主文档目录）是包含所有要提供给远程客户机的虚拟服务器文件的中心目录。

使用文档根目录可以轻松地将访问限制到虚拟服务器上的文件。还可以轻松地将文档移动到新目录 （可能位于其它硬盘上）而无需更改任何 URL，因为在 URL 中指定的路径相对于主文档目录。

例如，如果文档目录为 *install_dir*/docs，则请求 （例如 http://www.sun.com/products/info.html）将通知服务器在 *install_dir*/docs/info.html 中查找此文件。如果更改文档根目录 （即移动所有文件和子目录），则只需更改虚拟服务器使用的文档根目录，而不用将所有 URL 均映射到新目录，或以某种方式通知客户机在新目录中查找。

默认 Sun ONE Application Server 实例 (server1) 的文档根目录成为在 server1 应用程序服务器实例中创建的虚拟服务器的文档根目录。可以覆盖创建的每个虚拟服务器的文档根目录。

在虚拟服务器中使用 Sun ONE Application Server 的功能

Sun ONE Application Server 提供了很多可用于虚拟服务器的功能 （例如， SSL 和访问控制）。以下各节介绍了这些功能，并介绍了如何查找更多信息。

本节包括以下主题：

- 在虚拟服务器中使用 SSL
- 使用访问日志文件和服务器日志文件
- 在虚拟服务器中使用访问控制
- 在虚拟服务器中使用 CGI

在虚拟服务器中使用 SSL

如果要在虚拟服务器上使用 SSL，通常应使用基于 IP 地址的虚拟服务器。通常使用端口 443。在基于 URL 主机的虚拟服务器上不宜使用 SSL，因为 Sun ONE Application Server 必须先读取请求，然后才能确定将请求发送到哪个 URL 主机。服务器读取请求与进行安全信息交换的初始信号握手同时发生。

唯一的例外是基于 URL 主机的虚拟服务器都具有相同的 SSL 配置，包括使用 “通配符证书”的相同服务器证书。有关详细信息，请参见 *《Sun ONE Application Server Administrator's Guide to Security》*。

在虚拟服务器中实现 SSL 的方法之一是使用两个 HTTP 监听器，一个使用 SSL 并监听端口 443，另一个不使用 SSL。用户通常通过非 SSL HTTP 监听器访问虚拟服务器。当必须使用安全事务时，可以单击 Web 页上的按钮，以启动安全事务。然后，请求将通过安全的 HTTP 监听器。

由于 SSL 事务的速度比非 SSL 事务慢很多，因此只在必要时使用 SSL 事务。其它情况下将使用更快的非 SSL 连接。

有关在 Sun ONE Application Server 和虚拟服务器中设置和使用安全性的详细信息，请参见 *《Sun ONE Application Server Administrator's Guide to Security》*。有关在虚拟服务器中配置 SSL 的示例图表，请参见第 351 页上的 “示例 2：安全服务器”。

使用访问日志文件和服务器日志文件

访问日志文件用于记录对虚拟服务器的 HTTP 访问。创建新的虚拟服务器时，默认情况下，访问日志文件与应用程序服务器实例的日志文件是同一文件。很多情况下，用户可能希望每个单独的虚拟服务器都有自己的日志文件。要进行此设置，可以更改每个虚拟服务器的日志路径。如果要将所有虚拟服务器访问记录到同一访问日志文件中，可以更改服务器实例的日志设置，以便将虚拟服务器 ID 包含到日志文件中。有关更改应用程序服务器实例日志的详细信息，请参见第 5 章 “使用日志”。

服务器日志文件用于记录提示性信息和错误。创建新的虚拟服务器时，默认情况下，它的日志文件与应用程序服务器实例的日志文件相同。您可以更改每个虚拟服务器的日志文件。

在虚拟服务器中使用访问控制

使用虚拟服务器可以以虚拟服务器为单位设置访问控制。甚至可以对其进行配置，以便每个虚拟服务器可以使用 LDAP 数据库对用户和组进行验证。有关详细信息，请参见 *《Sun ONE Application Server Administrator's Guide to Security》*。

在虚拟服务器中使用 CGI

可以在虚拟服务器上使用 CGI。需要在每个虚拟服务器上设置用于存储 CGI 的目录并设置 CGI 的文件类型。有关 CGI 的详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Web Applications》*。

创建和配置 HTTP 监听器

服务器在可以处理请求前，必须先通过 HTTP 监听器接受该请求，然后再将其定向到正确的虚拟服务器。创建服务器实例 （在安装过程中或以后的操作中）的同时将自动创建一个 HTTP 监听器 http-listener-1。此 HTTP 监听器使用 IP 地址 0.0.0.0 和指定为应用程序服务器端口号的端口号。不能删除默认的 HTTP 监听器。

本节包括以下主题：

- 创建 HTTP 监听器
- 编辑 HTTP 监听器设置
- 删除 HTTP 监听器

创建 HTTP 监听器

使用管理界面创建 HTTP 监听器的步骤：

1. 在左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 单击 “HTTP Listeners”。
3. 单击 “New”。

4. 填写各个字段。

   HTTP 监听器必须有一个唯一的端口号和 IP 地址组合。既可以使用 IPV4 地址，也可以使用 IPV6 地址。如果要为基于 IP 地址的虚拟服务器创建 HTTP 监听器，请为 HTTP 监听器指定一个特殊的 IP 地址。

   “Return Server Name”字段在服务器发送给客户机的 URL 中指定主机名。这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，则此名称应为别名。

   默认虚拟服务器用于在未找到其它虚拟服务器的情况下回复 HTTP 监听器的请求。有关详细信息，请参见第 337 页上的“选择用于处理请求的虚拟服务器”。

   必须先启用 HTTP 监听器，然后才能使其接受请求。

   还可以启用安全性并配置此 HTTP 监听器的高级属性。要指定 IPV6，请在“Family”字段中使用值 inet6。如果此值为 inet6，系统将在服务器日志中的 IPv4 地址前加上前缀 ::ffff:。

5. 单击“OK”。

请注意，创建 HTTP 监听器时，必须在默认虚拟服务器字段中输入一个现有的虚拟服务器。可以使用同服务器实例一同创建的虚拟服务器，然后在创建附加虚拟服务器后，根据需要返回并对其进行更改。

要使用命令行界面创建 HTTP 监听器，请使用 asadmin 公用程序的 create-http-listener 命令。要获取已创建的所有 HTTP 监听器的列表，请使用命令 list-http-listeners。

要创建 HTTP 监听器，请使用以下语法：

asadmin create-http-listener --user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] --address *address* [--instance *instancename*] --listenerport *listener_port* --defaultvs *virtual_server* --servername *server_name* [--family *family*] [--acceptorthreads *acceptor_threads*] [--blockingenabled *blocking_enabled*] [--securityenabled *security_enabled*] [--enabled *enabled*] *listener_id*

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

编辑 HTTP 监听器设置

使用管理界面编辑 HTTP 监听器设置的步骤：

1. 在左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 打开 “HTTP Listeners”。
3. 单击要编辑的 HTTP 监听器。
4. 进行所需的更改，然后单击 “Save”。

有关详细信息，请参见联机帮助。

也可以在命令行界面中使用 asadmin 公用程序编辑 HTTP 监听器。使用 get 命令获取当前设置，并使用 set 命令将其设置为新值。

获取 HTTP 监听器的所有属性值的步骤：

asadmin> get *server_instance*.http-listener.*http_listener_name*.*

例如，要获取默认 HTTP 监听器的值：

asadmin> get server1.http-listener.http-listener-1.*

设置任意属性值的步骤：

asadmin> set *server_instance*.http-listener.*http_listener_name.attribute_name=value*

例如，要将 http-listener-1 的属性 defaultVirtualServer 设置为 server2：

asadmin> set server1.http-listener.http-listener-1.defaultVirtualServer=server2

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

删除 HTTP 监听器

使用管理界面删除 HTTP 监听器的步骤：

1. 在左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 单击 “HTTP Listeners”。
3. 单击要删除的 HTTP 监听器旁边的复选框。
4. 单击 “Delete”。

要使用命令行界面删除 HTTP 监听器，请使用 asadmin 公用程序的 delete-http-listener 命令，该命令的语法如下：

asadmin delete-http-listener ---user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] --instance *instance httplistener_id*

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

创建和配置虚拟服务器

设置 HTTP 监听器后，可以创建和使用虚拟服务器。

本节包括以下主题：

- 创建虚拟服务器
- 编辑虚拟服务器设置
- 删除虚拟服务器

创建虚拟服务器

使用管理界面创建虚拟服务器的步骤：

1. 在左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 单击 “Virtual Servers”。
3. 单击 “New”。
4. 填写必需字段和任何可选字段。
5. 单击 “Save”。

要使用命令行界面创建虚拟服务器，请使用 asadmin 公用程序的 create-virtual-server 命令，该命令的语法如下：

asadmin create-virtual-server --user *username* ---user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] [--instance *instancename*] --hosts *hosts* --mime *mime_types_file* [--httplisteners *http-listeners*] [--defaultwebmodule *default_web_module*] [--configfile *config_file*] [--defaultobj *default_object*] [--state *state*] [--acls *acls*] [--acceptlang *accept_language*] [--logfile *logfile*] [--property (*name=value*)[:*name=value*]*] *virtual_server_id*

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

创建虚拟服务器时，可以输入以下类型的设置：

- 必需的设置
- 可选的常规设置
- Web 应用程序设置
- CGI 设置
- 服务设置的 HTTP 质量

必需的设置

虚拟服务器必需的设置包括名称 (ID) 和一个或多个 URL 主机。

还须指定 MIME 类型文件。 MIME 类型文件包含文件扩展名到文件类型的映射。例如，在 MIME 类型文件中可以指定将所有扩展名为 .cgi 的文件作为 CGI 文件处理。

无需为每个虚拟服务器创建单独的 MIME 类型文件。而是根据需要创建多个 MIME 类型文件，并将它们与虚拟服务器关联。默认的 MIME 类型文件称作 mime1，且文件名为 mime.types。

有关 MIME 类型文件的详细信息，请参见第 330 页上的 “配置 MIME 类型”。

可选的常规设置

除必需字段以外，还可以设置可选字段。

HTTP 监听器

HTTP 监听器可处理到虚拟服务器的连接。必须指定一个 HTTP 监听器，远程客户机才能访问虚拟服务器。

ACL

应用于虚拟服务器的访问控制列表 (ACL)。有关详细信息，请参见 *《Sun ONE Application Server Administrator's Guide to Security》*。

接受语言消息头

客户机在使用 HTTP 1.1 与服务器进行通信时可以发送用于描述接受的语言的消息头信息。可以对服务器进行配置以分析此语言信息。

例如，如果存储日文和英文文档，则可以选择分析接受语言消息头。如果客户机将日文用作接受语言消息头与服务器进行通信，它们将接收到日文版的页面。如果客户机将英文用作接受语言消息头与服务器进行通信，它们将接收到英文版的页面。

如果不支持多语言，则不应分析接受语言消息头。

状态

此状态是虚拟服务器的状态，与应用程序服务器实例处于 “On”还是 “Off”状态无关。如果此页面中显示的虚拟服务器的状态为 “On”，则虚拟服务器只能接受请求 （如果应用程序服务器实例也处于 “On”状态）。

这对于默认应用程序服务器实例的默认虚拟服务器也同样适用。如果关闭应用程序服务器实例，默认虚拟服务器仍设置为 “On”，但不会接受任何连接。

有效状态为 “On”、“Off”或 “Disabled”。设置为 “On”的虚拟服务器可接受连接。

您不能关闭或禁用应用程序服务器实例的默认虚拟服务器。

日志文件

日志文件 （也称作服务器日志文件）用于记录提示性信息和错误。访问日志文件用于记录对虚拟服务器的 HTTP 访问。

文档根目录

文档根目录 （有时称作主文档目录）是包含所有要提供给远程客户机的虚拟服务器文件的中心目录。有关详细信息，请参见第 338 页上的 “文档根目录”。

Web 应用程序设置

Web 应用程序是 Servlet、JavaServer Page、HTML 文档和其它 Web 资源（可能包括图像文件、压缩的归档文件和其它数据）的集合。 Web 应用程序可以打包至一个归档文件 （WAR 文件）中，也可存在于打开的目录结构中。

Sun ONE Application Server 7 支持 Servlet 2.3 API 规范，该规范允许在 Web 应用程序中包含 Servlet 和 JSP。此外， Sun ONE Application Server 7 还支持非 J2EE 应用程序组件 SHTML 和 CGI。

创建虚拟服务器时，可为虚拟服务器指定一个默认 Web 模块。默认 Web 模块可响应所有无法解析为其它已部署到虚拟服务器的 Web 模块的请求。如果不指定默认 Web 模块，系统将使用包含空上下文根的 Web 模块。如果不存在包含空上下文根的 Web 模块，系统将创建并使用默认 Web 模块。

部署 Web 应用程序时，应指定一个虚拟服务器。部署后的 Web 应用程序将显示在可用 Web 模块（可将这些模块选作虚拟服务器的默认 Web 模块）列表中。将 Web 模块指定为虚拟服务器的默认 Web 模块时，虚拟服务器将自动添加到虚拟服务器的 Web 应用程序列表中。

CGI 设置

创建虚拟服务器时创建的 CGI 设置可以控制用户和组 （CGI 程序运行时的身份）、执行 CGI 之前要更改为的目录 (chroot) 和在执行 chroot 后更改为的目录。

在 UNIX 上还可以设置精度，即一个用于确定 CGI 程序相对于服务器优先级的增量。通常，服务器在精度值为 0 的情况下运行，精度增量在 0 （CGI 程序与服务器在同一优先级下运行）到 19 （CGI 程序的运行优先级比服务器低很多）之间。

服务设置的 HTTP 质量

服务质量是指为虚拟服务器设置的性能限制。例如， ISP 可能会根据虚拟服务器允许使用的带宽收取不同数额的费用。这些设置既可以被强制执行 （即只允许使用指定的带宽和最大数目的连接），也可以不被强制执行。如果未强制执行设置，则当超出限制时，系统将向日志文件中记录一个信息。有关详细信息，请参见第 138 页上的 “使用 CLI 管理事务服务”。

除了可以通过管理界面更改这些设置以外，还可以使用命令行界面的 asadmin 公用程序。要使用命令行界面的 asadmin 公用程序配置服务质量，请使用以下命令：

- create-http-qos
- delete-http-qos

这些命令的语法如下：

asadmin create-http-qos --user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] [--virtualserver *virtual_server_id*] [--bwlimit *bandwidth_limit*] [--enforcebwlimit *enforce_bandwidth_limit*] [--connlimit *connection_limit*] [--enforceconnlimit *enforce_connection_limit*] *instance_name*

asadmin delete-http-qos --user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] [--virtualserver *virtual_server_id*] *instance_name*

如果指定一个虚拟服务器，则这些命令将创建或删除此虚拟服务器的服务质量信息。如果不指定虚拟服务器，则此命令将影响服务器实例。

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

编辑虚拟服务器设置

设置虚拟服务器后，即可对其进行编辑。有关编辑虚拟服务器设置的信息，请参见以下主题：

- 使用管理界面编辑常规设置
- 使用命令行界面编辑常规设置
- 编辑 CGI 设置
- 编辑文档处理设置、文档目录设置和 HTTP/HTML 设置

使用管理界面编辑常规设置

可以在创建虚拟服务器时设置虚拟服务器的常规设置。要更改这些设置，请执行以下步骤：

1. 在左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器。
4. 进行所需的更改。

   可以更改的内容包括服务质量设置、添加 ACL、与内容相关的设置 （例如文档根目录和接受语言消息头）、与 CGI 相关的设置 （例如用户、组、精度和 chroot 设置）和默认的 Web 模块。
5. 单击 “Save”。

有关其中某些设置的详细信息，请参见第 343 页上的 “创建和配置虚拟服务器”。或参见联机帮助。

使用命令行界面编辑常规设置

也可以使用命令行界面中的 asadmin 公用程序编辑这些设置。使用 get 命令获取当前设置，并使用 set 命令将其设置为新值。

要获取虚拟服务器的所有属性，请使用以下语法：

asadmin> get *instance_name*.virtual-server.*vserver_id*.*

例如：

asadmin> get server1.virtual-server.vs1.*

如果要获取应用程序服务器实例 server1 的所有属性，请使用以下语法：

asadmin> get server1.virtual-server.server1.*

要设置某个属性 （例如，接受语言消息头），请使用以下语法：

asadmin> set server1.virtual-server.server1.virtualserver.acceptLanguage=false

注意 可以使用命令行界面设置 “General”页面上所有字段的值。但是，您不能使用命令行界面设置其它选项卡页面 （例如 CGI 选项卡页面）的字段值。

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

编辑 CGI 设置

有关编辑 CGI 的信息，请参见 《*Sun ONE Application Server Developers Guide to Web Applications*》。

编辑文档处理设置、文档目录设置和 HTTP/HTML 设置

有关更改这些设置的信息，请参见第 16 章 “管理虚拟服务器内容”。

删除虚拟服务器

删除虚拟服务器的步骤：

1. 在管理界面左侧窗格的应用程序服务器实例中，打开 “HTTP Server”。
2. 单击 “Virtual Servers”。
3. 单击要删除的虚拟服务器旁边的复选框。
4. 单击 “Delete”。

   不能使用管理界面删除所有虚拟服务器。

要使用命令行界面删除虚拟服务器，请使用 asadmin 公用程序的 delete-virtual-server 命令。

用法如下：

asadmin delete-virtual-server --user *username* [--password *password*] [--host *hostname*] [--port *adminport*] [--secure | -s] [--passwordfile *file_name*] --instance *instance virtualserver_id*

有关命令语法的详细信息，请参见命令行界面帮助。有关 asadmin 用法的详细信息，请参见附录 A “使用命令行界面”。

部署虚拟服务器

Sun ONE Application Server 的虚拟服务器体系结构非常灵活。应用程序服务器实例可以拥有任意数量的 HTTP 监听器 （安全的和不安全的）。可以将任意数量的虚拟服务器与这些 HTTP 监听器关联。您既可以使用基于 IP 地址的虚拟服务器，也可以使用基于 URL 主机的虚拟服务器。

每个虚拟服务器都可以 （而非必须）拥有自己的 ACL 列表、 mime.types 文件和 Java Web 应用程序集。

此设计提供了最大的灵活性，使您可以为各种应用程序配置服务器。以下示例介绍了一些适用于 Sun ONE Application Server 的可能配置。

- 示例 1：默认配置
- 示例 2：安全服务器
- 示例 3：内部网宿主
- 示例 4：海量宿主

示例 1：默认配置

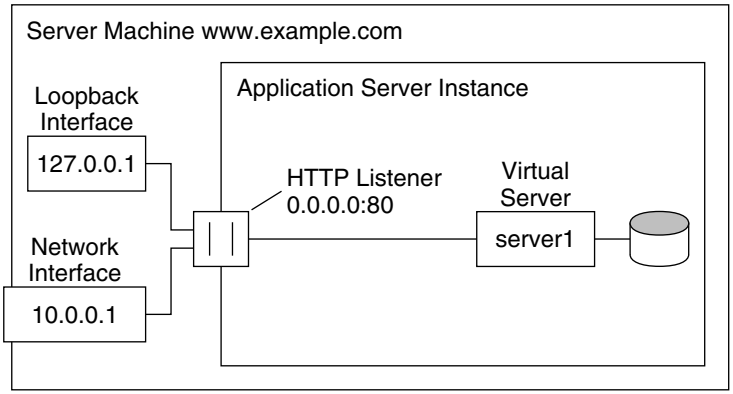
默认配置是一个应用程序服务器实例。它只使用一个 HTTP 监听器监听计算机配置的任何 IP 地址的端口 80、 1024 或所选的任何端口。

本地网络中的某些机制为计算机配置的每个地址都建立了名称 - 地址映射。以下示例中的计算机有两个网络接口：地址 127.0.0.1 上的回送接口 （即使在没有网卡的情况下仍然存在的接口）和地址 10.0.0.1 上的以太网接口。

名称 example.com 通过 DNS 映射为 10.0.0.1。 HTTP 监听器配置为监听计算机配置的任何地址的端口 80 ("0.0.0.0:80")。

由于默认配置中没有基于 IP 地址的虚拟服务器，因此默认监听器是唯一的 HTTP 监听器。所有连接都传送到虚拟服务器 server1。

默认配置



DNS

www.example.com	10.0.0.1

在此配置中，到以下地址的连接将到达服务器并由虚拟服务器 VS1 处理。

- http://127.0.0.1/ （在 example.com 上启动）
- http://localhost/ （在 example.com 上启动）
- http://example.com/
- http://10.0.0.1/

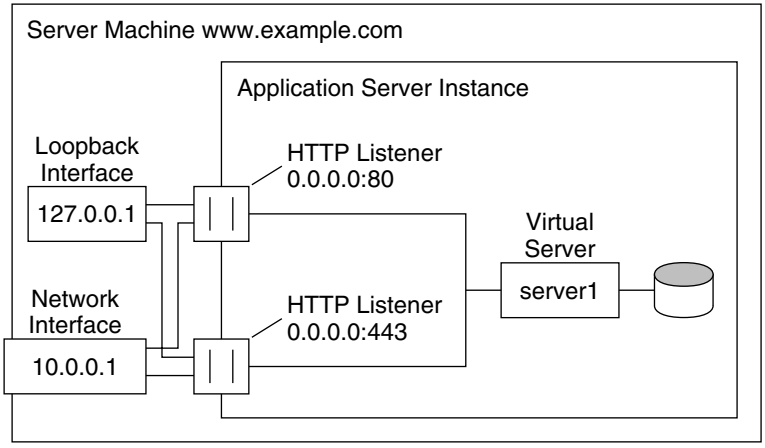
此配置适用于传统的 HTTP 服务器。无需添加附加的虚拟服务器或 HTTP 监听器。通过更改 server1 的设置配置服务器的设置。

示例 2：安全服务器

如果要在默认配置中使用 SSL，只需将 HTTP 监听器更改为安全模式。

还可以添加一个为 0.0.0.0:443 配置的新的安全 HTTP 监听器。虚拟服务器现在有两个 HTTP 监听器，其中一个使用安全 HTTP 监听器，另一个则不使用。现在，服务器将提供同时具有或不具有 SSL 的相同内容，也就是说 http://example.com/ 和 https://example.com/ 提供相同的内容。

安全服务器



DNS

www.example.com	10.0.0.1

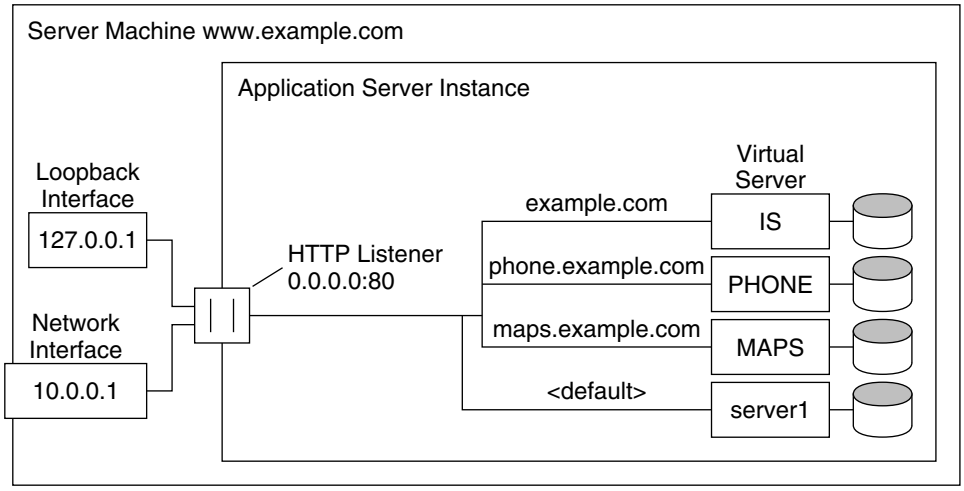
请注意， SSL 参数被附加到 HTTP 监听器。

示例 3：内部网宿主

一个更复杂的 Sun ONE Application Server 配置是服务器在内部网部署中支持几个虚拟服务器。例如，假设您有三个内部站点，员工可在这些站点中查找其他用户的电话号码、查看校园地图以及跟踪发送到信息服务部门的请求的状态。以前 （在本示例中），这些站点以三个不同的计算机为宿主，这些计算机映射为名称 phone.example.com、 maps.example.com 和 is.example.com。

为了将硬件和管理开销减少到最低程度，用户希望将这三个站点合并为计算机 example.com 上的一个应用程序服务器。可以使用以下两种方法进行此设置：使用基于 URL 主机或基于 IP 地址的虚拟服务器。两者都有明显的优点和缺点。

使用基于 URL 主机的虚拟服务器的内部网宿主



DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.1
phone.example.com	10.0.0.1
maps.example.com	10.0.0.1

尽管基于 URL 主机的虚拟服务器易于设置，但它们具有以下缺点：

- 在此配置中支持 SSL 需要使用通配符证书进行非标准设置。有关详细信息，请参见 《*Sun ONE Application Server Administrator's Guide to Security*》。
- 基于 URL 主机的虚拟服务器不能与传统的 HTTP 客户机一起使用。

基于 IP 地址的虚拟服务器的优点包括：

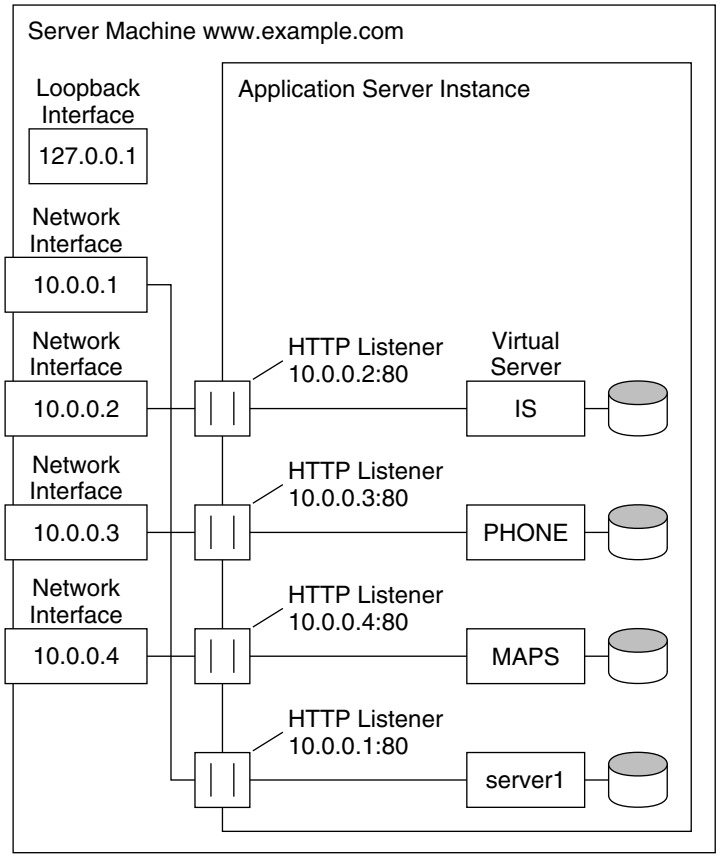
- 它们可以与不支持 HTTP/1.1 Host 消息头的旧客户机一起使用。
- 提供 SSL 支持比较简便。

缺点包括：

- 它们要求更改主机上的配置 （实际或虚拟网络接口的配置）。
- 它们不能缩放至具有上千个虚拟服务器的配置。

两种配置都要求为三种名称设置名称 - 地址映射。在基于 IP 地址的配置中，每个名称都映射为不同的地址。必须对主机进行设置，才能接收所有这些地址上的连接。在基于 URL 主机的配置中，所有名称都可以映射为同一地址 （计算机最初具有的地址）。

使用基于 IP 地址的虚拟服务器的内部网宿主



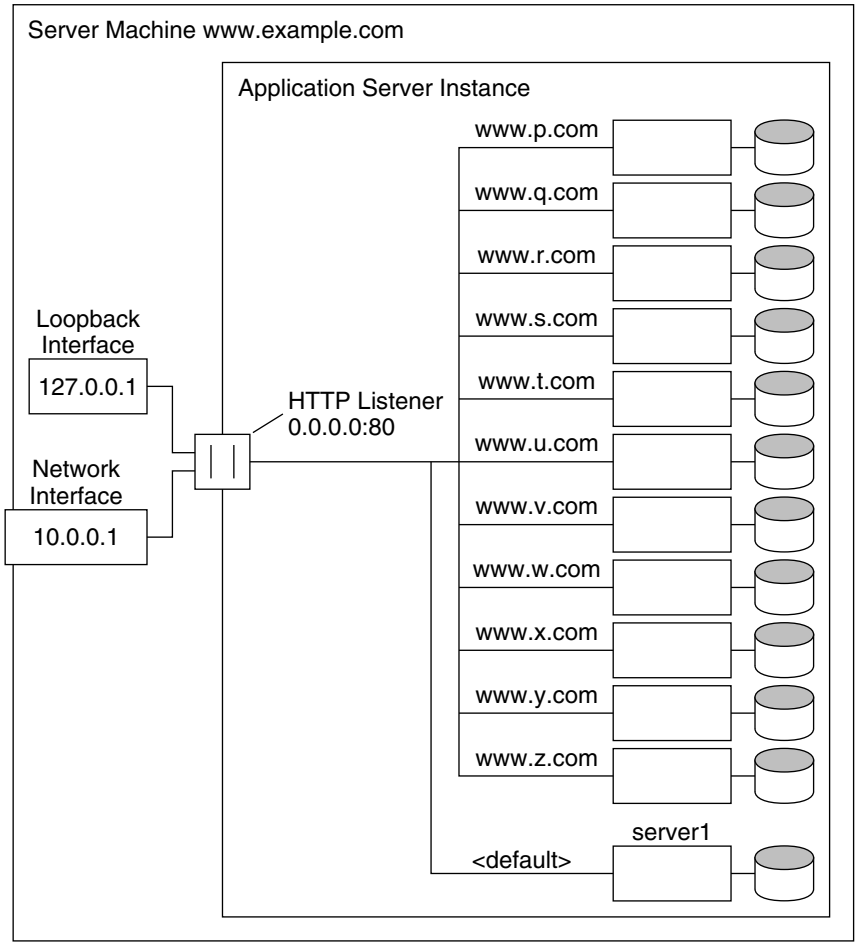
DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.2
phone.example.com	10.0.0.3
maps.example.com	10.0.0.4

示例 4：海量宿主

海量宿主是一个可以启用许多低通信量虚拟服务器的配置。例如， ISP （承载许多低通信量的个人主页）便属于海量宿主。虚拟服务器通常基于 URL 主机。

海量宿主



DNS

www.example.com	10.0.0.1
www.p.com	10.0.0.1
www.q.com	10.0.0.1
www.r.com	10.0.0.1
. . .	
www.z.com	10.0.0.1

注意，默认虚拟服务器 server1 仍然存在。

第 16 章

管理虚拟服务器内容

本章介绍了如何配置和管理虚拟服务器提供的文件。

本章包括以下主题：

- 更改文档根目录
- 设置其它文档目录
- 启用远程文件操作
- 使用 htaccess
- 限制符号链接 (UNIX)
- 自定义用户公有信息目录 (UNIX)
- 设置文档首选项
- 自定义错误响应
- 更改国际字符集
- 设置文档页脚
- 配置 URL 转发
- 设置服务器分析的 HTML
- 设置高速缓存控制指令
- 使用更强大的加密算法

更改文档根目录

文档根目录是您在其中存储了希望供远程客户机使用的所有文件的中心目录。

您在添加虚拟服务器时，使用绝对路径指定文档根目录。有关文档根目录的详细信息，请参见第 338 页上的 “文档根目录”。

使用管理界面更改该文档根目录，以使其使用不同的路径的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “General”标签。
5. 在 “Document Root”字段中，输入目录的绝对路径。

   您需要手动创建该目录。
6. 单击 “OK”。

有关详细信息，请参见联机帮助。

注意 通常，每个虚拟服务器有其自己的文档根目录。

设置其它文档目录

多数时候，虚拟实例或服务器实例的文档位于文档根目录中。但是，有时候您可能希望从文档根目录之外的目录提供文档。这可以通过设置其它文档目录来完成。通过从文档根目录之外的目录提供文档，您无需赋予其他用户访问您的主文档根目录的权限，他们也可以管理文档组。

使用管理界面添加其它文档目录的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Directories”标签。
5. 单击 “Additional Doc Directories”。

6. 选择要映射的 URL 前缀。

   客户机在需要文档时将此 URL 发送到服务器。

7. 指定将 URL 映射到的目录。

8. 单击 “OK”。

有关详细信息，请参见联机帮助。

您应该限制对其它文档目录的访问，这样用户就无法对其进行写入操作。

启用远程文件操作

在启用远程文件操作后，客户机可以在您的服务器上进行以下操作：上载文件、删除文件、创建目录、删除目录、列出目录内容以及重命名文件。虚拟服务器的配置文件 obj.conf 中包括的命令将在您启用远程文件操作时被激活。这些命令激活后，远程浏览器则可以更改服务器上的文档。您应该使用访问控制来限制对这些资源的写入操作，以防止未经授权的更改操作。

注意，启用远程文件操作不会影响使用内容管理系统 （例如 Microsoft Frontpage）。

UNIX：您必须拥有访问文件的正确权限，否则该功能将无法使用；这就是说，文档根目录的用户必须与服务器的用户相同。

使用管理界面来启用远程文件操作的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。

2. 打开 “Virtual Servers”。

3. 单击要编辑的虚拟服务器的名称。

4. 单击 “Doc Directories”标签。

5. 单击 “Remote File Manipulation”。

6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。

7. 选择激活远程文件操作。

8. 单击 “OK”。

有关详细信息，请参见联机帮助。

使用 htaccess

htaccess 文件是动态配置文件，它用于存储配置选项的子集。您可以将 htaccess 文件和 Sun ONE Application Server 标准访问控制结合使用 （标准访问控制总是前于任何 htaccess 访问控制进行应用）。

有关 htaccess 的详细信息，请参见 《*Sun ONE Application Server Administrator's Guide to Security*》。

限制符号链接 (UNIX)

您可以在服务器上限制文件系统链接的使用。文件系统链接是对存储在其它目录和文件系统中的文件的参考。使用参考，用户可以象访问当前目录中的文件一样访问远程文件。文件系统链接有两种类型：

- 硬链接 — 硬链接实际上是指向同一数据块集合的两个文件名；原始文件和链接是相同的。因此，硬链接不能位于不同的文件系统中。
- 符号 （软）链接 — 符号链接包括两个文件，一个是包括数据的原始文件，另一个是指向原始文件的链接文件。符号链接比硬链接更灵活。符号链接可以用于不同的文件系统，还可以链接到目录

有关硬链接和符号链接的详细信息，请参见 UNIX 系统文档。

文件系统链接是用于创建指向位于主文档目录之外文档的指针的简便方法，并且任何人都可以创建文件系统链接。因此，您可能会担心有人会创建指向敏感文件 （例如，机密文件或系统密码文件）的指针。

使用管理界面来限制符号链接的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Directories”标签。
5. 单击 “Symbolic Links”。
6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。
7. 选择是否启用软链接和/或硬链接以及起始的目录。
8. 单击 “OK”。

有关详细信息，请参见联机帮助。

自定义用户公有信息目录 (UNIX)

有时候用户希望维护他们自己的 Web 页面。您可以配置公有信息目录，使服务器上的所有用户可以创建主页和其它文档而不需要您介入操作。

注意 虽然 “User Document Directories”页面显示在 Windows 系统的管理界面中，但是该功能并不可用。

通过使用该系统，客户机可以使用服务器将其识别为公有信息目录的 URL 来访问服务器。例如，假设您选择了前缀 ~ 和目录 public_html。如果收到一个对 http://www.sun.com/~jdoe/aboutjane.html 的请求，服务器将认为 ~jdoe 指向一个用户的公共信息目录。服务器在该系统的用户数据库中查找 jdoe 并找到 Jane 的主目录。服务器随后查找 ~/jdoe/public_html/aboutjane.html。

本节包括以下主题：

- 配置公有信息目录
- 限制内容发布
- 启动时装入整个密码文件

配置公有信息目录

使用管理界面配置虚拟服务器来使用公有目录的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Handling”标签。
5. 编辑 “User Doc Directories”。
6. 选择用户 URL 前缀。

   该前缀通常为 ~，因为该字符是用于访问用户主目录的标准的 UNIX 前缀。
7. 选择服务器要在用户主目录中查找 HTML 文件的子目录。

   通常，目录为 public_html。

8. 指定密码文件。

   服务器需要获知可以在何处查找列出您的系统用户的文件。服务器使用该文件来确定有效的用户名并找到其主目录。如果您将系统密码文件用于此用途，则服务器使用标准库调用来查找用户。或者，您也可以创建另一个用户文件来查找用户。您可以指定该用户文件的绝对路径。

   该文件中的每一行应具有以下结构（/etc/passwd 文件中不需要的元素前面带有 * 符号）：

   username:*:*:groupid:*:homedir:*

9. 选择是否在启动时装入密码数据库。

   有关详细信息，请参见第 362 页上的 “启动时装入整个密码文件”。

10. 单击 “OK”。

有关详细信息，请参见联机帮助。

为用户提供独立的目录的另一种方法是：创建一个映射到所有用户都可以修改的中心目录的 URL。

限制内容发布

某些时候，系统管理员可能希望限制某些用户帐户，使其无法通过用户文档目录来发布内容。要限制一个用户使其无法进行发布，在 /etc/passwd 文件中该用户主目录路径的末尾添加一个斜杠：

jdoe::1234:1234:John Doe:/home/jdoe:/bin/sh

成为：

jdoe::1234:1234:John Doe:/home/jdoe/:/bin/sh

进行修改后， Sun ONE Application Server 将不支持来自该用户的目录的页面。请求该 URI 的浏览器收到“404 File Not Found”错误，并且访问日志将记录一个 404 错误。

如果后来您又决定允许此用户发布内容，则从该 /etc/passwd 条目中删除添加的斜杠符号，然后重新启动应用程序服务器实例。

启动时装入整个密码文件

您还可以选择在启动时装入整个密码文件。如果选择此选项，则服务器在启动时将密码文件装入内存，以便用户可以更快地进行查找。但是，如果密码文件非常大，则这个选项会占用过多的内存。

设置文档首选项

本节包括以下主题：

- 输入索引文件名
- 选择目录索引
- 指定服务器主页
- 指定默认的 MIME 类型

使用管理界面设置文档首选项的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Handling”标签。
5. 单击 “Doc Preferences”。
6. 选择相应的字段值，如下一部分中所述。
7. 单击 “OK”。

下一部分中详细地讨论了您可以设置的首选项。有关详细信息，请参见联机帮助。

输入索引文件名

如果某个文档名称没有在 URL 中指定，服务器将自动显示索引文件。默认的索引文件是 index.html 和 home.html。如果指定的索引文件多于一个，服务器将按照名称在此字段中显示的顺序进行查找，直到找到一个 文件。例如，如果索引文件为 index.html 和 home.html，则服务器将查找 index.html，如果未找到该文件则查找 home.html。

选择目录索引

一个文档目录可能有几个子目录。例如，可能有一个目录称作 products，另一个子目录称作 people 等等。通常，使客户机可以访问这些目录的概述 （或索引）会对他们很有帮助。

服务器通过搜索一个名为 index.html 或 home.html 的索引文件 （您将该文件当作目录内容的概述进行创建和维护）来为目录创建索引。有关详细信息，请参见第 364 页上的 “输入索引文件名”。您可以通过将任何文件命名为默认名称中的一个，来将其指定为目录的索引文件，这意味着您也可以使用 CGI 程序作为索引。

如果索引文件未找到，服务器将在文档根目录中生成一个列出了所有文件的索引文件。

警告 如果服务器在防火墙外，请关闭目录索引，以确保目录结构和文件名称不可访问。

指定服务器主页

当终端用户第一次访问服务器时，他们看到的第一个文件通常被称作主页。通常，这个文件包括了服务器的常规信息和指向其它文档的链接。

默认情况下，服务器查找 “Document Preferences”页面的 “Index Filename”字段中指定的索引文件，并将其用作主页。您也可以指定一个文件作为主页。

指定默认的 MIME 类型

文档发送到客户机时，服务器中的某一部分识别文档的类型，因而客户机可以正确地提供文档。但是，服务器有时无法确定文档的正确类型，因为服务器中没有定义该文档的扩展名。在这种情况下，则发送默认值。

默认值通常为 text/plain，但是您应该将其设置为服务器上存储的最常见的文件类型。以下列出了一些常用的 MIME 类型：

- text/plain
- text/richtext
- image/jpeg
- application/x-tar
- application/x-gzip
- text/html
- image/tiff
- image/gif
- application/postscript
- audio/basic

自定义错误响应

您可以指定自定义错误响应，它用于在虚拟服务器出错时向客户机发送详细消息。您可以指定要发送的文件，或指定要运行的 CGI 程序。

例如，您可以更改服务器收到特定目录的错误时的响应方式。如果客户机尝试连接受访问控制保护的服务器内容，您可以返回一个其中包括如何获得说明信息的错误文件。

在启用自定义错误响应之前，必须创建用于响应错误而发送的 HTML 文件，或者创建用于响应错误而运行的 CGI 程序。创建完成之后，启用管理界面中的响应。

使用管理界面来启用自定义错误响应的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Handling”标签。
5. 单击 “Error Responses”。
6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。

7. 对于每个要更改的错误代码，指定包含该错误响应的文件或 CGI 的绝对路径。
8. 单击“OK”。

有关详细信息，请参见联机帮助。

更改国际字符集

文档的字符集一部分取决于编写文档所用的语言。您可以通过选择资源并输入该资源的字符集，来覆盖用于文档、文档集或目录的客户机的默认字符集设置。

浏览器可以在 HTTP 中使用 MIME 类型的 charset 参数来更改其字符集。如果服务器在其响应中包括此参数，浏览器将相应地更改其字符集。请参见以下示例：

- Content-Type:text/html;charset=iso-8859-1
- Content-Type:text/html;charset=iso-2022-jp

RFC 1700 中指定了以下 charset 名称 （以 x- 开头的名称除外）：

- us-ascii
- iso-8859-1
- iso-2022-jp
- x-sjis
- x-euc-jp
- x-mac-roman

另外，以下是得到认可的 us-ascii 的别名：

- ansi_x3.4-1968
- iso-ir-6
- ansi_x3.4-1986
- iso_646.irv:1991
- ascii
- iso646-us
- us
- ibm367
- cp367

以下是得到认可的 iso_8859-1 的别名：

- latin1
- iso_8859-1
- iso_8859-1:1987
- iso-ir-100
- ibm819
- cp819

使用管理界面更改字符集的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “Doc Handling”标签。
5. 单击 “International Characters”。
6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。
7. 为整个服务器或部分服务器设置字符集。

   如果将该字段保留为空，字符集将设置为 NONE。
8. 单击 “OK”。

有关详细信息，请参见联机帮助。

设置文档页脚

您可以为服务器的某一部分中的所有文档指定文档页脚，它包括最近修改的时间。除了 CGI 脚本的输出信息或经分析的 HTML (.shtml) 文件之外的所有文件都可包含页脚。如果需要将文档页脚显示在 CGI 脚本的输出信息或经分析的 HTML 文件中，则将页脚文本输入到单独的文件，并添加一个代码行或新的服务器端语句，以便将此文件附加到页面输出信息中。

使用管理界面设置文档页脚的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。

4. 单击 “Doc Handling”标签。
5. 单击 “Doc Footer”。
6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。

   如果选择了一个目录，则文档页脚仅在服务器收到该目录或该目录中任何文件的 URL 时才进行应用。
7. 指定希望包含页脚的文件的类型。
8. 指定日期格式。
9. 输入希望显示在页脚中的文本。

   文档页脚字符数目最多为 765。如果希望页脚中包含最近一次修改文档的日期，请键入字符串 :LASTMOD:。

有关详细信息，请参见联机帮助。

配置 URL 转发

URL 转发使您可以将文档请求重定向到另一个服务器。转发 URL 或重定向是服务器通知用户 URL 已经更改 （例如， URL 由于文件已移动到其它目录或其它服务器而更改）的一种方法。还可以使用重定向将对某服务器中某文档的用户请求无缝地发送到另一台服务器中的文档。

例如，如果将 http://www.sun.com/info/movies 转发至前缀 film.sun.com，则 URL http://www.sun.com/info/movies 将重定向到 http://film.sun.com/info/movies。

有时，您可能希望将对一个子目录中所有文档的请求重定向到指定的 URL。例如，如果必须删除某个目录 （因为该目录产生的通信量过大，或者由于某种原因该目录中的文档不再提供），则可以将对其中任何文档的请求定向到一个解释文档为何不再可用的页面。例如，可以将 /info/movies 中的前缀重定向到 http://www.sun.com/explain.html。

使用管理界面配置 URL 转发的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “HTTP/HTML”标签。

5. 单击 “URL Forwarding”。
6. 键入要重定向的 URL 前缀，以及是否要将其重定向到另一个前缀或者静态 URL。
7. 单击 “OK”。

有关详细信息，请参见联机帮助。

设置服务器分析的 HTML

通常情况下， HTML 发送到客户机时就像它存储在磁盘上一样，无需服务器进行任何干预。但是，服务器可以在发送文档之前搜索 HTML 文件以查找特定的命令（也就是说，服务器可以分析 HTML）。如果希望服务器分析这些文件并在文档中插入请求特定的信息或文件，则必须首先启用 HTML 分析。

使用管理界面来设置 HTML 分析的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “HTTP/HTML”标签。
5. 选择 “Parse HTML”。
6. 从资源检出器中选择 “Entire Server”将所做的更改应用到整个虚拟服务器，或者浏览到虚拟服务器中的特定目录。

   如果您选择一个目录，则服务器仅在收到对该目录和该目录中任何文件的 URL 时才分析 HTML。
7. 选择是否激活服务器分析的 HTML。

   您可以为 HTML 文件而不是 exec 标记进行激活（也可以为 HTML 文件和 exec 标记进行激活），这样 HTML 文件可以在服务器上执行其它程序。
8. 选择要分析的文件。

   您可以选择是否只分析扩展名为 .shtml 的文件还是分析所有 HTML 文件（分析所有 HTML 文件将会降低性能）。如果使用的是 UNIX，则也可以选择在执行权限打开的情况下分析 UNIX 文件，但是这样可能存在安全隐患。
9. 单击 “OK”。

有关设置服务器以接受分析的 HTML 的详细信息，请参见联机帮助。

有关使用服务器分析的 HTML 的详细信息，请参见 《*Sun ONE Application Server Developer's Guide to Web Applications*》。

设置高速缓存控制指令

高速缓存控制指令是 Sun ONE Application Server 用于控制代理服务器对哪些信息进行高速缓存的一种方法。使用高速缓存控制指令，可以覆盖代理的默认高速缓存，这样可以使敏感信息以后不会被高速缓存，同时可能也不会被检索到。要使这些指令正常工作，代理服务器必须遵从 HTTP 1.1。

有关 HTTP 1.1 的详细信息，请参见超文本传输协议 -- HTTP/1.1 规范 (RFC 2068)，位于以下位置：

http://www.ietf.org/

使用管理界面来设置高速缓存控制指令的步骤：

1. 在左侧的窗格 （用于该应用程序服务器实例）中，打开 “HTTP Server”。
2. 打开 “Virtual Servers”。
3. 单击要编辑的虚拟服务器的名称。
4. 单击 “HTTP/HTML”标签。
5. 单击 “Cache Control Directives”。
6. 在字段中进行输入。以下列出了响应目录的有效值：
   - **Public。**该响应可以被所有高速缓存进行缓存。这是默认选项。
   - **Private。**该响应只能被专用 （非共享）的高速缓存进行缓存。
   - **No Cache。**该响应不可高速缓存到任何地方。
   - **No Store。**高速缓存不应将请求或响应存储到非易失性存储器中的任何地方。
   - **Must Revalidate。**高速缓存项必须在初始服务器中重新验证。
   - **Maximum Age (sec)。**客户机不接受寿命长于该设置的响应。
7. 单击 “OK”。

有关详细信息，请参见联机帮助。

使用更强大的加密算法

有关设置更强大的加密算法的详细信息，请参见 《*Sun ONE Application Server Administrator's Guide to Security*》。

第 4 部分

附录

附录 A “使用命令行界面”

附录 B “第三方版权声明”

使用命令行界面

本附录提供了在系统提示符下以单模式 （即在命令提示符下一次运行一条命令）、以多模式 （即可以运行多条命令，而不必重新输入环境级别信息）以及在脚本和程序中使用命令行界面 （即 asadmin 公用程序）的有关说明。您可以使用命令行界面代替管理界面屏幕。

本附录包括以下主题：

- 关于命令行界面
- 使用 asadmin
- 安全性考虑
- 并行访问注意事项
- 命令参考

关于命令行界面

本节包括以下主题：

- 关于 asadmin 公用程序
- 关于 Ant 任务
- 关于其它命令行公用程序

关于 asadmin 公用程序

asadmin 公用程序可以执行所有配置和管理任务。您可以使用此公用程序代替管理界面。

关于 Ant 任务

许多开发者使用 Ant 来帮助加快 J2EE 应用程序的开发过程。他们将 asadmin 公用程序编写在 Ant 脚本中以执行某些任务。然后使用 Ant 任务来构建应用程序、部署及取消部署模块和应用程序，以及控制 Sun ONE Application Server。

有关 Ant 任务的详细信息，请参见 *《Sun ONE Application Server Developer's Guide》*。

有关 Ant 的详细信息，请访问 Jakarta Project 站点 http://jakarta.apache.org/ant/。

关于其它命令行公用程序

Sun ONE Application Server 还包含有其它命令行公用程序。下表列出了这些公用程序并给出了每个公用程序的简要说明。

其它命令行公用程序

公用程序	定义
appclient	启动应用程序客户机容器并调用封装在应用程序 JAR 文件中的客户机应用程序。
capture-schema	获取数据库模式和映射信息。
flexanlg	生成有关服务器的统计数据。
htpasswd	创建用户验证文件。
package-appclient	封装应用程序客户机容器库和 JAR 文件。有关详细信息，请参见 *《Sun ONE Application Server Developer's Guide to Clients》*。
verifier	用 DTD 验证 J2EE 部署描述符。有关详细信息，请参见 *《Sun ONE Application Server Developer's Guide》*。
wscompile	提供服务定义接口并生成客户机占位程序或服务器端骨架，或为提供的接口生成一组 WSDL。
wsdeploy	生成可部署的 WAR 文件。

有关这些公用程序的详细信息，请参见各自的联机帮助。

使用 asadmin

asadmin 公用程序有一组用于执行管理任务的命令。您可以使用这些命令执行大多数使用管理界面所能执行的任务。您可以在 *install_dir*/bin 下找到并运行 asadmin 公用程序。在 Windows 上，双击 asadmin.bat 文件，然后 asadmin 公用程序将在命令窗口中启动并以多模式运行。

请注意，使用命令行无法设置某些 HTTP 服务器相关属性和管理服务器属性，这些属性必须使用管理界面进行设置。您可以设置存储在 server.xml 配置文件中的所有属性，但是不能设置存储在 init.conf 或 obj.conf 中的属性。有关配置文件的详细信息，请参见《*Sun ONE Application Server Administrator's Configuration File Reference*》。

有关单个命令的详细信息，请参见第 393 页上的 “命令参考”和命令的帮助信息。

本节包括以下主题：

- 了解命令语法
- 使用单模式和多模式
- 使用交互式和非交互式选项
- 使用环境命令
- 使用密码文件选项
- 以本地或远程方式运行 asadmin
- 使用命令行调用
- 使用换码符
- 使用 get 和 set 命令
- 使用帮助
- 查看输出和错误消息

了解命令语法

asadmin 公用程序的语法如下：

asadmin command -short-option argument --long-option argument operand

Command

命令是所执行的操作或任务。命令区分大小写。

选项

选项指定公用程序如何执行一条命令。选项区分大小写。请注意，短选项之前有一个短划线 (-)，长选项之前有两个短划线 (--)。对于许多选项，既可以使用长选项形式，也可以使用短选项形式。例如， user 可以是 --user 或 -u。有些选项是必需的，有些是可选的。在命令语法中，可选的选项放在方括号内。运行一条命令时，其中必须包含所有必需选项，否则将返回错误消息，并且命令无法执行。

有关可用长选项和短选项的名称列表，请参见第 393 页上的 “命令参考”中的 “短选项和长选项、默认值以及环境变量”表。

大多数选项都要求提供变量值，例如， --port *port_number*。但布尔选项除外，它用于切换某功能的打开或关闭，不需要变量值。

您也可以将选项保存在环境变量中。有关详细信息，请参见第 381 页上的 “使用环境命令”。有关选项的等效环境变量的完整列表，请参见第 424 页上的 “长选项和短选项格式、默认值以及等效的环境变量”。

布尔选项

布尔选项用于切换开或关 （例如，使用 --interactive 将打开交互模式，此时系统会提示您输入选项；使用 --no-interactive 将关闭交互模式）。将 --no- 置于长选项之前可以将该选项切换至关闭状态，而指定短选项的名称则始终会将该选项设置为与默认值相反的值。

您可以将短的布尔选项编组。例如，可以使用 -Ie 来指定交互 （短选项 -I）和回显（短选项 -e）。

Operand

输入空格或制表符可以关闭操作数。在命令语法中，操作数可以按任意顺序出现。您可以使用后面不带任何选项的 -- 来分隔选项和操作数。 -- 之后的任何变量都被视为操作数，即使它们以短划线 (-) 开头。例如，在

asadmin> create-jvm-options --instance server1 -- -Xmx1500m 中，

-XMx1500m 被视为操作数，即使它以短划线开头。

语法示例

asadmin create-instance [--user *admin_user*] [--password *admin_password*] [-H *host_name*] [--port *port_number*] [--sysuser *sys_user*] [--domain *domain_name*] [--local=*true/false*] [--passwordfile *file_name*] [--secure | -s] --instanceport *instance_port instance_name*

在此语法示例中， -H 是 hostname 的短选项， --user 是长选项，其变量为 *admin_user*， *instance_name* 是操作数。可选的选项放在方括号内。

以下示例显示了带有实际值的命令语法，其中没有使用某些可选选项。

asadmin create-instance --user admin --password *password* -H austen --port 4848 --instanceport 1024 server2

使用单模式和多模式

您可以在单模式或多模式下运行 asadmin。在单模式下，可以在命令提示符下一次运行一条命令。在多模式下，可以运行多条命令，而无需重复输入环境级别信息。

使用来自某个文件的输入时，如果命令失败，在单模式下，程序将退出；在多模式下，程序将返回到 asadmin 提示符。

单模式

如果在命令提示符下调用命令行界面中的某条命令，则是在单模式下运行。命令行界面将运行命令，然后退出并返回到命令提示符。要在命令提示符下运行命令行界面，请转至 *install_dir*/appserv/bin 目录，然后在命令提示符下键入命令：

> asadmin *command options arguments*

例如：

> asadmin create-instance --user admin --password *password* -H austen --port 4848 --instanceport 1024 server2

多模式

多模式允许您在开始时设置环境，以便可以运行多条命令而无需重复输入某些环境级别信息（例如，服务器名称、端口和密码）。使用多模式的一个显著优点是可以明显加快命令的输入和执行速度，因为这时 asadmin 驻留在内存中。如果这些环境变量是在操作系统级别设置的，多模式会拾取这些设置。 asadmin 公用程序将一直使用这些设置，直至您更改它们。

在 Windows 上，如果运行 asadmin.bat 文件，您将自动处于多模式下。

在 UNIX 上，要从命令行以多模式启动 asadmin 公用程序，请键入以下内容：

```
> asadmin multimode
```

处于多模式时，命令提示符将变为 asadmin。然后您可以在 asadmin 提示符下键入命令。这时无需使用公用程序名。例如：

```
asadmin> create-instance --user admin --password password -H austen --port 4848
--instanceport 1024 server2
```

键入 exit 或 quit 可以退出多模式。这时将返回到命令提示符。

多重多模式

您也可以使用以下命令在多模式会话中调用多模式：

```
asadmin> multimode
```

当退出第二个多模式环境后，您将返回到初始的多模式环境。

例如，如果您在多模式下管理 server1，并且希望管理 server2 以比较两者，则可以在 server1 的多模式下调用 server2 的多模式。因为不需要退出当前的多模式会话，所以可以保持您的环境设置。当退出用于 server2 的多模式会话后，您将返回到 server1 的多模式环境。

使用交互式和非交互式选项

使用命令行界面时，您可以选择交互或非交互使用模式。如果选择交互模式，系统会在您未指定密码时提示您输入密码。默认情况下，系统将启用交互模式。

您可以通过使用 export 命令设置交互环境变量来禁用和启用交互模式。有关详细信息，请参见 “export 命令中使用的环境变量”表。

在单模式下，您可以在任何情况下使用交互式选项。在多模式下，当您在命令提示符下一次运行一条命令，以及从某个文件中运行时，可以使用交互式选项。但是，多模式下来自某个输入流的命令以及由另一个程序调用的命令无法在交互模式下运行。

使用环境命令

asadmin 公用程序包含一组环境变量，您可以使用环境命令对其进行设置。在多模式下，一旦设置了这些变量，就不需要再重新设置您的环境，直至退出多模式。您也可以在操作系统级别设置这些环境变量。如果这样设置，在进入多模式时，这些变量会被自动拾取，并且在退出多模式后仍会继续保持。

环境变量是 “名称 / 值”对，您可以在任何时候通过分配来设置。环境变量由 AS_ADMIN_ 后跟大写的选项名组成。例如，要设置管理服务器用户，可以键入以下内容：

export AS_ADMIN_USER=*administrator*

其中 *administrator* 是管理员的用户名。

这也使 AS_ADMIN_USER 的值可用于 asadmin 命令，例如：

```
asadmin multimode
asadmin> export AS_ADMIN_HOST=austen
```

只要您处于此多模式会话中，管理服务器的主机名将一直设置为 austen，除非您重新分配。

您也可以在一个步骤中设置和导出多个环境变量的值，例如：

```
asadmin> export AS_ADMIN_PORT=4848 AS_ADMIN_USER=admin
```

要查看当前的环境变量设置，请使用不带变量的 export 命令：

```
asadmin> export
AS_ADMIN_HOST=austen
AS_ADMIN_PORT=4848
AS_ADMIN_USER=admin
```

使用 unset 命令可以从环境中删除变量及其值。例如：

```
asadmin> unset AS_ADMIN_HOST
```

您可以覆盖为某个环境变量设置的值。要执行此操作，可以重新设置该变量，也可以在某个 asadmin 命令中设置一个不同的值。例如：

```
asadmin> export AS_ADMIN_HOST=dickens
asadmin> show-instance-status --host austen instance-name
```

此示例显示了管理服务器主机 austen 中某实例的状态，因为该值覆盖了较早的主机值 dickens。

如果不使用导出的变量，您必须在大多数命令中提供以下选项，或者使用默认值（有关默认值的列表，请参见第 424 页上的 “长选项和短选项格式、默认值以及等效的环境变量”）：

- --host
- --port
- --user
- --password 或 --passwordfile
- --secure=true （如果需要安全保护）
- --instance （如果需要）

下面的 “export 命令中使用的环境变量”表描述了某些可在 export 命令中使用的环境变量。这些变量是最常用的变量，因为它们是专用于设置环境的。第一列显示了环境变量的名称，第二列显示了其用途以及当未设置值时所使用的默认值。有关环境变量的完整列表，请参见第 424 页上的 “长选项和短选项格式、默认值以及等效的环境变量”。

export 命令中使用的环境变量

环境变量	用途
AS_ADMIN_HOST	管理服务器的主机名。如果未指定值，则使用本地主机。
AS_ADMIN_PORT	管理服务器的端口号。如果未指定值，则使用 4848。
AS_ADMIN_USER	运行命令的用户的用户名。
AS_ADMIN_PASSWORD	运行命令的用户的密码。用户名和密码用于验证用户，以确定是否允许此用户管理服务器。这与通过管理界面访问管理服务器时的验证相同。
AS_ADMIN_SECURE	=true （如果需要安全保护）。
AS_ADMIN_INSTANCE	设置 Sun ONE Application Server 的实例。任何使用实例名作为变量的后续命令 （不包括将其用作操作数的命令）都将使用此指定实例。

使用密码文件选项

如果不希望在命令行上键入密码或为密码设置环境变量，可以创建一个密码文件，然后在命令行上使用它作为一个选项。

每个具有 password 选项的命令也都具有 passwordfile 选项，可以用后者代替前者。密码文件包含以下行：

AS_ADMIN_PASSWORD=*value*

AS_ADMIN_ADMINPASSWORD=*value*

AS_ADMIN_USERPASSWORD=*value*

如果使用 passwordfile 选项，文件中的密码将导出到多模式环境中，并且未指定 password 选项的后续命令将使用这些值。

如果在命令行上同时指定了密码 (password) 和密码文件 (passwordfile) 选项，密码文件中的值将导出到多模式环境中，但是当前命令将使用密码选项指定的密码，因为密码选项优先于密码文件。

以本地或远程方式运行 asadmin

asadmin 公用程序通常通过管理服务器来发送其命令。因此，在安装了 Sun ONE Application Server 的系统上不需要运行 asadmin。但是，为使大多数 asadmin 命令能够工作，必须运行管理服务器。

某些命令具有使其以本地方式运行的选项，例如 create-instance 命令。如果在 create-instance 命令中使用 --local=true 选项，则必须在安装了服务器的计算机上运行此命令，但是不需要为创建实例运行管理服务器。

某些命令必须以本地方式运行。例如，start-appserv （用于启动管理服务器及其所有实例）不能以远程方式运行，因为在使用此命令启动管理服务器之前，管理服务器不会运行。

有关管理服务器的详细信息，请参见第 2 章 “设置管理服务器首选项”。

以下命令既可以以本地方式运行也可以以远程方式运行：

- create-instance
- delete-instance
- list-instances
- start-instance
- stop-instance
- display-license
- version
- stop-domain
- restart-instance
- list-domains

对于这些命令，您可以选择以本地方式运行命令而不必指定本地 (local) 选项。默认情况下，如果您在命令语法中为用户、密码、主机或端口指定了值，该命令将被视为远程命令 （虽然您仍然可以为这些选项指定本地值）。如果没有为这些选项指定值，该命令将默认为以本地方式运行。

以本地方式执行命令时，如果有域 (domain) 选项，则必须为该命令指定域选项 （除非只有一个域）；以远程方式执行命令时，将忽略域选项 （如果指定了域选项）。

使用命令行调用

您可以用多种方式调用命令行，如以下主题所述：

- 从命令行使用 asadmin
- 利用来自文件 （脚本）的输入使用 asadmin
- 通过标准输入 （管道）使用 asadmin

从命令行使用 asadmin

使用命令的最简单方式是在命令行上一次运行一条命令，您可以键入公用程序、命令及其选项和变量。在多模式下，您可以键入多条命令而无需重复键入公用程序名和环境选项 （如果已经设置了环境变量）。您可以以交互方式 （系统会提示输入所需的其它输入信息，例如，密码）或非交互方式运行单模式或多模式命令。

有关单模式和多模式的详细信息，请参见第 379 页上的 “使用单模式和多模式”。

有关以交互方式使用命令的详细信息，请参见第 380 页上的 “使用交互式和非交互式选项”。

从命令行运行命令的示例

> asadmin create-instance --user admin --password *password* --host austen --port 4848 --instanceport 1024 server2

该命令完成后，您将返回到操作系统提示符。

利用来自文件 （脚本）的输入使用 asadmin

您可以创建一个其中包含许多 asadmin 命令的脚本。使用脚本，您可以成批处理命令、设置在指定时间运行的工作，还可以简化和自动执行任务。

要调用文件中的脚本，请使用以下语法：

> asadmin multimode --file *filename*

以下是文件中的一个简单脚本的示例，您可以按上述语法调用：

```
# 创建新实例并启动它。
export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=mypassword
AS_ADMIN_HOST=austen AS_ADMIN_PORT=4848
create-instance --instanceport 9000 austen3
start-instance austen3
```

此脚本设置了环境，创建了名为 austen3 的实例并启动了新实例。以数字记号 (#) 开头的行是注释行，执行时将被忽略。

通过标准输入 （管道）使用 asadmin

您可以使用以下语法通过管道为 asadmin 公用程序提供输入：

cat *filename* | asadmin multimode

此语法在 Windows 上可能无效。

使用换码符

在命令语法中使用某些字符 （冒号 :、星号 * 和反斜线 \）时会导致错误，除非使用换码符将其关闭。能否使用换码符取决于所用的平台，以及使用的是单模式还是多模式。

注意 您不需要为 get 和 set 命令中的冒号使用换码符。

本节包括以下主题：

- UNIX 上单模式下的换码符
- Windows 上单模式下的换码符
- 所有平台上单模式下的换码符
- 所有平台上多模式下的换码符

UNIX 上单模式下的换码符

在 Solaris 上，您可以使用双反斜线 (\\) 或双引号 (" ") 为受限制的字符换码。

使用反斜线 (\\) 换码

例如，在使用其值中包含冒号的选项创建 JDBC 连接池时，可以使用反斜线 （示例中假设已经为某些属性设置了环境变量）：

```
asadmin create-jdbc-connection-pool  --instance server1 --datasourceclassname
oracle.jdbc.pool.OracleDataSource --failconnection=true --isconnectvalidatereq=true
--property
url=jdbc\\:oracle\\:thin\\:@asperfsol8\\:1521\\:V8i:user=staging_lookup_app:password
=staging_lookup_app OraclePoollookup
```

使用引号换码

要在上述示例中使用引号，需要将值放在双引号 (") 内，同时使用反斜线为双引号换码。

```
asadmin create-jdbc-connection-pool  --instance server1 --datasourceclassname
oracle.jdbc.pool.OracleDataSource --failconnection=true --isconnectvalidatereq=true
--property
url=\"jdbc:oracle:thin:@asperfsol8:1521:V8i\":user=staging_lookup_app:password=staging
_lookup_app OraclePoollookup
```

您也可以使用第 386 页上的 “所有平台上单模式下的换码符”中介绍的方法。

Windows 上单模式下的换码符

在 Windows 上，您可以使用反斜线字符进行换码。例如，在使用其值中包含冒号的选项创建 JDBC 连接池时，可以使用反斜线 （示例中假设已经为某些属性设置了环境变量）：

```
asadmin create-jdbc-connection-pool  --instance server1 --datasourceclassname
oracle.jdbc.pool.OracleDataSource --failconnection=true --isconnectvalidatereq=true
--property
url=jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i:user=staging_lookup_app:password=stagi
ng_lookup_app OraclePoollookup
```

您也可以使用第 386 页上的 “所有平台上单模式下的换码符”中介绍的方法。

所有平台上单模式下的换码符

在任意平台上，您可以使用反斜线为字符换码，然后将包含换码字符的值放在双引号内。例如，在使用其值中包含冒号的选项创建 JDBC 连接池时，可以如下所示使用换码符 （示例中假设已经为某些属性设置了环境变量）：

```
asadmin create-jdbc-connection-pool  --instance server1 --datasourceclassname
oracle.jdbc.pool.OracleDataSource --failconnection=true --isconnectvalidatereq=true
--property
url="jdbc\:oracle\:thin\:@iasperfsol8\:1521\:V8i":user=staging_lookup_app:password=sta
ging_lookup_app OraclePoollookup
```

所有平台上多模式下的换码符

在多模式下，您可以使用以下语法，其中只需要引号，不需要斜线或反斜线：

```
asadmin> create-jdbc-connection-pool  --instance server1 --datasourceclassname
oracle.jdbc.pool.OracleDataSource --failconnection=true --isconnectvalidatereq=true
--property
url="jdbc:oracle:thin:@asperfsol8:1521:V8i":user=staging_lookup_app:password=staging_lo
okup_app OraclePoollookup
```

使用 get 和 set 命令

使用 get 和 set 命令可以访问和更改 Sun ONE Application Server 中的配置设置。多数情况下， asadmin 命令仅设置了*必需的*属性。使用 set 命令可以更改可选属性的值。

get 和 set 命令

命令	变量	用途
get	(scope) 其中 scope 代表某个属性，并且是一个有效名称。	获取属性的值。
set	(scope=*value*) 其中 scope 代表某个属性，并且是一个有效名称，*value* 是要为该属性设置的值。	设置属性的值。
reconfig	instance-name	运行任何修改了配置文件的命令后，需要运行 reconfig 以便将更改应用到服务器。有关应用更改/重新配置服务器的详细信息，请参见第 73 页上的 “应用对应用程序服务器实例所做的更改”。

您可以在一条命令中用空格分开各个属性以便获取或设置多个属性值。例如：

```
set server1.appReloadPollInterval=20 server1.mime.mime1.file=mime.types
```

您也可以使用 AS_ADMIN_PREFIX 环境变量设置一个将为后续的 get 和 set 命令使用的前缀。之后，在后续 get 和 set 命令中的前缀字符串和操作数之间会自动插入一个句点 (.)。例如：

```
asadmin>export AS_ADMIN_PREFIX=server1
asadmin>get *
server1.locale = en_US
server1.appReloadPollInterval = 2
server1.name = server1
...
```

因为 get 和 set 命令使用句点作为分隔符，所以如果某项目的名称中包含句点，则必须在句点前面使用反斜线换码符 (\)。以下示例显示了一个服务器实例名 server2.sun.com，其中在句点前面使用了反斜线：

```
get server2\.sun\.com.*
```

如果未包含反斜线，将收到一条错误消息。

get 和 set 命令示例

以下示例显示了如何使用 get 命令获取属性的值，以及如何使用 set 命令来设置值。

MDB 容器服务示例

如果应用程序服务器实例是 server1，则可以使用您的环境设置，在多模式下运行以下命令以获取所有 MDB 容器属性的值：

```
asadmin> get server1.mdb-container.*
```

以下是此命令的输出示例，其中显示了属性的当前值：

```
server1.mdb-container.logLevel = null
server1.mdb-container.steadyPoolSize = 10
server1.mdb-container.idleInPoolTimeoutInSeconds = 600
server1.mdb-container.maxPoolSize = 60
server1.mdb-container.monitoringEnabled = false
server1.mdb-container.poolResizeQuantity = 2
```

要仅获取 MDB 容器 monitoringEnabled 属性的值，请使用以下命令：

```
asadmin> get server1.mdb-container.montioringEnabled
```

要将 monitoringEnabled 属性的值设置为 True，请使用以下命令：

```
asadmin> set server1.mdb-container.montioringEnabled=true
```

JMS 资源示例

要配置任何资源，属性应如下所示：

instancename.resource.primary_key_value.attribute_name

例如：

asadmin> get server1.jms-resource.myjms.*

这将获取名为 myjms 的 JMS 目标资源的所有属性。例如：

server1.jms-resource.myjms.resType = javax.jms.Topic
server1.jms-resource.myjms.enabled = true
server1.jms-resource.myjms.name = myjms
server1.jms-resource.myjms.description = null

要获取单个属性 （例如， resType）的值：

asadmin> get server1.jms-resource.myjms.resType

要设置某个属性 （例如， description）：

asadmin> set server1.jms-resource.myjms.description=mydescription

此示例将 description 属性设置为 mydescription。

获取和设置多个值的示例

您可以通过一条命令获取和设置多个值。要同时设置两个属性，请将这两个属性用空格分开。例如：

set server1.appReloadPollInterval=20 server1.mime.mime1.file=mime.types

您也可以使用环境变量 AS_ADMIN_PREFIX 来设置用于若干 get 和 set 命令的前缀。

使用 get 和 set 命令进行监视

您也可以使用 get 和 set 命令监视正在运行的服务器。此外，还可以使用 list 命令进行监视。您可以将 monitor 选项设置为 True 或 False。如果设置为 True，将监视指定的属性。有关使用命令行界面监视 Sun ONE Application Server 的详细信息，请参见第 119 页上的 “使用 CLI 提取监视数据”。

使用帮助

您可以在命令提示符下键入 -h 或 --help 以获得每条 asadmin 命令的帮助。例如，要获得有关 asadmin 的帮助，请键入以下内容：

asadmin --help

您将看到一个包含所有 asadmin 命令的列表。

要获得某个特定 asadmin 命令的帮助，请键入以下内容：

asadmin *command* -h

或

asadmin *command* --help

帮助信息包括命令提要、命令说明、语法信息、示例以及相关命令列表。

请注意，在命令中的任何位置使用 -h 或 --help 都将获得此命令的帮助，而不会执行命令。

您也可以在 UNIX 环境中访问手册页形式的命令行帮助页面。对于非捆绑的安装，请将 *install_dir*/man 添加到 MANPATH 环境变量中。执行此操作后，您便可以访问 Sun ONE Application Server 公用程序的手册页。例如，可以在命令提示符下键入 man asadmin。

查看输出和错误消息

如果命令成功执行，您将看到一条消息，通知您所完成的操作。如果命令失败，将显示一条错误消息。

本节包括以下主题：

- 查看退出状态
- 查看用法

查看退出状态

除了显示一条错误消息外， asadmin 命令还始终会退出并包含一个退出状态。如果命令成功执行，退出状态为 0，如果命令失败，退出状态为 1。

UNIX 上的退出状态

您可以在命令提示符下键入 echo $? 以检查退出状态。

您也可以在脚本中使用退出代码。例如，以下 Korn shell 脚本使用退出状态来指示是否成功执行了 list-instances 命令：

```
#!/bin/ksh
asadmin list-instances
if  [[ $? = 0 ]]
      then
            echo "success"
      else
            echo "error"
fi
```

Windows 上的退出状态

在 Windows 上，您可以在 .bat 脚本中检查退出状态。例如，以下两个脚本分别显示了成功的脚本及返回的输出，以及未成功的脚本及返回的输出：

成功条件

myscript.bat

```
------------
echo off
echo Processing Command
call asadmin list-instances --domain domain1
if not %errorlevel% EQU 0  goto end
echo Command Succesful
goto program-end
:end
echo Command Failed
:program-end
```

输出：

```
Processing Command
admin-server <not running>
server1 <not running>
Command Succesful
```

错误条件

myscript.bat

```
-------------
echo off
echo Processing Command
call asadmin list-instances
if not %errorlevel% EQU 0  goto end
echo Command Succesful
goto program-end
:end
echo Command Failed
:program-end
```

输出：

```
Processing Command
No default domain.Need to enter a domain.
Command Failed
```

查看用法

如果键入命令而不带有变量，将显示一条错误消息，其中包含该命令的语法。例如：

```
asadmin> create-instance

接收的操作数数量无效

用法 :create-instance [--user admin_user] [--password admin_password] [--host localhost]
[--port 4848] [--sysuser sys_user] [--domain domain_name] [--local=false] [--passwordfile
file_name] [secure | -s] --instanceport instanceport instancename
```

安全性考虑

从命令行运行命令行界面时，必须为所有命令提供密码。如果在多模式下运行，则必须在一开始设置环境时提供密码。如果退出多模式，则再次启动多模式时，必须再次设置环境，其中包括密码。您需要使用环境命令来设置密码。有关详细信息，请参见第 381 页上的 “使用环境命令”。

您也可以设置一个密码文件，这样便无需在命令行键入密码。有关详细信息，请参见第 382 页上的 “使用密码文件选项”。

如果没有有效的用户名和密码验证信息，将不会执行命令。

命令行界面具有您为 Sun ONE Application Server 设置的安全性措施。有关 Sun ONE Application Server 中安全性的详细信息，请参见 《*Sun ONE Application Server Administrator's Security Guide*》。

并行访问注意事项

有时可能会有多个用户同时试图使用命令行界面和 / 或管理界面来配置某个服务器。如果发生这种情况，第二个配置请求将排队等候，直至第一个请求完成。如果请求的等候时间过长，将发生超时。

对于某些命令，只有在使用了 reconfig 命令之后更改才会生效。这意味着在将更改应用到服务器之前，可能有多个用户对某个属性进行了编辑。有关 reconfig 的详细信息，请参见第 73 页上的 “应用对应用程序服务器实例所做的更改”。

命令参考

本节包括以下主题：

- 命令列表
- 带点的名称和属性列表
- 长选项和短选项格式、默认值以及等效的环境变量

命令列表

下表显示了所有 asadmin 命令及其用途。有关命令的语法和用法的详细信息，请参见联机帮助。

左列显示了命令名，右列显示了其用途。

asadmin 命令

命令	用途
add-resources	添加一个或多个 JDBC、 JMS 或 JavaMail 类型的资源。
create-acl	创建 ACL （访问控制列表）。
create-authdb	创建验证数据库。
create-auth-realm	创建验证领域。
create-custom-resource	创建自定义资源。
create-domain	创建域。
create-file-user	在密钥文件中创建文件领域用户。
create-http-listener	创建 HTTP 监听器。
create-http-qos	为应用程序服务器实例或虚拟服务器创建 HTTP 服务质量设置。
create-iiop-listener	创建 IIOP 监听器。
create-instance	创建应用程序服务器实例。
create-javamail-resource	创建 Java 邮件资源。
create-jdbc-connection-pool	创建 JDBC 连接池。
create-jdbc-resource	创建 JDBC 资源。
create-jmsdest	创建 JMS （Java 消息服务）目标。
create-jms-resource	创建 JMS 资源。

asadmin 命令

命令	用途
create-jndi-resource	创建 JNDI 资源。
create-jvm-options	在 Java 配置或事件探查器元素中创建 JVM 选项。
create-lifecycle-module	创建生命周期模块。
create-mime	创建 MIME 类型文件。
create-persistence-resource	创建 Persistence Manager 工厂资源。
create-profiler	为 JVM 创建事件探查器。
create-ssl	为 HTTP 监听器、 IIOP 监听器或 IIOP 服务创建 SSL 设置。
create-virtual-server	创建虚拟服务器。
delete-acl	删除 ACL。
delete-authdb	删除验证数据库。
delete-auth-realm	删除验证领域。
delete-custom-resource	删除自定义资源。
delete-domain	删除域。此命令只能以本地方式执行。
delete-file-user	从密钥文件中删除文件领域用户。
delete-http-listener	删除 HTTP 监听器。
delete-http-qos	删除应用程序服务器实例或虚拟服务器的 HTTP 服务质量设置。
delete-iiop-listener	删除 IIOP 监听器。
delete-instance	删除应用程序服务器实例。
delete-javamail-resource	删除 Java 邮件资源。
delete-jdbc-connection-pool	删除 JDBC 连接池。
delete-jdbc-resource	删除 JDBC 资源。
delete-jmsdest	删除 JMS 目标。
delete-jms-resource	删除 JMS 资源。
delete-jndi-resource	删除 JNDI 资源。
delete-jvm-options	从 Java 配置或事件探查器元素中删除 JVM 选项。
delete-lifecycle-module	删除生命周期模块。
delete-mime	删除 MIME 类型文件。

asadmin 命令

命令	用途
delete-persistence-resource	删除 Persistence Manager 工厂资源。
delete-profiler	删除 JVM 事件探查器。
delete-ssl	删除 HTTP 监听器、 IIOP 监听器或 IIOP 服务的 SSL 设置。
delete-virtual server	删除虚拟服务器。
deploy	在应用程序服务器实例中部署 EJB、 WEB、连接器、appclient 或应用程序组件。
deploydir	在应用程序服务器实例中部署目录中的 EJB、 WEB、连接器、 appclient 或应用程序组件。
disable	禁用应用程序服务器实例中已部署的组件。
display-license	显示许可证信息。此命令只能以本地方式执行。
enable	启用 （允许运行）应用程序服务器实例中已部署的组件。
export	导出 asadmin 环境变量的值，以便可以为后续的 asadmin 命令使用。
get	获取属性的值。
help	显示给定命令的帮助信息 （说明、用法、语法、示例），或 asadmin 的一般帮助信息。
install-license	安装许可证文件。此命令只能以本地方式执行。
jms-ping	对 JMS 提供商执行 ping 命令，以查看它是否正在运行。
list	列出可配置的元素。
list-acls	列出应用程序服务器实例的 ACL。
list-authdbs	列出验证数据库。
list-auth-realms	列出验证领域。
list-components	列出服务器实例已部署的组件。
list-custom-resources	列出服务器实例中的自定义资源。
list-domains	列出域。
list-file-users	列出服务器实例中的所有文件领域用户。
list-file-groups	列出指定文件领域用户的所有组。如果未指定用户，则列出服务器实例的所有组。
list-http-listeners	列出服务器实例的 HTTP 监听器。
list-instances	列出域中的应用程序服务器实例。

asadmin 命令

命令	用途
list-iiop-listeners	列出服务器实例的 IIOP 监听器。
list-javamail-resources	列出服务器实例的 Java 邮件资源。
list-jdbc-connection-pools	列出服务器实例的 JDBC 连接池。
list-jdbc-resources	列出服务器实例的 JDBC 资源。
list-jmsdest	列出服务器实例的 JMS 目标。
list-jms-resources	列出服务器实例的 JMS 资源。
list-jndi-resources	列出服务器实例的 JNDI 资源。
list-lifecycle-modules	列出服务器实例的生命周期模块。
list-mimes	列出服务器实例的 MIME 类型文件。
list-persistence-resources	列出服务器实例的 Persistence Manager 工厂资源。
list-profilers	列出服务器实例的 JVM 事件探查器。
list-components	列出部署的模块 （或部署的应用程序的模块）中的一个或多个 EJB （或 Servlet）。
list-virtual-servers	列出服务器实例的虚拟服务器。
multimode	允许您在保持环境设置并保持在 asadmin 中时，执行多条命令。
reconfig	将更改应用到服务器。大多数更改在应用之后才会生效。
restart-instance	重新启动服务器实例。
set	设置属性的值。
show-component-status	显示已部署的组件的状态。
show-instance-status	显示服务器实例的状态 （即它是否正在运行）。
shutdown	关闭管理服务器。
start-appserv	启动管理服务器和所有服务器实例。此命令只能以本地方式执行。
start-domain	启动域中的所有实例。此命令只能以本地方式执行。
start-instance	启动服务器实例。
stop-appserv	停止管理服务器和所有服务器实例。此命令只能以本地方式执行。
stop-domain	停止域中的所有实例。
stop-instance	停止服务器实例。

asadmin 命令

命令	用途
undeploy	删除服务器实例中已部署的组件。
unset	取消为 asadmin 设置的导出的环境变量。
update-file-user	更新现有文件领域用户。
version	显示 Sun ONE Application Server 的版本信息。

带点的名称和属性列表

在使用 get 和 set 命令获取和设置属性时，需要知道 asadmin 用于表示服务、资源等的名称，以便使用此名称获取特定对象的属性。

因为用于使用这些名称的语法会用句点来分隔名称，所以这些名称被称为带点的名称。

asadmin 中使用的带点的名称

下表列出了使用 asadmin 配置项目时所使用的名称。这些名称可以分为以下类别：

- 服务名称
- 资源名称
- 应用程序名称
- 其它名称

服务名称

下表显示了用于获取和设置服务的属性的服务名称：

用于命令行界面的服务名称

服务	带点的名称
JMS 服务配置	jms-service
事务服务配置	transaction-service
MDB 容器配置	mdb-container
EJB 容器配置	ejb-container
Web 容器配置	web-container

用于命令行界面的服务名称

服务	带点的名称
JVM 配置	java-config
ORB 配置	orb 或 iiop-service
ORB 监听器配置	orblistener 或 iiop-listener 请注意， orblistener 或 iiop-listener 本身并不是有效名称，它们都要求在后面提供监听器的名称。例如： ORB 监听器配置 orblistener.<listener name> 或 iiop-listener.<listener name>
日志配置	log-service
安全性配置	security-service
HTTP 配置	http-service

资源名称

下表显示了用于获取和设置资源的属性的资源名称。请注意，这些名称本身并不是有效名称，它们都要求在后面提供资源的名称。

用于命令行界面的资源名称

资源	带点的名称
JDBC 资源配置	jdbc-resource
JNDI 资源配置	jndi-resource
JDBC 连接池资源配置	jdbc-connection-pool
自定义资源配置	custom-resource
JMS 资源配置	jms-resource
Persistence Manager 工厂资源配置	persistence-manager-factory-resource
Java 邮件资源配置	mail-resource

应用程序名称

下表显示了用于获取和设置应用程序相关配置的属性的带点的名称。请注意，这些名称本身并不是有效名称，它们都要求在后面提供应用程序的名称。

用于命令行界面的应用程序名称

应用程序组件	带点的名称
应用程序配置	application
EJB 模块配置	ejb-module
Web 模块配置	web-module
连接器模块配置	connector-module

其它名称

下表显示了可以使用 get 和 set 进行配置的其它项目的带点的名称。请注意，这些名称本身并不是有效名称，它们都要求在后面提供应用程序的名称；例如，http-listener.*listener_name*、 lifecycle-module.*module-name* 等。

用于命令行界面的其它项目的名称

项目	带点的名称
HTTP 监听器	http-listener 或 http-server.http-listener
MIME 类型文件	mime
ACL	acl
虚拟服务器	virtual-server
验证数据库	auth-db
安全性领域	authrealm
生命周期模块	lifecycle-module
事件探查器配置	profiler
服务器配置	server configuration (服务器实例的名称)

属性

以下各节显示了上面列出的每个命名项目的属性并提供了用法示例。请注意，某些属性是只读的 （即只能用于 get 命令，不能用于 set 命令）。

注意 以下各节中的示例假设已在环境变量中定义了用户、密码、主机和端口，因此未在语法中列出这些选项。

jms-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JMS 服务属性

server.xml 名称	**asadmin 名称**
port	port
admin-username	adminUserName
admin-password	adminPassword
log-level	logLevel
enabled	enabled
init-timeout-in-seconds	initTimeoutInSeconds
start-args	startArgs

要获取实例 (server1) 的所有属性：

```
asadmin> get server1.jms-service.*
```

要获取名为 adminPassword 的属性：

```
asadmin> get server1.jms-service.adminPassword
```

要将名为 adminPassword 的属性的值设置为 admin：

```
asadmin> set server1.jms-service.adminPassword=admin
```

transaction-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

事务服务属性

server.xml 名称	asadmin 名称
automatic-recovery	automaticTransactionRecovery
timeout-in-seconds	transactionRecoveryTimeout
tx-log-dir	transactionLogFile
heuristic-decision	heuristicDecision
keypoint-interval	keypointInterval
log-level	logLevel
monitoring-enabled	monitoringEnabled

要获取实例 (server1) 的所有属性：

asadmin> get server1.transaction-service.*

要获取名为 transactionRecoveryTimeout 的属性：

asadmin> get server1.transaction-service.transactionRecoveryTimeout

要将名为 transactionRecoveryTimeout 的属性的值设置为 49：

asadmin> set server1.transaction-service.transactionRecoveryTimeout=49

mdb-container

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

MDB 容器属性

server.xml 名称	asadmin 名称
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
log-level	logLevel
monitoring-enabled	monitoringEnabled

要获取实例 (server1) 的所有属性：

asadmin> get server1.mdb-container.*

要获取名为 steadyPoolSize 的属性：

asadmin> get server1.mdb-container.steadyPoolSize

要将名为 steadyPoolSize 的属性的值设置为 10：

asadmin> set server1.mdb-container.steadyPoolSize=10

ejb-container

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

EJB 容器属性

server.xml 名称	**asadmin 名称**
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
cache-resize-quantity	cacheResizeQuantity
max-cache-size	maxCacheSize
pool-idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
cache-idle-timeout-in-seconds	idleInCacheTimeoutInSeconds
removal-timeout-in-seconds	removalTimeoutInSeconds
victim-selection-policy	victimSelectionPolicy
commit-option	commitOption
log-level	logLevel
monitoring-enabled	monitoringEnabled

要获取实例 (server1) 的所有属性：

asadmin> get server1.ejb-container.*

要获取名为 maxPoolSize 的属性：

asadmin> get server1.ejb-container.maxPoolSize

要将名为 maxPoolSize 的属性的值设置为 12：

asadmin> set server1.ejb-container.maxPoolSize=12

web-container

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

Web 容器属性

server.xml 名称	asadmin 名称
log-level	logLevel
monitoring-enabled	monitoringEnabled （未使用）

要获取实例 (server1) 的所有属性：

asadmin> get server1.web-container.*

要获取名为 logLevel 的属性：

asadmin> get server1.web-container.logLevel

要将名为 monitoringEnabled 的属性的值设置为 WARNING：

asadmin> set server1.web-container.logLevel=WARNING

java-config

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JVM 属性

server.xml 名称	asadmin 名称
java-home	javahome
debug-enabled	debugEnabled
debug-options	debugOptions
javac-options	javacoptions
rmic-options	rmicoptions
classpath-prefix	classpathprefix
server-classpath	serverClasspath
classpath-suffix	classpathsuffix
native-library-path-prefix	libpathprefix

JVM 属性

server.xml 名称	asadmin 名称
native-library-path-suffix	libpathsuffix
env-classpath-ignored	envpathignore

要获取实例 (server1) 的所有属性：

asadmin> get server1.java-config.*

要获取名为 classpathprefix 的属性：

asadmin> get server1.java-config.classpathprefix

要将名为 classpathprefix 的属性的值设置为 com.sun：

asadmin> set server1.java-config.classpathprefix=com.sun

orb 或 iiop-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

ORB/IIOP 服务属性

server.xml 名称	asadmin 名称
message-fragment-size	msgSize
steady-thread-pool-size	minThreads
max-thread-pool-size	maxThreads
max-connections	maxConnections
idle-thread-timeout-in-seconds	idleThreadTimeout
log-level	log
monitoring-enabled	monitor
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback

ORB/IIOP 服务属性

server.xml 名称	asadmin 名称
client-auth-enabled	clientAuth

要获取实例 (server1) 的所有属性：

asadmin> get server1.orb.*

或

asadmin> get server1.iiop-service.*

要获取名为 msgSize 的属性：

asadmin> get server1.orb.msgSize

或

asadmin> get server1.iiop-service.msgSize

要将名为 idleThreadTimeout 的属性的值设置为 300：

asadmin> set server1.orb.idleThreadTimeout=300

或

asadmin> set server1.iiop-service.idleThreadTimeout=300

orblistener 或 iiop-listener

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

IIOP 监听器属性

server.xml 名称	asadmin 名称
id	id
address	address
port	port
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3

IIOP 监听器属性

server.xml 名称	asadmin 名称
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

要获取实例 (server1) 的所有属性：

asadmin> get server1.orblistener.*orb_listener_id*.*

或

asadmin> get server1.iiop-listener.*orb_listener_id*.*

要获取名为 port 的属性：

asadmin> get server1.orblistener.*orb_listener_id*.port

或

asadmin> get server1.iiop-listener.*orb_listener_id*.port

要将名为 address 的属性的值设置为 bluestar：

asadmin> set server1.orblistener.*orb_listener_id*.address=bluestar

或

asadmin> set server1.iiop-listener.*orb_listener_id*.address=bluestar

log-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

日志配置属性

server.xml 名称	asadmin 名称
file	file
level	level
log-stdout	stdout
log-stderr	stderr
echo-log-messages-to-stderr	echoToStderr

日志配置属性

server.xml 名称	asadmin 名称
create-console	createConsole
log-virtual-server-id	LogVirtualServerId
use-system-logging	useSystemLogging

要获取实例 (server1) 的所有属性：

asadmin> get server1.log-service.*

要获取名为 level 的属性：

asadmin> get server1.log-service.level

要将名为 echoToStderr 的属性的值设置为 True：

asadmin> set server1.log-service.echoToStderr=true

security-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

安全性领域配置属性

server.xml 名称	asadmin 名称
default-realm	defaultRealm
default-principal	defaultPrinicpal
default-principal-password	defaultPrinicpalPassword
anonymous-role	anonymousRole
audit-enabled	auditEnabled
log-level	logLevel

要获取实例 (server1) 的所有属性：

asadmin> get server1.security-service.*

要获取名为 anonymousRole 的属性：

asadmin> get server1.security-service.anonymousRole

要将名为 encryptPasswords 的属性的值设置为 True：

asadmin> set server1.security-service.auditEnabled=true

http-service

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

HTTP 服务属性

server.xml 名称	**asadmin 名称**
qos-metrics-interval-in-seconds	qos-metrics-interval-in-seconds
qos-recompute-time-interval-in-millis	qos-recompute-time-interval-in-millis
qos-enabled	qos-enabled
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit

要获取实例 (server1) 的所有属性：

asadmin> get server1.http-service.*

要获取名为 bandwidthLimit 的属性：

asadmin> get server1.http-service.bandwidthLimit

要将名为 qos-enabled 的属性的值设置为 True：

asadmin> set server1.http-service.qos-enabled=true

jdbc-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JDBC 资源属性

server.xml 名称	**asadmin 名称**
jndi-name	name
pool-name	pool
enabled	enabled

JDBC 资源属性

server.xml 名称	asadmin 名称
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.jdbc-resource.*jdbc_resource_name*.*

要获取名为 pool 的属性：

asadmin> get server1.jdbc-resource.*jdbc_resource_name*.pool

要将名为 enabled 的属性的值设置为 True：

asadmin> set server1.jdbc-resource.*jdbc_resource_name*.enabled=true

jndi-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JNDI 资源属性

server.xml 名称	asadmin 名称
jndi-name	name
jndi-lookup-name	LookupName
res-type	resType
factory-class	factory
enabled	enabled
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.jndi-resource.*jndi_name*.*

要获取名为 factory 的属性：

asadmin> get server1.jndi-resource.*jndi_name*.factory

要将名为 factory 的属性的值设置为 com.sun：

asadmin> set server1.jndi-resource.*jndi_name*.factory=com.sun

jdbc-connection-pool

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JDBC 连接池属性

server.xml 名称	**asadmin 名称**
name	name
datasource-classname	dsClassName
res-type	resType
description	description
steady-pool-size	steadyPoolSize
max-pool-size	maxPoolSize
max-wait-time-in-millis	maxWaitTime
pool-resize-quantity	resizeValue
idle-timeout-in-seconds	idleTimeout
transaction-isolation-level	transactionIsolationLevel
is-isolation-level-guaranteed	isIsolationLevelGuaranteed
connection-validation-method	validationMethod
is-connection-validation-required	isValidationRequired
fail-all-connections	failAll
validation-table-name	validationTable

要获取实例 (server1) 的所有属性：

asadmin> get server1.jdbc-connection-pool.*pool_name*.*

要获取名为 dsClassName 的属性：

asadmin> get server1.jdbc-connection-pool.*pool_name*.dsClassName

要将名为 resizeValue 的属性的值设置为 2：

asadmin> set server1.jdbc-connection-pool.*pool_name*.resizeValue=2

custom-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

自定义资源属性

server.xml 名称	asadmin 名称
jndi-name	name
res-type	resType
factory-class	factory
enabled	enabled
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.custom-resource.*jndi_name*.*

要获取名为 factory 的属性：

asadmin> get server1.custom-resource.*jndi_name*.factory

要设置名为 factory 的属性：

asadmin> set server1.custom-resource.*jndi_name*.factory=myclass

jms-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JMS 资源属性

server.xml 名称	asadmin 名称
jndi-name	name
res-type	resType
enabled	enabled
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.jms-resource.*jms_resource_name*.*

要获取名为 res-type 的属性：

asadmin> get server1.jms-resource.*jms_resource_name*.resType

要将名为 enabled 的属性的值设置为 True：

asadmin> set server1.jms-resource.*jms_resource_name*.enabled=true

persistence-manager-factory-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

Persistence Manager 工厂资源属性

server.xml 名称	**asadmin 名称**
jndi-name	jndiName
jdbc-resource-jndi-name	JdbcResourceJndiName
factory-class	factoryClass
enabled	enabled
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.persistence-manager-factory-resource.*jndi_name*

要获取名为 factoryClass 的属性：

asadmin> get server1.persistence-manager-factory-resource.*jndi_name*.factoryClass

要将名为 enabled 的属性的值设置为 True：

asadmin> set server1.persistence-manager-factory-resource.*jndi_name*.enabled=true

mail-resource

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

Java 邮件资源属性

server.xml 名称	**asadmin 名称**
jndi-name	name
enabled	enabled
store-protocol	storeProtocol
store-protocol-class	storeProtocolClass
transport-protocol	transportProtocol
transport-protocol-class	transportProtocolClass
host	host
user	user
from	from
debug	debug
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.mail-resource.*jndi_name*.*

要获取名为 host 的属性：

asadmin> get server1.mail-resource.*jndi_name*.host

要将名为 enabled 的属性的值设置为 True：

asadmin> set server1.mail-resource.*jndi_name*.enabled=true

application

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

应用程序属性

server.xml 名称	**asadmin 名称**
name	name
location	location
virtual-servers	virtualServers
description	description
enabled	enabled

要获取实例 (server1) 的所有属性：

asadmin> get server1.application.*application_name*.*

要获取应用程序中名为 location 的属性：

asadmin> get server1.application.*application_name*.location

要设置名为 location 的属性：

asadmin> set server1.application.*application_name*.location=
"/export/home/as7se/as1/repository/applications/ASConverter"

ejb-module

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

EJB 模块属性

server.xml 名称	**asadmin 名称**
name	name
location	location
description	description
enabled	enabled

要获取实例 (server1) 中独立 EJB 模块的所有属性：

asadmin> get server1.ejb-module.*ejb_jar_name*.*

要获取实例 (server1) 的应用程序中的 EJB 模块的所有属性：

asadmin> get server1.j2ee-application.*application_name*.ejb-module.*ejb_jar_name*.*

或

asadmin>get server1.application.*application_name*.ejb-module.*ejb_jar_name*.*

要获取独立 EJB 模块的名为 location 的属性：

asadmin> get server1.ejb-module.*ejb_jar_name*.location

要获取应用程序中 EJB 模块的名为 location 的属性：

asadmin> get server1.j2ee-application.*application_name*.ejb-module.*ejb_jar_name*.location

或

asadmin> get server1.application.*application_name*.ejb-module.*ejb_jar_name*.location

要设置独立 EJB 模块的名为 location 的属性：

asadmin> set server1.ejb-module.*ejb_jar_name*.location="/export/home/as7se/as1/repository/modules/*ejb_jar_name*"

要设置捆绑到应用程序的 EJB 模块中名为 location 的属性：

asadmin> set server1.j2ee-application.*application_name*.ejb-module.*ejb_jar_name*.location="/export/home/as7se/as1/repository/modules/*ejb_jar_name*"

或

asadmin>set server1.application.*application_name*.ejb-module.*ejb_jar_name*.location="/export/home/as7se/as1/repository/modules/*ejb_jar_name*"

web-module

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

Web 模块属性

server.xml 名称	**asadmin 名称**
name	name
location	location
context-root	contextRoot

Web 模块属性

server.xml 名称	asadmin 名称
virtual-servers	virtualServers
description	description
enabled	enabled

要获取实例 (server1) 中独立 Web 模块的所有属性：

asadmin> get server1.web-module.*web_war_name*.*

要获取实例 (server1) 的应用程序中的 Web 模块的所有属性：

asadmin> get server1.web-module.*application_name*.*web_war_name*.*

要获取独立 Web 模块的名为 location 的属性：

asadmin> get server1.web-module.*web_war_name*.location

要获取应用程序中 Web 模块的名为 location 的属性：

asadmin> get server1.web-module.*application_name*.*web_war_name*.location

要设置独立 Web 模块的名为 location 的属性：

asadmin> set server1.web-module.war-ic.location=
"/export/home/as7se/as1/repository/modules/*web_war_name*"

要设置捆绑到应用程序的 Web 模块中名为 location 的属性：

asadmin> set server1.web-module.*application_name*.*web_war_name*.location=
"/export/home/as7se/as1/repository/modules/*web_war_name*"

connector-module

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

连接器模块属性

server.xml 名称	asadmin 名称
name	name
location	location
description	description
enabled	enabled

要获取实例 (server1) 中独立连接器模块的所有属性：

asadmin> get server1.connector-module.*connector_rar_name*.*

要获取独立连接器模块的名为 location 的属性：

asadmin> get server1.connector-module.*connector_rar_name*.location

要设置独立连接器模块的名为 location 的属性：

asadmin> set server1.connector-module.*connector_rar_name*.location=
"/export/home/as7se/as1/repository/modules/*connector_rar_name*"

http-listener 或 http-server.http-listener

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

HTTP 监听器属性

server.xml 名称	**asadmin 名称**
id	id
address	address
port	port
family	family
acceptor-threads	acceptorThreads
blocking-enabled	blockingEnabled
security-enabled	securityEnabled
default-virtual-server	defaultVirtualServer
server-name	serverName
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

要获取实例 (server1) 的所有属性：

asadmin> get server1.http-listener.*http_listener_name*.*

或

asadmin> get server1.http-server.http-listener.*http_listener_name*.*

要获取名为 factory 的属性：

asadmin> get server1.httpl-listener.*http_listener_name*.address

或

asadmin> get server1.http-server.http-listener.*http_listener_name*.address

要将名为 address 的属性的值设置为 IP 地址 0.0.0.0：

asadmin> set server1.http-listener.*http_listener_name*.address=0.0.0.0

或

asadmin> set server1.http-server.http-listener.*http_listener_name*.address=0.0.0.0

mime

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

MIME 类型属性

server.xml 名称	**asadmin 名称**
id	id
file	file

要获取实例 (server1) 的所有属性：

asadmin> get server1.mime.*mime_name*.*

要获取名为 file 的属性：

asadmin> get server1.mime.*mime_name*.file

要将名为 file 的属性的值设置为 mime.types：

asadmin> set server1.mime.*mime_name*.file=mime.types

acl

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

ACL 属性

server.xml 名称	asadmin 名称
id	id
file	file

要获取实例 (server1) 的所有属性：

asadmin> get server1.acl.*acl_name*.*

要获取名为 file 的属性：

asadmin> get server1.acl.*acl_name*.file

要设置名为 file 的属性：

asadmin> set server1.acl.*acl_name*.file=com/as1.acl

virtual-server

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

虚拟服务器属性

server.xml 名称	asadmin 名称
id	id
http-listeners	httpListeners
config-file	configFile
default-object	defaultObject
accept-language	acceptLanguage
log-file	logFile
default-web-module	defaultWebModule
hosts	hosts
mime	mime
state	state
acls	acls

虚拟服务器属性

server.xml 名称	asadmin 名称
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit
property name="dir" value=	property.dir
property name="nice" value=	property.nice
property name="user" value=	property.user
property name="group" value=	property.group
property name="chroot" value=	property.chroot
property name="docroot" value=	property.docroot
property name="accesslog" value=	property.accesslog

要获取实例 (server1) 的所有属性：

asadmin> get *instance_name*.virtual-server.*vserver_id*.*

例如：

asadmin> get server1.virtual-server.server1.*

要获取虚拟服务器 server1 的名为 httpListeners 的属性：

asadmin> get server1.virtual-server.server1.httpListeners

要将名为 acceptLanguage 的属性的值设置为 False：

asadmin> set server1.virtual-acceptLanguage=false

auth-db

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

验证数据库属性

server.xml 名称	asadmin 名称
id	id
database	database
basedn	basedn

验证数据库属性

server.xml 名称	asadmin 名称
certmaps	certmaps

要获取实例的所有属性：

asadmin> get instancename.virtual-server.*vserver_id*.auth-db.*authdb_id*.*

例如，对于实例 server1、虚拟服务器 server1：

asadmin> get server1.virtual-server.server1.auth-db.*authdb_id*.*

要获取名为 database 的属性：

asadmin> get server1.virtual-server.server1.auth-db.*authdb_id*.database

要设置名为 database 的属性：

asadmin> set server1.virtual-server.server1.auth-db.*authdb_id*.database=Oracle

authrealm

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

验证领域属性

server.xml 名称	asadmin 名称
name	name
classname	classname

要获取实例 (server1) 的所有属性：

asadmin> get server1.authrealm.*authrealm_id*.*

要获取名为 classname 的属性：

asadmin> get server1.authrealm.*authrealm_id*.classname

要设置名为 classname 的属性：

asadmin> set
server1.authrealm.*authrealm_id*.classname=com.sun.as.security.auth.realm.sharedpassword.SharedPasswordRealm

lifecycle-module

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

生命周期模块属性

server.xml 名称	asadmin 名称
name	name
enabled	enabled
class-name	className
classpath	classPath
load-order	loadOrder
is-failure-fatal	isFailureFatal
description	description

要获取实例 (server1) 的所有属性：

asadmin> get server1.lifecycle-module.*lifecycle_module_id*.*

要获取生命周期模块的名为 className 的属性：

asadmin> get server1.lifecycle-module.*lifecycle_module_id*.className

要设置名为 className 的属性：

asadmin> set
server1.lifecycle-module.*lifecycle_module_id*.className=com.*lifecycle_module_id*.lifecycle

profiler

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

JVM 事件探查器配置属性

server.xml 名称	asadmin 名称
name	name
classpath	classPath
native-library-path	nativeLibraryPath
enabled	enabled

要获取实例 (server1) 的所有属性：

asadmin> get server1.profiler.*

要获取名为 enabled 的属性：

asadmin> get server1.profiler.enabled

要将名为 enabled 的属性的值设置为 False：

asadmin> set server1.profiler.enabled=false

server configuration (服务器实例的名称)

下表的左列显示了属性的 server.xml 名称，右列显示了 asadmin 使用的名称。

服务器配置属性

server.xml 名称	**asadmin 名称**
instance-name	name
locale	locale
log-root	logRoot
session-store	sessionStore
application-root	applicationRoot
dynamic-reload-enabled	appDynamicReloadEnabled
dynamic-reload-poll-interval-in-seconds	appReloadPollInterval

要获取实例 (server1) 的所有属性：

asadmin> get server1.*

要获取名为 logRoot 的属性：

asadmin> get server1.logRoot

要设置名为 logRoot 的属性：

asadmin> set server1.logRoot="/space/log"

长选项和短选项格式、默认值以及等效的环境变量

下表列出了命令行选项的长格式和短格式。如果未列出短格式，则表明该选项没有短格式。

短选项和长选项、默认值以及环境变量

选项名	长格式	短格式	默认值	环境变量
acceptlang	--acceptlang			AS_ADMIN_ACCEPT_
acceptorthreads	--acceptorthreads			AS_ADMIN_ACCEPTOR_THREADS
acls	--acls			AS_ADMIN_ACLS
address	--address			AS_ADMIN_ADDRESS
adminpassword	--adminpassword			AS_ADMIN_ADMINPASSWD
adminport	--adminport		4848	AS_ADMIN_ADMINPORT
adminuser	--adminuser			AS_ADMIN_ADMINUSER
basedn	--basedn			AS_ADMIN_BASEDN
blockingenabled	--blockingenabled			AS_ADMIN_BLOCKINGENABLED
bwlimit	--bwlimit			AS_ADMIN_BWLIMIT
certmaps	--certmaps			AS_ADMIN_CERTMAPS
certname	--certname			AS_ADMIN_CERTNAME
classname	--classname			AS_ADMIN_CLASSNAME
classpath	--classpath			AS_ADMIN_CLASSPATH
clientauthenabled	--clientauthenabled			AS_ADMIN_CLIENTAUTHENABLED
configfile	--configfile			AS_ADMIN_CONFIGFILE
connectionpoolid	--connectionpoolid			AS_ADMIN_CONNECTIONPOOLID
connlimit	--connlimit			AS_ADMIN_CONNLIMIT
contextroot	--contextroot			AS_ADMIN_CONTEXTROOT
database	--database			AS_ADMIN_DATABASE
debug	--debug		false	AS_ADMIN_DEBUG
defaultobj	--defaultobj			AS_ADMIN_DEFAULTOBJ

短选项和长选项、默认值以及环境变量

选项名	长格式	短格式	默认值	环境变量
defaultwebmodule	--defaultwebmodule			AS_ADMIN_DEFAULTWEBMODULE
description	--description			AS_ADMIN_DESCRIPTION
discardmanualchanges	--discardmanualchanges	-d	False	AS_ADMIN_DISCARDMANUALCHANGES
echo	--echo	-e	false	AS_ADMIN_ECHO
enabled	--enabled			AS_ADMIN_ENABLED
enforcebwlimit	--enforcebwlimit			AS_ADMIN_ENFORCEBWLIMIT
enforceconnlimit	--enforceconnlimit			AS_ADMIN_ENFORCECONNLIMIT
failconnection	--failconnection		false	AS_ADMIN_FAILCONNECTION
failurefatal	--failurefatal		false	AS_ADMIN_FAILUREFATAL
family	--family			AS_ADMIN_FAMILY
file	--file	-f		AS_ADMIN_FILE
force	--force	-F	true	AS_ADMIN_FORCE
help	--help	-h		AS_ADMIN_HELP
host	--host	-H		AS_ADMIN_HOST
hosts	--hosts			AS_ADMIN_HOSTS
httplistenerid	--httplistenerid			AS_ADMIN_HTTPLISTENERID
httplisteners	--httplisteners			AS_HTTP_LISTENERS
idletimeout	--idletimeout		300	AS_ADMIN_IDLETIMEOUT
instance	--instance	-i	server1	AS_ADMIN_INSTANCE
instanceport	--instanceport			AS_ADMIN_INSTANCEPORT
interactive	--interactive	-I	true	AS_AMDIN_INTERACTIVE
isconnectvalidaterequired	--isconnectvalidaterequired		false	AS_ADMIN_ISCONNECTVALIDATEREQUIRED
jdbcjndiname	--jdbcjndiname	-a		AS_ADMIN_JDBCJNDINAME
jndilookupname	--jndilookupname	-l		AS_ADMIN_JNDILOOKUPNAME
keepmanualchanges	--keepmanualchanges	-k	false	AS_ADMIN_KEEPMANUALCHANGES

短选项和长选项、默认值以及环境变量

选项名	长格式	短格式	默认值	环境变量
loadorder	--loadorder			AS_ADMIN_LOADORDER
local	--local	-l	false	
logfile	--logfile			AS_ADMIN_LOGFILE
maxpoolsize	--maxpoolsize		32	AS_ADMIN_MAXPOOLSIZE
maxwait	--maxwait		6000	AS_ADMIN_MAXWAIT
mime	--mime			AS_ADMIN_MIME
mimefile	--mimefile			AS_ADMIN_MIMEFILE
monitor	--monitor	-m	false	AS_ADMIN_MONITOR
name	--name	-n		AS_ADMIN_NAME
nativelibpath	--nativelibpath			AS_ADMIN_NATIVELIBPATH
objtype	--objtype	-o		AS_ADMIN_OBJTYPE
password	--password	-w		AS_ADMIN_PASSWORD
poolresize	--poolresize		2	AS_ADMIN_POOLRESIZE
port	--port	-p	8000	AS_ADMIN_PORT
prefix	--prefix	-x		AS_ADMIN_PREFIX
printprompt	--printprompt	-P	true	AS_ADMIN_PROMPT
property	--property			AS_ADMIN_PROPERTY
securityenabled	--securityenabled			AS_ADMIN_SECURITYENABLED
servername	--servername			AS_ADMIN_SERVERNAME
ssl2ciphers	--ssl2ciphers			AS_ADMIN_SSL2CIPHERS
ssl2enabled	--ssl2enabled			AS_ADMIN_SSL2ENABLED
ssl3enabled	--ssl3enabled			AS_ADMIN_SSL3ENABLED
ssl3tlsciphers	--ssl3tlsciphers			AS_ADMIN_SSL3TLSCIPHERS
state	--state			AS_ADMIN_STATE
steadypoolsize	--steadypoolsize	8		AS_ADMIN_STEADYPOOLSIZE
storeprotocol	--storeprotocol			AS_ADMIN_STOREPROTOCOL
storeprotocolclass	--storeprotocolclass			AS_ADMIN_STOREPROTOCOLCLASS
tlsenabled	--tlsenabled			AS_ADMIN_TLSENABLED

短选项和长选项、默认值以及环境变量

选项名	长格式	短格式	默认值	环境变量
tlsrollbackenabled	--tlsrollbackenabled			AS_ADMIN_TLSROLLBACKENABLED
transprotocol	--transprotocol		smtp	AS_ADMIN_TRANSPROTOCOL
type	--type			S_ADMIN_TRANSPROTOCOLCLASS
upload	--upload	-U	true	AS_ADMIN_TYPE
url	--url			AS_ADMIN_URL
user	--user	-u		AS_ADMIN_USER
validationmethod	--validationmethod		auto-commit	AS_ADMIN_VALIDATIONMETHOD
validationtable	--validationtable			AS_ADMIN_VALIDATIONTABLE
version	--version	-v		AS_AMDIN_VERSION
virtualserver	--virtualserver			AS_ADMIN_VIRTUALSERVER

第三方版权声明

本产品包含由 RSA Security, Inc 授权的代码。

本产品的一部分是使用 ANTLR 开发的。 ANTLR 1989-2000 由 jGuru.com 开发，网址是 http://www.ANTLR.org 和 http://www.jGuru.com。

本产品包括根据 “Sun 公共许可证”通过 Netbeans Project （网址为 http://www.netbeans.org）开发的软件。此软件 （如果可用）可在 www.netbeans.org 上找到。

本产品包括 Perl。 Perl （如果可用）可在 http://public.ActiveState.com/gsar/APC/ 上找到。

本产品包括通过 Exolab Project (http://www.exolab.org) 开发的软件。

本产品包括通过 DOM4J Project (http://dom4j.org/) 开发的软件。

本产品包括通过 Apache Foundation 开发的软件。版权所有 (c) 1999-2001 Apache Software Foundation.。保留所有权利。

本产品包括由 Regents of University of Californiania 开发的软件。版权所有 (c) 1991, 1993 Regents of University of California.。保留所有权利。

本产品包括由 International Business Machines Corporation 开发的软件。版权所有 (c) 1995-2001 International Business Machines Corporation 和其它公司。保留所有权利。 IBM 代码根据 ICU 许可证获取 , 如下所示。

ICU 许可证 — ICU 1.8.1 和更高版本。

版权和权限声明

词汇表

本词汇表提供了用于说明 Sun ONE Application Server 部署和开发环境的常用术语的定义。要查看标准 J2EE 术语的词汇表，请参见以下位置的 J2EE 词汇表：

http://java.sun.com/j2ee/glossary.html

ACL 访问控制列表。 ACL 是一个文本文件，包含用于标识可访问 Sun ONE Application Server 中存储的资源的用户列表。*参见*通用 ACL。

API 应用程序编程接口。一组指令，计算机程序可使用这些指令与其它用于解释此 API 的软件或硬件进行通信。

Bean 管理的持久性 实体 Bean 变量和数据存储库之间的数据传送。数据访问逻辑通常由开发者使用 Java 数据库连接 (JDBC) 技术或其它数据访问技术提供。*参见*容器管理的持久性。

Bean 管理的事务 其中企业 Bean 的事务分界由开发者通过编程方式控制。*参见*容器管理的事务。

BLOB 二进制大对象。一种用于存储和检索复杂对象字段的数据类型。 BLOB 是二进制或可序列化的对象 （例如，图像），它们转换为大的字节数组，然后被序列化到容器管理的持久性字段中。

BMP *参见* Bean 管理的持久性。

BMT *参见* Bean 管理的事务。

CA 参见认证机构或连接器体系结构。

CKL 损坏的密钥列表。由认证机构发布的列表，用于指示客户机用户或服务器用户不应再信任的任何证书。这种情况下，密钥已经损坏。*参见* CRL。

CLI 命令行界面。允许您在用户提示处键入可执行指令的界面。*参见*管理界面。

CMP *参见*容器管理的持久性。

CMR *参见*容器管理的关系。

CMT *参见*容器管理的事务。

cookie 一个小的信息集合，可以传输至调用 Web 浏览器，然后在此浏览器的每个后续调用中检索此信息集合，以便服务器能够识别来自同一客户机的调用。 Cookie 特定于域，并可利用与应用程序和服务器之间进行的其它数据交换相同的 Web 服务器安全功能。

CORBA 通用对象请求代理体系结构。面向对象的分布式计算的标准体系结构定义。

COSNaming 服务 基于 IIOP 的命名服务。

CosNaming 提供商 为了支持全局 JNDI 名称空间 （可由 IIOP 应用程序客户机访问）， Sun ONE Application Server 包含基于 J2EE 的 CosNaming 提供商，以支持 CORBA 引用 （远程 EJB 引用）的绑定。

create 方法 用于在创建时定制企业 Bean 的方法。

CRL 证书撤回列表。由认证机构发布的列表，用于指示客户机用户或服务器用户不应再信任的任何证书。这种情况下，证书已经撤回。*参见* CKL。

DataSource 对象 DataSource 对象包含一组属性，用于标识和说明它所表示的真实世界数据源。

DN 独特名称。目录服务器中条目名称的字符串表示法。

DN 属性 独特名称属性。一个包含关联的用户、组或对象的标识信息的文本字符串。

DTD 文档类型定义。一组 XML 文件的结构和属性的说明。

EAR 文件 企业归档文件。包含 J2EE 应用程序的归档文件。 EAR 文件的扩展名为 .ear。*参见* JAR 文件。

e-commerce 电子商务。描述通过 Internet 进行的商业活动的术语。

EIS 企业信息系统。也可以解释为封装的企业应用程序、事务系统或用户应用程序。通常简写为 EIS。 EIS 的示例包括：R/3、 PeopleSoft、 Tuxedo 和 CICS。

EJB QL　EJB 查询语言。用于在由容器管理的关系定义的实体 Bean 网络中提供导航的查询语言。

EJB 技术　企业 Bean 是封装了应用程序商业逻辑的服务器端组件。商业逻辑是用于实现应用程序功能的代码。例如，在库存控制应用程序中，企业 Bean 可在 checkInventoryLevel 和 orderProduct 方法中实现商业逻辑。通过调用这些方法，远程客户机可以访问由应用程序提供的库存服务。*参见*容器、实体 Bean、消息驱动的 Bean 和会话 Bean。

EJB 容器　*参见*容器。

ejbc 公用程序　企业 Bean 的编译程序。它检查所有 EJB 类和接口是否与 EJB 规范兼容，并生成占位程序和骨架。

ERP　企业资源规划。支持企业资源规划的多模块软件系统。 ERP 系统通常包含关系数据库和应用程序，用于管理采购、库存、人事、客户服务、装运、财务规划以及商业的其它重要方面。

facade　其中使用应用程序特定的状态会话 Bean 来管理各种企业 Java Bean (EJB)。

factory 类　用于创建 persistence manager 的类。*参见*连接工厂。

finder 方法　允许客户机在全局可用目录中查找 Bean 或 Bean 集合的方法。

FQDN　全限定域名。系统的完整名称，包含主机名和域名。

仅限使用特定计算机的客户机使用，来限制对它们的访问。

HTML　超文本标记语言。一种用于创建可由 Web 浏览器显示的文档的编码标记语言。每个文本块都由指示文本类型的代码括起。

HTML 页面　以 HTML 编码并用于在 Web 浏览器中显示的页面。

HTTP　超文本传输协议。从远程主机获取超文本对象的 Internet 协议。此协议基于 TCP/IP。

HTTP servlet　用于扩展 javax.servlet.HttpServlet 的 servlet。这些 servlet 内置了对 HTTP 协议的支持。与普通 servlet 相对。

HTTPS　超文本传输协议安全。用于安全事务的 HTTP。

IDE　集成开发环境。允许您在一个使用方便的界面中创建、汇编、部署和调试代码的软件。

IIOP Internet Inter-ORB 协议。传输层协议，由 IIOP 上的远程方法调用 (RMI) 和通用对象请求代理体系结构 (CORBA) 使用。

IIOP 监听器 IIOP 监听器是一种监听套接字，用于监听指定的端口，并接收从基于 CORBA 的客户机应用程序传入的连接。

IMAP Internet 消息访问协议。

IP 地址 用于 TCP/IP 网络上的计算机或其它设备的结构化的数字标识符。 IP 地址格式是由句号分隔的四个数字组成的 32 位数字地址。每个数字介于 0 到 255 之间（例如， 123.231.32.2 可以作为一个 IP 地址）。

J2EE Java 2 Enterprise Edition。用于开发和部署基于 Web 的多层企业应用程序的环境。 J2EE 平台由一组服务、应用程序编程接口 (API) 以及提供开发这些应用程序的功能的协议组成。

JAF JavaBeans Activation Framework (JAF) 将对 MIME 数据类型的支持集成到 Java 平台中。参见 Mime 类型。

JAR 文件 Java 归档文件。用于将多个文件聚集到一个文件的文件。 JAR 文件的扩展名为 .jar。

JAR 文件格式 Java 归档文件格式。将多个文件聚集到一个文件且与平台无关的文件格式。可以将多个小应用程序及其所需的组件 （类文件、图像、声音和其它资源文件）捆绑到一个 JAR 文件中，然后在一个 HTTP 事务中将其下载到浏览器。 JAR 文件格式还支持文件压缩和数字签名。

JAR 文件合同 指定企业 Bean 软件包中必须包含的信息的 Java 归档合同。

Java IDL Java 接口定义语言。使用 Java 编程语言编写的 API，用于提供基于标准的兼容性，以及与通用对象请求代理体系结构 (CORBA) 的连接。

JavaBean 可移植的、与平台无关的可重用组件模型。

JavaMail 会话 应用程序与邮件存储库交互时使用的对象。应用程序代码使用 JNDI 服务定位使用 JNDI 名称的 JavaMail 会话资源对象。

JAXM 用于 XML 消息传送的 Java API。允许应用程序使用 SOAP 标准发送和接收面向文档的 XML 消息。这些消息可以带附件，也可以不带附件。

JAXP 用于 XML 处理的 Java API。支持使用 DOM、 SAX 和 XSLT 处理 XML 文档的 Java API。使应用程序能够独立于特定的 XML 处理实现分析和传送 XML 文档。

JAXR 用于 XML 注册表的 Java API。提供统一和标准的 Java API，用于访问不同类型的 XML 注册表。使用户能够建立、部署和发现 Web 服务。

JAX-RPC 用于基于 XML 的远程过程调用的 Java API。使开发者能够根据基于 XML 的 RPC 协议建立可互操作的 Web 应用程序和 Web 服务。

JDBC Java 数据库连接。基于标准的类和接口的集合，使开发者能够创建数据识别组件。 JDBC 以平台无关和供应商无关的方式实现了与数据源的连接和交互。

JDBC 连接池 将用于指定数据库连接的 JDBC 数据源属性与连接池属性组合在一起的池。

JDBC 资源 用于通过现有的 JDBC 连接池将应用程序服务器中运行的应用程序与数据库连接在一起的资源。由 JNDI 名称 （由应用程序使用）和现有 JDBC 连接池的名称组成。

JDK Java 开发工具。包含开发者为 Java 2 Platform 之前的 Java 平台版本生成应用程序所需的 API 和工具的软件。*参见* JDK。

JMS Java 消息服务。用于定义 JMS 客户机访问 JMS 消息服务设备方法的接口和语义的标准集。这些接口提供了 Java 程序创建、发送、接收和读取消息的标准方法。

JMS 管理的对象 预先配置的 JMS 对象 （连接工厂或目标），由管理员创建，供一个或多个 JMS 客户机使用。使用被管理对象使 JMS 客户机具有提供商无关性，即它将 JMS 客户机与提供商的专用部分隔开。这些对象被管理员置于 JNDI 名称空间中，并由 JMS 客户机使用 JNDI 查找进行访问。

JMS 客户机 一个应用程序（或软件组件），用于与其它使用 JMS 消息服务的 JMS 客户机进行交互以交换消息。

JMS 连接工厂 JMS 管理的对象， JMS 客户机用它来创建指向 JMS 消息服务的连接。

JMS 目标 JMS 消息服务中的物理目标，生成的消息传送给此目标以进行路由，然后传输给使用者。此物理目标由 JMS 管理的对象进行标识和封装， JMS 客户机使用此对象指定生成的消息的目标和/或所使用的消息的来源。

JMS 提供商 一种产品，用于为消息传送系统实现 JMS 接口并添加完整产品所需的管理和控制功能。

JMS 消息 JMS 客户机使用的异步请求、报告或事件。消息包含消息头 （可在其中添加附加字段）和正文。消息头指定标准字段和可选属性。消息正文包含传输的数据。

JMS 服务　用于为 JMS 消息传送系统提供传送服务（包括与 JMS 客户机的连接、消息路由和传送、持久性、安全性和记录）的软件。消息服务维护 JMS 客户机向其发送消息以及从中将消息传送给使用的客户机的物理目标。

JNDI　Java 命名和目录接口。这是对 Java 平台的标准扩展，它为启用了 Java 技术的应用程序提供了统一的接口，以便在企业中集成多个命名和目录服务。作为 Java 企业 API 集的一部分， JNDI 实现了到异构的企业命名和目录服务的无缝连接。

JNDI 名称　用于访问已经在 JNDI 命名服务中注册的资源的名称。

JRE　Java 运行时环境。 Java 开发工具 (JDK) 的一个子集，其中包括 Java 虚拟机、Java 核心类以及为使用 Java 编程语言编写的应用程序提供运行时支持的支持文件。*参见* JDK。

JSP　JavaServer Page。使用 HTML 或 XML 标记、 JSP 标记和 Java 代码编写的文本页面。 JSP 将标准浏览器页面的布局功能与编程语言的功能组合在一起。

jspc 公用程序　JSP 编译器。它检查所有 JSP 是否符合 JSP 规范。

JTA　Java 事务 API。允许应用程序和 J2EE 服务器访问事务的 API。

JTS　Java 事务服务。用于处理事务的 Java 服务。

LDAP　轻量目录访问协议。 LDAP 是运行在 TCP/IP 上的开放式目录访问协议。它可缩放至全局大小并可包含数百万个条目。使用提供的 LDAP 服务器 Sun ONE Application Server，可以将所有企业信息存储在一个集中管理的目录信息储存库中，以便任何应用程序服务器可通过网络进行访问。

LDIF　LDAP 数据交换格式。用于以文本格式表示 Sun ONE Application Server 条目的格式。

MDB　*参见*消息驱动的 Bean。

MIME 数据类型　MIME （多用途 Internet 邮件扩展）类型控制系统支持的多媒体文件的类型。

NTV　名称、类型、值。

O/R 映射工具　对象 - 关系 [数据库] 工具。Sun ONE Application Server 管理界面中的映射工具，用于创建实体 Bean 的 XML 部署描述符。

Persistence Manager　负责安装在容器中的实体 Bean 持久性的实体。

POP3　邮局协议。

prepared 命令 SQL 中的一条数据库命令，用于通过预编译使重复执行更加有效。prepared 命令可包含参数。 prepared 语句中包含一个或多个 prepared 命令。

prepared 语句 一个用于封装重复使用的 QUERY、UPDATE 或 INSERT 语句以获取数据的类。 prepared 语句中包含一个或多个 prepared 命令。

QOS QOS（服务质量）是指为服务器实例或虚拟服务器设置的性能限制。例如，如果您是 ISP，则可能希望根据提供的带宽数量对虚拟服务器收取不同数额的费用。可以在两个方面进行限制：带宽数量和连接数目。

RAR 文件 资源归档文件。包含资源适配器的 JAR 归档文件。

RDB 关系数据库。

RDBMS 关系数据库管理系统。

ResultSet 一个用于实现 java.sql.ResultSet 接口的对象。 ResultSets 用于封装从数据库或其它表格数据源中检索的行集合。

RMI 远程方法调用。 Java API 标准集，使开发者可以编写可向远程进程传递对象的远程接口。

RMIC 远程方法调用编译器。

RowSet 一个对象，用于封装从数据库或其它表格数据源中检索的行集合。RowSet 扩展了 java.sql.ResultSet 接口，使 ResultSet 可以用作 JavaBeans 组件。

RPC 远程过程调用。用于访问远程对象或服务的机制。

SAF 服务器应用程序函数。参与请求处理和其它服务器活动的函数

servlet Servlet 类的实例。 servlet 是运行在服务器中的可重用应用程序。在 Sun ONE Application Server 中， servlet 用作应用程序中每个交互的中心分发程序，用于执行表示逻辑、调用商业逻辑以及调用或执行表示布局。

servlet 引擎 一个用于处理所有 servlet 元功能的内部对象。从整体上看，是一组为 servlet 提供服务的过程，包括实例化和执行。

servlet 运行程序 servlet 引擎的一部分，它使用请求对象和响应对象调用 servlet。*参见* servlet 引擎。

SMTP 简单邮件传输协议

SNMP SNMP （简单网络管理协议）是用于交换有关网络活动的数据的协议。利用 SNMP，可以在被管理的设备和网络管理站 (NMS) 之间传送数据。被管理的设备即运行 SNMP 的任何设备：主机、路由器、 Web 服务器和网络上的其它服务器。NMS 是用于远程管理网络的计算机。

SOAP 简单对象访问协议 (SOAP) 组合使用基于 XML 的数据结构和超文本传输协议 (HTTP) 定义一个标准方法，用于调用分布在 Internet 中不同操作环境下的对象。

SQL 结构化查询语言。关系数据库应用程序中常用的语言。SQL2 和 SQL3 表示语言版本。

SSL 安全套接字层。用于在 Internet 上提供安全通信的协议。

Sun ONE Application Server 轻量目录访问协议 (LDAP) 的 Sun ONE 版本。Sun ONE Application Server 的每个实例都使用 Sun ONE Application Server 存储共享的服务器信息 （包括有关用户和组的信息）。*参见* LDAP。

Sun ONE Message Queue 用于实现 Java 消息服务 (JMS) 开放标准的 Sun ONE 企业消息传送系统：它是一个 JMS 提供商。

TLS 传输层安全性。在传输层提供加密和认证的协议，以便数据可在安全的通道中流动，而不要求对客户机和服务器应用程序进行明显的更改。

UDDI 通用说明、发现和集成。提供世界范围的 Web 服务注册表用于发现和集成。

URI 通用资源标识符。说明某个域中的特定资源。本地描述为基本目录的子集，以便 /ham/burger 成为基本目录，且 URI 指定 toppings/cheese.html。相应的 URL 应为 http://domain:port/toppings/cheese.html。

URL 统一资源定位器。唯一标识 HTML 页或其它资源的地址。 Web 浏览器使用 URL 指定要显示的页。 URL 说明了传输协议 （例如， HTTP、 FTP）、域 （例如，www.my-domain.com）和 URI （可选）。

WAR 文件 Web 归档文件。包含 Web 模块的 Java 归档文件。WAR 文件的扩展名为 .war。

Web 服务 通过 Web 提供的服务。一种独立的、自行描述的模块应用程序，可以接收和处理 Internet 或内部网中的系统请求，并返回响应。

Web 服务器 用于存储和管理 HTML 页和 Web 应用程序 （而不是整个 J2EE 应用程序）的主机。 Web 服务器响应 Web 浏览器的用户请求。

Web 服务器插件　Web 服务器插件是 HTTP 反向代理插件，用于指示 Sun One Web Server 或 Sun ONE Application Server 将某些 HTTP 请求转发给其它服务器。

Web 高速缓存　一个 Sun ONE Application Server 功能，使 servlet 或 JSP 在特定的时间范围内将结果缓存，以提高性能。在此时间范围内对此 servlet 或 JSP 的后续调用将获得高速缓存的结果，以便 servlet 或 JSP 无需再次执行。

Web 连接器插件　Web 服务器的扩展，使其可以与 Sun ONE Application Server 进行通信。

Web 模块　单独部署的 Web 应用程序。*参见* Web 服务。

Web 容器　*参见*容器。

Web 应用程序　servlet、JavaServer Page、HTML 文档和其它 Web 资源的集合，其中可包含图像文件、压缩的归档文件以及其它数据。Web 应用程序可以打包至一个归档文件（WAR 文件）中，或存在于打开的目录结构中。Sun ONE Application Server 还支持某些非 Java Web 应用程序技术（例如，SHTML 和 CGI）。

WSDL　Web 服务说明语言。一种基于 XML 的语言，用于以标准化的方式定义 Web 服务。它从本质上说明了 Web 服务的三个基本属性：Web 服务的定义、访问此 Web 服务的方法以及此 Web 服务的位置。

XA 协议　一个用于分布式事务的数据库行业标准协议。

XML　可扩展标记语言。一种使用 HTML 样式标记标识文档中使用的各种信息以及格式化文档的语言。

安全套接字层　*参见* SSL。

安全性　确保应用程序资源只能由授权客户机访问的屏蔽机制。

版本更新　*参见*动态重新装入。

备份存储库　数据存储库，通常是指文件系统或数据库。备份存储库可由后台线程（或清除器线程）监视，以删除不需要的条目。

本地会话　仅对一个服务器可见的用户会话。

本地接口　一个接口，为与会话 Bean 或实体 Bean 处于相同 Java 虚拟机 (JVM) 中的客户机提供访问此 Bean 的机制。

本地事务　对于一个数据库而言是本地的并且被限制在一个进程中的事务。本地事务仅对一个后端有效。本地事务通常使用 JDBC API 进行分界。*参见*全局事务。

本地数据库连接　本地连接中的事务上下文对于当前进程和当前数据源而言是本地的，并且不跨进程和数据源分发。

表　一个命名组，包含数据库的行和列中的相关数据。

表示布局　Web 页内容的格式。

表示逻辑　在应用程序中创建页面的操作，包括处理请求、生成响应内容以及为客户机格式化页面。通常由 Web 应用程序处理。

部署　向应用程序服务器分发应用程序所需的文件，以使应用程序可以在应用程序服务器中运行的过程。*参见*证书。

部署描述符　与每个用于描述部署方法的模块和应用程序一同提供的 XML 文件。部署描述符指示部署工具使用特定的容器选项部署模块或应用程序，并说明部署者必须满足的特定配置要求。

参数　由客户机发送的名称/值对（包括窗体字段数据、HTTP 消息头信息等内容），并封装在请求对象中。与属性相对。广义上讲，是指 Java 方法或可在数据库中使用的命令的参数。

程序安全性　在代码中明确控制安全性，而不是允许组件的容器 （例如， Bean 的容器或 servlet 引擎）处理安全性的过程。与声明的安全性相对。

程序员分界的事务　*参见* Bean 管理的事务。

池处理　提供一些预先配置的资源，以提高性能的过程。如果资源位于池中，则组件可以使用池中的现有实例，而不必创建新的实例。在 Sun ONE Application Server 中，数据库连接、 servlet 实例和企业 Bean 实例都可以放入池中。

持久性　对于企业 Bean 而言，是指在实体 Bean 的实例变量和基础数据库之间传输实体 Bean 状态的协议。与瞬时性相对。对于会话而言，是指会话存储机制。

持久状态　其中，对象的状态保存在持久存储器 （通常为数据库）中。

窗体操作处理程序　在 servlet 或应用程序逻辑中专门定义的方法，可根据窗体中的命名按钮执行操作。

存储过程　以 SQL 编写并存储在数据库中的语句块。可以使用存储过程执行任何类型的数据库操作（例如，修改、插入或删除记录）。使用存储过程可以减少网络上发送的信息数量，从而提高数据库性能。

代理　用于管理 JMS 消息路由、传送、持久性、安全性以及记录的 Sun ONE Message Queue 实体，它提供了一个界面，使管理员能够监视和调整性能以及资源的使用情况。

代理　一种使用对象的构成作为实现策略的面向对象的技术。一个对象（负责操作结果）代理另一个对象（其被代理者）的实现。例如，类加载器通常代理某些类对其父类进行的装入操作。

单个签入　指可以在单个虚拟服务器实例中的多个 J2EE 应用程序间共享用户验证状态的情况。

点对点传送模型　生成方将消息插入特定的队列；使用方从用于保存其消息的队列中提取消息。消息仅发送给一个消息使用者。

动态重新部署　在不重新启动服务器的情况下重新部署组件的过程。

动态重新装入　在不重新启动服务器的情况下更新和重新装入组件的过程。默认情况下，servlet、JavaServer Page (JSP) 和企业 Bean 组件可以动态重新装入。也称作版本更新。

独特名称　*参见* DN、 DN 属性。

队列　由管理员创建的对象，用于实现点对点传送模型。队列通常可用于存放消息，即使使用其消息的客户机处于不活动状态。队列可作为消息生成者和使用者之间的中转站。

对象持久性　*参见*持久性。

钝化作用　一种从内存中释放 Bean 的资源、但不销毁此 Bean 的方法。这样，Bean 将具有持久性，且可在不耗费实例化开销的前提下被重新调用。

发布 / 订阅传送模型　发布者和订阅者通常匿名，并且可以动态发布或订阅主题。系统将来自某个主题的多个发布者的消息分发给它的多个订阅者。

防火墙　允许网络管理员限制通过网络的信息流，以增强安全性的电子边界。

访问控制　用于通过控制 Sun ONE Application Server 的访问对象来确保 Sun ONE Application Server 安全性的方法。

分布式事务　可应用到位于不同服务器中的多个异构数据库的单个事务。

封装　用于本地化模块中的内容。由于对象封装了数据和实现，因此对象的用户可以将对象看作是能够提供服务的黑箱。可以添加、删除或更改实例变量和方法，但如果对象提供的服务保持不变，则使用对象的代码可以继续使用此对象，而不必被重新编写。

服务器实例　Sun ONE Application Server 可以在同一台计算机中的同一安装中包含多个实例。每个实例都有自己的目录结构、配置和部署的应用程序。每个实例还可以包含多个虚拟服务器。*参见*虚拟服务器。

高速缓存的行集合　CachedRowSet 对象允许您在数据源中检索数据，然后在检查和修改数据的同时，从数据源中分离此数据。高速缓存的行集合将同时记录检索到的初始数据和应用程序对此数据所做的任何更改。如果应用程序尝试更新初始数据源，则行集合将被重新连接到数据源，且只将已更改的行重新合并到数据库。

高速缓存控制指令　Sun ONE Application Server 通过高速缓存控制指令控制代理服务器高速缓存的信息。使用高速缓存控制指令，可以覆盖代理的默认高速缓存，以防以后将敏感信息高速缓存，或对其进行检索。代理服务器必须遵循 HTTP 1.1 协议，才能使这些指令正常运行。

隔离级别　*参见*事务隔离级别。

公共密钥加密　一种加密格式，其中的每个用户都有一个公共密钥和一个专用密钥。消息通过接收方的公共密钥以加密方式发送；接收者使用专用密钥将消息解密。使用此方法可避免除此用户以外的任何人获取专用密钥。

故障转移　一个恢复过程，使 Bean 能够在服务器崩溃时，不出现任何丢失。

管理服务器　应用程序服务器实例，专门用于提供 Sun ONE Application Server 的管理功能，包括部署、基于浏览器的管理以及从命令行界面 (CLI) 和集成开发环境 (IDE) 进行的访问。

管理界面　用于配置和管理 Sun ONE Application Server 的基于浏览器的窗体集。*参见* CLI。

管理信息库 (MIB)　树状结构，用于定义主 SNMP 代理能够访问的变量。MIB 提供了对 HTTP 服务器的网络配置、状态和统计数据的访问。使用 SNMP 可以从网络管理工作站 (NMS) 查看此信息。*参见*网络管理站 (NMS) 和 SNMP。

管理域　多个管理域是 Sun ONE Application Server 的功能之一，允许不同的管理用户创建和管理各自的域。域是指使用单个系统中安装的通用二进制代码集创建的一组实例。

行　包含表中每一列的值的单个数据记录。

回滚　事务的取消操作。

会话　一个对象， servlet 使用此对象记录用户在多个 HTTP 请求中与 Web 应用程序进行的交互。

会话 Bean　由客户机创建的企业 Bean ；通常仅存在于单个客户机 - 服务器会话期间。会话 Bean 执行客户机操作 （例如，计算或访问其它 EJB）。由于会话 Bean 可能具有事务性，因此如果系统发生崩溃，它将无法恢复。会话 Bean 对象可以是无状态的 （与特定客户机无关联），也可以是有状态的 （与特定客户机关联），即它们可以在方法和事务之间维护会话状态。*参见*状态会话 Bean、无状态会话 Bean。

会话 cookie　返回至包含用户会话标识符的客户机的 cookie。*参见*粘性 cookie。

会话超时　指定的时间长度，超过此时间长度后， Sun ONE Application Server 可使用户会话无效。*参见*状态。

会话状态　其中的对象状态根据与同一客户机重复交互的结果而更改。*参见*持久状态。

激活　将企业 Bean 的状态从辅助存储器传送到内存的过程。

加密　转换信息，使除预期接收者以外的任何人都无法理解它的过程。

加密算法　一种用于加密或解密的算法 （数学函数）。

监听器　使用发布对象注册的类，指示事件发生时应执行的操作。

解密　转换加密信息，使其重新可被识别的过程。

进程　活动程序的执行序列。进程由一个或多个线程组成。

句柄　标识企业 Bean 的对象。客户机可以序列化句柄，然后将其取消序列化，以获取对 Bean 的引用。

角色　应用程序中主题的功能性分组，由已部署环境中的一个或多个组表示。*参见*用户、组。

可插拔验证　一种允许 J2EE 应用程序使用 J2SE 平台中的 Java 验证和授权服务 (JAAS) 功能的机制。开发者可以插入自己的验证机制。

可重用组件　一个创建后可在多个容量中使用的组件 （例如，由多个资源或应用程序使用）。

可调用的语句　一个类，用于为支持从存储过程返回结果集的数据库封装数据库过程或函数调用。

可分发的会话　可以在群集中的所有服务器中分发的用户会话。

可序列化的对象　可以被解构和重构的对象，以便将对象存储或分发到多个服务器。

客户机合同　一种合同，用于确定客户机和 EJB 容器之间的通信规则、为使用企业 Bean 的应用程序建立统一开发模型以及通过标准化与客户机的关系确保更充分地重复使用企业 Bean。

客户机验证　验证客户机证书的过程，即以加密方式验证证书签名和指向信任 CA 列表中的 CA 的证书链。参见验证、认证机构。

控制描述符　一组企业 Bean 配置条目，允许您为 Bean 方法指定可选单个属性覆盖，并指定企业 Bean 事务和安全性属性。

类加载器　用于根据特定规则装入 Java 类的 Java 组件。*参见*类路径。

类路径　用于标识存储 Java 类的目录和 JAR 文件的路径。*参见*类加载器。

粒度等级　将应用程序划分为若干部分的方法。*高级粒度*意味着应用程序被分为许多较小的、定义更严格的企业 Java Bean (EJB)。*低级粒度*意味着应用程序被分为较少的部分，从而生成较大的程序。

连接池　通过高速缓存以及重复使用物理连接，避免连接开销，并允许在较大数目的线程之间共享较少数目的连接，从而实现了高效的数据库访问。*参见* JDBC 连接池。

连接工厂　一个用于生成连接对象以便 J2EE 组件能够访问资源的对象。用于创建 JMS 连接 （TopicConnection 或 QueueConnection）以便应用程序代码可以利用提供的 JMS 实现。应用程序代码使用 JNDI 服务定位使用 JNDI 名称的连接工厂对象。

连接器　标准的容器扩展机制，用于提供到 EIS 的连接。连接器特定于 EIS，由资源适配器和应用程序开发工具组成，用于建立 EIS 连接。资源适配器通过其对连接器体系结构中定义的系统级合同的支持，插入到容器中。

连接器体系结构　用于将 J2EE 应用程序与 EIS 集成的体系结构。此体系结构包含两个部分：EIS 供应商提供的资源适配器以及允许插入此资源适配器的 J2EE 服务器。此体系结构定义了一组合同（例如，事务、安全性和资源管理），资源适配器必须支持这些合同才能插入到 J2EE 服务器中。

列　数据库表中的字段。

领域　一个范围，针对此范围定义了通用安全策略并由安全服务的安全管理员强制执行。在 J2EE 规范中，也称作*安全策略域*或*安全域*。

流　用于管理通过 HTTP 进行数据通信的技术。对结果进行流处理后，可以立即使用数据的第一部分。如果未对结果进行流处理，则必须在接收整个结果后，才能使用其中的任何部分。使用流技术可以更高效地返回大量数据，从而显著提高应用程序的性能。

密钥对文件　*参见*信任数据库。

模块　已单独部署在应用程序之外的 Web 应用程序、企业 Bean、消息驱动的 Bean、应用程序客户机或连接器。*参见*应用程序、组件、生命周期模块。

模式　基础数据库的结构，包括表格名称、列的名称和类型、索引信息和关系（主键和外键）信息。

目标资源　表示“主题”目标或“队列”目标的对象。应用程序使用此对象对“队列”进行读/写操作，或发布/订阅“主题”。应用程序代码使用 JNDI 服务定位使用 JNDI 名称的 JMS 资源对象。

目录服务器　参见 Sun ONE Application Server。

配置　调整服务器或为组件提供元数据的过程。通常情况下，特定组件的配置保存在组件的部署描述符文件中。*参见*管理服务器、部署描述符。

普通 servlet　用于扩展 javax.servlet.GenericServlet 的 servlet。普通 servlet 与协议无关，即它们不包含对 HTTP 或任何其它传输协议的固有支持。与 HTTP servlet 相对。

请求对象　一个包含客户机生成的页面和会话数据，并作为输入参数传递给 servlet 或 JavaServer 页 (JSP) 的对象。

全局事务　一个由事务管理器管理和调整，并可以跨多个数据库和进程的事务。事务管理器通常使用 XA 协议与数据库后端进行交互。*参见*本地事务。

全局数据库连接　可用于多个组件的数据库连接。要求使用资源管理器。

权限　授予或拒绝授予用户或组的一组权限。*参见* 激活。

认证机构　通过 Internet 出售证书的公司，或负责为公司的内部网或外部网颁发证书的部门。

容器　一个用于向特定类型的 J2EE 组件提供生命周期管理、安全性、部署和运行时服务的实体。Sun ONE Application Server 提供 Web 容器和 EJB 容器，并支持应用程序客户机容器。*参见*组件。

容器管理的持久性　其中 EJB 容器负责实体 Bean 持久性。数据在实体 Bean 的变量和数据存储库之间传输，其中的数据访问逻辑由 Sun ONE Application Server 提供。*参见* Bean 管理的持久性。

容器管理的关系　类对中的字段之间的关系，其中关系一方的操作会影响另一方。

容器管理的事务　其中，企业 Bean 的事务分界由 EJB 容器以声明方式指定并自动控制。*参见* Bean 管理的事务。

软件包　存储在通用目录中的相关类的集合。它们通常原封不动地打包到一个 Java 归档 JAR 文件中。*参见*证书、部署。

商业逻辑　用于实现应用程序的关键商业规则 （而不是数据集成或表示逻辑）的代码。

审计　用于记录重要事件，以供随后进行检查的方法，通常在出错或出现安全漏洞的情况下使用。

生命周期模块　为响应服务器生命周期中的事件而监听并执行其任务的模块。

生命周期事件　服务器生命周期的一个阶段 （例如，启动或停机）。

声明的安全性　在组件的配置文件中声明安全性属性，并允许组件容器 （例如，Bean 的容器或 servlet 引擎）隐式管理安全性。这种类型的安全性不需要任何程序控制。与程序安全性相对。*参见*容器管理的持久性。

声明的事务　参见容器管理的事务。

实体 Bean　与物理数据（例如，数据库中的某一行）有关的企业 Bean。实体 Bean 由于被捆绑到永久数据中，因此生存期很长。实体 Bean 通常具有事务性，可为多个用户识别。*参见*消息驱动的 Bean、只读 Bean、会话 Bean。

事件　触发模块或应用程序响应的命名操作。

试探性决定　特定事务使用的事务模式。事务必须提交或回滚。

事务　一组数据库命令，以组的形式执行。所有相关命令都必须成功执行才能使整个事务成功执行。

事务隔离级别　确定数据库中并行事务的相互可见范围。

事务管理器　控制全局事务的对象，通常使用 XA 协议。*参见*全局事务。

事务恢复　自动或手动恢复分布式事务。

事务上下文　事务的范围，可以为局部或全局。*参见*本地事务、全局事务。

事务属性　事务属性控制事务的范围。

授权　用于确定对方法或资源的访问的过程。 J2EE 平台中的授权取决于通过验证与请求关联的用户是否在给定的安全角色中。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，但仅允许雇员查看自己的个人信息。

属性　可由 servlet 设置的请求对象中的 “名称 - 值” 对。还包括用于修改 XML 文件中的元素的 “名称 - 值” 对。与*参数*相对。更多情况下，属性是指元数据单元。

属性　定义应用程序组件行为的单个属性。在 server.xml 文件中，属性是包含名称/值对的元素。

数据访问逻辑　与数据源交互有关的商业逻辑。

数据库　关系数据库管理系统 (RDBMS) 的一般术语。一个软件包，用于创建和操作大量相关的、经过组织的数据。

数据库连接　数据库连接是与数据库或其它数据源的通信链接。组件可以同时创建和操作多个数据库连接，以访问数据。

数据源　数据来源 （例如，数据库）的句柄。数据源通过 iPlanet Application Server 进行注册，然后以编程方式进行检索，以建立与数据源的连接并与数据源进行交互。数据源定义指定连接到数据源的方法。

数字签名　用于验证消息和签名者的电子安全机制。

瞬时性　在未被使用时释放资源的协议。与持久性相对。

提交　通过向数据库发送所需命令来完成事务。*参见*回滚、事务。

通用 ACL　Sun ONE Application Server 中的命名列表，使用户或组与一个或多个权限相关。可以随意定义和访问此列表，以记录任何权限集。

外部 JDNI 资源　允许 JNDI 服务用作到达远程 JNDI 服务器的桥梁。

网络管理站 (NMS)　用于远程管理特定网络的计算机。NMS 软件通常可提供图像，以显示收集的数据，或使用这些数据确保服务器在特定范围内运行。*参见* SNMP。

文档根目录　文档根目录（有时称作主文档目录）是包含所有要提供给远程客户机的虚拟服务器文件的中心目录。

文件高速缓存　文件高速缓存包含有关文件和静态文件内容的信息。默认情况下，文件高速缓存处于打开状态。

无状态会话 Bean　用于表示无状态服务的会话 Bean。无状态会话 Bean 是完全瞬态的，它封装了特定客户机在有限的时间范围内所需的商业逻辑的临时块。

系统管理员　管理 Sun ONE Application Server 软件以及部署 Sun ONE Application Server 应用程序的人员。

线程　进程内部的执行序列。进程可允许多个线程同时进行，这种进程称作多线程进程。如果进程顺序执行每个线程，则它是单线程进程。

响应对象　一个对象，用于引用调用客户机并提供生成客户机输出的方法。

消息传送　由企业应用程序使用的异步请求、报告或事件的系统，允许松耦合的应用程序可靠而安全地传输信息。

消息驱动的 Bean　作为异步消息使用者的企业 Bean。消息驱动的 Bean 不包含特定客户机的状态，但其实例变量可以包含客户机消息处理的状态，包括指向 EJB 对象的开放数据库连接和对象引用。客户机通过向消息驱动的 Bean 作为其消息监听器的目标发送消息，来访问此消息驱动的 Bean。

小应用程序　用 Java 语言编写并在 Web 浏览器中运行的小应用程序。通常情况下，小应用程序由 Web 页调用或嵌入到 Web 页中，以提供特殊功能。而 *servlet* 则是运行在服务器上的小应用程序。

协同定位　为避免远程过程调用以及提高性能，将组件与相关组件放在同一内存空间中。

信任数据库　包含公共密钥和专用密钥的安全性文件，也称作 MDB。

虚拟服务器　虚拟 Web 服务器，提供面向特定 URL 的内容。多个虚拟服务器可以使用相同或不同的主机名、端口号或 IP 地址来提供内容。HTTP 服务可以根据 URL 将传入的 Web 请求定向至不同的虚拟服务器。也称作虚拟主机。可以将 Web 应用程序指定给特定的虚拟服务器。服务器实例可以具有多个虚拟服务器。*参见*服务器实例。

验证　一个实体 （例如，用户）向另一个实体 （例如，应用程序）证明自己正以特定标识 （用户的安全标识）的身份运行的过程。 Sun ONE Application Server 支持基本验证、基于窗体的验证以及 SSL 双向验证。*参见*客户机验证、摘要验证、主机 IP 验证和可插拔验证。

异步通信　一种通信模式，在这种模式下，消息发送方在继续执行其它工作之前无需等待发送方法返回。

应用程序　一组使用 J2EE 应用程序部署描述符封装到 .ear 文件中的组件。*参见*组件、模块。

应用程序层　J2EE 应用程序在概念上的划分。*客户机层*：用户界面 (UI)。最终用户与客户机软件 （例如， Web 浏览器）进行交互，以使用应用程序。*服务器层*：组成应用程序的商业逻辑和表示逻辑，在应用程序的组件中定义。*数据层*：使应用程序能够与数据源进行交互的数据访问逻辑。

应用程序服务器　运行商业应用程序的可靠、安全和可缩放的软件平台。应用程序服务器通常向应用程序提供高级服务 （例如，组件生命周期、位置以及分发和事务资源访问）。

应用程序客户机容器　*参见*容器。

映射　将面向对象的模型捆绑到关系数据模型 （通常是指关系数据库模式）的功能。将模式转换为不同结构的过程。也称为用户到安全角色的映射。

用户　使用应用程序的个人。从编程角度看，用户由用户名、密码和一组使应用程序能够识别客户机的属性组成。*参见*组、角色。

用户会话　由服务器跟踪的一系列用户应用程序交互。会话维护用户状态、永久对象和标识验证。

域注册表　域注册表是单个数据结构，其中包含在安装 Sun ONE Application Server 时创建和配置的所有域的域特定信息 （例如，域名、域位置、域端口、域主机）。

元数据　有关组件的信息 （例如，组件名称以及组件行为的规范）。

元素　较大集合的成员，例如，数组中的数据单元或逻辑元素。在 XML 文件中，为基础结构单元。 XML 元素包含子元素或数据，还可能包含属性。

远程接口　企业 JavaBean 的两个接口之一。远程接口定义由客户机调用的商业方法。

运行时系统　程序运行的软件环境。运行时系统包含装入以 Java 编程语言编写的程序、动态链接本机方法、管理内存以及处理异常所需的所有代码。还包括 Java 虚拟机的实现，它可以作为 Java 解释器。

摘要验证　一种验证方式，允许用户根据用户名和密码进行验证，而不必以明文形式发送用户名和密码。

粘性 cookie　一个返回至客户机以强制此客户机总是连接到同一服务器进程的 cookie。*参见*会话 cookie。

只读 Bean　EJB 客户机从不对其进行修改的实体 Bean。*参见*实体 Bean。

主机 IP 验证　一种安全机制，用于通过将管理服务器或 Web 站点上的文件和目录仅限使用特定计算机的客户机使用，来限制对它们的访问。

主接口　一种机制，用于定义使客户机能够创建和删除企业 Bean 的方法。

证书　数字数据，用于指定个人、公司或其它实体的名称，并证明证书中包含的公共密钥属于此实体。客户机和服务器都可以拥有证书。

主密钥　使客户机能够定位特定实体 Bean 的唯一标识符。

主密钥类名　一个变量，用于指定 Bean 的主密钥的全限定类名。用于 JNDI 查找。

主题　由管理员创建的对象，用于实现发布/订阅传送模型。主题可被看作内容分层结构中的节点，用于收集和分发与此主题相关的消息。将主题用作媒介，可以将消息发布者与消息订阅者分开。

主要　作为验证结果指定给实体的标识。

专用密钥　*参见*公共密钥加密。

装配　将分散的应用程序组件组合为一个可以部署的单元的过程。*参见*部署。

状态　**1.** 实体在任何给定时间的环境或条件。 **2.** 分布式数据存储机制，可用于通过 Sun ONE Application Server 特征接口 IState2 存储应用程序的状态。*参见*会话状态、持久状态。

状态会话 Bean　一个表示与特定客户机之间的会话的会话 Bean，它自动在多个客户机调用的方法中维护状态。

资源管理器　一个用作资源（例如，数据库或消息代理）与资源客户机（例如，Sun ONE Application Server 进程）之间的辅助设备的对象。控制全局可用的数据源。

资源引用　部署描述符中的元素，用于标识资源的组件编码名称。

组　以某一方式相关的一组用户。组成员关系通常由本地系统管理员维护。*参见*用户、角色。

组件　Web 应用程序、企业 Bean、消息驱动的 Bean、应用程序客户机或连接器。*参见*应用程序、模块。

组件合同　用于建立企业 Bean 与其容器之间的关系的合同。

索引

英文

A

B

C

D

F

G

H

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z