Sun Desktop Manager 1.0 安装指南

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

文件号码 819–6095–10

060406@14774

目录

前言

本文档介绍了部署 Sun™ Desktop Manager 1.0 所需的安装步骤和配置步骤。

概述

Sun Desktop Manager 旨在为桌面主机提供集中配置。可以为组织或域结构中的各种元素指定设置，以便管理员可以高效地管理多组用户或主机。

本书结构

第 1 章提供了 Sun Desktop Manager 的简要概述。

第 2 章讨论了 Sun Desktop Manager 服务器端安装。

第 3 章提供了有关安装 Java Desktop System Configuration Agent 的信息。

第 4 章提供了有关 Java Web Console 的安装信息。

附录 A 包含配置参数信息。

附录 B 讨论了如何在 Desktop Manager 中使用 OpenLDAP 和 Active Directory。

附录 C 提供了有关组织映射的信息。

相关文档

- 《Sun Desktop Manager 1.0 管理指南》
- 《Sun Desktop Manager 1.0 Developer Guide》

文档、支持和培训

Sun 功能	URL	描述
文档	http://www.sun.com/documentation/	下载 PDF 及 HTML 格式的文档，购买印刷文档
支持和培训	http://sunsolve.sun.com	获取技术支持、下载修补程序，以及学习 Sun 提供的课程

◆ ◆ ◆ 第 1 章

概述和概念

本文档介绍了部署 Sun™ Desktop Manager 1.0 所需的安装步骤和配置步骤。有关 Sun Desktop Manager 的更全面的概述，请参见 《Sun Desktop Manager 1.0 管理指南》 。

Sun Desktop Manager 概述

Sun Desktop Manager 为桌面主机提供了集中配置。可以为组织或域结构中的各种元素指定设置，以便管理员可以高效地管理多组用户或主机。

图 1–1 Desktop Manager 体系结构

Desktop Manager 包括以下主要组件：

- 配置系统信息库
- 管理工具
- Desktop Manager 模板
- Configuration Agent
- 配置适配器

配置数据集中存储在配置系统信息库中。可以使用管理工具（由基于 Web 的 Desktop Manager 图形用户界面和命令行界面组成）对配置数据进行管理（创建/删除/修改/指定/取消指定）。基于 Web 的管理工具将使用模板在 Web 浏览器中显示配置数据。

Configuration Agent 为用户应用程序从配置系统信息库中检索配置数据。它可以缓存从中心配置系统信息库中检索到的信息。

管理工具完全独立于 Configuration Agent，这意味着管理工具只能对配置系统信息库进行操作。

用户应用程序（使用配置适配器）通过 Configuration Agent 来查询配置数据。

对于以下配置系统，产品直接支持设置的检索和应用：

- GConf. Gnome 配置框架
- StarSuite Registry
- Mozilla Preferences
- Java Preferences

◆ ◆ ◆ 第 2 章

管理应用程序安装

本章提供了有关如何安装 Sun Desktop Manager 服务器端组件的说明。

Sun Desktop Manager

Desktop Manager 提供了一个可在 Java Web Console 上运行的基于 Web 的管理工具。此用户界面允许管理员遍历组织的层次结构，以便为桌面应用程序定义策略。可以为层次结构中的每一项（例如组织、角色、用户、域和主机）定义这些策略。Desktop Manager 使用多个配置模板来显示特定于不同桌面应用程序（如 Gnome、Mozilla、StarSuite 和 Evolution）的设置。

▼ 安装

开始之前 Desktop Manager 需要安装 Java Web Console 2.2.5 或更高版本后才能运行。请确保系统上已经安装了有效版本。要确定是否已安装了有效版本，请以超级用户 (root) 身份登录并执行以下命令：

```
# smcwebserver status
```

注 – Java Web Console 2.2.4 是 Solaris™ 10 操作系统的一部分，但 Desktop Manager 需要安装 2.2.5 或更高版本。server/console 目录中的 Desktop Manager 归档文件提供了 2.2.5 版。通过在该目录中运行 `./setup` 可以安装此程序。

如果系统中未安装 Java Web Console，或者已安装的版本对于 Desktop Manager 无效，请参阅第 4 章 中的说明，以便先安装或更新 Java Web Console 。然后，请返回到本章继续安装 Desktop Manager。

1 **下载 Desktop Manager Zip 归档文件，然后将其内容解压缩到临时目录中。**

```
# unzip SunDesktopMgr-1.0.zip
```

2. **以超级用户 (root) 身份登录，然后通过以下命令执行安装脚本：**

```
# cd SunDesktopMgr-1.0/<platform>/server/manager
# ./setup
```

3. **查看安装脚本的输出中是否存在错误。**

   如果安装成功完成，安装脚本将自动重新启动 Java Web Console，然后您可以使用 Web 浏览器访问 Desktop Manager。

▼ 操作

1. **在浏览器中键入以下 URL：**

```
https://<hostname>.<domainname>:6789
```

2. **在登录屏幕中，键入一个现有 Unix 用户的用户名和密码。**

   将打开 Java Web Console。

3. **在控制台应用程序启动页中，单击 Desktop Manager 链接。**

   - **要跳过控制台应用程序启动页而直接转到 Desktop Manager，请在浏览器中输入以下 URL：**

```
https://<hostname>.<domainname>:6789/apoc
```

▼ 删除 Desktop Manager

◗ **要从 Java Web Console 中删除 Desktop Manager，请转到为安装创建的临时目录，以超级用户 (root) 身份登录，然后执行以下命令**

```
# cd server/manager
# ./setup -u
```

迁移问题

Desktop Manager 与 Java Desktop System Configuration Manager 的先前版本（1.0 和 1.1 发行版）兼容。但是也存在一些差别，您应该对此有所了解。

在先前的 Configuration Manager 版本中，所有的配置文件数据都存储在一个指定的 LDAP 服务器中。此 LDAP 服务器是作为整个 Configuration Manager 安装过程的一部分进行配置的。此过程还包括 LDAP 登录模块的配置，该模块封装了 LDAP 服务器验证。

对于 Desktop Manager，所有必要的配置步骤目前均可通过向导完成，因此在安装过程中无需再执行任何类型的配置。Desktop Manager 目前还支持多个配置系统信息库。因此，您可以对存储于多个不同的 LDAP 服务器、基于文件的系统信息库等位置的策略数据进行管理。无需再对特定的 LDAP 登录模块进行配置。

不同版本的 LDAP 架构是相同的。如果已经为先前版本的 Configuration Manager 配置了 LDAP 服务器，则在切换到 Desktop Manager 时无需进行任何更改。因此，无需更新客户端 (Java Desktop System Configuration Manager1.1. Agent) 或 LDAP 端即可使用 Desktop Manager。

注 – 在安装 Desktop Manager 之前，应首先删除系统中所有先前的 Configuration Manager 或 Desktop Manager 安装。要删除先前的安装，请（以超级用户身份）执行以下命令：

```
# cd server/manager
# ./setup -u
```

Desktop Manager 安装完成后，即可创建一个指向现有 LDAP 服务器的配置系统信息库：

▼ 创建配置系统信息库

1. **在浏览器中键入以下 URL**

   ```
   https://<hostname>.<domainname>:6789
   ```

2. **在登录屏幕中键入一个现有 Unix 用户的用户名和密码。**

   将打开 Java Web Console。

3. **在控制台应用程序启动页中，单击 Sun Desktop Manager 1.0 链接。**

4. **单击“新建”按钮以启动配置系统信息库向导。**

   向导将指导您完成所需步骤，以配置基于 LDAP 的配置系统信息库。

注意 – 向导会自动提供一个选项，用于将现有策略数据迁移到新的 2.0 格式。此迁移操作是可选的，主要用于提高新版本的 Sun Desktop Manager 1.0 代理的性能。如果仍需在环境中支持 Java Desktop System Configuration Manager 1.1 代理，请不要执行此迁移操作。

Desktop Manager 疑难解答

无法安装

症状：在 Java Web Console 安装结束时出现一条消息，指出因没有注册的应用程序而无法启动。

可能的原因：未安装任何应用程序，包括 Desktop Manager。.

解决方案：安装 Desktop Manager，然后启动 Java Web Console。

连接被拒绝

症状：尝试打开一个正确的 URL（如 `http://<hostname>.<domainname>:6789`），但是收到连接被拒绝的消息。

可能的原因：Java Web Console 未在服务器上运行。

解决方案：要启动 Java Web Console，请以超级用户身份登录，然后执行以下命令：

```
#smcwebserver status
#smcwebserver start
```

无法登录

症状：在 Java Web Console 登录页上，用户/密码组合被拒绝。

可能的原因：相应的 UNIX 用户帐户不存在。

解决方案：检查在系统上是否配置了相应的 UNIX 用户名和密码。如有必要，请创建一个本地 UNIX 用户帐户以用于测试。

无 Desktop Manager 链接

症状：Java Web Console 应用程序列表页未显示 Sun Desktop Manager 链接。

可能的原因：未安装 Desktop Manager 模块。

解决方案：要检查是否在 Java Web Console 中安装了 Desktop Manager，请以超级用户身份登录，然后执行以下命令：

```
# smreg list -a
```

如果列表中未包含 `com.sun.apoc.manager_<version>` 应用程序，则需要重新安装 Desktop Manager。

Null 指针异常、Tomcat/Java 错误或显示空白页

症状：启动了 Desktop Manager，但是只显示空白页或错误消息。

可能的原因：如果错误显示 `NoClassDefFoundError:sun/tools/javac/Main`，则说明 Java Web Console 使用了错误的 Java 版本。

解决方案：通过执行 `# smreg list -p` 并查看 `java.home` 属性，可以检查当前的 Java Web Console Java 环境。此属性必须指向有效的 Java 主目录，并且该主目录必须为 JDK。如果此值设置错误，则需要执行以下命令：

```
# smreg add -p java.home=<JAVA_HOME>
```

注 – *<JAVA_HOME>* 必须指向有效安装，例如，可以在其 `bin` 子目录中找到 `javac`。

然后，您需要使用以下命令重新启动 Java Web Console：

```
# smcwebserver restart
```

无法连接到 SSL LDAP 服务器

症状：在系统信息库创建向导中提供了 LDAP 服务器的详细信息之后（包括选中“使用 SSL”复选框），按“下一步”按钮时显示一个消息框，指出无法连接该服务器。

可能的原因：提供了错误的端口号，未将 LDAP 服务器配置为侦听该端口上使用 SSL 的连接，或者 Java Web Console 密钥库中缺少合适的证书。

解决方案：首先检查是否已将 LDAP 服务器配置为侦听相应端口（在向导中指定）上的 SSL 连接请求。如果此设置正确，请确保 Java Web Console 密钥库（位于 `/etc/opt/webconsole/keystore`）中具有证书颁发机构或 LDAP 服务器证书。 可以使用命令 `keytool -import -file <certificate file> -keystore /etc/opt/webconsole/keystore` 添加证书。该密钥库的默认密码为 **`changeit`**。必须使用命令 `smcwebserver restart` 重新启动 Java Web Console，才能在 Desktop Manager 中显示更改。

无法写入到目录中

症状：创建基于文件的后端或混合式后端时，显示错误消息“无法写入到目录中！”

可能的原因：noaccess 用户无正确权限。

解决方案：为 noaccess 用户指定写入权限。

◆ ◆ ◆ 第 3 章

3 客户端组件

要从 Desktop Manager 访问配置数据，桌面客户端需要安装 Java Desktop System Configuration Agent。Configuration Agent 可以与远程配置数据系统信息库和适配器进行通信，还能将数据集成到特定的配置系统中。当前支持的配置系统为：GConf、Java Preferences、Mozilla Preferences 和 StarSuite Registry。

Solaris 10 操作系统随附有 Configuration Agent 的一个版本。但是，Desktop Manager 需要安装该工具的较新版本。此较新版本将在安装 Desktop Manager 客户端组件及相关修补程序时一起安装。

安装 Desktop Manager 客户端组件：

1. 下载 Desktop Manager 的 Zip 归档文件，并将其内容解压缩到临时目录中。

   ```
   # unzip SunDesktopMgr-1.0.zip
   ```

2. 安装建议的修补程序。

   在 SunDesktopMgr-1.0/<platform>/client/Patches 目录中提供了这些修补程序。请按照每个修补程序的安装说明进行操作。

3. 以超级用户 (root) 身份登录，然后通过以下命令执行安装脚本：

   ```
   # cd SunDesktopMgr-1.0/<platform>/client
   # ./setup
   ```

Configuration Agent

Configuration Agent 是多个不同软件包之一，下表列出了这些软件包：

Solaris 软件包名称	描述
SUNWapbas	配置共享库

Solaris 软件包名称	描述
SUNWapmsc	Configuration Agent 的其他文件
SUNWapoc	Configuration Agent
SUNWapdc	Configuration Agent 向导

安装这些软件包时，将安装此 API 需要的所有文件。可以手动安装这些软件包，也可以通过 Java Desktop System 安装来完成。安装完成之后，必须在系统上配置并启用 Configuration Agent。

注 – 在 Java Desktop System 安装中，Configuration Agent 软件包将作为 Solaris 的一部分进行安装。但是，Desktop Manager 会在安装过程中对这些文件进行修补，以便提供正确的功能级别。

要访问远程配置数据，Configuration Agent 需要某些最基本的引导信息，例如 LDAP 服务器的主机名和端口。这些信息保存在一组属性文件（如 `policymgr.properties`、`apocd.properties` 和 `os.properties`）中。这些文件存储在本地 `/etc/apoc` 目录中。可以手动编辑这些属性文件（请参见附录 A），也可以使用 Configuration Agent 的配置向导。

该配置向导提供了一个图形用户界面，可以引导您完成 Configuration Agent 的必要设置。对于该向导的每个页面，都存在一个相应的帮助屏幕。您可以通过 `/usr/bin/apoc-config` 脚本以超级用户 (root) 的身份启动该向导。

注 – 也可以启动该向导而不启动图形界面。例如，通过执行 `/usr/bin/apoc-config -nodisplay` 在控制台模式下启动该向导。

引导信息

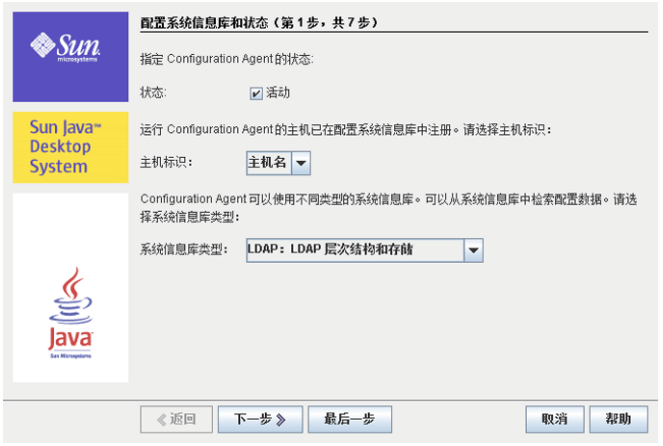

图 3–1 Configuration Agent，配置系统信息库

注 – 在括号中指明了关联的属性文件关键字（在适当的位置）。

- 状态：Configuration Agent 的状态。可以使用该复选框激活或取消激活 Configuration Agent。要使用配置系统信息库，必须激活 Configuration Agent。激活操作自动包括对 Solaris 上服务管理工具 (smf(5)) 的必要注册。
- 主机标识符 (HostIdentifierType)：可以是 "HostName" 或 "IPAddress"。搜索特定于主机的策略数据时，Configuration Agent 将通过主机名或 IP 地址来识别当前主机。请根据主机的标识方式（在选定类型的上下文中）来选择正确的值。
- 上下文类型：使用此设置可以向 Configuration Agent 指明，组织层次结构和配置数据是在 LDAP 存储、基于文件的存储还是混合式存储中定义的。

注 – 要手动启用或禁用 Configuration Agent，请以 **root** 身份登录，然后键入 `/usr/lib/apoc/apocd enable`（如果要启用）或 `/usr/lib/apoc/apocd disable`（如果要禁用）。

图 3–2 Configuration Agent, LDAP 层次结构和基于文件的存储

注 – 图 3–2 中的屏幕将随着在前一个屏幕中所选的上下文类型而变化。如果选择了 LDAP 或混合式上下文类型，则必须提供服务器标识符、服务器端口和后缀。如果选择了基于文件或混合式上下文类型，则必须提供配置设置 URL。

- 服务器标识符：LDAP 服务器的主机名。
- 服务器端口：LDAP 服务器的端口号。
- 后缀：LDAP 系统信息库的基本 DN。
- 配置设置 URL：用于指定基于文件系统信息库位置的 URL。

  URL 列表可用于指定备用系统信息库，以便在无法连接到第一个系统信息库时使用。此列表可由一个或多个 URL 组成（URL 之间以空格分隔），每个 URL 的格式为 `file://<filepath>` 、`http://<host>:<port>/<filepath>` 或 `https://<host>:<port>/<filepath>`。有关详细信息，请参见附录 A。

  注 – 代理将首先尝试使用 SSL 连接来访问 LDAP 服务器。如果失败，代理会尝试普通的 SSL 连接。

  要成功进行 SSL 连接，Java 运行时环境密钥库中必须具有合适的证书。标准 JRE 的密钥库位于 `<installation directory>/lib/security/cacerts` 中，而标准 JDK 的密钥库位于 `<installation directory>/jre/lib/security/cacerts` 中。必须将证书颁发机构或 LDAP 服务器证书添加到密钥库中，方法是使用命令 `keytool -import -file <certificate file> -keystore <cacerts file location>`。该密钥库的默认密码为 **`changeit`**。

图 3–3 Configuration Agent，验证机制

- Configuration Agent 的验证类型：可以是“匿名”或“简单”。如果选择“匿名”，将自动禁用“限定的用户名”和“密码”字段。
- 限定的用户名 (AuthDn)：具有读取和搜索系统信息库权限的用户的完整 DN。
- 密码 (Password)：已注册的 LDAP 用户的密码

注 – 如果在目录中启用了匿名访问，则可以将“限定的用户名”和“密码”设置留空。

- 应用程序的验证类型 (AuthType)：可以是“匿名”或“GSSAPI”，这取决于 LDAP 服务器验证用户的方式。

注 – 有关详细信息，请参见第 24 页中的 “数据访问/用户验证”。

端口设置

Configuration Agent 使用以下两个端口：

- 代理端口 (DaemonPort)：代理使用该端口与客户端应用程序进行通信（默认值为 **38900**）。
- 管理端口 (DaemonAdminPort)：在与代理通信时，代理控制器程序 (apocd) 所用的端口（默认值为 **38901**）。

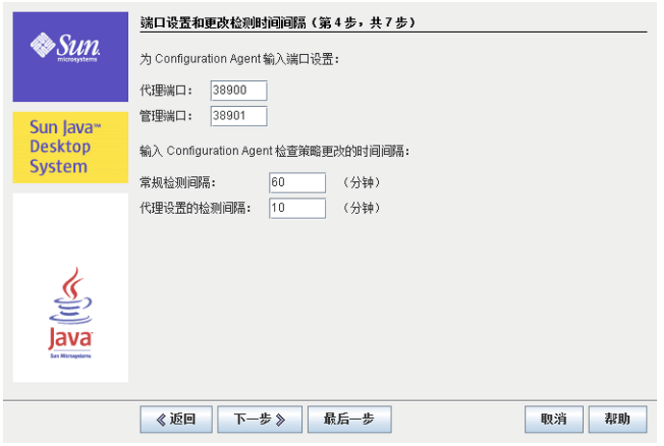

图 3–4 Configuration Agent，端口设置

更改检测间隔

Configuration Agent 使用以下两种间隔定期检查配置数据中的更改：

- 常规检测间隔 (ChangeDetectionInterval)：桌面应用程序（客户端）的配置数据的两个更改检测周期之间的间隔（以分钟为单位）。

  注 – 指定 **-1** 将关闭更改检测。

- 代理设置的间隔 (DaemonChangeDetectionInterval)：代理特定的配置设置的两个更改检测周期之间的间隔（以分钟为单位）。

  注 – 指定 **-1** 将关闭更改检测。

可以使用常规检测间隔来调整远程配置数据更改至客户端应用程序的传播。为此设置指定的值表示最长经过多少分钟后，远程更改才反映在客户端应用程序中。

值越小，Configuration Agent 和 LDAP 服务器活动就越多。因此，调整此设置的值时应谨慎。例如，在初始部署阶段，可以将该值设置为一分钟，这样就可以测试远程配置对客户端应用程序的影响。完成测试后，请将此设置还原为初始值。

操作设置

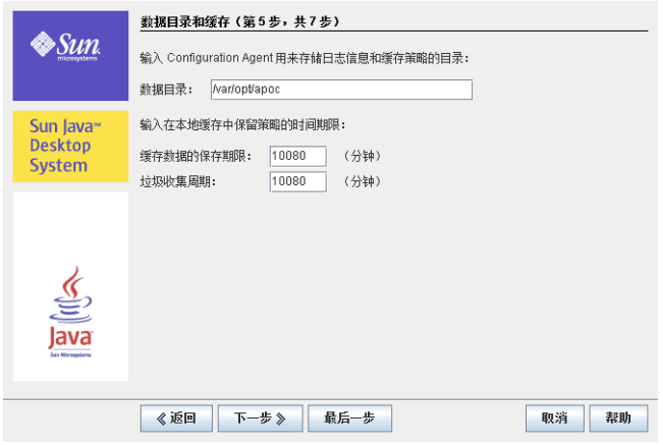

图 3–5 Configuration Agent，数据目录

可以配置以下设置：

- 数据目录 (DataDir)：用于存储运行时数据的目录。默认值为 **`/var/opt/apoc`**。
- 缓存数据的存储期限 (TimeToLive)：非脱机配置数据在本地数据库中的保留时间（以分钟为单位）。

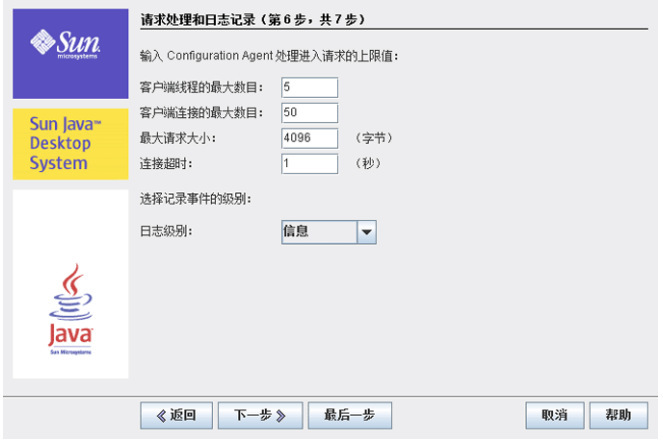

图 3–6 Configuration Agent，请求处理和日志记录

- 垃圾收集周期 (GarbageCollectionInterval)：本地配置数据库中两个垃圾收集周期之间的间隔（以分钟为单位）。
- 客户端线程的最大数目 (MaxClientThreads)：可以同时处理的客户端请求的最大数目。

- 客户端连接的最大数目 (MaxClientConnections)：客户端连接的最大数目。
- 最大请求大小 (MaxRequestSize)：客户端请求的最大大小。
- 连接超时 (ConnectTimeout)：表示 LDAP 服务器响应连接请求时所允许的间隔。默认值为一秒。
- 日志级别 (LogLevel)：代理日志文件中的详细程度。此日志级别应与 Java 记录程序的级别保持一致。这些级别如下所示（按严重性递减顺序）：
  - 严重
  - 警告
  - 信息
  - 配置
  - 良好
  - 较好
  - 最好

注 – 大多数操作设置（“数据目录”和“连接超时”设置除外）也可以通过 LDAP 服务器中存储的相应策略进行集中维护。如果要使用此功能，请不要通过向导更改相应的设置。相反，应使用 Desktop Manager 中的 Configuration Agent 策略来集中指定操作设置。

应用代理设置

已通过 Desktop Manager 存储在 LDAP 服务器中的操作设置（“数据目录”和“连接超时”除外）将在代理配置的下一个更改检测周期自动生效（请参见 DaemonChangeDetectionInterval）。

图 3–7 Configuration Agent，“汇总”页

在本地更改的所有其他设置均需要重新装入 Configuration Agent，或者重新启动 Configuration Agent。如果使用配置向导，将自动执行重新装入和重新启动操作。

注 – 要手动重新启动 Configuration Agent，请确保未运行任何相关的客户端应用程序，然后以 root 身份登录，并键入命令 `/usr/lib/apoc/apocd restart`。

其他代理设置

注 – 在配置向导中未提供以下设置。

- 应用本地策略 (ApplyLocalPolicy)：使用此设置可以指明本地主机上的策略数据是否可用于客户端应用程序。"true" 值表示本地策略数据可用于客户端应用程序。"false" 值表示本地策略数据无法用于客户端应用程序。有关详细信息，请参见第 23 页中的 “使用本地策略”。

使用本地策略

可以配置 Configuration Agent 来应用部署于本地的策略 （替代全局策略或作为其补充） 中的设置。可以使用以下步骤部署此类本地策略：

▼ 部署本地策略

1. **使用 Desktop Manager 创建具有所需策略设置的配置文件。**
2. **使用 Desktop Manager 将配置文件导出为一个 Zip 文件。**
3. **在客户端主机上，创建 `${DataDir}/Policies/profiles/PROFILE_REPOSITORY_default` 目录（如果此目录尚未存在）。**

   `${DataDir}` 与 Configuration Agent 数据目录的值（默认值为 `/var/opt/apoc`）相对应。
4. **将先前导出的 Zip 文件复制到 `${DataDir}/Policies/profiles/PROFILE_REPOSITORY_default` 中。**
5. **确保将 Configuration Agent 配置为应用可用的本地策略（有关详细信息，请参见第 23 页中的“ 其他代理设置 ”）。**

注 – 如果更改了 Configuration Agent 的 "ApplyLocalPolicy" 设置，则需要重新加载 Configuration Agent（方法是以 `root` 身份登录，然后键入命令 `/usr/lib/apoc/apocd reload`）。

在下一个 Configuration Agent 更改检测周期中，所有以此方式部署的本地策略都将可用于客户端。

自动重新启动 Configuration Agent

如果失败，Configuration Agent 将自动重新启动。服务管理工具 (smf(5)) 负责作出此项决策。如果服务管理工具认为不应该重新启动（例如，如果失败次数太多），则会将 Configuration Agent 置于维护模式。

如果无法重新启动 Configuration Agent，则应暂时禁用此代理（方法是以 root 身份登录，然后执行命令 `/usr/lib/apoc/apocd disable`），纠正导致代理失败的所有问题，然后通过执行命令 `/usr/lib/apoc/apocd enable` 重新启用代理。

数据访问/用户验证

Configuration Agent 将基于桌面用户的登录 ID 从 LDAP 服务器中检索信息。组织映射文件中的 `User/UniqueIdAttribute` 设置会将登录 ID 映射到 LDAP 服务器中的用户元素。Configuration Agent 还将检索有关主机的信息，例如主机的名称或 IP 地址。这些信息将通过组织映射文件中的 `Host/UniqueIdAttribute` 设置映射到 LDAP 服务器中的主机元素。有关组织映射的详细信息，请参见附录 C。

有两种方式访问 LDAP 服务器：匿名方式或使用 GSSAPI。对于匿名访问，无需在桌面上执行任何操作。对于 GSSAPI 方式，必须在桌面上获得 Kerberos 凭证。要将 Kerberos 凭证的获得与用户登录集成在一起，必须在 Java Desktop System 主机上安装和配置 `pam_krb5` 模块。

您可以使用 `gdm` 将 Kerberos 同用户登录集成在一起，例如使用以下 `/etc/pam.d/gdm` 文件：

```
#%PAM-1.0
auth   required    pam_unix2.so  nullok #set_secrpc
auth   optional  pam_krb5.so use_first_pass missing_keytab_ok ccache=SAFE putenv_direct
account required    pam_unix2.so
password required    pam_unix2.so  #strict=false
session required    pam_unix2.so  # trace or none
session required    pam_devperm.so
session optional    pam_console.so
```

如果您以这种方式将 Kerberos 同用户登录集成在一起，则您应启用屏幕保护程序的 Kerberos 支持。例如，使用以下 `/etc/pam.d/xscreensaver` 文件：

```
auth required pamkrb5.so use_first_pass missing_keytab_ok
ccache=SAFE putenv_direct
```

适配器

应用程序适配器是 Desktop Manager 所支持的配置系统的扩展。适配器允许各种应用程序使用集中配置数据（具体取决于配置系统）。支持的配置系统包括：

- GConf：Gnome 配置系统，用于桌面和大多数 Gnome 应用程序，如 Evolution
- StarSuite Registry：StarSuite 和 OpenOffice.org 使用的配置系统
- Mozilla Preferences：Mozilla 使用的配置系统
- Java Preferences：提供给 Java 应用程序的配置 API

此外还提供了桌面定义适配器，它向用户桌面中添加了桌面启动器、菜单项和启动程序。

GConf 适配器

GConf 适配器是用于 Solaris 的 `SUNWapoc-adapter-gconf` 软件包的一部分。从相应的 packageAdapter 安装适配器时，将更新 `/etc/gconf/2/path` 中的 GConf 数据源路径，以包括 Desktop Manager 源。适配器提供两种数据源，它们为：

- "apoc:readonly:"：用来访问策略的非保护设置。在用户设置之后、本地默认设置之前插入此数据源。
- "apoc:readonly:mandatory@"：用来访问策略的保护设置。在本地强制设置之后、用户设置之前插入此数据源。

GConf 适配器配置

GConf 适配器是作为安装的一部分进行配置的，但其操作取决于在 GConf 路径文件 (`/etc/GConf/2/path`) 中是否包含表示强制性集中设置和默认设置的两个数据源。安装完系统之后，即使此路径文件中包含正确的信息（使 GConf 能够按预期那样使用集中设置），管理员也要确保以 "apoc" 为前缀的数据源仍存在于该文件中，以便在需要时可以修改此路径以包含其他自定义数据源。此外，还要确保表示强制性集中设置的数据源位于本地强制性设置和用户设置之间，而表示默认集中设置的数据源位于用户设置和本地默认设置之间。

Java Preferences 适配器

Java Preferences 适配器是用于 Solaris 的 `SUNWapcj` 软件包的一部分。

Java Preferences 适配器配置

Java Preferences 适配器是作为 Preferences API 实现提供的，此实现必须用作其他现有实现（例如随 JRE 一起提供的默认基于文件的系统）的包装器。要在使用 Preferences API 的 Java 应用程序中使用集中配置，必须为该应用程序编写启动脚本（使用 `/usr/lib/apoc/apocjlaunch` 脚本作为帮助程序）。此脚本需要定义几个环境变量，然后在其末尾添加 `apocjlaunch` 脚本（此脚本可在必要的环境中启动 Java 应用程序）。必须设置的环境变量包括：

- `JAVA`：包含 Java 运行时可执行文件的路径
- `APPLICATION`：包含该应用程序常规 Java 运行时调用的结尾部分。例如，*classname [arguments]* 用于单一类启动，`-jar` *jarname [arguments]* 用于 jar 归档文件启动。

可以设置的其他可选环境变量包括：

- `CLASSPATH`：以冒号分隔的 jar 文件或类文件列表，这些文件需要作为应用程序类路径的一部分
- `DEFINES`：包含定义语句的字符串，这些定义语句需要作为应用程序启动的一部分
- `PREFFACTORY`：应用程序需要使用的基础 Preferences API 实现中的工厂类名称

Mozilla 适配器

Mozilla 适配器是用于 Solaris 的 `SUNWmozapoc-adapter` 软件包的一部分。

Mozilla 适配器配置

Mozilla 适配器是作为此产品安装的一部分进行安装的，且不需要其他任何配置。

StarSuite 适配器

StarSuite 适配器包含在标准的 StarSuite 安装中，并且允许您访问配置文件配置数据，而无需进行任何特殊的修改。

StarSuite 适配器配置

StarSuite 适配器是作为此产品安装的一部分进行安装的，且不需要其他任何配置。

桌面定义适配器

桌面定义适配器由以下软件包组成：

软件包名称	描述
SUNWapleg	配置访问二进制文件
SUNWardsa	桌面定义适配器
SUNWardsa-misc	适配器的系统集成

这些软件包将在安装 Desktop Manager 客户端组件时进行安装，且不需要任何其他设置。

桌面定义适配器配置

桌面定义适配器由安装进程配置（以便在用户登录时使用），且不需要任何其他设置。

删除适配器

删除这些产品时会同时删除 Mozilla 适配器和 StarSuite 适配器。通过删除安装部分所提到的软件包，可以使用相应的软件包管理系统工具来删除 GConf、Java Preferences 和桌面定义适配器。

删除 Java Preferences 适配器后，不应再使用为启动 Java 应用程序（该程序使用 Preferences API）而编写的启动脚本。这些脚本中的 Java 调用将失败，因为缺少了某些必需的类。

适配器疑难解答

导致在相应的应用程序中看不到集中配置数据的大多数问题都可能来源于 Configuration Agent，因为它是所有适配器用于检索数据的通用机制。

如果对于给定设置（或其中的组），对集中配置所做的更改看上去未能生效，则可能是因为用户已经在应用程序中为此设置明确设置了一个值（通常使用产品的“选项”或“首选项”对话框）。在这种情况下，除非将集中设置定义为受保护（即管理员强制指定一个值，且不允许用户修改），否则用户首选项将优先于使用 Desktop Manager 设置的值。

Configuration Agent 疑难解答

本部分解答了一些与 Configuration Agent 的特性和使用方法相关的问题，并提供了一些用于解决代理问题的提示。

问题和解答

什么是 Configuration Agent，它是如何工作的？

Configuration Agent 是一种策略缓存和传送应用程序。设计和构建此应用程序的目的是确保能够集中配置桌面客户端应用程序，同时不会对这些应用程序及其所在主机的性能造成显著影响。这一点是通过以下方式实现的：

- 在本地可用缓存中缓存任意已下载的策略，以备客户端将来使用
- 共享所有可以且应该共享的昂贵资源（例如，与保存策略的 LDAP 服务器的连接）

客户端应用程序与 Configuration Agent 进行交互的典型方案极其简单，其过程可描述如下：

1. 用户启动一种相关的桌面客户端应用程序（gconfd、Mozilla 或 StarSuite）
2. 客户端应用程序连接到 Configuration Agent
3. 客户端应用程序从 Configuration Agent 中请求所需的策略数据
4. Configuration Agent 在缓存中搜索请求的策略数据
5. 如果在缓存中未找到策略数据，Configuration Agent 将从预配置的策略系统信息库中下载所需数据，然后将其存储在缓存中
6. 将策略数据发送给发出请求的客户端应用程序
7. Configuration Agent 监视策略系统信息库中是否存在对策略数据进行的任何修改
8. 如果检测到修改，Configuration Agent 会刷新缓存以保持最新状态，并向客户端应用程序通知此修改。

如何获得和安装 Configuration Agent？

Configuration Agent 随 Solaris 10 一起提供，并在默认情况下随 Solaris 10 一起安装

我刚刚安装了 Solaris 10，接下来应该怎么做？

默认情况下，Configuration Agent 处于禁用状态且未进行配置。要使用 Configuration Agent，必须至少对其进行最低配置，然后启用该程序。完成以上步骤后，桌面客户端应用程序将在下一次启动时自动开始使用提供的所有策略。

我要对 Configuration Agent 进行配置，应该如何做？

要正确配置 Configuration Agent，请使用 Configuration Agent 向导。可以通过执行（以 root 身份）`/usr/bin/apoc-config` 命令来启动向导。向导会指导您逐步完成正确配置代理所需的步骤。多数情况下，完成向导过程中唯一必需的信息是策略系统信息库的位置。

也可以通过手动编辑 Configuration Agent 的配置文件来配置 Configuration Agent。但不建议这样做，因为使用此方法配置代理很容易发生错误。另外，Configuration Agent 向导还包含其他逻辑，用于确定进行特定的配置更改后是否需要重新启动或重新加载代理。

我要启用 Configuration Agent，应该如何做？

可以使用三种可能的机制来启用代理：

1. 使用 Configuration Agent 向导 (`/usr/bin/apoc-config`)，将“代理状态”设置为“活动”。
2. 使用 Configuration Agent 控制器程序 (`/usr/lib/apoc/apocd`)，以 root 身份执行以下命令：

   ```
   /usr/lib/apoc/apocd enable
   ```
3. 使用smf(5)，以超级用户身份执行以下命令：

   ```
   /usr/sbin/svcadm enable svc:/network/apocd/udp
   ```

我已经配置并启用了 Configuration Agent，如何知道它是否正在工作？

了解 Configuration Agent 是否正确配置且正常工作的最简单方法是：使用 Desktop Manager 创建一个策略，将其指定给一个用户，然后以该用户身份登录到桌面计算机，再验证创建的策略设置是否正被使用。在桌面会话中有很多容易检测到的策略设置，例如背景和主题。

启用 Configuration Agent 的意义何在？

Configuration Agent 是一种符合 smf(5) 的服务，因此，启用该代理的意义来源于 smf(5)。启用代理之后，代理即可开始提供服务。启用代理后会发生以下行为：

- 代理启动
- 在启用代理之后启动的所有桌面客户端应用程序均可检索策略数据
- 代理会在系统启动期间自动重新启动

如何判断 Configuration Agent 是否已启用？

可以使用以下任一方法来确定 Configuration Agent 是否已启用：

- 使用 Configuration Agent 的控制器程序。 以超级用户身份登录并键入以下命令：

  ```
  /usr/lib/apoc/apocd is-enabled
  ```

  如果代理已启用，控制器程序会返回以下消息：

  ```
  Checking Configuration Agent enabled status ... Enabled
  ```

  否则，控制器程序会返回以下消息：

```
Checking Configuration Agent enabled status ... Not enabled
```

- 使用 smf(5) 执行以下命令：

  ```
  /usr/bin/svcs svc:/network/apocd/udp:default
  ```

  如果代理已启用，svcs 会返回以下消息：

  ```
  STATE          STIME    FMRI
  online          8:36:04 svc:/network/apocd/udp:default
  ```

  如果代理被禁用，svcs 会返回以下消息：

  ```
  STATE          STIME    FMRI
  disabled       15:58:34 svc:/network/apocd/udp:default
  ```

  如果代理处于维护模式，svcs 会返回以下消息：

  ```
  STATE          STIME    FMRI
  maintenance     8:38:42 svc:/network/apocd/udp:default
  ```

如何判断 Configuration Agent 是否正在运行？

可以使用以下任一方法来确定 Configuration Agent 是否正在运行：

- 以超级用户身份登录并执行 Configuration Agent 控制器程序：

  ```
  /usr/lib/apoc/apocd status
  ```

  如果代理已启用，控制器程序会返回以下消息：

  ```
  Checking Configuration Agent status ... Running
  ```

  否则，控制器程序会返回以下消息：

  ```
  Checking Configuration Agent status ... Not running
  ```

- 执行以下命令：

  ```
  /usr/bin/svcs svc:/network/apocd/udp:default
  ```

  如果代理正在运行，svcs 会返回以下消息：

  ```
  STATE          STIME    FMRI
  online          8:36:04 svc:/network/apocd/udp:default
  ```

  如果代理未运行，svcs 会返回以下消息：

  ```
  STATE          STIME    FMRI
  disabled       15:58:34 svc:/network/apocd/udp:default
  ```

如果代理处于维护模式，svcs 会返回以下消息：

```
STATE          STIME    FMRI
maintenance     8:38:42 svc:/network/apocd/udp:default
```

- 执行以下命令：

```
ps -ef | grep apoc
```

如果 Configuration Agent 正在运行，在 ps 输出中应显示以下相关的 Java 进程：

```
 daemon 29295 29294  0 13:05:22?  0:03 java -Djava.library.path=/usr/lib/apoc
 -cp /usr/share/lib/apoc/apocd.jar:/usr/s
 daemon 29294   1   0 13:05:22?  0:00 sh -c java
-Djava.library.path=/usr/lib/apoc -cp /usr/share/lib/apoc/apocd.jar:
 root 29345 28134  0 13:08:59 pts/1   0:00 grep apoc
```

日志文件在什么位置？

需要解决 Configuration Agent 问题时，可以参考以下日志文件：

- smf(5) 日志文件：
  - /var/svc/log/network-apocd-udp:default.log 文件记录了与尝试启动和停止 Configuration Agent 特定实例相关的事件。此文件还包含 Configuration Agent 控制器程序 (/usr/lib/apoc/apocd) 写入标准输出的消息，以及 JVM 或 Configuration Agent 的输出消息。
  - /var/svc/log/svc.startd.log 日志文件记录了较高级别的 smf(5) 事件。例如，如果在短时间内多次尝试启动 Configuration Agent 失败，smf(5) 可能会认为 Configuration Agent 无法启动。在这种情况下，smf(5) 会将 Configuration Agent 置于维护模式，并将此结果写入一个日志条目。

  当遇到与启动 Configuration Agent 相关的问题时，以上两种日志文件通常都很有用。
- Configuration Agent 日志：

  Configuration Agent 将日志消息写入默认日志目录 /var/opt/apoc/Logs 中的日志文件。Configuration Agent 的“数据目录”为 /var/opt/apoc 。可以使用 Configuration Agent 向导 (/usr/bin/apoc-config) 应用程序来更改此目录的位置。可以通过更改“日志级别”（使用 Configuration Agent 向导）来更改日志消息的详细程度。如果您认为未正确配置 Configuration Agent，或者遇到一些其他类型的代理失败事件，则可以在参考代理日志文件之前使用 Configuration Agent 向导将日志级别设置为“最详细” 。此步骤可以确保您获得最全面的可用日志信息。
- 系统日志：

  也可以在 SunRay 计算机上查看 /var/adm/messages 日志文件或 /var/opt/SUNWut/log/messages 日志文件，以诊断 Configuration Agent 的各种问题。

如何提高代理日志记录机制的详细程度？

请参见第 31 页中的 “日志文件在什么位置？”

什么是维护模式？

当 smf(5) 检测到与启动或重新启动代理相关的问题时，就会将 Configuration Agent 置于维护模式。如果 smf(5) 启动代理失败，它将多次尝试重新启动，直到代理启动成功，否则，smf(5) 将认为代理无法启动。如果是后一种情况，smf(5) 会将代理置于维护模式，以指明您需要解决代理所遇到的问题。解决这些问题之后，您可以清除代理的 smf(5) 状态以返回到正常运行模式。

如何退出维护模式（清除 smf(5) 状态）？

以超级用户身份登录并执行 `/usr/sbin/svcadm clear svc:/network/apocd/udp` 命令。

Configuration Agent 异常停止运行时会出现什么情况？

smf(5) 会检测到代理已停止运行并尝试重新启动代理。如果由于某种原因导致连续尝试重新启动失败，smf(5) 会将代理置于维护模式。如果代理重新启动成功，则不会影响正在运行的桌面客户端应用程序。任何此类客户端应用程序都会在代理重新启动后自动重新连接到代理。

如果启用/启动代理，是否需要重新启动桌面客户端应用程序？

实际操作取决于启动特定桌面客户端应用程序时代理是否已启用/正在运行。如果代理已启用/正在运行，则客户端应用程序会建立与代理的连接，并在连接断开后尝试重新建立连接。也就是说，每次启动、启用或禁用代理时，客户端应用程序都会一直尝试重新连接代理（在代理运行之后）。如果在客户端应用程序启动时代理未启用/正在运行，则客户端应用程序不会使用 Configuration Agent，甚至在代理启动之后也不会尝试与其连接。综上所述可以得出以下结论：

- 代理已启用/正在运行时启动的桌面客户端应用程序不需要重新启动。
- 代理未启用/正在运行时启动的桌面客户端应用程序必须重新启动。

我的桌面客户端应用程序似乎未使用已配置的策略，应该怎么做？

与 Configuration Agent 相关的最常见问题是：看不到已配置的策略对桌面客户端应用程序产生的效果。造成这种问题最常见的原因是：代理配置不正确、策略系统信息库配置不正确或策略系统信息库不可用。以下指导方法可以帮助您发现和解决上述问题：

- 确保已配置代理。
- 确保代理已启用/正在运行。 如果需要启动代理，必须同时重新启动当前打开的桌面客户端应用程序。
- 如果问题仍然存在，可暂时提高代理日志记录机制的详细程度，然后重新启动代理（如有可能），这样即可获得自代理启动之后的完整而详细的日志消息。
- 如果代理无法正常启动，请参考第 33 页中的 “启动 Configuration Agent 时出现的问题”部分。
- 如果代理正常启动，但桌面客户端应用程序未使用可用的策略，请参考“从正在运行的 Configuration Agent 获取策略时出现的问题”部分。
- 如果仍然无法解决问题，请与技术支持联系。

启动 Configuration Agent 时出现的问题

如果无法启动 Configuration Agent，并且确定已配置并启用了 Configuration Agent，则需要参考日志文件。以下各部分描述了关于此问题的最常见错误。

代理数据目录无效或无法访问

Configuration Agent 数据目录由代理创建并使用，用于存储日志文件、策略缓存等。此目录的默认位置为 `/var/opt/apoc`。

如果将数据目录设置为无法访问的位置（即， `/dev/null/cant/write/here`），则 Configuration Agent 会在 smf(5) 日志中生成以下错误消息。要解决此问题，请使用 Configuration Agent 向导 (`/usr/bin/apoc-config`) 将数据目录指向一个可访问的位置。

```
[ Nov 17 14:35:38 Executing start method ("/usr/lib/apoc/apocd svcStart") ]
Starting Configuration Agent ... Warning: Cannot create Log directory
   '/dev/null/cant/write/here/Logs'
Warning:Failed to create log file handler
Nov 17, 2005 2:35:39 PM com.sun.apoc.daemon.misc.APOCLogger config
CONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 38901
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /dev/null/cant/write/here
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 38900
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:35:39 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException
    at com.sun.apoc.daemon.apocd.Daemon.initAuthDir(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.init(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
[ Nov 17 14:36:08 Method or service exit timed out.  Killing contract 980 ]
[ Nov 17 14:36:08 Method "start" failed due to signal KILL ]
```

使用处于忙碌状态的客户端请求端口

Configuration Agent 使用 TCP/IP 套接字连接与桌面客户端应用程序进行通信。 默认情况下，这些连接都是通过端口 38900 建立的。

如果将 Configuration Agent 配置为使用端口 1234（此端口已由其他服务使用），则会生成以下错误消息。 错误消息将记录在 Configuration Agent 日志中。 要解决此问题，请使用 Configuration Agent 向导 (`/usr/bin/apoc-config`) 将代理端口设置更改为未使用的端口号。

```
Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger config
CONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 38901
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /var/opt/apoc
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 1234
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger info
INFO: Daemon starting
Nov 17, 2005 2:50:59 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Garbage collection scheduled ( interval = 10080 minutes )
Nov 17, 2005 2:50:59 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: java.net.BindException: Address already in use
    at com.sun.apoc.daemon.transport.ChannelManager.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.run(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind(Native Method)
    at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:119)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:59)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:52)
```

使用处于忙碌状态的管理端口

Configuration Agent 使用 TCP/IP 套接字连接与 Configuration Agent 控制器程序 (/usr/lib/apoc/apocd) 进行通信。默认情况下，这些连接都是通过端口 38901 建立的。

如果将 Configuration Agent 配置为使用端口 1234（此端口已由其他服务使用），Configuration Agent 日志中将出现以下错误消息。要解决此问题，请使用 Configuration Agent 向导 (/usr/bin/apoc-config) 将管理端口设置更改为未使用的端口号。

```
ONFIG: Daemon configuration:
 MaxRequestSize = 4096
 DaemonAdminPort = 1234
 ThreadTimeToLive = 5
 DaemonChangeDetectionInterval = 10
 IdleThreadDetectionInterval = 15
 PROVIDER_URL =
 DataDir = /var/opt/apoc
 ApplyLocalPolicy = true
 ChangeDetectionInterval = 60
 MaxClientConnections = 50
 GarbageCollectionInterval = 10080
 InitialChangeDetectionDelay = 10
 TimeToLive = 10080
 ConnectionReadTimeout = 5000
 DaemonPort = 38900
 LogLevel = FINEST
 MaxClientThreads = 5


Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger info
INFO: Daemon starting
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Garbage collection scheduled ( interval = 10080 minutes )
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Client manager started
Nov 17, 2005 2:55:11 PM com.sun.apoc.daemon.misc.APOCLogger fine
FINE: Channel manager started
Nov 17, 2005 2:55:11 PM Daemon main
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: java.net.BindException: Address already in use
    at com.sun.apoc.daemon.admin.AdminManager.initChannel(Unknown Source)
    at com.sun.apoc.daemon.admin.AdminManager.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.run(Unknown Source)
    at com.sun.apoc.daemon.apocd.Daemon.main(Unknown Source)
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind(Native Method)
    at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:119)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:59)
```

```
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:52)
    ... 4 more
```

从正在运行的 Configuration Agent 获取策略时出现的问题

配置系统信息库规范缺少或无效

Configuration Agent 需要连接到有效的配置系统信息库，以便下载和缓存策略信息。如果未在代理配置中正确标识配置系统信息库（例如，使用了无效格式或未指定系统信息库），则当桌面客户端应用程序启动时，会在 Configuration Agent 日志中记录类似以下内容的错误消息。要解决此问题，请使用 Configuration Agent 向导 (`/usr/bin/apoc-config`) 来标识要使用的配置系统信息库。

```
FINER: New client added
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authenticating user geoffh
Nov 18, 2005 1:59:22 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 1:59:23 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.environment.InvalidParameterException: The parameter organisation
PROVIDER_URL#protocol (null) is not valid, the value must be comprised in
{ldaps,ldap,https,http,file}.
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
Caused by: com.sun.apoc.daemon.misc.APOCException:
 com.sun.apoc.spi.environment.InvalidParameterException:
 The parameter organisation PROVIDER_URL#protocol (null) is not valid,
 the value must be comprised in {ldaps,ldap,https,http,file}.
```

```
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyMgr(Unknown Source)
    ... 14 more
Caused by: com.sun.apoc.spi.environment.InvalidParameterException: The parameter
organisation PROVIDER_URL#protocol (null) is not valid, the value must be comprised in
{ldaps,ldap,https,http,file}.
    at com.sun.apoc.spi.PolicyMgrFactoryImpl.createPolicyMgr(Unknown Source)
    ... 15 more
Nov 18, 2005 1:59:23 PM PolicyBackend openPolicyBackend
```

无法连接到策略系统信息库

Configuration Agent 需要连接到有效的配置系统信息库，以便下载和缓存策略信息。如果无法建立连接，则当桌面客户端应用程序启动时，会在 Configuration Agent 日志中记录类似以下内容的错误消息。在以下情况中，主机 sobuild 不存在、无法连接或无法通过端口 389 访问 LDAP 服务器。要解决此问题，请使用 Agent Configuration 向导 (`/usr/bin/apoc-config`) 确保正确标识了策略系统信息库，并在此基础上确保可以访问该策略系统信息库。例如，对于 LDAP 系统信息库，必须确保 LDAP 服务器正在运行、LDAP 服务器所在的计算机在网络上可用，并且指定的端口为 LDAP 服务器正在使用的端口。

如果尝试使用 SSL 连接来访问 LDAP 服务器，请确保与 Java 运行时环境（用于运行 Configuration Agent）关联的密钥库中包含合适的证书。有关 `apoc-config` 的详细信息，请参见 第 15 页中的 “Configuration Agent” 部分。

```
FINER: New client added
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authenticating user geoffh
Nov 18, 2005 2:17:43 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 2:17:43 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.OpenConnectionException: An error occured while connecting to
ldap://sobuild:389.
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
```

```
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
Caused by: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.OpenConnectionException: An error occured while
connecting to ldap://sobuild:389. at
com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyMgr(Unknown Source)
    ... 14 more
Caused by: com.sun.apoc.spi.OpenConnectionException: An error occured while
connecting to ldap://noSuchHost:389.
    at com.sun.apoc.spi.ldap.LdapClientContext.prepareConnection(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapClientContext.connect(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapConnectionHandler.openAuthorizedContext(Unknown Source)
    at com.sun.apoc.spi.ldap.LdapConnectionHandler.connect(Unknown Source)
    at com.sun.apoc.spi.ldap.entities.LdapOrganizationProvider.open(Unknown Source)
    at com.sun.apoc.spi.PolicyMgrFactoryImpl.createPolicyMgr(Unknown Source)
    ... 15 more
Caused by: netscape.ldap.LDAPException: failed to connect to server sobuild:389 (91);
Cannot connect to the LDAP server
    at netscape.ldap.LDAPConnSetupMgr.connectServer(LDAPConnSetupMgr.java:422)
    at netscape.ldap.LDAPConnSetupMgr.openSerial(LDAPConnSetupMgr.java:350)
    at netscape.ldap.LDAPConnSetupMgr.connect(LDAPConnSetupMgr.java:244)
    at netscape.ldap.LDAPConnSetupMgr.access$0(LDAPConnSetupMgr.java:241)
    at netscape.ldap.LDAPConnSetupMgr$1.run(LDAPConnSetupMgr.java:179)
    at java.lang.Thread.run(Thread.java:595)
Nov 18, 2005 2:17:44 PM PolicyBackend openPolicyBackend
```

连接到未配置的策略系统信息库

要使 Configuration Agent 能够在策略系统信息库中找到策略数据，必须先正确配置策略系统信息库。如果指定了未配置或错误配置的策略系统信息库，则当桌面客户端应用程序启动时，会在 Configuration Agent 日志中记录类似以下内容的错误消息。要解决此问题，请参见

```
FINER: New client added
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: CreateSession transaction started
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finer
FINER: Creating new client session
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authenticating user geoffh
Nov 18, 2005 2:36:55 PM com.sun.apoc.daemon.misc.APOCLogger finest
FINEST: Authentication successful
Nov 18, 2005 2:36:55 PM PolicyBackend openPolicyBackend
FINER: THROW
com.sun.apoc.daemon.misc.APOCException: com.sun.apoc.daemon.misc.APOCException:
com.sun.apoc.spi.environment.RemoteEnvironmentException: Error on reading the
configuration data on LDAP server ldap://sobuild:389.
```

```
    at com.sun.apoc.daemon.apocd.PolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.HostPolicyBackend.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.PolicyBackendFactory.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.openPolicyBackend(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache$DataSource.open(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.createDataSources(Unknown Source)
    at com.sun.apoc.daemon.apocd.Cache.<init>(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.createNewCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.CacheFactory.openCache(Unknown Source)
    at com.sun.apoc.daemon.apocd.Session.<init>(Unknown Source)
    at com.sun.apoc.daemon.transaction.CreateSessionTransaction.executeTransaction
(Unknown Source)
    at com.sun.apoc.daemon.transaction.Transaction.execute(Unknown Source)
    at com.sun.apoc.daemon.apocd.ClientEventHandler.handleEvent(Unknown Source)
    at com.sun.apoc.daemon.apocd.EventWorkerThread.run(Unknown Source)
```

我在 Configuration Agent 日志中看到一条关于“客户端最大连接数”的消息。这是什么意思？

由 Configuration Agent 启用的每个桌面客户端应用程序（gconfd、Mozilla 或 StarSuite）在运行时都会打开一个与 Configuration Agent 的连接。这些连接的限制是在代理配置中指定的。默认的连接数限制为 50。如果一台计算机上有多个用户，则可能需要增加此限制的值，方法是使用 Configuration Agent 向导 (`/usr/bin/apoc-config`) 更改“最大客户端连接数”设置。如果 Configuration Agent 达到最大连接数，则会在 Configuration Agent 日志中记录与以下内容类似的错误消息：

```
Nov 18, 2005 3:20:55 PM com.sun.apoc.daemon.misc.APOCLogger warning
WARNING: The maximum number of client connections ( 50 ) has been reached.
No new client connections can be established at this time.
```

我使用 Desktop Manager 修改了一些策略，但是在我的客户端计算机上看不到这些修改

设计 Configuration Agent 时的假设之一是，Desktop Manager 所创建的策略数据是相对静态的（即，不会发生频繁更改）。此假设导致生成了一个步骤，即代理间歇性地参考策略系统信息库，以查看是否进行过修改。默认情况下，代理每小时为所有正在运行的桌面应用程序检查一次系统信息库。因此，使用 Desktop Manager 进行更改之后，至多需要等一小时，正在运行的桌面应用程序即会收到更改通知。如果需要，可以使用 Agent Configuration 向导 (`/usr/bin/apoc-config`) 更改“常规检测间隔”的值，以提高系统信息库检查频率。另外，也可以强制 Configuration Agent 为所有连接的应用程序刷新策略数据，方法是以超级用户身份登录，然后执行 `/usr/lib/apoc/apocd change-detect` 命令。

◆ ◆ ◆ 第 4 章

4 Java Web Console

Java Web Console 旨在为 Sun Microsystems 生成一个通用的、基于 Web 的系统管理解决方案。该控制台提供了一种环境，供用户访问具有一致用户界面的系统管理应用程序。

此控制台之所以基于 Web 模型，有许多原因。但主要原因是使系统管理员可以使用 Web 浏览器来访问系统管理应用程序。

Java Web Console 具有以下特点 :

- 通用的验证和授权
- 通用日志
- 所有系统管理应用程序通过基于 HTTPS 的同一端口共用一个入口点
- 通用的外观

控制台的优点是，管理员登录一次便可使用控制台中的任何应用程序。

安装

系统要求

Java Web Console 支持多个客户端和服务器操作系统，还支持若干个浏览器。

客户端

- Solaris 10 上的 Netscape™ 6.2x 和 7.x
- Windows 98、98 SE、ME、2000 和 XP 上的 Netscape 6.2x 和 7.x
- Windows 98、98 SE、ME、2000 和 XP 上的 Internet Explorer 5.5x 和 6.x
- Solaris 上的 Mozilla 1.4x
- Solaris 上的 Firefox 1.0

服务器

- Solaris 10
- Red Hat Application Server 2.1、3.0
- SuSE Linux 8.0 或更高版本
- J2SE 1.4.1_03 或更高版本

  如果在服务器上检测到 J2SE 1.4.1 或更低版本，安装程序将提示您使用 Java Desktop System Management Tools CD 中的 J2SE 版本对已安装产品进行升级。
- Tomcat：4.0.3 或更高版本

  Tomcat 包含在 Java Desktop System Management Tools CD 中

安装 Java Web Console

Java Web Console 2.2.4 是 Solaris 10 操作系统的一部分，但 Desktop Manager 需要 2.2.5 版本。`server/console` 目录中的 Desktop Manager 归档文件提供了 2.2.5 版本。可以通过在该目录中运行 `./setup` 进行安装。

如果已经安装了 Java Web Console 3.0，则需要卸载 3.0 版，然后从 `server/console` 目录中安装 Java Web Console 2.2.5（如上所述）。

运行控制台

要注册新的应用程序，通常只需停止并重新启动 Java Web Console 服务器。

注意 – 在首次启动 Java Web Console 之前，请确保已经完成了 Desktop Manager 安装。除非已经在控制台中至少部署了一个应用程序，否则 Java Web Console 将不会成功运行。

- 要启动 Java Web Console，请键入 `smcwebserver start`。
- 要停止 Java Web Console，请键入 `smcwebserver stop`。
- 要重新启动 Java Web Console，请键入 `smcwebserver restart`。
- 要访问 Java Web Console，请在浏览器中输入以下 URL：
  `https://`*<hostname>*`.`*<domainname>*`:6789`

Java Web Console 本身即可支持基于 Unix 的验证和基于角色的访问控制 (Role-Based Access Control, RBAC)。但是，您也可以配置其他验证机制，例如 LDAP 验证。

注 – 默认的会话超时时间为 15 分钟。可以使用 `smreg` 命令配置超时长度。例如，要将超时长度设置为 5 分钟，请键入 `smreg add -p -c session.timeout.value=5`。

有关 Java Web Console 命令的详细信息，请参见 smcwebserver 和 smreg 的手册页。

删除 Java Web Console

注意 – 如果使用的是 Solaris，由于 Java Web Console 是操作系统的一部分，因此无法将其删除。

Java Web Console 疑难解答

无法安装 Java Web Console

症状：在安装结束时出现一条消息，指出由于没有注册的应用程序而无法启动 Java Web Console。

可能的原因：安装 Desktop Manager 模块之后，该模块启动了 Java Web Console。

连接被拒绝

症状：尝试打开一个正确的 URL（例如 `https://<your.server>:6789`），但是连接被拒绝。

可能的原因： Java Web Console 未在服务器上运行。

无法登录

注 – 默认情况下不安装 LDAP 登录模块。因此，不会将登录证书与存储在 LDAP 服务器中的登录证书进行比较，只需要常规的系统登录证书。此疑难解答部分仅适用于手动安装 LDAP 登录模块的情况。

症状：已经进入 Web Console 的登录页，但用户名/密码组合被拒绝。

可能的原因：

- 未运行 LDAP 服务器。
- 未正确配置 Web Console LDAP 验证模块。
- LDAP 服务器上不存在此用户。
- 此用户在 LDAP 服务器上具有不同的密码。

无 Desktop Manager 链接

症状：已经登录到 Web Console，但应用程序列表页中未包含 Desktop Manager。

可能的原因：

- 未安装 Desktop Manager 模块

Null 指针异常、Tomcat/Java 错误或显示空白页

症状：打开 Desktop Manager 后未显示任何值，只显示空白页或一些错误消息。

可能的原因：如果错误消息中包含 `NoClassDefFoundError:sun/tools/javac/Main`，则说明 Java Web Console 使用了错误的 Java 安装。

其他问题

如果 Web 服务器未正常运行，日志文件可能会提供相关信息。这些日志文件位于 `/var/log/webconsole/` 中。可以通过使用 `smreg` 命令提高日志的详细程度：

```
smreg add -p debug.trace.level=3
smreg add -p debug.trace.options=tmp
```

可以使用以下命令恢复原始设置：

```
smreg add -p debug.trace.level=0
smreg add -p debug.trace.options=m
```

可以使用以下命令触发配置数据库的完全转储：

```
smreg list
```

运行 Desktop Manager 的 Web 服务器有时可能未正确关闭，从而使其端口一直处于使用状态。这将阻止启动新的 Web 服务器。如果 `smcwebserver start/restart` 命令报告错误消息、执行完 `smcwebserver stop` 命令后仍可访问 Desktop Manager，或者新启动的服务器仍按先前方式运行，请检查 6789 端口是否仍处于使用状态（使用 `netstat -a | grep 6789` 命令），或者检查 Web 服务器是否仍在运行（使用 `ps -ef | grep java` 命令）。如果出现上述任何一种情况，应该中止相关的进程，不再使用 6789 端口。

◆ ◆ ◆ 附 录 A

配置参数

可以为以下 Desktop Manager 组件定义配置参数：

- Desktop Manager，在用于定义配置系统信息库的文件（位于 `/etc/opt/SUNWapcmg/`）中定义参数。
- Configuration Agent，在 `/etc/apoc/policymgr.properties` 文件中定义参数。
- Desktop Manager CLI，在 `$HOME/pgtool.properties` 文件中定义参数，CLI 只支持纯 LDAP 系统信息库。

可以为参数添加前缀，以指出应用这些参数的系统信息库提供程序。每种提供程序都会首先考虑带有前缀的参数。如果未定义此类参数，则使用不带前缀的参数。

表 A–1 前缀

前缀值	系统信息库提供程序
ORGANIZATION_	组织树
DOMAIN_	域树
PROFILE_	配置文件
ASSIGNMENT_	指定
LDAP_META_CONF_	映射数据（如果使用 LDAP 系统信息库）

表 A–2 参数

名称	描述	可能值	默认值

PROVIDER_URL	用于指定系统信息库连接的 URL。URL 列表可用于指定备用系统信息库，以便在无法连接到第一个系统信息库时使用。	由一个或多个以空格分隔的 URL 所组成的列表，每个 URL 使用以下某种格式：
		`ldap://<host>:<port>/<baseDN>`
		`ldaps://<host>:<port>/<baseDN>`
		`file://<filepath>`
		`http://<host>:<port>/<filepath>`
		`https://<host>:<port>/<filepath>`
			无，强制性参数

SECURITY_PRINCIPAL	用于连接系统信息库的用户名。	对系统信息库具有读取和搜索访问权限的用户的用户名（匿名连接时不指定此值）。	无，匿名连接
SECURITY_CREDENTIALS	在 SECURITY_PRINCIPAL 中定义的用户密码。	杂乱或明文密码。	无
SECURITY_CREDENTIALS_ENCODING	指出在 SECURITY_PRINCIPAL 中定义的密码是否为杂乱密码。警告：密码的杂乱处理只是对密码的一种掩饰，而不是任何类型的安全加密。	如果密码进行了杂乱处理，则此值为 "scramble"（生成配置数据时由向导自动完成）。如果密码以明文显示，则此值为 "none"；如果要编辑密码，请使用此值。	"none"
MAX_SEARCH_RESULT	对任意系统信息库进行搜索后显示的最大结果数。注意：前缀方案不适用于此参数。	正数，0 表示无限制。	100

以下参数只适用于 LDAP 系统信息库。

表 A–3 特定于 LDAP 的参数

名称	描述	可能值	默认值
AuthDn	首次访问 LDAP 系统信息库时使用的用户全限定 DN，用于检索在 SECURITY_PRINCIPAL 中定义的用户。	对系统信息库具有读取和搜索访问权限的用户的用户名（匿名连接时不指定此值）。	无，匿名访问

表 A–3 特定于 LDAP 的参数 （续）

名称	描述	可能值	默认值
密码	AuthDN 的密码。	杂乱或明文密码。	无
Password_ENCODING	指出在 Password 中定义的密码是否为杂乱密码。警告：密码的杂乱处理只是对密码的一种掩饰，而不是任何类型的安全加密。	如果密码进行了杂乱处理，则此值为 "scramble"（生成配置数据时由向导自动完成）。如果密码以明文显示，则此值为 "none"；如果要编辑密码，请使用此值。	"none"
连接超时	连接创建超时（以秒为单位）。	正数，0 表示无时间限制。	1

示例 A–1 混合式后端的示例

以下示例为混合式后端，其中有关主机和用户的信息来自现有的 LDAP 系统信息库，而配置文件及其指定则存储于文件系统中。

```
#Organization, Domain, MetaConf
PROVIDER_URL = ldap://server1.sun.com:389/o=apoc ldap://server2.sun.com:389/o=apoc
SECURITY_PRINCIPAL = jmonroe
SECURITY_CREDENTIALS = JmonroE
SECURITY_CREDENTIALS_ENCODING = none
AuthDn = cn=reader,ou=special users,o=apoc
Password = lakjflajf
Password_ENCODING = scramble
ConnectTimeout = 5

#Profile
PROFILE_PROVIDER_URL = file:///path/to/repository

#Assignment
ASSIGNMENT_PROVIDER_URL = file:///path/to/repository
```

◆ ◆ ◆ 附 录 B

在 Desktop Manager 中使用 OpenLDAP 和 Active Directory

在 Desktop Manager 中使用 OpenLDAP 服务器

要将 OpenLDAP 服务器用作 Desktop Manager 数据的系统信息库，必须扩展服务器模式，以支持用来存储配置数据的对象类和属性。可以在 `/usr/share/webconsole/apoc/deploy` 目录中找到名为 `apoc.schema` 的自定义模式文件。

必须将此文件复制到 OpenLDAP 配置目录 (`/etc/openldap`) 的 `schema` 子目录中，并通过在 OpenLDAP 配置目录中的 `slapd.conf` 文件中添加此自定义模式文件来将其添加到 OpenLDAP 模式中。要执行此操作，可以在该文件所包含的模式序列结尾处插入一行 `include /etc/openldap/schema/apoc.schema`。有关扩展 OpenLDAP 服务器模式的详细信息，请参阅该服务器的手册。

扩展了 OpenLDAP 服务器模式之后，即可使用 Desktop Manager 中的“添加配置系统信息库”向导来完成其余的配置。

注 – Desktop Manager Agent 将通过提供所需数据的用户的 DN（但不提供密码），尝试匿名连接 OpenLDAP 服务器。OpenLDAP 服务器的某些发行版在默认情况下可能会禁用这种匿名验证模式，在这种情况下，必须通过在通用服务器参数中添加一行 `allow bind_anon_cred` 来启用此模式，通用服务器参数是在 OpenLDAP 配置目录 (`/etc/openldap`) 中的 `slapd.conf` 文件中定义的。有关该参数的详细信息，请参阅该服务器的手册。

在 Desktop Manager 中使用 Active Directory 服务器

要将 Active Directory 服务器用作 Desktop Manager 数据的系统信息库，必须扩展服务器模式，以支持用来存储配置数据的对象类和属性。可以在 `/usr/share/webconsole/apoc/deploy` 目录中找到名为 `apoc-ad.ldf` 的模式扩展文件。

必须通过以下步骤将 `apoc-ad.ldf` 文件导入 Active Directory 模式：

1. 启用模式扩展。有关如何执行该操作的详细信息，请参阅 Active Directory 文档。

2. 在命令提示符下执行以下命令：**`ldifde -i -c "DC=Sun,DC=COM" <`*`BaseDN`*`> -f apoc-ad-registry.ldf`** 。

注 – 将 <*BaseDN*> 替换为 Active Directory 的基本 DN。

扩展了 Active Directory 服务器模式之后，即可使用 Desktop Manager 中的“添加配置系统信息库”向导来完成其余的配置。

如果“添加配置系统信息库”向导提示需要 LDAP 证书，请提供对此树具有读取权限的用户的完整 DN 和密码。此用户不能将 Active Directory 用作任何其他目的。有关如何设置此类用户的详细信息，请参阅 Active Directory 文档。此外，Active Directory 的域名对于运行 Desktop Manager 的计算机来说必须是已知的。要实现此目的，可以在该计算机的 `/etc/hosts` 文件中添加一行内容，将 Active Directory 服务器的 IP 地址映射到其域名。

要从桌面主机检索配置数据，Active Directory 的域名对于该主机来说也必须是已知的。可以通过两种方法验证桌面用户：匿名验证和使用 GSSAPI 进行验证。

- 如果要使用匿名连接进行验证，则必须配置 Active Directory 服务器以将读取权限授予每个用户。有关如何执行该操作的详细信息，请参阅 Active Directory 文档。
- 要使用 GSSAPI 进行验证，用户必须通过 Active Directory 的验证，并且系统中必须具有用户证书。可以通过在系统中配置 Kerberos 验证来实现此目的，因为这会在登录时生成这些证书。请参阅系统的管理员指南，以了解有关如何执行此操作的详细信息。

◆ ◆ ◆ 附 录 C

组织映射

组织映射

要定义 LDAP 条目和 Desktop Manager 元素之间的映射，必须编辑 `Organization` 映射文件。必须为各个键提供与 LDAP 系统信息库的版式相匹配的值。

用户元素通过一个所有元素都使用的对象类和一个属性（其值在整个系统信息库中必须是唯一的）进行标识。可提供显示名称的格式，此格式将影响用户在管理应用程序中如何显示；且如果组织中用户条目使用容器条目，则可以选择定义该条目。键名及其默认值包括：

```
# 所有用户条目使用的对象类
User/ObjectClass=inetorgperson
# 属性，用户条目中该属性的值在系统信息库中是唯一的
User/UniqueIdAttribute=uid
# 用户条目的组织条目中的可选容器，
# 如果未使用，则删除该行
User/Container=ou=People
# 管理应用程序中的显示名称的格式
User/DisplayNameFormat=sn, givenname
```

角色元素通过其可能使用的对象类列表和相应的命名属性进行标识。这些列表使用 `<item1>,<item2>,...,<itemN>` 格式，并且必须对齐。也就是说，这些列表必须具有相同数目的项，并且第 n 个对象类必须与第 n 个命名属性一同使用。通过两个键定义了角色和用户之间的关系以及角色和主机之间的关系。*VirtualMemberAttribute* 键必须指定可以从用户条目或主机条目中查询属性值的属性。该键还必须含有条目所属角色的完整 DN。*MemberAttribute* 键必须从用户或主机条目中为搜索过滤器指定一个属性。该键还必须含有用户或主机所属角色的完整 DN。*VirtualMemberAttribute* 键可以是服务类虚拟属性，但 *MemberAttribute* 键必须是可以在过滤器中使用的真实属性。键名及其默认值包括：

```
# 角色的对象类列表
Role/ObjectClass=nsRoleDefinition
# 相应命名属性的对齐列表
Role/NamingAttribute=cn
```

```
# 包含用户/主机所属角色的 DN
# 的真实属性（可在过滤器中使用）
Role/MemberAttribute=nsRoleDN
# 对用户/主机的查询返回用户/主机
# 所属角色的 DN 的属性
Role/VirtualMemberAttribute=nsRole
```

组织元素的标识方式与角色类似，它也使用了两个对齐的列表，即对象类列表和相应命名属性列表。键名及其默认值包括：

```
# 组织的对象类列表
Organization/ObjectClass=organization
# 相应命名属性的对齐列表
Organization/NamingAttribute=o
```

域元素的标识方式与组织元素类似。键名及其默认值包括：

```
# 域的对象类列表
Domain/ObjectClass=ipNetwork
# 相应命名属性的对齐列表
Domain/NamingAttribute=cn
```

主机元素的标识方式与用户元素类似。键名及其默认值包括：

```
# 所有主机条目使用的对象类
Host/ObjectClass=ipHost
# 属性，主机条目中该属性的值在系统信息库中是唯一的
Host/UniqueIdAttribute=cn
# 主机条目的域条目中的可选容器，
# 如果未使用，则删除该行
Host/Container=ou=Hosts
```