在 Oracle® Solaris 11.2 中复制和创建软件包系统信息库

文件号码 E53763-02
2014 年 9 月

ORACLE

版权所有 © 2011, 2014, Oracle 和/或其附属公司。保留所有权利。

本软件和相关文档是根据许可证协议提供的，该许可证协议中规定了关于使用和公开本软件和相关文档的各种限制，并受知识产权法的保护。除非在许可证协议中明确许可或适用法律明确授权，否则不得以任何形式、任何方式使用、拷贝、复制、翻译、广播、修改、授权、传播、分发、展示、执行、发布或显示本软件和相关文档的任何部分。除非法律要求实现互操作，否则严禁对本软件进行逆向工程设计、反汇编或反编译。

此文档所含信息可能随时被修改，恕不另行通知，我们不保证该信息没有错误。如果贵方发现任何问题，请书面通知我们。

如果将本软件或相关文档交付给美国政府，或者交付给以美国政府名义获得许可证的任何机构，必须符合以下规定：

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

本软件或硬件是为了在各种信息管理应用领域内的一般使用而开发的。它不应被应用于任何存在危险或潜在危险的应用领域，也不是为此而开发的，其中包括可能会产生人身伤害的应用领域。如果在危险应用领域内使用本软件或硬件，贵方应负责采取所有适当的防范措施，包括备份、冗余和其它确保安全使用本软件或硬件的措施。对于因在危险应用领域内使用本软件或硬件所造成的一切损失或损害，Oracle Corporation 及其附属公司概不负责。

Oracle 和 Java 是 Oracle 和/或其附属公司的注册商标。其他名称可能是各自所有者的商标。

Intel 和 Intel Xeon 是 Intel Corporation 的商标或注册商标。所有 SPARC 商标均是 SPARC International, Inc 的商标或注册商标，并应按照许可证的规定使用。AMD、Opteron、AMD 徽标以及 AMD Opteron 徽标是 Advanced Micro Devices 的商标或注册商标。UNIX 是 The Open Group 的注册商标。

本软件或硬件以及文档可能提供了访问第三方内容、产品和服务的方式或有关这些内容、产品和服务的信息。对于第三方内容、产品和服务，Oracle Corporation 及其附属公司明确表示不承担任何种类的担保，亦不对其承担任何责任。对于因访问或使用第三方内容、产品或服务所造成的任何损失、成本或损害，Oracle Corporation 及其附属公司概不负责。

目录

示例

使用本文档

- **概述** – 介绍如何使用 Oracle Solaris 映像包管理系统 (Image Packaging System, IPS) 功能创建、复制、更新和维护软件包系统信息库以及使其可供访问。
- **目标读者** – 安装和管理软件的系统管理员，或者帮助其他用户安装和管理软件的系统管理员。
- **必备知识** – 具有使用 Oracle Solaris 服务管理工具 (Service Management Facility, SMF) 功能的经验及管理 NFS 和 Web 服务器的经验。

产品文档库

位于 http://www.oracle.com/pls/topic/lookup?ctx=E56344 的文档库中包含此产品的最新信息和已知问题。

获得 Oracle 支持

Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关本文档的反馈。

♦ ♦ ♦ 第 1 章

1 映像包管理系统软件包系统信息库

Oracle Solaris 11 软件在映像包管理系统 (Image Packaging System, IPS) 软件包中分发。IPS 软件包存储在 IPS 软件包系统信息库中，后者由 IPS 发布者填充。

本指南介绍了如何使用 Oracle Solaris 映像包管理系统 (Image Packaging System, IPS) 功能创建软件包系统信息库。使用 IPS 工具，您可以轻松复制现有的系统信息库或者为自己的软件包创建自己的系统信息库，并且可以轻松更新系统信息库中的软件包。您可以为系统信息库的用户提供文件接口以及 HTTP 或 HTTPS 接口。本指南还介绍了如何自动更新系统信息库以及如何克隆系统信息库，并说明了 Apache Web 服务器配置（例如高速缓存、负载平衡和配置 HTTPS 访问）。

本章介绍以下内容：

- 创建本地 IPS 软件包系统信息库供内部使用的可能原因
- 创建软件包系统信息库的最佳做法
- 托管系统信息库的系统要求

本地 IPS 系统信息库

出于以下原因，您可能需要本地 IPS 系统信息库：

- **性能和安全。**您不希望客户机系统转到 Internet 检索新的软件包或更新现有软件包。
- **更改控制。**您希望确保明年可以执行与今天相同的安装。您希望轻松地控制系统可更新到的版本。
- **定制软件包。**您希望交付定制 IPS 软件包。

创建和使用本地 IPS 软件包系统信息库的最佳做法

使用以下最佳做法维护系统信息库的可用性并最大程度地减少错误。

包含所有支持系统信息库更新 (Support Repository Update, SRU) 的所有内容。

使用所有支持更新内容使本地系统信息库保持最新状态。支持更新包含安全更新和其他重要修复。Oracle Solaris OS 软件包系统信息库的每个次要发行版和更新都作为完整的一套软件包发布。而 SRU 仅作为已更改软件包的零散更新发布。

- 勿将支持更新中的一部分软件包添加到系统信息库。请将支持更新的所有内容添加到本地系统信息库。
- 不跳过支持更新。在每个系统信息库中累积应用所有适用的支持更新。
- 勿删除 Oracle 发布者提供的软件包。
- 使用 `svc:/application/pkg/mirror` 服务管理工具 (Service Management Facility, SMF) 服务自动更新 Oracle 支持系统信息库中的本地主系统信息库。有关说明，请参见如何从 Internet 自动复制系统信息库 [19]。

用户可以通过指定要安装的 `entire` incorporation 软件包的版本来更新到低于系统信息库中的最新版本的版本。请参见《在 Oracle Solaris 11.2 中添加和更新软件 》中的第 4 章 “更新或升级 Oracle Solaris 映像”。

每次更新系统信息库时都进行验证。

每当更改系统信息库的内容或属性值时，使用 `pkgrepo verify` 命令。`pkgrepo verify` 命令可验证系统信息库内容的以下属性是否正确：

- 文件校验和。
- 文件权限。检查系统信息库文件和目录以及指向系统信息库的路径，以确保 `pkg5srv` 用户可以读取系统信息库内容。
- 软件包清单权限。
- 软件包清单内容。
- 软件包签名。

在共享位置创建系统信息库。

共享位置是不在任何可引导环境 (bootable environment, BE) 中的位置。共享位置的示例包括 `/var/share` 和 `/export`。在共享位置创建系统信息库具有以下优点：

- 可以轻松地从其他现有 BE 中访问系统信息库。
- 通过更新或克隆现有 BE 来创建新 BE 时，不会因为存在系统信息库的多个副本而浪费空间。
- 无需浪费时间和 I/O 资源来重新应用已在其他 BE 中应用过的系统信息库更新。

如果要使用非全局区域，则必须可以从全局区域访问在非全局区域中配置的发布者的所有位置，即使未在全局区域中配置该发布者也是如此。

在其自己的 ZFS 文件系统中创建每个系统信息库。

使用单独的 ZFS 文件系统可以执行以下操作：

- 获得更好的性能。
- 为单独的文件系统设置特征。例如，将 `atime` 设置为 `off` 可在更新系统信息库时获得更好的性能。`atime` 属性控制是否在读取文件时更新文件的访问时间。关闭此属性可避免在读取文件时产生写入流量。

- 管理资源使用情况。为每个系统信息库数据集指定适当的磁盘配额，以确保大型系统信息库更新不会占用完池中的所有空间。如果您是自动执行更新（如如何从 Internet 自动复制系统信息库 [19]中所述），则此最佳做法尤其重要。
- 创建快照。

每次更新系统信息库时都生成快照。

每次更新系统信息库时都生成系统信息库文件系统的快照具有以下优点：

- 可从快照回滚到系统信息库的先前版本。
- 可从快照更新系统信息库以最大限度地减少用户中断。

提供高可用性。

- 在不同位置中维护系统信息库的克隆。有关说明，请参见“维护多个相同的本地系统信息库” [32]。
- 配置 Web 服务器以提供多个系统信息库并对其进行高速缓存和负载平衡。有关信息，请参见第 5 章 在 Web 服务器后运行 Depot 服务器。

保证本地系统信息库安全。

有关说明，请参见“配置系统信息库的 HTTPS 访问” [49]。

系统要求

托管 IPS 软件包系统信息库的系统可以是基于 x86 或基于 SPARC 的系统。

操作系统

运行 Oracle Solaris 11 11/11 的系统信息库服务器支持所有 Oracle Solaris 11 更新软件包。

磁盘空间

要托管 Oracle Solaris 11.2 发行版系统信息库的副本，系统信息库服务器必须具有 16 GB 的空闲空间。

由于最佳做法是使用所有支持更新使本地系统信息库保持最新状态，因此计划每年针对支持更新使用 10-15 GB 的额外空间。其他软件（例如 Oracle Solaris Studio 或 Oracle Solaris Cluster）当然也要求在软件包系统信息库中有额外空间。

如果一个系统托管多个 IPS 系统信息库，请为每个系统信息库创建一个单独的 ZFS 文件系统，以便单独回滚和恢复每个系统信息库。

系统信息库管理特权

可以使用以下方法之一获取创建和配置软件包系统信息库所需的特权。有关配置文件和角色的更多信息，包括如何确定需要哪个配置文件和角色，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》。

权限配置文件

可使用 `profiles` 命令列出分配给您的权限配置文件。以下配置文件对于维护本地软件包系统信息库非常有用：

ZFS File System Management（ZFS 文件系统管理）

使用此权限配置文件可以运行 `zfs` 命令。

Software Installation（软件安装）

使用此权限配置文件可以运行 `pkg` 命令。

Service Management（服务管理）

使用此权限配置文件可以运行 SMF 命令（例如 `svccfg`）。

角色

使用 `roles` 命令列出分配给您的角色。如果您具有 `root` 角色，则可以使用 `root` 口令通过 `su` 命令来承担 `root` 角色。

sudo 命令

根据站点的安全策略，可以使用 `sudo` 命令和用户口令来执行特权命令。

♦ ♦ ♦ 第 2 章

2 复制 IPS 软件包系统信息库

本章介绍了创建 Oracle Solaris IPS 软件包系统信息库副本的两种方法：可以使用介质或 Oracle Solaris 下载站点上的系统信息库文件，也可以通过 Internet 手动或自动检索系统信息库内容。在所有情况下，首先在共享位置为本地软件包系统信息库创建单独的 ZFS 文件系统。创建系统信息库之后，对该系统信息库进行验证并生成快照。

本章还介绍了与复制系统信息库相关的性能和故障排除信息。

复制系统信息库的性能注意事项

如果从 Oracle Solaris 下载站点下载系统信息库文件，或者如果使用“从 Internet 复制系统信息库” [18]中显示的 `pkgrecv` 命令从 Internet 位置检索系统信息库内容，请参考以下配置以提高传输性能：

- 确保 ZFS 存储池容量小于 80%。使用 `zpool list` 命令查看池容量。
- 如果使用代理，请检查代理的性能。
- 关闭使用大量内存的应用程序。
- 确保临时目录中有足够的空闲空间。在操作期间，`pkgrecv` 命令使用 `$TMPDIR` 作为临时存储目录。如果未设置 `TMPDIR`，`pkgrecv` 将使用 `/var/tmp` 作为临时存储的目录。请确保 `$TMPDIR` 或 `/var/tmp` 具有足够的空闲空间，适合所执行的 `pkgrecv` 操作的规模。
- 如果使用 `pkgrecv` 命令复制大型系统信息库，请考虑使用 `--clone` 选项。使用 `--clone` 选项速度更快，并且占用的内存更少。请参见如何克隆本地 IPS 软件包系统信息库 [33]。
- 如果使用 `pkgrecv` 命令创建或更新大型系统信息库，请考虑对目标系统信息库使用 SSD。软件包检索完成之后，可以根据需要移动系统信息库。

本地软件包系统信息库的故障排除

下列方法可避免出现问题或者可以帮助找到可能遇到的问题的原因：

- 验证系统信息库源文件。如果使用 .zip 文件创建系统信息库，请通过使用校验和来确认系统上的文件是否正确，如如何从 zip 文件复制系统信息库 [14]中所述。
- 验证已安装的系统信息库。使用 pkgrepo verify 命令检查已安装的系统信息库。

  pkgrepo verify 会报告以下权限问题：

  - 文件权限。为避免基于文件系统的系统信息库的目录权限和文件权限出现问题，请确保 pkg5srv 用户具有读取系统信息库的权限。
  - 目录权限。确保系统信息库中的所有目录都具有执行权限。

  如果 pkgrepo verify 命令报告其他类型的错误，请尝试使用 pkgrepo fix 命令修复这些错误。有关更多信息，请参见 pkgrepo(1) 手册页。
- 检查发布者源。确保为每个映像中的每个发布者设置相应的源。要更新已安装的软件包，请安装依赖于已安装软件包的软件包，或者安装非全局区域，设置为发布者源的系统信息库必须至少包含在其中设置发布者的相应映像中所安装的相同软件。请参见如何使用户能够使用文件接口检索软件包 [23]中的步骤 3。有关设置发布者以及排除软件包安装问题的信息，请参见《在 Oracle Solaris 11.2 中添加和更新软件》。
- 检查 Web 服务器配置。如果将 Apache Web 服务器配置为访问系统信息库，请将 Web 服务器配置为不对已编码的正斜杠进行解码。请参见“必需的 Apache 配置设置” [43]中的说明。对已编码的正斜杠进行解码可导致 "package not found"（找不到软件包）错误。
- 不要创建只能从非全局区域访问的系统信息库。必须能够从全局区域访问在非全局区域中配置的发布者的所有位置，即使未在全局区域中配置该发布者也是如此。

从文件复制系统信息库

本节介绍如何从一个或多个系统信息库文件创建 Oracle Solaris 软件包系统信息库的本地副本。系统信息库文件可能位于介质中，也可能在 Oracle Solaris 下载站点上提供。系统信息库文件可能是 zip 文件或 iso 文件。

▼ 如何从 zip 文件复制系统信息库

1. **为新系统信息库创建 ZFS 文件系统。**

   在共享位置创建系统信息库。创建系统信息库文件系统时，请将 atime 设置为 off。请参见“创建和使用本地 IPS 软件包系统信息库的最佳做法” [9]。

```
$ zfs create -o atime=off rpool/export/IPSpkgrepos
$ zfs create rpool/export/IPSpkgrepos/Solaris
$ zfs get atime rpool/export/IPSpkgrepos/Solaris
NAME                              PROPERTY  VALUE  SOURCE
rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
```

2. 获取软件包系统信息库文件。

   可以从下载系统安装映像的位置下载 Oracle Solaris IPS 软件包系统信息库 .zip 文件，或者在介质包中找到系统信息库 DVD。连同 .zip 文件一起下载 install-repo.ksh 脚本和 .txt 文件（README 及校验和文件）。

   ```
   $ ls
   install-repo.ksh           sol-11_2-ga-repo-3of4.zip
   README-zipped-repo.txt     sol-11_2-ga-repo-4of4.zip
   sol-11_2-ga-repo-1of4.zip  sol-11_2-ga-repo.txt
   sol-11_2-ga-repo-2of4.zip
   ```

3. 确保脚本文件是可执行文件。

   ```
   $ chmod +x install-repo.ksh
   ```

4. 运行系统信息库安装脚本。

   系统信息库安装脚本 install-repo.ksh 将每个系统信息库 .zip 文件解压缩到指定目录。该脚本有选择地执行以下额外任务：

   - 验证下载的 .zip 文件的校验和。如果验证校验和时未指定 -c 选项，请在运行系统信息库安装脚本之前手动验证校验和。运行以下 digest 命令，并将输出与 .md5 文件中的相应校验和进行比较：

     ```
     $ digest -a md5 file
     ```

   - 如果指定目标已经包含系统信息库，则将系统信息库内容添加到现有内容。
   - 验证最终系统信息库。如果验证系统信息库时未指定 -v 选项，请在运行系统信息库安装脚本之后，使用 pkgrepo 命令的 info、list 和 verify 子命令验证系统信息库。
   - 创建 ISO 映像文件以供挂载和分发。如果使用 -I 选项创建 .iso 文件，则 .iso 文件和说明如何使用 .iso 文件的 README 文件位于指定的目标目录中。

5. 验证系统信息库内容。

   如果未在上一步骤中指定 -v 选项，则使用 pkgrepo 命令的 info、list 和 verify 子命令检查是否已正确复制系统信息库。如果 pkgrepo verify 命令报告了错误，请尝试使用 pkgrepo fix 命令修复这些错误。有关更多信息，请参见 pkgrepo(1) 手册页。

6. 生成新系统信息库快照。

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

例 2-1 从 zip 文件创建新系统信息库

在本示例中，解压缩 zip 文件之前，不存在系统信息库。脚本可以采用以下选项：

-s 可选。指定 .zip 文件所在目录的完整路径。缺省值：当前目录。

-d 必填。指定希望系统信息库所在的目录的完整路径。

`-i` 可选。指定用于填充此系统信息库的文件。源目录可以包含多个 `.zip` 文件集。缺省值：源目录中可用的最新映像。

`-c` 可选。将 `.zip` 文件的校验和与指定文件中的校验和进行比较。如果指定不带参数的 `-c`，则使用的缺省文件为源目录中的 `-i` 映像的 `.md5` 文件。

`-v` 可选。验证最终系统信息库。

`-I` 可选。在源目录中创建系统信息库的 ISO 映像。还请在源目录中保留 `mkiso.log` 日志文件。

`-h` 可选。显示用法消息。

```
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ ls /export/IPSpkgrepos/Solaris
COPYRIGHT         NOTICES           pkg5.repository   publisher         README-iso.txt
```

系统信息库重建和验证可能需要一些时间才能完成，但是在看到 "Repository can be found in"（在以下位置中可以找到系统信息库）消息后，即表明可以检索系统信息库内容。

例 2-2 从 zip 文件添加现有系统信息库

在本示例中，将系统信息库 zip 文件的内容添加到现有软件包系统信息库中的内容。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4764     online           2014-03-18T05:30:57.221021Z
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
IPS repository exists at destination /export/IPSpkgrepos/Solaris
Current version: 0.175.2.0.0.35.0
Do you want to add to this repository? (y/n) y
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
```

```
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository rebuild.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-06-02T18:11:55.640930Z
```

▼ 如何从 iso 文件复制系统信息库

1. **为新系统信息库创建 ZFS 文件系统。**

   在共享位置创建系统信息库。创建系统信息库文件系统时，请将 atime 设置为 off。请参见“创建和使用本地 IPS 软件包系统信息库的最佳做法” [9]。

   ```
   $ zfs create -o atime=off rpool/export/IPSpkgrepos
   $ zfs create rpool/export/IPSpkgrepos/Solaris
   $ zfs get atime rpool/export/IPSpkgrepos/Solaris
   NAME                              PROPERTY  VALUE  SOURCE
   rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
   ```

2. **获取软件包系统信息库映像文件。**

   使用 -I 选项从系统信息库 .zip 文件创建 .iso 文件，如例 2-1 “从 zip 文件创建新系统信息库”中所述。

3. **挂载映像文件。**

   挂载系统信息库 .iso 文件以访问其内容。

   ```
   $ mount -F hsfs /path/sol-11_2-repo.iso /mnt
   ```

   要避免系统信息库服务器系统每次重新启动时都重新挂载 .iso 映像，请按照下一步骤中所述复制系统信息库文件内容。

4. **将系统信息库内容复制到新位置。**

   要提高系统信息库的访问性能并避免系统每次重新启动时都重新挂载 .iso 映像，请将系统信息库文件从 /mnt/repo 复制到 ZFS 文件系统。可以使用 rsync 命令或 tar 命令执行此复制。

   - **使用 rsync 命令。**

     如果使用 rsync 命令，请确保指定 /mnt/repo/（包括末尾的斜杠字符）而非 /mnt/repo 以复制 repo 目录中的文件和子目录。请参见 rsync(1) 手册页。

     ```
     $ rsync -aP /mnt/repo/ /export/IPSpkgrepos/Solaris
     ```

- **使用 tar 命令。**
  使用以下示例中所示的 tar 命令可以更快速地将系统信息库从已挂载文件系统复制到系统信息库 ZFS 文件系统。

  ```
  $ cd /mnt/repo; tar cf - . | (cd /export/IPSpkgrepos/Solaris; tar xfp -)
  ```

5. **取消挂载映像文件。**
   确保您不再位于 /mnt 目录下。

   ```
   $ umount /mnt
   ```

6. **验证新系统信息库内容。**
   使用 pkgrepo 命令的 info、list 和 verify 子命令检查是否已正确复制系统信息库。如果 pkgrepo verify 命令报告了错误，请尝试使用 pkgrepo fix 命令修复这些错误。有关更多信息，请参见 pkgrepo(1) 手册页。

7. **生成新系统信息库快照。**

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

从 Internet 复制系统信息库

本节描述如何通过从 Internet 位置复制系统信息库来创建 Oracle Solaris 软件包系统信息库的本地副本。第一个过程说明从命令行发出复制命令。第二个过程说明使用 SMF 服务自动复制和更新系统信息库。

▼ 如何从 Internet 显式复制系统信息库

1. **为新系统信息库创建 ZFS 文件系统。**
   在共享位置创建系统信息库。创建系统信息库文件系统时，请将 atime 设置为 off。请参见“创建和使用本地 IPS 软件包系统信息库的最佳做法” [9]。

   ```
   $ zfs create -o atime=off rpool/export/IPSpkgrepos
   $ zfs create rpool/export/IPSpkgrepos/Solaris
   $ zfs get atime rpool/export/IPSpkgrepos/Solaris
   NAME                              PROPERTY  VALUE  SOURCE
   rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
   ```

2. **创建必需的系统信息库基础结构。**
   创建必需的 pkg(5) 系统信息库基础结构，以便复制系统信息库。前面方法中使用的映像文件包含系统信息库基础结构，因此不需要执行这一步骤。在按照此方法所述使用

pkgrecv 命令复制系统信息库内容时，需要创建系统信息库基础结构，然后将系统信息库内容复制到该基础结构中。请参见 pkg(5) 和 pkgrepo(1) 手册页。

```
$ pkgrepo create /export/IPSpkgrepos/Solaris
```

3. 将系统信息库内容复制到新位置。

   使用 pkgrecv 命令可复制系统信息库。此操作可能会影响网络性能。完成此操作所需的时间取决于网络带宽和连接速度。另请参见“复制系统信息库的性能注意事项” [13]。如果稍后更新此系统信息库，则仅传送更改内容，此过程所需的时间也会大大减少。

   以下命令从 -s 选项指定的软件包系统信息库中检索所有软件包的所有版本，放入 -d 选项指定的系统信息库中。如果要从安全站点复制，请确保已安装必需的 SSL 证书和密钥，并指定必需的证书和密钥选项。

   ```
   $ pkgrecv -s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris \
   --key /path-to-ssl_key --cert /path-to-ssl_cert '*'
   ```

   有关 -m 和 --clone 选项的信息，请参见 pkgrecv(1) 手册页。不应将 -m latest 选项用于此目的。使用太稀疏的系统信息库可能会导致在用户尝试更新其映像时出现错误。

4. 验证新系统信息库内容。

   使用 pkgrepo 命令的 info、list 和 verify 子命令检查是否已正确复制系统信息库。如果 pkgrepo verify 命令报告了错误，请尝试使用 pkgrepo fix 命令修复这些错误。有关更多信息，请参见 pkgrepo(1) 手册页。

5. 生成新系统信息库快照。

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

▼ 如何从 Internet 自动复制系统信息库

缺省情况下，svc:/application/pkg/mirror SMF 服务从 /var/share/pkg/repositories/solaris 的相应映像中定义的 solaris 发布者源定期地执行 pkgrecv 操作。pkgrecv 操作在每月某一天的凌晨 2:30 开始。要更改该缺省行为，请按照此过程所述对服务进行配置。

在每次成功运行此服务结束时，系统信息库目录都会刷新。无需刷新系统信息库以构建搜索索引。

由于此服务定期运行，因此将创建系统信息库并保持更新状态。无需按照本文档中所示的系统信息库手动更新说明进行操作。

其他系统可将其 solaris 发布者源设置为此自动更新的系统信息库，或者设置为此系统信息库的克隆。仅一个系统需要具有 Internet 发布者源并运行 mirror 服务以自动接收更新。

1. 设置发布者源。

   缺省情况下，mirror 服务会传输根目录为 / 的映像中配置的 solaris 发布者的软件包。虽然无法直接在 mirror 服务配置中指定发布者源，但是您可以配置从中检索此信息的映像根目录。在该映像根目录下，使用 pkg set-publisher 配置要用作镜像系统信息库的 pkgrecv 传输源的发布者源。

   a. （可选）设置映像根目录。

      如果要用于 mirror 服务的发布者配置与要在此映像中使用的发布者配置不同，请在共享位置（未包含在任何 BE 中）创建用户映像，并将 mirror 服务中 config/ref_image 属性的值重置为该新映像，如以下示例所示。mirror 服务将使用 config/ref_image 映像中的发布者配置。

      ```
      $ svccfg -s pkg/mirror:default setprop config/ref_image = /var/share/pkg/
      mirror_svc_ref_image
      $ pkg image-create /var/share/pkg/mirror_svc_ref_image
      ```

   b. （可选）设置发布者。

      如果要使用除 solaris 发布者外的其他发布者中的软件包更新镜像系统信息库，请重置 mirror 服务中的 config/publishers 属性的值，如以下介绍添加 ha-cluster 和 solarisstudio 发布者的示例中所示。

      ```
      $ svccfg -s pkg/mirror:default setprop config/publishers = solaris,ha-
      cluster,solarisstudio
      ```

   c. 设置发布者源。

      由于此服务定期运行，因此应将发布者源设置为提供常规更新的系统信息库。对于 Oracle 产品，可能需要将发布者源设置为某个支持系统信息库，以检索支持系统信息库更新 (Support Repository Update, SRU)。在以下示例中，仅当在备用映像根目录中配置发布者时才需要 -R 选项。可能不需要 -k 和 -c 选项，具体取决于源 URI。

      ```
      $ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
      -g https://pkg.oracle.com/solaris/support/ -k ssl_key -c ssl_cert solaris
      $ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
      -g https://pkg.oracle.com/ha-cluster/support/ -k ssl_key -c ssl_cert ha-cluster
      $ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
      -g https://pkg.oracle.com/solarisstudio/support/ -k ssl_key -c ssl_cert solarisstudio
      ```

      使用以下命令之一验证在新映像中配置的发布者：

      ```
      $ pkg -R /var/share/pkg/mirror_svc_ref_image publisher
      $ pkg -R /var/share/pkg/mirror_svc_ref_image publisher solaris ha-cluster
       solarisstudio
      ```

2. （可选）配置 mirror 服务的其他属性。

   可能需要修改 mirror 服务的其他属性，例如服务运行的时间或镜像系统信息库的位置。

可能需要更改服务运行的时间，以更接近于匹配所镜像的发布者源的预期更新时间。要更改服务运行的时间，请修改 `config/crontab_period` 属性的值。

要更改镜像系统信息库的位置，请修改 `config/repository` 属性的值。如果更改镜像系统信息库的位置，请将系统信息库保存在共享位置。请参见“创建和使用本地 IPS 软件包系统信息库的最佳做法” [9]。缺省位置 `/var/share/pkg/repositories/solaris` 是未包含在任何 BE 中的共享位置。

3. 启用 mirror 服务。

   使用 `svcs mirror` 命令检查 `mirror` 服务的状态。

   - **该服务处于禁用状态，而您想使用此服务。**

     a. **如果更改了配置，请刷新服务实例。**

        如果更改了 `mirror` 服务的任何配置（如前面步骤的 `svccfg setprop` 命令中所示），请刷新该服务以将更改后的值提交到正在运行的快照。如果 `svcprop -p config mirror` 命令的输出未显示所需的值，请确保 `svccfg -s mirror:default listprop config` 命令的输出显示了所需的值。使用 `svcadm refresh mirror:default` 或 `svccfg -s mirror:default refresh` 将更改后的值提交到该服务的正在运行的快照。再次使用 `svcprop -p config mirror` 命令以确认已按照所需方式配置了该服务。

     b. **启用服务实例。**

        使用以下命令启用 mirror 服务：

        ```
        $ svcadm enable mirror:default
        ```

        使用 `svcs mirror` 命令确认 `mirror` 服务处于联机状态。该服务将按照在 `config/crontab_period` 属性中设置的时间运行。

   - **该服务处于联机状态，而您想立即运行此服务。**

     如果该服务处于联机状态，请刷新此服务以立即运行它。应该会显示 `pkg5srv` 用户正在运行 `svc-pkg-mirror` 方法和 `pkgrecv` 命令。

   - **该服务处于联机状态，而您不想使用此服务。**

     使用 `svcadm disable mirror` 命令禁用此服务。您可能希望只在一个系统上运行此服务以维护主系统信息库。而在其他系统上，您可能希望禁用此服务。

   - **该服务处于维护状态或者已降级。**

     使用 `svcs -xvL mirror` 命令获取更多信息以诊断并修复问题。

4. 验证系统信息库内容。

在 mirror 服务结束运行后，使用 pkgrepo 命令的 info、list 和 verify 子命令检查是否已正确复制或更新系统信息库。如果 pkgrepo verify 命令报告了错误，请尝试使用 pkgrepo fix 命令修复这些错误。有关更多信息，请参见 pkgrepo(1) 手册页。

检查 mirror 服务的 config/crontab_period 属性的值以查看服务运行的时间。当该服务正在运行时，svcs -p mirror 命令将服务状态显示为 online* 并显示此服务启动的进程。一直等到服务状态显示为 online 且不存在与该服务关联的进程，然后再验证系统信息库。

5. 生成新系统信息库快照。

```
$ zfs snapshot rpool/VARSHARE/pkg/repositories/solaris@sol-11_2_0
```

接下来的步骤 您可能不希望同时复制多个发布者中的内容。为此，无需在一个 config/publishers 属性中设置多个发布者，您可以创建 pkg/mirror 服务的多个实例。例如，可以将 config/publishers 属性设置为 solaris（对于 default 实例）、ha-cluster（对于新 pkg/mirror:ha-cluster 实例）和 solarisstudio（对于新 pkg/mirror:solarisstudio 实例）。同样，可以为每个实例设置不同的 config/crontab_period。可以将每个发布者中的内容存储在一个系统信息库中（如本过程中所示），也可以为每个 pkg/mirror 实例设置单独的 config/repository 值。

另请参见 有关 SMF 命令的更多信息，请参见《在 Oracle Solaris 11.2 中管理系统服务》。

♦ ♦ ♦ 第 3 章

3 提供对系统信息库的访问

本章介绍如何使客户机能够通过使用文件接口或 HTTP 接口来检索本地系统信息库中的软件包。一个系统信息库可以同时针对这两种访问方式进行相应的配置。

使用户能够使用文件接口检索软件包

本节描述如何从本地网络上的目录提供本地系统信息库软件包。

▼ 如何使用户能够使用文件接口检索软件包

1. **配置 NFS 共享。**

   要使客户机能够通过使用 NFS 访问本地系统信息库，请创建并发布 NFS 共享。

   ```
   $ zfs share -o share.nfs=on rpool/export/IPSpkgrepos%ipsrepo
   ```

   有关更多信息（例如可以设置的其他 `share.nfs` 属性），请参见 `zfs_share(1M)` 手册页。

2. **确认已发布共享。**

   请使用以下测试之一来确认是否已发布共享：

   - **在共享文件系统表中搜索系统信息库。**

     ```
     $ grep repo /etc/dfs/sharetab
     /export/IPSpkgrepos     ipsrepo nfs     sec=sys,rw
     ```

   - **确定是否可以从远程系统访问系统信息库。**

     ```
     $ dfshares solaris
     RESOURCE                                  SERVER ACCESS  TRANSPORT
        solaris:/export/IPSpkgrepos           solaris  -       -
     ```

3. **设置发布者源。**

要使客户机系统能够从本地文件系统信息库获取软件包，请设置发布者的源。

a. **确定发布者的名称。**

使用以下命令确定系统信息库中发布者的名称：

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

b. **检查此发布者源的适用性。**

要更新已安装的软件包，请安装依赖于已安装软件包的软件包，或者安装非全局区域，设置为发布者源的系统信息库必须至少包含在其中设置发布者的相应映像中所安装的相同软件。系统信息库也可以包含更旧或更新的软件，但必须包含该映像中安装的相同软件。

以下命令显示指定的系统信息库不是此映像的合适发布者源：

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                   IFO
entire                0.5.11-0.175.2.0.0.36.0   i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release
 entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

以下命令显示指定的系统信息库是此映像的合适发布者源：

```
$ pkgrepo list -Hs /export/IPSpkgrepos/Solaris entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

c. **设置发布者源。**

使用前面步骤中的系统信息库位置和发布者名称，运行以下命令设置发布者的源：

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

`-G '*'` 删除 `solaris` 发布者的所有现有源。

`-M '*'` 删除 `solaris` 发布者的所有现有镜像。

`-g` 将新建的本地系统信息库的 URI 添加为 `solaris` 发布者的新源。

有关配置发布者的更多信息，请参见《在 Oracle Solaris 11.2 中添加和更新软件 》中的“配置发布者”。

如果重置其他映像中的发布者源，请再次执行适用性测试：其他映像可能具有所安装软件的不同版本，且可能无法使用此系统信息库。如果重置其他系统上的映像中的发布者源，请对 `-g` 参数使用完整路径。

使用户能够使用 HTTP 接口检索软件包

本节描述如何使用软件包库 (depot) 服务器提供本地系统信息库软件包。

▼ 如何使用户能够使用 HTTP 接口检索软件包

软件包库 (depot) 服务器 `pkg.depotd` 提供对包含在软件包系统信息库中的数据的网络访问。`svc:/application/pkg/server` SMF 服务会调用 `pkg.depotd` 守护进程。为了使客户机能够通过使用 HTTP 访问本地系统信息库，以下过程将说明如何配置 `pkg/server` 服务。您可以配置服务的 `default` 实例。以下过程说明了如何创建和配置新实例。

1. **创建 depot 服务器实例。**

   使用 `add` 子命令添加 `pkg/server` 服务的新实例，名称为 `solaris`。

   ```
   $ svccfg -s pkg/server add solaris
   ```

2. **设置系统信息库的路径。**

   设置服务的此实例可找到系统信息库数据的路径。

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/inst_root=/export/IPSpkgrepos/Solaris
   ```

3. **（可选）设置端口号。**

   设置 depot 服务器实例应侦听传入软件包请求的端口号。缺省情况下，`pkg.depotd` 在端口 80 上侦听连接。要更改端口，请重置 `pkg/port` 属性。

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/port=81
   ```

4. **（可选）设置其他属性。**

   有关 `pkg/server` 属性的完整列表，请参见 `pkg.depotd(1M)` 手册页。

   要设置多个服务属性，请使用以下命令一次编辑所有属性。对于要更改的行，请注意删除开始处的注释标记 (#)。

   ```
   $ svccfg -s pkg/server:solaris editprop
   ```

5. **启动系统信息库服务。**

   重新启动软件包库 (depot) 服务器服务。

   ```
   $ svcadm refresh pkg/server:solaris
   $ svcadm enable pkg/server:solaris
   ```

6. **测试系统信息库服务器是否正在运行。**

要确定系统信息库服务器是否正在运行，请打开浏览器窗口定位到 `localhost` 位置。缺省情况下，`pkg.depotd` 在端口 80 上侦听连接。如果更改了端口，请打开浏览器窗口定位到 `localhost:`*port_number* 位置。

7. **设置发布者源。**

   要使客户机系统能够从本地文件系统信息库获取软件包，请设置发布者的源。

   a. **确定发布者的名称。**

      使用以下命令确定系统信息库中发布者的名称：

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

   b. **检查此发布者源的适用性。**

      要更新已安装的软件包，请安装依赖于已安装软件包的软件包，或者安装非全局区域，设置为发布者源的系统信息库必须至少包含在其中设置发布者的相应映像中所安装的相同软件。系统信息库也可以包含更旧或更新的软件，但必须包含该映像中安装的相同软件。

      以下命令显示指定的系统信息库不是此映像的合适发布者源：

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                    IFO
entire                0.5.11-0.175.2.0.0.36.0    i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release
 entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

      以下命令显示指定的系统信息库是此映像的合适发布者源：

```
$ pkgrepo list -Hs http://localhost:81/ entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

   c. **设置发布者源。**

      将发布者源设置为以下某一值：

      - `pkg/inst_root` 位置。

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

      - `pkg/port` 位置。

```
$ pkg set-publisher -G '*' -M '*' -g http://localhost:81/ solaris
```

      `-G '*'` 删除 `solaris` 发布者的所有现有源。

`-M '*'`	删除 `solaris` 发布者的所有现有镜像。
`-g`	将新建的本地系统信息库的 URI 添加为 `solaris` 发布者的新源。

有关配置发布者的更多信息，请参见《在 Oracle Solaris 11.2 中添加和更新软件》中的“配置发布者”。

如果重置其他映像中的发布者源，请再次执行适用性测试：其他映像可能具有所安装软件的不同版本，且可能无法使用此系统信息库。

另请参见

- “使用 Web 服务器访问方式提供多个系统信息库” [39]介绍了如何从多个位置或从单个位置提供多个系统信息库。
- “在一个域中有多个系统信息库” [47]介绍了如何在具有不同前缀的一个域名下运行多个系统信息库。
- “配置系统信息库的 HTTPS 访问” [49]介绍了如何配置安全系统信息库访问。

♦ ♦ ♦ 第 4 章

维护本地 IPS 软件包系统信息库

本章描述如何更新 IPS 系统信息库中的软件包、如何设置或更新系统信息库的属性以及如何将软件包从另一个源添加到系统信息库中。

更新本地系统信息库

本节中的过程说明了更新 IPS 软件包系统信息库的以下最佳做法：

- 使用相应发行版的所有支持更新使每个系统信息库保持最新状态。支持更新包含安全更新和其他重要修复。
  - 不要尝试从支持更新中选择特定修复进行应用。勿将支持更新中的一部分软件包添加到系统信息库。请将支持更新的所有内容添加到本地系统信息库。`pkgrecv` 命令的缺省行为是检索所有软件包的所有版本。
  - 不跳过支持更新。在每个系统信息库中累积应用所有适用的支持更新。

  用户可以通过指定要安装的 `entire` incorporation 软件包的版本来更新到低于系统信息库中的最新版本的版本。请参见《在 Oracle Solaris 11.2 中添加和更新软件 》中的第 4 章 “更新或升级 Oracle Solaris 映像”。
- 更新系统信息库的副本。此做法有助于确保系统不会在更新系统信息库时访问系统信息库。在更新系统信息库之前创建系统信息库的快照，克隆该快照，执行更新，并将原始系统信息库替换为更新后的克隆。

如果要维护具有相同内容的软件包系统信息库的多个副本，请使用以下过程更新这些相同系统信息库中的一个。有关从此主系统信息库更新其他系统信息库的过程，请参见“维护多个相同的本地系统信息库” [32]。

▼ 如何更新本地 IPS 软件包系统信息库

注 - 如果使用 `svc:/application/pkg/mirror` SMF 服务定期更新系统信息库，则无需执行此过程。有关使用 `mirror` 服务的说明，请参见如何从 Internet 自动复制系统信息库 [19]。

1. 创建软件包系统信息库的 ZFS 快照。
   确保具有要更新的系统信息库的当前快照。

```
$ zfs list -t all -r rpool/export/IPSpkgrepos/Solaris
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
```

   如果已具有该系统信息库的快照，请使用 `zfs diff` 命令检查该快照是否与系统信息库数据集相同。

```
$ zfs diff rpool/export/IPSpkgrepos/Solaris@initial
$
```

   如果 `zfs diff` 命令未生成输出，则该快照与其父数据集相同，您可以将该快照用于更新。

   如果 `zfs diff` 命令生成了输出，或者如果您没有系统信息库的快照，则按照如何从 Internet 显式复制系统信息库 [18]的步骤 6中所示创建新快照。然后将此新快照用于更新。

2. 创建软件包系统信息库的 ZFS 克隆。
   克隆系统信息库快照以创建可以更新的系统信息库的副本。

```
$ zfs clone rpool/export/IPSpkgrepos/Solaris@initial rpool/export/IPSpkgrepos/Solaris_tmp
$ zfs list -r rpool/export/IPSpkgrepos/Solaris/
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
rpool/export/IPSpkgrepos/Solaris_tmp          76K  78.4G  17.6G  /export/IPSpkgrepos/
Solaris_tmp
```

3. 更新软件包系统信息库的 ZFS 克隆。
   正如可从文件或在 HTTP 位置中创建原始系统信息库一样，您可以从文件或在 HTTP 位置中更新系统信息库。

   - 从 **zip** 文件更新。
     请参见例 2-2 “从 zip 文件添加现有系统信息库”。如果指定目标已包含软件包系统信息库，则 `zip` 文件的内容将添加到现有系统信息库的内容。

   - 从 ISO 文件更新。

     a. 挂载 ISO 映像。

```
$ mount -F hsfs ./sol-11_2-incr-repo.iso /mnt
```

     b. 将 ISO 文件内容复制到系统信息库克隆。
        使用 `rsync` 或 `tar`，如如何从 iso 文件复制系统信息库 [17]中所示。

```
$ rsync -aP /mnt/repo/ /export/IPSpkgrepos/Solaris_tmp
```

c. 取消挂载 ISO 映像。

- 从系统信息库更新。
  将其他系统信息库中的内容复制到系统信息库克隆。如果要从安全站点复制，请确保已安装必需的 SSL 证书和密钥，并指定必需的证书和密钥选项。

  ```
  $ pkgrecv -s https://pkg.oracle.com/solaris/support \
  -d /export/IPSpkgrepos/Solaris_tmp \
  --key /path-to-ssl_key --cert /path-to-ssl_cert '*'
  ```

  有关 pkgrecv 命令的更多信息，请参见 pkgrecv(1) 手册页。因为仅对更改过的软件包进行更新，因此更新系统信息库的时间可能比填充原始系统信息库的时间少很多。请参见“复制系统信息库的性能注意事项” [13]中的性能提示。
  如果 pkgrecv 操作中断，请按照“继续执行中断的软件包接收” [32]中的说明进行操作。

4. 将工作系统信息库替换为更新后的克隆。

   ```
   $ svcadm disable -st pkg/server:solaris
   $ zfs promote rpool/export/IPSpkgrepos/Solaris_tmp
   $ zfs rename rpool/export/IPSpkgrepos/Solaris rpool/export/IPSpkgrepos/Solaris_old
   $ zfs rename rpool/export/IPSpkgrepos/Solaris_tmp rpool/export/IPSpkgrepos/Solaris
   ```

   有关 svcadm 命令的更多信息，请参见 svcadm(1M) 手册页。

5. 验证更新后的系统信息库。
   使用 pkgrepo verify 命令验证更新后的系统信息库。有关 pkgrepo verify 和 pkgrepo fix 命令的更多信息，请参见 pkgrepo(1) 手册页。

6. 将新软件包编入目录并更新搜索索引。
   将在新更新的系统信息库中找到的所有新软件包编入目录，并更新所有搜索索引。

   ```
   $ pkgrepo refresh -s rpool/export/IPSpkgrepos/Solaris
   ```

7. 创建软件包系统信息库的 ZFS 快照（即新更新的克隆）。

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@S11U2SRU1
   ```

8. 重新启动 SMF 服务。
   如果要通过 HTTP 接口提供系统信息库，请重新启动 SMF 服务。重新启动该服务时务必指定相应的服务实例。

   ```
   $ svcadm restart pkg/server:solaris
   ```

9. 删除旧的系统信息库。

如果更新后的系统信息库正常工作，让您满意，则可以删除旧的系统信息库。

```
$ zfs destroy rpool/export/IPSpkgrepos/Solaris_old
```

继续执行中断的软件包接收

如果 pkgrecv 操作中断，请使用 -c 选项检索已下载的内容并继续下载内容。传输中断时，信息性消息中将显示 *cache_dir* 的值，如以下示例中所示：

```
PROCESS                    ITEMS       GET (MB)        SEND (MB)
...
pkgrecv: http protocol error: code: 503 reason: Service Unavailable
URL: 'https://pkg.oracle.com/solaris/support/file/file_hash

pkgrecv: Cached files were preserved in the following directory:
        /var/tmp/pkgrecv-fOGaIg
Use pkgrecv -c to resume the interrupted download.
$ pkgrecv -c /var/tmp/pkgrecv-fOGaIg \
-s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris_tmp \
--key /path/to/ssl_key --cert /path/to/ssl_cert '*'
Processing packages for publisher solaris ...
Retrieving and evaluating 156 package(s)...
```

维护多个相同的本地系统信息库

您可能希望维护具有相同内容的软件包系统信息库的多个副本以实现以下目标：

- 通过在不同节点上维护副本来提高系统信息库的可用性。
- 提高系统信息库的访问性能（如果具有多个用户或用户分布在广泛地域）。

使用如何更新本地 IPS 软件包系统信息库 [29]过程更新软件包系统信息库之一。然后使用如何克隆本地 IPS 软件包系统信息库 [33]过程从首先更新的系统信息库更新其他相同系统信息库。这两个过程非常相似，一个重要区别是使用 pkgrecv 命令的方式。克隆过程中显示的 pkgrecv 操作完全复制源系统信息库文件，具有以下效果：

- 克隆系统信息库的目录的时间戳与源系统信息库的目录的时间戳完全相同。如果已对系统信息库进行负载平衡，则所有系统信息库中的目录应完全相同，以避免在负载平衡器将客户机从一个节点切换到另一个节点时出现问题。有关负载平衡的信息，请参见“配置负载平衡” [47]。
- 将从目标系统信息库中删除存在于目标系统信息库中但不在源系统信息库中的软件包。勿将稀疏系统信息库用作克隆操作的源，除非目标是仅创建该稀疏系统信息库的精确副本。

▼ 如何克隆本地 IPS 软件包系统信息库

有关这些步骤的详细信息，请参见如何更新本地 IPS 软件包系统信息库 [29]。

1. 复制目标系统信息库。
   确保具有目标系统信息库的当前快照。创建此快照的 ZFS 克隆。
2. 更新目标系统信息库的副本。
   使用 `pkgrecv` 命令将以前更新的本地软件包系统信息库克隆到目标系统信息库的副本。有关 `pkgrecv` 克隆操作的更多信息，请参见 `pkgrecv(1)` 手册页。

   ```
   $ pkgrecv -s /net/host1/export/IPSpkgrepos/Solaris \
   -d /net/host2/export/IPSpkgrepos/Solaris_tmp --clone
   ```

3. 将工作目标系统信息库替换为更新后的克隆。
4. 验证更新后的系统信息库。
   使用 `pkgrepo verify` 命令验证更新后的目标系统信息库。
5. 生成新更新系统信息库快照。
6. 重新启动 SMF 服务。
   如果要通过 HTTP 接口提供系统信息库，请重新启动 SMF 服务。重新启动该服务时务必指定相应的服务实例。
7. 删除旧的系统信息库。
   如果更新后的系统信息库正常工作，让您满意，则删除旧的系统信息库。

另请参见 如果要通过 HTTP 接口提供系统信息库，请参见以下相关文档：

- “使用 Web 服务器访问方式提供多个系统信息库” [39]介绍了如何使用在不同端口上运行的多个 `pkg.depotd` 守护进程来提供多个系统信息库。
- “在一个域中有多个系统信息库” [47]介绍了如何在具有不同前缀的一个域名下运行多个系统信息库。

检查和设置系统信息库属性

本节描述如何显示有关 IPS 系统信息库的信息以及如何更改系统信息库属性值。

查看适用于整个系统信息库的属性

以下命令显示本地系统信息库识别的软件包发布者的列表。STATUS 列指示当前是否正在处理此发布者的软件包数据。

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4506     online           2013-07-11T23:32:46.379726Z
```

以下命令显示适用于整个系统信息库的属性信息。有关系统信息库属性的完整列表及其说明（包括属性值的规范），请参见 pkgrepo(1) 手册页。

```
$ pkgrepo get -s /export/IPSpkgrepos/Solaris
SECTION    PROPERTY                    VALUE
publisher  prefix                      solaris
repository check-certificate-revocation False
repository signature-required-names    ()
repository trust-anchor-directory      /etc/certs/CA/
repository version                     4
```

`publisher/prefix`

缺省发布者的名称。虽然系统信息库可以包含多个发布者的软件包，但只能将其中一个发布者设置为缺省发布者。此缺省发布者名称用于以下目的：

- 当未在 `pkg` 命令的软件包 FMRI 中指定发布者时标识软件包
- 将软件包发布到系统信息库（使用 `pkgsend`(1) 命令）且未在软件包清单中指定发布者时，为软件包指定发布者

`repository/check-certificate-revocation`

用于检查证书的标志。如果设置为 `True`，`pkgrepo verify` 命令将尝试确定证书自发布以来是否已撤销。此值必须与《在 Oracle Solaris 11.2 中添加和更新软件 》中的“其他映像属性”中和 pkg(1) 手册页中介绍的 `check-certificate-revocation` 映像属性的值匹配。

`repository/signature-required-names`

在验证软件包签名时必须视为证书通用名称的名称列表。此列表由 `pkgrepo verify` 命令使用。此值必须与《在 Oracle Solaris 11.2 中添加和更新软件 》中的“签名的软件包的映像属性”中和 pkg(1) 手册页中介绍的 `signature-required-names` 映像属性的值匹配。

`repository/trust-anchor-directory`

目录的绝对路径名称，包含此系统信息库中的软件包的信任锚。缺省设置为 `/etc/certs/CA/`。此值必须与《在 Oracle Solaris 11.2 中添加和更新软件 》中的“其他映像属性”中和 pkg(1) 手册页中介绍的 `trust-anchor-directory` 映像属性的值匹配。

`repository/version`

系统信息库的格式版本。不能使用“修改系统信息库属性值” [36]中显示的 `pkgrepo set` 命令设置此值。可以使用 `pkgrepo create` 命令设置此值。缺省情况下创建版本 4 系统信息库。版本 4 系统信息库支持存储多个发布者的软件包。

查看系统信息库发布者属性

以下命令显示有关本地系统信息库中 `solaris` 发布者的属性信息。括号表明该值可以是值列表。

```
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris
PUBLISHER SECTION    PROPERTY         VALUE
solaris   publisher  alias
solaris   publisher  prefix           solaris
solaris   repository collection-type  core
solaris   repository description      ""
solaris   repository legal-uris       ()
solaris   repository mirrors          ()
solaris   repository name             ""
solaris   repository origins          ()
solaris   repository refresh-seconds  ""
solaris   repository registration-uri ""
solaris   repository related-uris     ()
```

`publisher/prefix`

在 `-p` 选项中指定的发布者的名称。如果未指定 `-p` 选项，则此值为该系统信息库的缺省发布者的名称，如上一节中所述。

`repository/collection-type`

此系统信息库中的软件包的类型。如果值为 `core`，则此系统信息库包含其中的软件包声明的所有依赖项。如果值为 `supplemental`，则此系统信息库不包含其中的软件包声明的所有依赖项。

`repository/description`

此系统信息库的目的和内容。如果可以通过 HTTP 接口访问此系统信息库，则此值显示在主页顶部附近的 "About"（关于）部分中。

`repository/legal-uris`

文档位置列表，提供有关系统信息库的法律信息。

`repository/mirrors`

系统信息库位置列表，这些系统信息库包含与此系统信息库相同的软件包内容。

`repository/name`

此系统信息库的名称。如果可以通过 HTTP 接口访问此系统信息库，则此值显示在主页顶部以及窗口标题中。

`repository/origins`

系统信息库位置列表，这些系统信息库包含与此系统信息库相同的软件包内容和元数据。

`repository/refresh-seconds`

客户机在两次检查此系统信息库中的更新软件包数据之间等待的秒数。

`repository/registration-uri`

资源位置，必须使用该资源才能获取用于访问此系统信息库的证书。

`repository/related-uris`

系统信息库位置列表，这些系统信息库包含其他可能有用的软件包。

以下命令显示有关 `pkg.oracle.com` 系统信息库中的指定 *section/property* 的信息。

```
$ pkgrepo get -p solaris -s http://pkg.oracle.com/solaris/release \
repository/name repository/description
PUBLISHER SECTION    PROPERTY    VALUE
solaris   repository description This\ repository\ serves\ the\ Oracle\ Solaris\ 11\ Package\
 repository.
solaris   repository name        Oracle\ Solaris\ 11\ Package\ Repository
```

修改系统信息库属性值

在“查看系统信息库发布者属性” [35]中显示，没有为本地系统信息库中的 `solaris` 发布者设置系统信息库名称和说明属性值。如果可以通过 HTTP 接口访问此系统信息库，并且您使用浏览器查看此系统信息库的内容，则将看到缺省名称但没有其说明。设置这些值之后，发布者的 `repository/name` 值会作为页标题显示在页顶部附近，发布者的 `repository/description` 值会显示在名称正下方的 "About"（关于）部分中。设置这些值时必须使用 `-p` 选项至少指定一个发布者。如果此系统信息库包含多个发布者的内容，则可以为每个发布者设置不同的值，或者可以指定 `-p all`。

```
$ pkgrepo set -p solaris -s /export/IPSpkgrepos/Solaris \
repository/description="Local copy of the Oracle Solaris 11 repository." \
repository/name="Oracle Solaris 11"
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris repository/name repository/
description
PUBLISHER SECTION    PROPERTY         VALUE
solaris   repository description      Local\ copy\ of\ the\ Oracle\ Solaris\ 11\ repository.
solaris   repository name             Oracle\ Solaris\ 11
```

定制本地系统信息库

可以使用 pkgrecv 命令将软件包及其发布者数据添加到系统信息库。可以使用 pkgrepo 命令从系统信息库中删除软件包和发布者。

向系统信息库添加软件包

可以向系统信息库添加发布者。例如，可以在一个系统信息库中维护 solaris、ha-cluster 和 solarisstudio 软件包。

如果添加定制软件包，请用定制发布者名称发布这些软件包。不要作为现有发布者（例如 solaris）发布定制软件包。如果发布未指定发布者的软件包，这些软件包将添加给系统信息库的缺省发布者。请将具有正确缺省发布者的定制软件包发布到测试系统信息库。然后使用 pkgrecv 命令将这些软件包及其发布者信息添加到生产系统信息库。有关说明，请参见《在 Oracle Solaris 11.2 中使用映像包管理系统打包和交付软件 》中的“发布软件包”。

在以下示例中，ISVproducts.p5p 软件包归档文件中的 isvpub 发布者数据以及所有软件包将添加到本地系统信息库。软件包归档文件是包含发布者信息以及该发布者提供的一个或多个软件包的文件。请参见《在 Oracle Solaris 11.2 中使用映像包管理系统打包和交付软件 》中的“交付为软件包归档文件”。大多数 pkgrepo 操作不可用于软件包归档文件。软件包归档文件包含软件包但不包含系统信息库配置。不过，pkgrepo list 和 pkgrepo contents 命令可处理软件包归档文件。pkgrepo contents 命令将在“检查系统信息库中的软件包” [38]中讨论。

在 pkgrepo list 输出中将显示发布者，因为其不是此映像中按搜索顺序排在第一位的发布者。

```
$ pkgrepo -s /tmp/ISVproducts.p5p list
PUBLISHER NAME                                            O VERSION
isvpub    isvtool                                           1.1,5.11:20131120T021902Z
isvpub    isvtool                                           1.0,5.11:20131120T010105Z
```

以下 pkgrecv 命令从源系统信息库中检索所有软件包。如果列出要检索的软件包的名称，或者指定 '*' 以外的模式，则应指定 -r 选项以确保检索所有必需的依赖性软件包。

```
$ pkgrecv -s /tmp/ISVproducts.p5p -d /export/IPSpkgrepos/Solaris '*'
Processing packages for publisher isvpub ...
Retrieving and evaluating 2 package(s)...
PROCESS        ITEMS      GET (MB)      SEND (MB)
Completed        2/2       0.0/0.0          0.0/0
```

更改系统信息库的内容后，刷新系统信息库并重新启动为此系统信息库配置的所有软件包库 (depot) 服务器服务实例。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris refresh -p isvpub
Initiating repository refresh.
$ svcadm refresh pkg/server:solaris
$ svcadm restart pkg/server:solaris
```

以下 pkgrepo info 命令显示一个软件包，因为检索的两个软件包是同一个软件包的不同版本。pkgrepo list 命令则显示两个软件包。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-01-02T19:19:06.983979Z
isvpub    1        online           2014-03-20T23:24:37.196773Z
$ pkgrepo -s /export/IPSpkgrepos/Solaris list -p isvpub
PUBLISHER NAME                                    O VERSION
isvpub    isvtool                                   1.1,5.11:20131120T021902Z
isvpub    isvtool                                   1.0,5.11:20131120T010105Z
```

使用 pkg set-publisher 命令添加 isvpub 发布者的新系统信息库位置。

如果可以通过 HTTP 接口访问此系统信息库，并且您使用浏览器查看此系统信息库的内容，则可以通过指定在该位置的发布者来查看此新软件包。例如，可以指定 http://localhost:81/isvpub/。

检查系统信息库中的软件包

除了“向系统信息库添加软件包” [37]中显示的 pkgrepo info 和 pkgrepo list 命令，还可以使用 pkgrepo contents 命令检查系统信息库中的软件包的内容。

对于单个软件包，pkgrepo contents 命令的输出与 pkg contents -m 命令的输出相同。pkgrepo contents 命令显示的输出是针对指定系统信息库中的每个匹配软件包，而 pkg contents 命令显示的输出仅针对可安装在此映像中的匹配软件包的各个版本。如果指定 -t 选项，则 pkgrepo contents 命令仅显示指定的操作。

以下示例不需要指定软件包的版本，因为指定的系统信息库中仅存在此软件包的一个版本。此软件包中包含 depend 操作以提供安装和运行 Oracle Database 12 所需的一系列 Oracle Solaris 软件包。

```
$ pkgrepo -s http://pkg.oracle.com/solaris/release/ \
contents -t depend oracle-rdbms-server-12cR1-preinstall
depend fmri=x11/library/libxi type=group
depend fmri=x11/library/libxtst type=group
depend fmri=x11/session/xauth type=group
depend fmri=compress/unzip type=require
depend fmri=developer/assembler type=require
depend fmri=developer/build/make type=require
```

从系统信息库中删除软件包

勿删除 Oracle 发布者提供的软件包。《在 Oracle Solaris 11.2 中添加和更新软件 》中介绍了仅安装所需软件包而避免安装不需要的软件包的方法。

可以使用 `pkgrepo remove` 命令删除不是由 Oracle 发布者提供的软件包。可以使用 `pkgrepo remove-publisher` 命令删除发布者以及该发布者提供的所有软件包。有关详细信息，请参见 `pkgrepo(1)` 手册页。应对系统信息库的副本执行这些操作，如如何更新本地 IPS 软件包系统信息库 [29]中所述。

使用 Web 服务器访问方式提供多个系统信息库

本节中的过程说明如何扩展“使用户能够使用 HTTP 接口检索软件包” [25]中提供的信息，从而支持提供多个系统信息库。

以下是使用 HTTP 访问方式提供多个 IPS 软件包系统信息库的两种不同方法。对于两种方法，首先都是使用唯一系统信息库路径创建 `pkg/server` 服务的附加实例。

- 多个位置。用户通过查看各个单独位置的页来访问每个系统信息库。
- 单个位置。用户从一个位置访问所有系统信息库。

除了提供对多个系统信息库的访问之外，请记住，单个系统信息库还可以提供来自多个发布者的软件包，如“向系统信息库添加软件包” [37]中所示。

▼ 如何从单独位置提供多个系统信息库

在本示例中，除了 `Solaris` 系统信息库以外，还存在 `SolarisStudio` 系统信息库。可以使用端口 81 从 `http://localhost/` 访问 `Solaris` 系统信息库，如 `pkg/server` 服务的 `solaris` 实例中所指定的那样。请参见“使用户能够使用 HTTP 接口检索软件包” [25]。

1. **创建新 depot 服务器实例。**

   使用 `svccfg` 命令的 `add` 子命令添加 `pkg/server` 服务的新实例。

   ```
   $ svccfg -s pkg/server add studio
   ```

2. **检查是否已添加新实例。**

   ```
   $ svcs pkg/server
   STATE  STIME    FMRI
   online 14:54:16 svc:/application/pkg/server:default
   ```

```
online 14:54:20 svc:/application/pkg/server:studio
online 14:54:20 svc:/application/pkg/server:solaris
```

3. 设置系统信息库的路径。

   设置服务的此实例可找到系统信息库数据的路径。

   ```
   $ svccfg -s pkg/server:studio setprop pkg/inst_root=/export/IPSpkgrepos/SolarisStudio
   ```

4. （可选）设置新实例的端口号。

   ```
   $ svccfg -s pkg/server:studio setprop pkg/port=82
   ```

5. （可选）设置 Apache 代理库。

   有关设置 pkg/proxy_base 的示例，请参见“配置带有前缀的简单代理” [46]。

6. 设置系统信息库名称和说明。

   确保按照“修改系统信息库属性值” [36]中所示设置系统信息库名称和说明。

7. 启动系统信息库服务。

   重新启动软件包库 (depot) 服务器服务。

   ```
   $ svcadm refresh pkg/server:studio
   $ svcadm enable pkg/server:studio
   ```

8. 测试系统信息库服务器是否正在运行。

   打开浏览器窗口定位到 http://localhost:82/ 位置。

   如果未设置端口号，缺省值为 80。查看 http://localhost:80/ 或 http://localhost/ 上的系统信息库。

   如果端口号也由其他 pkg/server 实例使用，请将发布者名称附加到位置上以查看新软件包。例如，查看 http://localhost:81/solarisstudio/ 上的系统信息库。

9. 设置发布者源。

   将发布者源设置为以下某一值：

   - pkg/inst_root 位置。

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - pkg/port 位置。

     ```
     $ pkg set-publisher -G '*' -M '*' -g http://localhost:82/ solarisstudio
     ```

另请参见 有关在具有不同前缀（例如 http://pkg.example.com/solaris 和 http://pkg.example.com/studio）的一个域名下运行多个系统信息库的信息，请参见“在一个域中有多个系统信息库” [47]。

▼ 如何从单个位置提供多个系统信息库

此过程中的许多步骤与上一过程中的步骤相同。有关详细信息，请参见上一过程。

1. **创建新 depot 服务器实例。**

2. **设置系统信息库的路径。**
   由特定 pkg/depot 实例管理的每个 pkg/server 实例都必须具有唯一的 pkg/inst_root 值。

3. **检查新实例的 readonly 属性。**
   pkg/readonly 属性的缺省值为 true。如果此值已更改，请将该值重置为 true。

   ```
   $ svcprop -p pkg/readonly pkg/server:studio
   true
   ```

4. **设置新实例的 standalone 属性。**
   缺省情况下，pkg/standalone 属性的值为 true。pkg/depot 服务实例可以从相同位置提供其 pkg/standalone 属性已设置为 false 的所有 pkg/server 实例。

   ```
   $ svccfg -s pkg/server:studio
   svc:/application/pkg/server:studio> setprop pkg/standalone=false
   svc:/application/pkg/server:studio> refresh
   svc:/application/pkg/server:studio> select solaris
   svc:/application/pkg/server:solaris> setprop pkg/standalone=false
   svc:/application/pkg/server:solaris> refresh
   svc:/application/pkg/server:solaris> exit
   $
   ```

   确保 pkg/inst_root 属性的值对于其 pkg/standalone 属性设置为 false 的每个 pkg/server 实例都是唯一的。

5. **（可选）设置 pkg/depot 实例的端口号。**
   缺省情况下，svc:/application/pkg/depot:default 服务的端口号为 80。此端口号可以与将由此 pkg/depot 实例管理的任何 pkg/server 实例的端口号相同。要更改端口号，请设置 pkg/depot:default 的 config/port 属性。

6. **重新启动 pkg/depot 实例。**

   ```
   $ svcadm refresh pkg/depot:default
   $ svcadm restart pkg/depot:default
   ```

7. **测试系统信息库服务器是否正在运行。**
   当用户打开 http://localhost:80/ 位置时，将看到列有 solaris 发布者的 http://localhost/solaris 系统信息库，并看到列有 solarisstudio 发布者的 http://localhost/studio 系统信息库。

如果一个系统信息库提供多个发布者的软件包，将列出所有发布者。例如，用户可能会看到列出发布者为 solaris 和 isvpub 的 http://localhost/solaris 系统信息库。

8. **设置发布者源。**

   将发布者源设置为以下某一值：

   - 唯一 pkg/inst_root 位置。

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - 由 config/port 的值和 pkg/server 实例名称定义的位置。

     ```
     $ pkg set-publisher -G '*' -M '*' -g http://localhost:80/studio/ solarisstudio
     ```

接下来的步骤 如果按照“更新本地系统信息库” [29]和“定制本地系统信息库” [37]中的介绍更改由 pkg/depot 实例管理的系统信息库的内容，请执行以下两个步骤：

- 在系统信息库上运行 pkgrepo refresh。
- 在 pkg/depot 实例上运行 svcadm restart。

可以创建 pkg/depot 服务（其每个实例都托管一个或多个系统信息库）的附加实例。

要生成独立配置而不是配置 pkg/server 和 pkg/depot 服务实例，请参见 pkg.depot-config(1M) 手册页。

♦ ♦ ♦ 第 5 章

在 Web 服务器后运行 Depot 服务器

在 Apache Web 服务器实例后运行 depot 服务器具有以下优点：

- 允许在一个域名下托管多个系统信息库。使用 pkg(5) depot 服务器，您可以轻松地将本地网络中或 Internet 上的系统信息库变为可供访问。但是，depot 服务器不支持在一个域名或复杂前缀下提供多个系统信息库。要在一个域名下托管多个系统信息库，请在 Web 代理后运行 depot 服务器。
- 提高性能和可用性。在 Web 代理后运行 depot 服务器还可以提高服务器的性能和可用性，即在多个 depot 上启用负载平衡和启用内容高速缓存。
- 可以提供安全的系统信息库服务器。在支持客户机证书且启用安全套接字层 (Secure Sockets Layer, SSL) 协议的 Apache 实例后运行 depot 服务器。

Depot 服务器的 Apache 配置

本章中的示例将 Apache Web 服务器用作代理软件。通过启用 svc:/network/http:apache22 服务激活 Apache Web 服务器。有关其他信息，请参见 Apache HTTP 服务器 2.2 文档。

这些示例中所示的原则可应用于任何代理服务器软件。

Oracle Solaris 11.2 OS 在 web/server/apache-22 软件包中包含有 Apache Web 服务器，该软件包提供了基本 httpd.conf 文件，位于 /etc/apache2/2.2 中。通常，可以使用以下命令查找 httpd.conf 文件：

```
$ pkg search -Hl -o path ':file:path:*httpd.conf'
etc/apache2/2.2/httpd.conf
etc/apache2/2.2/original/httpd.conf
```

必需的 Apache 配置设置

如果在 Apache Web 服务器实例后运行软件包库 (depot) 服务器，请在 httpd.conf 文件中包含以下设置，以便不对已编码的正斜杠进行解码：

```
AllowEncodedSlashes NoDecode
```

由于正斜杠用于表示分层次的软件包名称，因此软件包名称可以包含 URL 编码的正斜杠。例如，软件包名称 `pkg://solaris/developer/build/make` 对于 Web 服务器变为 `http://pkg.oracle.com/solaris/release/manifest/0/developer%2Fbuild%2Fmake`。为防止这些正斜杠被解释为目录分隔符，请指示 Apache 不要对 `%2F` 编码斜杠进行解码。

省略此设置可能会导致 `404 Not Found` 错误，并且可能会对搜索功能产生非常严重的负面影响。

建议的常规 Apache 配置设置

以下设置会影响性能和安全。

减小元数据的线上传输大小。

HTTP 客户机可以通知服务器它们接受在 HTTP 请求中传递压缩数据。启用 Apache DEFLATE 过滤器可以大幅度减少元数据（如目录和清单）的线上传输大小。元数据（如目录和清单）通常可以压缩 90%。

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain
```

允许更多通过管道传输的请求。

增大 `MaxKeepAliveRequests` 值可允许客户机在不关闭连接的情况下发出更多的通过管道传输的请求。

```
MaxKeepAliveRequests 10000
```

设置响应的最长等待时间。

该代理超时值设置 Apache 等待后端 depot 响应的时间长度。对于大多数操作而言，30 秒已足够。返回大量结果的搜索所花费的时间会长得多。您可能需要增大超时值来满足此类搜索。

```
ProxyTimeout 30
```

禁用前向代理。

请确保前向代理已禁用。

```
ProxyRequests Off
```

为 Depot 服务器配置高速缓存

将 depot 服务器设置在高速缓存代理后所需的配置极少。除了目录属性文件（参见“目录属性文件的高速缓存注意事项” [45]）和系统信息库搜索结果（参见“搜索的高速缓

存注意事项" [46]），提供的所有文件都是唯一的，因此可以安全地无限期高速缓存（如有必要）。另外，所有 depot 响应均包含相应的 HTTP 标头，从而确保高速缓存中的文件不会因错误而变为过时文件。

有关将 Apache 配置为高速缓存代理的更多信息，请参见 Apache Caching Guide（高速缓存指南）。

使用 `CacheRoot` 指令指定用于包含缓存文件的目录。请确保 Apache 进程可写入指定的目录。即使 Apache 无法写入此目录，也不会显式输出错误消息。

```
CacheRoot /tank/proxycache
```

Apache 允许为特定目录启用高速缓存。您可能希望系统信息库服务器将服务器上的所有内容都放入高速缓存，如以下指令中所示。

```
CacheEnable disk /
```

使用 `CacheMaxFileSize` 指令设置要缓存的文件的最大大小。Apache 的缺省值 1 MB 对于大多数系统信息库来说可能太小。以下指令将缓存文件的最大大小设置为 1 GB。

```
CacheMaxFileSize 1000000000
```

调整盘上高速缓存的目录结构，以便获得与底层文件系统相适应的最佳性能。在 ZFS 数据集中，多个目录级别对性能的影响超过一个目录中有大量文件的影响。因此，请将目录级别配置为一，并在每个目录中包含大量文件。使用 `CacheDirLevels` 和 `CacheDirLength` 指令控制目录结构。将 `CacheDirLevels` 设置为 1。将 `CacheDirLength` 设置为一个合适的值，该值可以使目录数与每个目录中的文件数达到良好的平衡。下面设置的值 2 将生成 4096 个目录。有关更多信息，请参见 Apache Disk-based Caching（基于磁盘的高速缓存）文档。

```
CacheDirLevels 1
CacheDirLength 2
```

目录属性文件的高速缓存注意事项

系统信息库目录属性文件 (`catalog.attrs`) 包含系统信息库目录的当前状态。此文件可能大到有必要放入高速缓存。但是，如果后端系统信息库的目录发生更改，此文件将过时。您可以使用以下两种方法之一来解决此问题。

- 不缓存此文件。如果系统信息库服务器在高带宽环境（在此环境中无需考虑流量增加）中运行，则此解决方法最有效。以下截取了 `httpd.conf` 文件的部分内容，其中显示了如何指定不缓存 `catalog.attrs` 文件：

  ```
  <LocationMatch ".*/catalog.attrs">
          Header set Cache-Control no-cache
  </LocationMatch>
  ```

- 每次更新后端系统信息库的目录时在高速缓存中删改此文件。

搜索的高速缓存注意事项

搜索软件包系统信息库时会根据请求生成定制响应。因此，搜索结果不是很适合放入高速缓存。depot 服务器会设置相应的 HTTP 标头，以确保搜索结果在高速缓存中不会过时。但是，预计通过高速缓存节省的带宽很少。以下截取了 `httpd.conf` 文件的部分内容，其中显示了如何指定不缓存搜索结果。

```
<LocationMatch ".*/search/\d/.*">
        Header set Cache-Control no-cache
</LocationMatch>
```

配置带有前缀的简单代理

本示例显示非负载平衡 depot 服务器的基本配置。本示例将 `http://pkg.example.com/myrepo` 连接到 `internal.example.com:10000`。

如果需要设置此示例中未介绍的其他属性，其说明请参见“使用 Web 服务器访问方式提供多个系统信息库” [39]。

使用 `pkg/proxy_base` 设置配置 depot 服务器，在该设置中指定用于访问 depot 服务器的 URL。使用以下命令设置 `pkg/proxy_base`：

```
$ svccfg -s pkg/server add repo
$ svccfg -s pkg/server:repo setprop pkg/proxy_base = astring: http://pkg.example.com/myrepo
$ svcadm refresh pkg/server:repo
$ svcadm enable pkg/server:repo
```

`pkg`(5) 客户机在执行网络操作时会打开 20 个到 depot 服务器的并行连接。请确保在任何给定时间，depot 线程的数目与预期的服务器连接数匹配。使用以下命令设置每个 depot 服务器的线程数目：

```
$ svccfg -s pkg/server:repo setprop pkg/threads = 200
$ svcadm refresh pkg/server:repo
$ svcadm restart pkg/server:repo
```

使用 `nocanon` 隐藏 URL 的标准化。此设置对于搜索正常工作至关重要。另外，将后端连接的数目限制为 depot 服务器提供的线程的数目。以下截取了 `httpd.conf` 文件的部分内容，其中显示了如何代理一个 depot 服务器：

```
Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ http://internal.example.com:10000/ nocanon max=200
```

有关 Oracle Solaris SSL 内核代理以及使用 SSL 加密和加速 Web 服务器通信的信息，请参见《在 Oracle Solaris 11.2 中确保网络安全 》中的第 3 章 “Web 服务器和安全套接字层协议”。

在一个域中有多个系统信息库

在代理后运行 depot 服务器的最重要原因是可以在具有不同前缀的一个域名下轻松运行多个系统信息库。“配置带有前缀的简单代理” [46]中的示例可以轻松扩展为支持多个系统信息库。

在本示例中，一个域名的三个不同前缀连接到三个不同的软件包系统信息库：

- `http://pkg.example.com/repo_one` 连接到 `internal.example.com:10000`
- `http://pkg.example.com/repo_two` 连接到 `internal.example.com:20000`
- `http://pkg.example.com/xyz/repo_three` 连接到 `internal.example.com:30000`

`pkg`(5) depot 服务器采用 SMF 管理的服务。因此，要在同一个主机上运行多个 depot 服务器，只需要创建一个新的服务实例：

```
$ svccfg -s pkg/server add repo1
$ svccfg -s pkg/server:repo1 setprop pkg/property=value
$ ...
```

和前一个示例一样，每个 depot 服务器运行 200 个线程。

```
Redirect /repo_one http://pkg.example.com/repo_one/
ProxyPass /repo_one/ http://internal.example.com:10000/ nocanon max=200

Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200

Redirect /xyz/repo_three http://pkg.example.com/xyz/repo_three/
ProxyPass /xyz/repo_three/ http://internal.example.com:30000/ nocanon max=200
```

配置负载平衡

您可能希望在 Apache 负载平衡器后运行 depot 服务器。负载平衡的一个优点是提高系统信息库的可用性。本节显示负载平衡的两个示例。

如果已对系统信息库进行负载平衡，则所有系统信息库中的目录应完全相同，以避免在负载平衡器将客户机从一个节点切换到另一个节点时出现问题。要确保目录完全相同，请克隆参与负载平衡的系统信息库，如“维护多个相同的本地系统信息库” [32]中所述。

一个使用负载平衡的系统信息库服务器

该示例将 `http://pkg.example.com/myrepo` 连接到 `internal1.example.com:10000` 和 `internal2.example.com:10000`。

使用相应的 proxy_base 设置配置 depot 服务器，如“配置带有前缀的简单代理” [46]中所示。

将后端连接的数目限定为每个 depot 服务器运行的线程数目除以负载平衡器设置中的 depot 数目所得的值。如果不这样，Apache 打开的 depot 连接数目会超过可用连接数目，因此连接将被停止，从而降低性能。使用 max= 参数指定到每个 depot 的最大并行连接数目。以下示例中有两个 depot，每个 depot 运行 200 个线程。有关如何设置 depot 线程数目的示例，请参见“配置带有前缀的简单代理” [46]。

```
<Proxy balancer://pkg-example-com-myrepo>
        # depot on internal1
        BalancerMember http://internal1.example.com:10000 retry=5 max=100

        # depot on internal2
        BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ balancer://pkg-example-com-myrepo/ nocanon
```

一个负载平衡的和一个非负载平衡的系统信息库服务器

此示例显示了一个同时采用负载平衡和非负载平衡 depot 服务器设置的系统信息库服务器，并给出了需要添加到 httpd.conf 文件的所有指令。

在本示例中，一个域名的两个不同前缀连接到三个不同的软件包系统信息库：

- http://pkg.example.com/repo_one 连接到 internal1.example.com:10000 和 internal2.example.com:10000
- http://pkg.example.com/repo_two 连接到 internal1.example.com:20000

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain

AllowEncodedSlashes NoDecode

MaxKeepAliveRequests 10000

ProxyTimeout 30

ProxyRequests Off

<Proxy balancer://pkg-example-com-repo_one>
        # depot on internal1
        BalancerMember http://internal1.example.com:10000 retry=5 max=100

        # depot on internal2
        BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /repo_one http://pkg.example.com/repo_one/
```

```
ProxyPass /repo_one/ balancer://pkg-example-com-repo_one/ nocanon
Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200
```

配置系统信息库的 HTTPS 访问

任何客户机都可以从配置为通过 HTTP 提供软件包的系统信息库下载软件包。在某些情况下，需要限制访问。限制对系统信息库的访问的一种方法是，在支持客户机证书且启用了 SSL 的 Apache 实例后运行 depot 服务器。

使用 SSL 具有以下优点：

- 确保在客户机和服务器之间加密传送软件包数据
- 允许您基于客户机向服务器提供的证书授予对系统信息库的访问权限

要设置安全的系统信息库服务器，必须创建定制证书链：

1. 创建证书颁发机构 (certificate authority, CA)，这是证书链的头。
2. 从此 CA 向允许访问系统信息库的客户机颁发证书。

将 CA 的一个副本存储在系统信息库服务器上。每当客户机向服务器提供证书时，都会根据服务器上的 CA 对客户机证书进行验证，从而确定是否授予访问权限。

本节介绍的以下步骤用于创建证书链并将 Apache 前端配置为验证客户机证书：

- 创建密钥库
- 创建客户机证书的证书颁发机构
- 将 SSL 配置添加到 Apache 配置文件
- 创建自签名服务器证书颁发机构
- 创建 PKCS12 密钥库

有关 Oracle Solaris 中的 Apache Web 服务器特权的信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的“使用扩展特权锁定资源”。

创建密钥库

要管理证书和密钥，请创建密钥库。密钥库中存储 CA、CA 密钥以及客户机证书和密钥。

用于密钥库管理的工具为 `pktool`。有关更多信息，请参见 `pktool`(1) 手册页。

`pktool` 的缺省密钥库位置为 `/var/user/`*username*，其中 *username* 是当前系统用户的名称。如果密钥库由多个用户管理，则此密钥库缺省位置可能会存在问题。此外，IPS

软件包系统信息库管理应具有专用密钥库以避免造成证书混乱。要为 IPS 软件包系统信息库的 pktool 密钥库设置定制位置，请设置环境变量 SOFTTOKEN_DIR。根据需要重置 SOFTTOKEN_DIR 变量以管理多个密钥库。

使用以下命令为密钥库创建目录。如果多个用户需要管理密钥库，请相应地设置所有者、组和权限。

```
$ mkdir /path-to-keystore
$ export SOFTTOKEN_DIR=/path-to-keystore
```

对密钥库的访问受口令短语的保护，每次调用 pktool 命令时都必须输入该口令短语。新创建的密钥库的缺省口令短语为 changeme。务必将 changeme 口令短语更改为更安全的口令短语。

使用以下命令设置密钥库的口令短语 (PIN)：

```
$ pktool setpin
Enter token passphrase: changeme
Create new passphrase:
Re-enter new passphrase:
Passphrase changed.
$ ls /path-to-keystore
pkcs11_softtoken
```

创建客户机证书的证书颁发机构

CA 是证书链中的顶级证书。需要 CA 才能生成客户机证书并验证客户机提供的证书是否可以访问系统信息库。

第三方 CA 由一些可信任的公司（例如 VeriSign）管理。此可信管理方式允许客户机根据其某一 CA 来验证服务器的身份。本节中的示例不包括验证系统信息库服务器的身份。此示例只显示如何验证客户机证书。因此，此示例使用自签名证书创建 CA，但不使用任何第三方 CA。

CA 需要公用名称 (common name, CN)。如果只运行一个系统信息库，可能需要将 CN 设置为您组织的名称（例如 "Oracle Software Delivery"）。如果具有多个系统信息库，每个系统信息库都必须具有其自己的 CA。在这种情况下，将 CN 设置为唯一标识正为其创建 CA 的系统信息库的名称。例如，如果具有一个发行版系统信息库和一个支持系统信息库，则仅发行版 CA 中的证书允许访问发行版系统信息库，仅支持 CA 中的证书允许访问支持系统信息库。

要标识密钥库中的证书，请为证书设置描述性标签。比较好的做法是将证书标签设置为 *CN*_ca，其中 *CN* 是证书的 CN。

使用以下命令创建 CA 证书，其中 *name* 是证书 CN，*CAlabel* 是证书标签：

```
$ pktool gencert label=CAlabel subject="CN=name" serial=0x01
```

CA 将存储在密钥库中。使用以下命令显示密钥库的内容：

```
$ pktool list
```

按照“将 SSL 配置添加到 Apache 配置文件” [53]中所述配置 Apache 时，需要从密钥库中提取 CA 证书。使用以下命令将 CA 证书提取到名为 `ca_file.pem` 的文件：

```
$ pktool export objtype=cert label=CAlabel outformat=pem \
outfile=ca_file.pem
```

创建用于访问系统信息库的客户机证书

生成 CA 后，可以生成客户机证书。

生成证书签名请求

要生成客户机证书，请生成证书签名请求 (Certificate Signing Request, CSR)。CSR 包含安全地传递到服务器所需的所有信息。

如果只希望检查客户机是否拥有您颁发的有效证书，无需对任何信息进行编码。当客户机向服务器提供其证书时，服务器将根据 CA 对该证书进行确认，验证该客户机证书是否由您生成。但是，SSL 需要 CSR 的 `subject`。如果不需要将任何其他信息传递到服务器，只需将 `subject` 设置为颁发证书的国家/地区。例如，可以将 `subject` 设置为 `C=US`。

比较好的做法是将客户机的用户名编码到证书中，从而允许服务器标识客户机。用户名是您向其授予对系统信息库的访问权限的用户的名称。可以将 CN 用于此目的。为此 CSR 指定标签，以便您可以找到并提取最终证书的密钥（如“提取证书密钥” [52]中所述）。

使用以下命令生成 CSR：

```
$ pktool gencsr subject="C=US,CN=username" label=label format=pem \
outcsr=cert.csr
```

使用以下 OpenSSL 命令检查文件 `cert.csr` 中的 CSR：

```
$ openssl req -text -in cert.csr
```

对 CSR 进行签名

CSR 必须经过 CA 的签名才能创建证书。要对 CSR 进行签名，请提供以下信息：

- 将证书的 issuer 设置为使用 gencert 命令创建 CA 时用于 subject 的相同字符串，如“创建客户机证书的证书颁发机构” [50]中所述。
- 设置十六进制序列号。在此示例中，CA 序列号被指定为 0x01，因此应为第一个客户机证书提供序列号 0x02。对生成的每个新客户机证书递增序列号。

  每个 CA 及其子孙客户机证书都具有其自己的序列号集。如果在密钥库中配置了多个 CA，请小心地正确设置客户机证书序列号。
- 将 signkey 设置为密钥库中的 CA 的标签。
- 将 outcert 设置为证书文件的名称。比较好的做法是在要访问的系统信息库后指定证书和密钥。

使用以下命令对 CSR 进行签名：

```
$ pktool signcsr signkey=CAlabel csr=cert.csr  \
serial=0x02 outcert=reponame.crt.pem issuer="CN=name"
```

在文件 *reponame*.crt.pem 中创建证书。使用以下 OpenSSL 命令检查证书：

```
$ openssl x509 -text -in reponame.crt.pem
```

提取证书密钥

从密钥库提取此证书的密钥。将 label 设置为在“生成证书签名请求” [51]中运行 gencsr 以生成 CSR 时指定的相同标签值。使用以下命令从密钥库导出密钥：

```
$ pktool export objtype=key label=label outformat=pem \
outfile=reponame.key.pem
```

将证书和密钥传输到需要访问受 SSL 保护的系统信息库的客户机系统。

允许客户机系统访问受保护的系统信息库

要访问受 SSL 保护的系统信息库，客户机系统必须具有证书副本和密钥，并且必须在发布者配置中指定证书和密钥。

将证书 (*reponame*.crt.pem) 和密钥 (*reponame*.key.pem) 复制到每个客户机系统。例如，可以将它们复制到每个客户机上的 /var/pkg/ssl 目录。

使用以下命令在发布者配置中指定生成的证书和密钥：

```
$ pkg set-publisher -k reponame.key.pem -c reponame.crt.pem \
-p https://repolocation
```

请注意，仅 HTTPS 系统信息库 URI 支持 SSL 验证。文件系统信息库 URI 不支持 SSL 验证。

将 SSL 配置添加到 Apache 配置文件

要对系统信息库使用基于客户机证书的验证，首先请设置通用 depot 服务器 Apache 配置，如“Depot 服务器的 Apache 配置” [43]中所述。然后将以下 SSL 配置添加到 `httpd.conf` 文件的结尾：

```
# Let Apache listen on the standard HTTPS port
Listen 443

# VirtualHost configuration for request on port 443
<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server, needs to match your server certificate
        ServerName pkg-sec.example.com

        # enable SSL
        SSLEngine On

        # Location of the server certificate and key.
        # You either have to get one from a certificate signing authority like
        # VeriSign or create your own CA for testing purposes (see "Creating a
        # Self-Signed CA for Testing Purposes")
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile /path/to/server.key

        # Intermediate CA certificate file. Required if your server certificate
        # is not signed by a top-level CA directly but an intermediate authority
        # Comment out this section if you are using a test certificate or your
        # server certificate doesn't require it.
        # For more info:
        # http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslcertificatechainfile
        SSLCertificateChainFile /path/to/ca_intermediate.pem

        # CA certs for client verification.
        # This is where the CA certificate created in step 3 needs to go.
        # If you have multiple CAs for multiple repos, just concatenate the
        # CA certificate files
        SSLCACertificateFile /path/to/ca_cert.pem

        # If the client presents a certificate, verify it here. If it doesn't,
        # ignore.
        # This is required to be able to use client-certificate based and
        # anonymous SSL traffic on the same VirtualHost.
        # This statement could also go into the <Location> tags but putting it
        # here avoids re-negotiation which can cause security issues with older
        # servers/clients:
        # http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555
        SSLVerifyClient optional

        <Location /repo>
                SSLVerifyDepth 1
                # This is the SSL requirement for this location.
                # Requirements can be made based on various information encoded
```

```
                # in the certificate. Two variants are the most useful for use
                # with IPS repositories:
                # a) SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/reponame/ )
                #    only allow access if the CN in the client certificate matches
                #    "reponame", useful for different certificates for different
                #    repos
                #
                # b) SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
                #    grant access if clients certificate is signed by one of the
                #    CAs specified in SSLCACertificateFile
                SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )

                # proxy request to depot running at internal.example.com:12345
                ProxyPass http://internal.example.com:12345 nocanon max=500
        </Location>
</VirtualHost>
```

创建自签名服务器证书颁发机构

出于测试目的，可以使用自签名服务器证书颁发机构 (certificate authority, CA) 而不是第三方 CA。为 Apache 创建自签名服务器 CA 的步骤与“创建客户机证书的证书颁发机构” [50]中介绍的为客户机证书创建 CA 的步骤非常相似。

使用以下命令创建服务器 CA。将 `subject` 设置为服务器的 DNS 名称。

```
$ pktool gencert label=apacheCA subject="CN=apachetest" \
serial=0x01
```

使用以下命令为服务器 CA 创建 CSR。如果可以在多个名称下访问服务器或者您希望可以直接在服务器 IP 地址下使用服务器，请使用 `subjectAltNames` 指令，如 OpenSSL 文档的 Subject Alternative Name（“主题备用名称”）中所述。

```
$ pktool gencsr label=apache subject="CN=pkg-sec.internal.example.com" \
altname="IP=192.168.1.1,DNS=pkg-sec.internal.example.com" \
format=pem outcsr=apache.csr
```

使用以下命令对 CSR 进行签名。将 `server.crt` 用于 `SSLCertificateFile`。

```
$ pktool signcsr signkey=apacheCA csr=apache.csr serial=0x02 \
outcert=server.crt issuer="CN=apachetest"
```

使用以下命令提取密钥。将 `server.key` 用于 `SSLCertificateKeyFile`。

```
$ pktool export objtype=key label=apache outformat=pem \
outfile=server.key
```

为确保客户机将接受此服务器密钥，请将 CA 证书 (`apacheCA`) 添加到客户机系统上接受的 CA 目录并重新启动 `ca-certificates` 服务以创建 OpenSSL 的必需链接。

使用以下命令提取 CA 证书：

```
$ pktool export label=apacheCA objtype=cert outformat=pem \
outfile=test_server_ca.pem
```

将 CA 证书复制到客户机上的 CA 证书目录：

```
$ cp /path-to/test_server_ca.pem /etc/certs/CA/
```

重新启动 CA 证书服务：

```
$ svcadm refresh ca-certificates
```

继续之前，确保已链接新 CA 证书。刷新后，ca-certificate 服务将在 /etc/openssl/certs 目录中重新生成链接。运行以下命令以检查是否已链接新 CA 证书：

```
$ ls -l /etc/openssl/certs | grep test_server_ca.pem
lrwxrwxrwx   1 root     root          40 May  1 09:51 e89d96e0.0 -> ../../certs/CA/
test_server_ca.pem
```

由于散列值 e89d96e0.0 基于证书的主题，因此该值可能有所不同。

创建 PKCS12 密钥库以使用 Firefox 访问安全系统信息库

在“创建用于访问系统信息库的客户机证书” [51]中创建的 PEM 证书将运行以使用 pkg 客户机访问安全的系统信息库。但是，要访问浏览器用户界面 (browser user interface, BUI)，必须将证书和密钥转换为 Firefox 可以导入的格式。Firefox 接受 PKCS12 密钥库。

使用以下 OpenSSL 命令为 Firefox 创建 PKCS12 密钥库：

```
$ openssl pkcs12 -export -in /path-to/certificate.pem \
-inkey /path-to/key.pem -out name.p12
```

要导入刚创建的 PKCS12 密钥库，请选择以下 Firefox 菜单、选项卡和按钮："Edit"（编辑）> "Preferences"（首选项）> "Advanced"（高级）> "Encryption"（加密）> "View certificates"（查看证书）> "Authorities"（颁发机构）> "Import"（导入）。

一次导入一个证书。

完整安全系统信息库示例

此示例配置三个名为 repo1、repo2 和 repo3 的安全系统信息库。repo1 和 repo2 系统信息库是使用专用证书配置的。因此，repo1 的证书将不在 repo2 上运行，repo2 的证书将不在 repo1 上运行。repo3 系统信息库配置为接受两个证书之一。

该示例假定您已具有 Apache 实例的可用正确服务器证书。如果您没有 Apache 实例的服务器证书，请参见“创建自签名服务器证书颁发机构” [54]中的说明创建一个测试证书。

这三个系统信息库分别在 `https://pkg-sec.example.com/repo1`、`https://pkg-sec.example.com/repo2` 和 `https://pkg-sec.example.com/repo3` 下进行设置。这些系统信息库分别指向在端口 10001、10002 和 10003 上的 `http://internal.example.com` 设置的 depot 服务器。确保已按照“创建密钥库” [49]中所述正确设置 `SOFTTOKEN_DIR` 环境变量。

▼ 如何配置安全系统信息库

1. 为 **repo1** 创建 CA 证书。

```
$ pktool gencert label=repo1_ca subject="CN=repo1" serial=0x01
$ pktool export objtype=cert label=repo1_ca outformat=pem \
outfile=repo1_ca.pem
```

2. 为 **repo2** 创建 CA 证书。

```
$ pktool gencert label=repo2_ca subject="CN=repo2" serial=0x01
$ pktool export objtype=cert label=repo2_ca outformat=pem \
outfile=repo2_ca.pem
```

3. 创建合并的 CA 证书文件。

```
$ cat repo1_ca.pem > repo_cas.pem
$ cat repo2_ca.pem >> repo_cas.pem
$ cp repo_cas.pem /path-to-certs
```

4. 创建一个客户机证书/密钥对，以允许用户 **myuser** 访问系统信息库 **repo1**。

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo1_0001 format=pem \
outcsr=repo1_myuser.csr
$ pktool signcsr signkey=repo1_ca csr=repo1_myuser.csr  \
serial=0x02 outcert=repo1_myuser.crt.pem issuer="CN=repo1"
$ pktool export objtype=key label=repo1_0001 outformat=pem \
outfile=repo1_myuser.key.pem
$ cp repo1_myuser.key.pem /path-to-certs
$ cp repo1_myuser.crt.pem /path-to-certs
```

5. 创建一个客户机证书/密钥对，以允许用户 **myuser** 访问系统信息库 **repo2**。

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo2_0001 format=pem \
outcsr=repo2_myuser.csr
$ pktool signcsr signkey=repo2_ca csr=repo2_myuser.csr  \
serial=0x02 outcert=repo2_myuser.crt.pem issuer="CN=repo2"
$ pktool export objtype=key label=repo2_0001 outformat=pem \
outfile=repo2_myuser.key.pem
$ cp repo2_myuser.key.pem /path-to-certs
```

```
$ cp repo2_myuser.crt.pem /path-to-certs
```

6. 配置 Apache。
   将以下 SSL 配置添加到 `httpd.conf` 文件的结尾：

```
# Let Apache listen on the standard HTTPS port
Listen 443

<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server
        ServerName pkg-sec.example.com

        # enable SSL
        SSLEngine On

        # Location of the server certificate and key.
        # You either have to get one from a certificate signing authority like
        # VeriSign or create your own CA for testing purposes (see "Creating a
        # Self-Signed CA for Testing Purposes")
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile /path/to/server.key

        # Intermediate CA certificate file. Required if your server certificate
        # is not signed by a top-level CA directly but an intermediate authority.
        # Comment out this section if you don't need one or if you are using a
        # test certificate
        SSLCertificateChainFile /path/to/ca_intermediate.pem

        # CA certs for client verification.
        # This is where the CA certificate created in step 3 needs to go.
        # If you have multiple CAs for multiple repos, just concatenate the
        # CA certificate files
        SSLCACertificateFile /path/to/certs/repo_cas.pem

        # If the client presents a certificate, verify it here. If it doesn't,
        # ignore.
        # This is required to be able to use client-certificate based and
        # anonymous SSL traffic on the same VirtualHost.
        SSLVerifyClient optional

        <Location /repo1>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo1/ )
                # proxy request to depot running at internal.example.com:10001
                ProxyPass http://internal.example.com:10001 nocanon max=500
        </Location>

        <Location /repo2>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo2/ )
                # proxy request to depot running at internal.example.com:10002
                ProxyPass http://internal.example.com:10002 nocanon max=500
        </Location>
```

```
        <Location /repo3>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
                # proxy request to depot running at internal.example.com:10003
                ProxyPass http://internal.example.com:10003 nocanon max=500
        </Location>

</VirtualHost>
```

7. 测试对 **repo1** 的访问。

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo1/
```

8. 测试对 **repo2** 的访问。

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo2/
```

9. 测试对 **repo3** 的访问。

   使用 repo1 证书测试对 repo3 的访问。

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

   使用 repo2 证书测试对 repo3 的访问。

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

索引

Y

Z