Oracle® Solaris 11 Express Image Packaging System Guide

Part No: 820–6572–11
November 2010

110425@25097

Contents

Preface

The *Oracle Solaris 11 Express Image Packaging System Guide* describes the Oracle Solaris Image Packaging System (IPS). IPS tools enable you to install, upgrade, and remove software packages for the Oracle Solaris operating system. IPS also enables you to create your own software packages, create and manage package repositories, and mirror existing package repositories.

Who Should Use This Book

This book is for system administrators who install and manage software and manage system images.

How This Book Is Organized

- Chapter 1, "Introduction to the Image Packaging System," describes the Image Packaging System and components such as packages, publishers, and repositories.
- Chapter 2, "IPS Graphical User Interfaces," explains how to use Package Manager and Update Manager, including how to use Web Install.
- Chapter 3, "Working With Packages," shows many examples of how to manage software packages and package publishers.
- Chapter 4, "Creating and Managing Images," describes how to create and update images and how to set image properties.
- Appendix A, "IPS Command Reference," briefly describes IPS commands and shows the `pkg`(1) man page.
- Glossary defines IPS terms.

Related Documentation

In addition to these books, see the Package Manager online help and the `pkg`(1M) and `beadm`(1M) man pages.

- *Getting Started With Oracle Solaris 11 Express* explains how to install the Oracle Solaris OS and includes enough information about IPS to enable you to install additional software after the initial installation and keep your software updated.
- *Managing Boot Environments With Oracle Solaris 11 Express* provides complete information about boot environments and how to use and manage them.
- *Oracle Solaris 11 Express Automated Installer Guide* explains how to set up hands-free installations of clients on a network, including how to specify packages to install and publishers to use.

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (`http://docs.sun.com`)
- Support (`http://www.oracle.com/us/support/systems/index.html`)
- Training (`http://education.oracle.com`) – Click the Sun link in the left navigation bar.

Oracle Software Resources

Oracle Technology Network (`http://www.oracle.com/technetwork/index.html`) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (`http://forums.oracle.com`).
- Get hands-on step-by-step tutorials with Oracle By Example (`http://www.oracle.com/technetwork/tutorials/index.html`).
- Download Sample Code (`http://www.oracle.com/technology/sample_code/index.html`).

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

CHAPTER 1

Introduction to the Image Packaging System

The Oracle Solaris Image Packaging System (IPS) is a framework that enables you to install, update, and remove software packages for the Oracle Solaris operating system. IPS also enables you to create your own software packages, create and manage package repositories, and mirror existing package repositories.

Image Packaging System

Oracle Solaris software is distributed in IPS packages. IPS packages are stored in IPS package repositories, which are populated by IPS publishers. A subset of the capabilities that are available through the IPS command-line interface is available through the IPS graphical user interfaces: Package Manager and Update Manager.

IPS tools provide the following capabilities:

- List, search, install, update, and remove software packages.
- List, add, and remove package publishers. Change publisher attributes such as search priority and stickiness.
- Update your system to a new software release.
- Create mirrors of existing package repositories.
- Create new package repositories.
- Create and publish packages.
- List, create, rename, activate, and remove boot environments.

To use IPS, you must be running the Oracle Solaris 11 Express operating system. To install the Oracle Solaris 11 Express release, see the *Getting Started With Oracle Solaris 11 Express* guide.

IPS Concepts

This section defines terms and concepts that are used in the remainder of this guide.

IPS Packages

IPS manages software in units of packages. An IPS *package* is a collection of directories, files, links, drivers, dependencies, groups, users, and license information in a defined format. This collection represents the installable objects of a package. Packages have attributes such as package name and description.

You can use commands to view information about a package, or you can view the manifest for a package.

IPS supports both IPS packages and SVR4 packages.

There is no standard on-disk format for an IPS package. IPS does not have an equivalent for `.rpm`, SVR4 package, or `.nbm` files.

Fault Management Resource Identifiers

Each IPS package is represented by a Fault Management Resource Identifier (FMRI).

The FMRI includes descriptive information about the package, such as the package name, version information, and date, as shown in the following example:

```
pkg://solaris/library/libc@5.11,5.11-0.75:20071001T163427Z
```

- Scheme: `pkg`
- Publisher: `solaris`
- Category: `library`
- Package name: `libc`
- Version string
  - Component version: `5.11`
  - Build version: `5.11`
  - Branch version: `0.75`
  - Time the package was published, in ISO-8601 basic format: `20071001T163427Z`

Publishers and Repositories

- A *publisher* is a forward domain name that identifies a person, group of persons, or an organization that publishes one or more packages.

- A *repository* is a location where clients publish and retrieve packages. The location is described by a Universal Resource Identifier (URI). A repository is also called a depot server.
- A repository contains packages from a single publisher.
- A publisher can publish to multiple repositories.

Multiple repositories can contain packages with the same name. Publishers can be configured into a preferred search order in each image. When a package is selected for installation, the preferred publisher is searched first for that package, then the next most preferred publisher is searched until the package is found or all publishers have been searched.

A repository URI is not required to contain the name of the publisher. For example, the `solaris` publisher publishes to repositories hosted at `oracle.com`.

Repository Origins and Mirrors

A repository has an origin and zero or more mirrors.

- The repository *origin* is the location of a package repository that contains both package *metadata* (package manifests and catalogs) and package *content* (package files).
- A *mirror* is a location of a package repository that contains only package content.

Mirrors provide a subset of the data that origins provide. Mirrors can be used only for downloading package files. Package metadata is downloaded from the origin. IPS clients access the origin to obtain a publisher's catalog, even when the clients download package content from a mirror.

Images and Boot Environments

- An *image* is a location where packages can be installed.

  An image can be one of three types:

  - Full images are capable of providing a complete system.
  - Partial images are linked to a full image (the parent image), but do not provide a complete system on their own.
  - User images contain only relocatable packages.
- A *boot environment* (BE) is bootable instance of an image. You can maintain multiple BEs on your system, and each BE can have different software versions installed. When you boot your system, you have the option to boot into any of the BEs on the system.

A new BE is automatically created when you perform one of the following operations:

- Install the Oracle Solaris OS.
- Update particular key system packages such as some drivers and other kernel components. This can happen when you install, uninstall, update, change variant, or change facet.
- Use the `beadm create` command.

Often a new BE is created when you execute the `pkg update` command to update all packages that have updates available.

If a new BE is created, the system performs the following steps:

1. Creates a clone of the current BE that is a bootable image.

   The clone BE includes everything hierarchically under the main root dataset of the original BE. Shared file systems are not under the root dataset and are not cloned. Instead, the BE accesses the original shared file systems.
2. Updates the packages in the clone BE, but does not update any packages in the current BE.

   If zones are configured in the current BE, these existing zones are configured in the new BE. However, packages are not installed or updated in these zones. You must manually update each zone in the new BE.
3. Sets the new BE as the default boot choice the next time the system is booted. The current BE remains as an alternate boot choice.

Several IPS commands provide the following options to support BE creation. Check the applicable man page.

`--be-name` *BEname*	Assign *BEname* to the newly created BE. This option is ignored if a new BE is not created.
`--deny-new-be`	Do not create a new BE, even if a new BE is required by the components to be installed. If this option is specified and a new BE is required, the operation is not performed.
`--require-new-be`	Create a new BE, even if a new BE is not required.

If a new BE is required but not enough space is available to create a new BE, you might be able to delete existing unneeded BEs. For more information about BEs, see the *Managing Boot Environments With Oracle Solaris 11 Express* guide.

Package Variants and Facets

Software can have components that are optional and components that are mutually exclusive. Examples of optional components include locales and documentation. Examples of mutually exclusive components include SPARC or x86 and debug or non-debug binaries.

In IPS, optional components are called *facets* and mutually exclusive components are called *variants*. Both variants and facets appear as tags on IPS *actions*.

Actions represent the installable objects on a system. Actions are described in the manifest of a package. Every action consists primarily of its name and a key attribute. Together, these refer to a unique object as it follows a version history.

Variants and facets affect whether a particular action is selected or deselected for installation.

The following list shows some examples of facet and variant tags and their possible values.

Name	Values
`facet.locale.*`	`true`, `false`
`facet.doc.man`	`true`, `false`
`facet.doc`	`true`, `false`
`facet.devel.*`	`true`, `false`
`variant.arch`	`sparc`, `i386`, `zos`
`variant.debug.*`	`true`, `false`

Actions that are not tagged with facets or variants are included.

An action that is tagged with a variant that is not selected is excluded. An action that is tagged with one or more facets is excluded if none of the facets is selected.

- Variant tags are evaluated with AND. If any of the variant tags does not match, the action is not installed.
- Facet tags are evaluated with OR. If any of the facet tags match, the action is not excluded.

A single action can have multiple facet and variant tags. An example of a component with multiple facet and variant tags is an architecture-specific header file that is used by developers.

Variants and facets are set at the image level. An image with a variant set to a particular property can only have actions that match that variant installed on it. For example, you cannot install an x86 package into a SPARC image.

System administrators can perform the following operations on facets and variants:

- Display the values of all variants or facets set on the current image.
- Change variants and facets in the current image. This operation probably will update packages and might require a new BE.

CHAPTER 2

IPS Graphical User Interfaces

IPS includes two tools that provide graphical user interfaces (GUIs).

- Package Manager provides most package and publisher operations and some boot environment (BE) operations. If you are new to the Oracle Solaris OS and IPS technologies, use Package Manager to quickly download and install packages.
- Update Manager updates all packages that have updates available.

Using Package Manager

Package Manager provides a subset of the tasks that can be performed from the command line:

- List, search, install, update, and remove packages
- Add and configure package sources
- Activate, rename, and remove BEs

Start Package Manager in one of the following ways:

Tool bar	Click the Package Manager icon in the tool bar. The Package Manager icon is a box with a circling arrow.
Desktop icon	Double-click the Package Manager icon on the desktop.
Menu bar	Choose System>Administration>Package Manager.
Command line	`$ /usr/lib/pm-launch packagemanager`

For complete Package Manager documentation, choose Help>Contents from the Package Manager menu bar.

Package Manager Command Line Options

The following options are supported for the `packagemanager(1)` command.

TABLE 2–1 Package Manager Command Options

Option	Description
`--image-dir` or `-R` *dir*	Operate on the image rooted at *dir*. The default behavior is to operate on the current image. The following command operates on the image stored at `/aux0/example_root`: `$ /usr/lib/pm-launch packagemanager -R /aux0/example_root`
`--update-all` or `-U`	Update all installed packages that have updates available. Specifying this option is the same as choosing the Updates option in the Package Manager GUI. See "Using Update Manager" on page 18 for more information about updating all packages.
`--info-install` or `-i` *file*`.p5i`	Specify a `.p5i` file to run Package Manager in Web Install mode. The specified file must have the extension `.p5i`. See the "Using Web Install" on page 16 for more information.
`--help` or `-h`	Display command usage information.

Using Web Install

See the Package Manager Help for detailed information about the Web Install process.

Package Manager supports installing packages using a simple one-click Web Install process. The Web Install process uses a `.p5i` file. A `.p5i` file contains information to add publishers and add packages that can be installed from these publishers. The information in the `.p5i` file is read and used by the Web Install process.

Exporting Files Using Web Install

If you want other users to be able to install packages that you have installed on your system, you can export the installation instructions for those package files using the Web Install process. The Web Install process creates a `.p5i` file that consists of installation instructions for those packages and publishers to be installed.

To export the installation instructions for your selected packages and their publishers to a `.p5i` file, perform the following steps:

1. From the Package Manager Publisher drop-down menu, select the publisher from which you want to include the packages in the `.p5i` file.
2. In the Package Manager package list pane, select the package whose installation instructions you want to distribute.
3. Choose File>Export Selections to display the Export Selections Confirmation window.
4. Click the OK button to confirm the selections. The Export Selections window is displayed.

5. A default name for the `.p5i` file is provided. You can change this file name, but do not change the `.p5i` extension.
6. A default location for the `.p5i` file is provided. You can change the location.
7. Click the Save button to save the file name and location.

Using Web Install to Add Publishers and Install Packages

The Web Install process enables you to install packages through a `.p5i` file. This file might be on your desktop or on a web site.

1. Use one of the following methods to start Package Manager in Web Install mode:
   - Click on a `.p5i` file on your desktop.
   - Start Package Manager from the command line and specify a `.p5i` file:

     ```
     $ /usr/lib/pm-launch packagemanager ./wifile.p5i
     ```
   - Go to a URL location that contains a link to a `.p5i` file.

     If the `.p5i` file is located on a web server that has registered this MIME type, just click the link to the `.p5i` file.

     If the `.p5i` file is located on a web server that has not registered this MIME type, save the `.p5i` file to your desktop and then click on it.
2. The Install/Update window is displayed. The label at the top of the window is: “Package Manager Web Installer/The following will be added to your system.” The publishers and packages to be installed are listed. Click the Proceed button to continue with the installation.
3. If the specified package publisher is not already configured on your system, the Add Publisher window is displayed. The name and URI of the publisher are already entered.

   If the publishers to be added are secure publishers, an SSL key and certificate are required. Browse to locate the SSL Key and SSL Certificate on your system.

   The Adding Publisher Complete dialog displays if the publisher is added successfully. Click the OK button to continue with the installation.
4. If a `.p5i` file contains packages from a disabled publisher, Web Install opens an Enable Publisher dialog. Use this dialog to enable the publisher so that you can install the packages.

The Install/Update window now looks the same as when you select the Package Manager Install/Update option.

The application closes when all packages are installed.

Using Update Manager

Update Manager updates all installed packages to the newest version allowed by the constraints imposed on the system by installed packages and publisher configuration. This function is the same as the following functions:

- In the Package Manager GUI, choose the Updates button or the Package>Updates menu option.
- Use the Package Manager CLI.

  ```
  $ /usr/lib/pm-launch packagemanager --update-all
  ```
- Use the `pkg` command.

  ```
  # pkg update
  ```

Start Update Manager in one of the following ways:

Status bar	When updates are available, you should see a notification in the status bar. Click where indicated in the notification. The Update Manager icon is a stack of three boxes.
Menu bar	Choose System>Administration>Update Manager.
Command line	`$ /usr/lib/pm-launch pm-updatemanager`

The Updates window displays, and the update process starts:

1. The system refreshes all catalogs.
2. The system evaluates all installed packages to determine which packages have updates available.
   - If no packages have updates available, the message “No Updates Available” is displayed and processing stops.
   - If package updates are available, the packages to be updated are listed for your review. This is your last chance to click the Cancel button to abort the update.
3. Click the Proceed button to continue with the update. The system downloads and installs all package updates.

   The following packages are updated first if they have updates available. Then any other packages are updated.

   ```
   package/pkg
   package/pkg/packagemanger
   package/pkg/updatemanager
   ```

   By default, each package is updated from the publisher from which it was originally installed. If the original publisher is non-sticky, then a newer version of the package that is compatible with this image could be installed from another publisher. Use the Package Manager Manage Publishers window or the `pkg set-publisher` command to set a publisher as sticky or non-sticky.

A new BE might be created if certain packages are updated. See "Images and Boot Environments" on page 11.

If an error occurs at any time during the update process, the Details panel is expanded and the details of the error are displayed. An error status indicator is shown next to the failed stage.

4. If the system created a new BE for the update, you can edit the default BE name. When you are satisfied with the BE name, click the Restart Now button to restart your system immediately. Click the Restart Later button to restart your system at a later time. You must restart to boot into the new BE. The new BE will be your default boot choice. Your current BE will be available as an alternate boot choice.

Update Manager Command Line Options

The following options are supported for the `pm-updatemanager`(1) command.

TABLE 2–2 Update Manager Command Options

Option	Description
`--image-dir` or `-R` *dir*	Operate on the image rooted at *dir*. The default behavior is to operate on the current image. The following command updates the image at `/aux0/example_root`: `$` **`/usr/lib/pm-launch pm-updatemanager -R /aux0/example_root`**
`--help` or `-h`	Display command usage information.

◆ ◆ ◆ CHAPTER 3

Working With Packages

This chapter shows you how to install and manage packages on your system and how to manage package publishers.

See also Appendix A, "IPS Command Reference."

Package Management Tasks

The following table provides a list of package management tasks and references to instructions for accomplishing those tasks. Use the `pkg help` command to show all subcommands and options of the `pkg` command.

TABLE 3–1 Package Management Task Map

Package Management Task	Instructions
Add or update software (packages).	"How to Install or Update a Package" on page 22 and "How to Update a Package" on page 24
Perform a trial installation of packages to see the changes that will occur on the system when a package is installed, without actually installing the package.	"How to View an Installation Action Without Installing" on page 24
Verify whether a package was installed correctly and fix any reported installation errors.	"How to Verify a Package Installation" on page 24 and "How to Fix Package Installation Errors" on page 25
Remove packages.	"How to Uninstall Packages" on page 25 and "How to Uninstall a Package From an Inactive Boot Environment" on page 25
Search for a package installed on your system or search in a remote repository.	"How to Search for Packages" on page 26

TABLE 3–1 Package Management Task Map *(Continued)*

Package Management Task	Instructions
Show information about packages.	"How to Show the Contents of a Package" on page 27, "How to Show Information About Packages" on page 27, and "How to List Package State Information" on page 28
Display and change a variant or a facet.	"How to Display a Variant" on page 30, "How to Change a Variant" on page 30, "How to Display a Facet" on page 31, and "How to Change a Facet" on page 31
View operation history.	"How to View and Delete Operation History" on page 31
Show information about publishers.	"How to Display Publisher Information" on page 32
Manage publishers and repositories.	"How to Add, Modify, or Remove a Package Publisher" on page 33
Configure an image to install signed packages.	"Installing Signed Packages" on page 35

If you receive error messages about trouble contacting a repository or publisher, check the following issues.

- Ensure that you have Internet connectivity.
- Ensure the publisher origin URI is correct. Can you see the packages in the repository when you go to the URI using a web browser?
- If your system is connected to a proxy server, ensure that the proxy is working correctly.
- Ensure that name resolution is configured correctly.
- Do you need an SSL key and certificate to access the repository?

If your packaging operations are slow, your disk might be too full. You might want to remove unused BEs to recover some space. Use the `beadm list` command to list the existing BEs. See the `beadm`(1M) man page, the *Managing Boot Environments With Oracle Solaris 11 Express* guide, or the Package Manager online help for instructions about removing BEs.

Installing and Managing Packages

IPS enables you to download and install software packages from an IPS package repository.

▼ How to Install or Update a Package

By default, the newest version of a package that is compatible with the rest of the image is installed from the preferred publisher. If the package is already installed, the package is updated by installing the newest version of the package that is compatible with the rest of the image from the publisher that provided the currently installed version. If you have more than one publisher configured, you can control which publisher provides a package by setting publisher stickiness and search order or by specifying the publisher in the package FMRI. You can also specify the version you want to install in the package FMRI. See "Fault Management Resource Identifiers" on page 10 for a description of a package FMRI. A BE might be created when you install, update, or uninstall a package. See "Images and Boot Environments" on page 11 for information about creating new BEs and for descriptions of the `--be-name`, `--require-new-be`, and `--deny-new-be` options.

- **Use the `pkg install` command to install or update a package.**

  You can specify more than one *pkg-fmri* pattern.

  ```
  # pkg install pkg-fmri
  ```

Example 3–1

Installing a Package

This example installs the `installadm` package. The output displays the status of the download, number of packages that were installed, number of files that were installed, and the size of the download in megabytes.

```
# pkg install install/installadm
DOWNLOAD     PKGS       FILES    XFER (MB)
Completed     9/9   1067/1067      6.1/6.1

PHASE                         ACTIONS
Install Phase               1458/1458

PHASE                             ITEMS
Package State Update Phase          9/9
Image State Update Phase            2/2
```

Example 3–2

Installing a Package From a Specific Publisher

To install a package from a specific publisher, specify the publisher name in the *pkg-fmri*.

```
# pkg install pkg://example.com/developer/sunstudio12u1
```

Example 3–3

Installing a Specific Version of a Package

To install a specific version of a package, specify the version information in the *pkg-fmri*.

```
# pkg install pkg:/developer/sunstudio12u1@12.1.1,5.11-0.111:20100306T002245Z
```

▼ How to Update a Package

- **You can use either the `install` or `update` subcommand to update a package.**

  The `install` subcommand installs the package if the package is not already installed in the image. If you want to be sure to update only packages that are already installed, and not install any new packages, then use the `update` subcommand.

  You can specify more than one *pkg-fmri* pattern.

  ```
  # pkg update pkg-fmri
  ```

Caution – If you use the `pkg update` command with no *pkg-fmri* specified, all installed packages that have updates available are updated. See "Update an Image" on page 42.

If a newer version of an installed package is available and is compatible with the rest of the image, the package is updated to that version. See "How to Install or Update a Package" on page 22 for a more complete description of what package is installed from which publisher.

▼ How to View an Installation Action Without Installing

You can check exactly what will be installed before you actually install anything. Using the `-n` option, you can execute the install command without making any persistent change. The `-n` option shows the changes that would be made if you executed the command without the `-n` option. You can also use the `-n` option with the `update` and `uninstall` subcommands.

- **Use the `-n` option with the `pkg install` command to see what would be installed without actually performing the installation.**

  The `-v` option gives verbose information.

  ```
  # pkg install -nv pkg-fmri
  ```

Example 3–4 View an Installation Action Without Installing

The following command gives information about what would be installed but does not actually install anything.

```
# pkg install -n pkg:/developer/sunstudio12u1
```

▼ How to Verify a Package Installation

- **Use the `pkg verify` command to validate the installation of a package.**

  ```
  $ pkg verify pkg-fmri
  ```

You can specify more than one *pkg-fmri* pattern. Use the `-v` option to display information messages. Use the `-q` option to display only error messages.

Example 3–5 Verify the Installation of a Package

```
$ pkg verify -v e1000g
Verifying: PACKAGE                          STATUS
pkg://solaris/driver/network/e1000g         OK
```

▼ How to Fix Package Installation Errors

● **Use the `pkg fix` command to fix package installation errors reported by the `pkg verify` command.**

```
$ pkg fix --accept pkg-fmri
```

If you do not specify the `--accept` option, and any packages require a license to be accepted, the fix operation fails. Use the `--licenses` option to display all licenses for packages that are installed or updated by this fix operation.

▼ How to Uninstall Packages

● **Use the `pkg uninstall` command to uninstall existing packages.**

You can specify more than one *pkg-fmri* pattern. You must specify at least one *pkg-fmri* pattern.

```
# pkg uninstall pkg-fmri
```

Use the `-r` option to recursively uninstall any packages that contain `require` dependencies on the *pkg-fmri* package. See "Images and Boot Environments" on page 11 for information about creating new BEs and for descriptions of the `--be-name`, `--require-new-be`, and `--deny-new-be` options.

▼ How to Uninstall a Package From an Inactive Boot Environment

1 **Mount the inactive BE from which the package is to be uninstalled.**

```
# beadm mount inactive-be mntpt
```

2 **Uninstall the package.**

```
# pkg -R mntpt uninstall pkg-fmri
```

3 **Unmount the previously mounted BE.**

```
# beadm unmount inactive-be
```

▼ How to Search for Packages

● **Use the `pkg search` command to search for packages whose data matches the specified pattern.**

You can specify more than one *pattern*. Multiple patterns are ANDed together.

```
$ pkg search pattern
```

By default, repositories associated with all publishers configured for this image are searched. Use the `-l` option to search only packages that are installed in this image. By default, matches are displayed only for currently installed or newer package versions. Use the `-f` option to display all matched versions.

Example 3–6 Searching for a Package in the Installed Image

The following example searches for the `bash` package in the installed image. The INDEX column tells you where in the data the match was found.

```
$ pkg search -l bash
INDEX      ACTION VALUE                      PACKAGE
pkg.fmri   set    solaris/shell/bash         pkg:/shell/bash@4.0.28-0.149
basename   file   usr/bin/bash               pkg:/shell/bash@4.0.28-0.149
basename   dir    etc/bash                   pkg:/shell/bash@4.0.28-0.149
basename   dir    usr/share/bash             pkg:/shell/bash@4.0.28-0.149
```

Example 3–7 Searching For a Package in a Specified Repository

In this example, two repositories that are not configured for this image are searched.

```
$ pkg search -s http://pkg.example1.com/release \
-s http://pkg.example2.com/release ksh
```

Example 3–8 Searching For a Package That Delivers a Specific File

This example shows that the `libdhcpagent` library came from the `system/library` package.

```
$ pkg search -l /lib/libdhcpagent.so.1
INDEX      ACTION VALUE                 PACKAGE
path       file   lib/libdhcpagent.so.1 pkg:/system/library@0.5.11-0.149
```

Example 3–9 Searching Using Wildcards and Boolean Directives

Multiple patterns are ANDed by default. You can also specify OR, *, and ?.

```
$ pkg search netbeans AND plug*in OR ide
```

Example 3–10 Searching Using Fields

The search string can be specified as the following set of fields:

pkg_name:*action_type*:*key*:*token*

Missing fields are implicitly wildcarded. Explicit wildcards can be used in the *pkg_name* and *token* fields. The *action_type* and *key* fields must match exactly. See "Actions" in the `pkg(5)` man page for a list of possible action types. Examples of keys include `basename`, `description`, and `driver_name`.

```
$ pkg search -l 'depend::package/pkg'
INDEX       ACTION VALUE                     PACKAGE
incorporate depend package/pkg@0.5.11-0.150 pkg:/consolidation/ips/ips-incorporation@0.5.11-0.150
require     depend package/pkg               pkg:/package/pkg/package-manager@0.5.11-0.150
require     depend package/pkg               pkg:/system/zones/brand/ipkg@0.5.11-0.150
```

▼ How to Show the Contents of a Package

● **Use the `pkg contents` command to show the action attributes of a package.**

By default, only the path attribute is shown. Use the `-o` option to specify additional attribute values to display. You can specify more than one *pkg-fmri* pattern. If you do not specify any *pkg-fmri* pattern, data are shown for all installed packages. If the package is not installed, use the `-r` option to retrieve data from all publishers configured for this image. When you use the `-r` option, you must specify one or more *pkg-fmri* patterns.

```
$ pkg contents pkg-fmri
```

Example 3–11 Listing the Contents of a Package

In this example, the size and path of each file in the `e1000g` package are listed:

```
$ pkg contents -t file -o pkg.size,path network/e1000g
PKG.SIZE PATH
  471656 kernel/drv/amd64/e1000g
  323612 kernel/drv/e1000g
    4238 kernel/drv/e1000g.conf
```

▼ How to Show Information About Packages

● **Use the `pkg info` command to display information about a package.**

You can specify more than one *pkg-fmri* pattern. If you do not specify any *pkg-fmri* pattern, information about all installed packages is displayed. If the package is not installed, use the `-r` option to retrieve data from all publishers configured for this image. When you use the `-r` option, you must specify one or more *pkg-fmri* patterns.

```
$ pkg info pkg-fmri
```

Example 3–12 Displaying Information About a Specific Package

This example displays information about the `openoffice` package. This example uses the `-r` option to display information even if the package is not installed.

```
$ pkg info -r openoffice
          Name: openoffice
       Summary: OpenOffice.org 3.1.0
      Category: Applications/Office
         State: Not installed
     Publisher: solaris
       Version: 3.1.0
 Build Release: 5.11
        Branch: 0.111
Packaging Date: May 18, 2009 06:27:12 AM
          Size: 430.65 MB
FMRI: pkg://solaris/openoffice@3.1.0,5.11-0.111:20090518T062712Z
```

Example 3–13 Displaying Copyright and License Information About a Package

This example displays the copyright and license information for the `firefox` package.

```
$ pkg info --license firefox
Copyright (c) 2008, 2010, Oracle and/or its affiliates.  All rights reserved.

Copyright (c) 1998-2010 by Contributors. All rights reserved.
Firefox and the Firefox logos are trademarks of the Mozilla Foundation. All rights reserved.
```

▼ How to List Package State Information

Both the `info` and `list` subcommands display the package name and publisher and some version information. In addition to the package name and publisher, the `pkg info` command displays the package summary, category, and size. The `pkg list` command shows whether an update exists for the package, whether an update can be installed in this image, and whether a package is obsolete or renamed. The `pkg info` command displays several lines of information for each package. The `pkg list` command displays one line of information for each package.

● **Use the `pkg list` command to display state information about a package such as whether an update is available.**

$ **pkg list** *pkg-fmri*

You can specify more than one *pkg-fmri* pattern. If you do not specify any *pkg-fmri* pattern, all installed packages are listed. The `pkg list -u` command lists all installed packages that have newer versions available.

The `pkg list` command displays the following information:

NAME (PUBLISHER)	Name of the package. If the publisher is not the preferred publisher, then the publisher name is listed in parentheses after the package name.

VERSION	The release and branch versions of the package. If you specify the `-v` option, the VERSION column is not shown. Instead, the full package FMRI is shown in the NAME column.
STATE	The state of the package. The state can be either “installed” or “known.”
UFOXI	Flags that give information about how the package relates to other packages in the image.

U	A `u` in the `U` column indicates that a newer version of this package is available. The newer version might not be possible to install because of package dependencies or other constraints. See the `-a` option in Example 3–16.
O	An `o` in the `O` column indicates that this package is obsolete. An `r` in the `O` column indicates that this package has been renamed.

Example 3–14 Listing a Package That Is Installed

In this example, the package is installed from the preferred publisher, and the package has an update available.

```
$ pkg list firefox
NAME (PUBLISHER)       VERSION         STATE      UFOXI
web/browser/firefox    0.5.11-0.150    installed  u----
```

Example 3–15 Listing the Newest Versions of a Package

Use the `-n` option to list the newest versions of a package from all configured publishers.

```
$ pkg list -n firefox
NAME (PUBLISHER)                     VERSION         STATE      UFOXI
web/browser/firefox (example.com)    0.5.11-0.151    known      -----
web/browser/firefox                  0.5.11-0.150    installed  u----
```

Example 3–16 Listing a Package That Is Not Installed

Use the `-a` option to list installed packages and the newest version that is available for installation. Packages are available for installation if they are allowed by the installed incorporations and by the variants of the image. In this example, the package is not currently installed but is available for installation from the preferred publisher. The package has been renamed from `sunstudio12u1` to `developer/sunstudio12u1`.

```
$ pkg list -a sunstudio12u1
NAME (PUBLISHER)           VERSION         STATE      UFOXI
```

```
developer/sunstudio12u1    12.1.1-0.111    known      -----
sunstudio12u1              12.1.1-1        known      --r--
```

Example 3–17 Listing the Package Name and Summary

Use the `-s` option to display just the package name and summary.

```
$ pkg list -s developer/sunstudio12u1
NAME (PUBLISHER)           SUMMARY
developer/sunstudio12u1    Sun Studio - C, C++, & Fortran compilers and Tools
```

▼ How to Display a Variant

A variant is a mutually exclusive component of a package such as architecture. Variants appear as tags on IPS actions and affect whether that action is installable. If an action has any variant tags, all variant tags must match the selection criteria to install the action.

- **Use the `pkg variant` command to display the values of variants of an image.**

  `$ `**`pkg variant`** *variant_spec*

  You can specify more than one *variant_spec*.

Example 3–18 Displaying the Values of All Variants

```
$ pkg variant
VARIANT              VALUE
variant.solaris.zone global
variant.arch         i386
```

▼ How to Change a Variant

- **Use the `pkg change-variant` command to change the value of a variant.**

  `# `**`pkg change-variant -n --accept`** *variant_spec*=*instance*

  Use the `-n` option to see what would change if you performed the operation without `-n`, but make no actual changes. A new BE might be created. See "Images and Boot Environments" on page 11.

Example 3–19 Changing a Variant

```
# pkg change-variant -n --accept variant.debug=false
```

▼ How to Display a Facet

A facet is an optional component of a package such as a locale. Facets appear as tags on IPS actions and affect whether that action is installable. If an action has any facet tags, at least one facet tag must match the selection criteria to install the action.

- **Use the `pkg facet` command to display the current values of all facets defined in the current image.**

  ```
  $ pkg facet facet_spec
  ```

  You can specify more than one *facet_spec*.

Example 3–20 Displaying All Facets in the Current Image

```
$ pkg facet
FACETS       VALUE
facet.devel  FALSE
```

▼ How to Change a Facet

- **Use the `pkg change-facet` command to change the current values of a facet.**

  ```
  # pkg change-facet -n --accept facet_spec=True|False|None
  ```

  Use the `-n` option to see what would change if you performed the operation without `-n`, but make no actual changes. A new BE might be created. See "Images and Boot Environments" on page 11.

  If the facet value is set to None, the facet specification is removed from the current image.

Example 3–21 Changing a Facet in the Current Image

```
# pkg change-facet facet.devel=True
$ pkg facet facet.devel
FACETS       VALUE
facet.devel  TRUE
```

▼ How to View and Delete Operation History

1 **Use the `pkg history` command to view the command history in the current image.**

Use the `-l` option to display more information, including the outcome of the command, the time the command completed, the version and name of the client used, the name of the user who performed the operation, and any errors encountered while executing the command. Use the `-n` option to display only the specified number of most recent operations.

```
$ pkg history
TIME                OPERATION                 CLIENT    OUTCOME
2010-10-16T16:32:46 update-publisher          pkg       Succeeded
2010-10-16T16:34:42 refresh-publishers        pkg       Succeeded
2010-10-16T16:36:04 rebuild-image-catalogs    pkg       Succeeded
2010-10-16T16:38:17 install                   pkg       Succeeded
```

2 **Use the `pkg purge-history` command to delete all command history information.**

```
# pkg purge-history
```

Managing Package Publishers

IPS enables you to configure package publishers and repositories.

▼ How to Display Publisher Information

● **Use the `pkg publisher` command to display information about package publishers configured for this image.**

Example 3–22 Display All Publishers

If no options or arguments are specified, all publishers configured for this image are shown.

```
$ pkg publisher
PUBLISHER                   TYPE    STATUS   URI
solaris        (preferred)  origin  online   http://pkg.oracle.com/solaris/release
example.com    (non-sticky) origin  online   http://pkg.example.com/release
```

Example 3–23 Display Only the Preferred Publisher

```
$ pkg publisher -P
PUBLISHER                   TYPE    STATUS   URI
solaris        (preferred)  origin  online    http://pkg.oracle.com/solaris/release
```

Example 3–24 Display Information About a Specific Publisher

This example displays information about the `solaris` publisher.

```
$ pkg publisher solaris
      Publisher: solaris
          Alias:
     Origin URI: http://pkg.oracle.com/solaris/release
```

```
        SSL Key: None
       SSL Cert: None
    Client UUID: 99e58a54-1119-11df-98e6-00262236a2ae
Catalog Updated: October 17, 2010 08:11:06 PM
        Enabled: Yes
```

▼ How to Add, Modify, or Remove a Package Publisher

● **Use the `pkg set-publisher` command to perform the following operations:**

- Add a publisher. To remove a publisher, use the `pkg unset-publisher` command. The preferred publisher cannot be removed.
- Specify the preferred publisher.
- Enable or disable a publisher. The preferred publisher cannot be disabled. A newly-added publisher is enabled by default.
- Set publisher stickiness. A newly-added publisher is sticky by default. If a publisher is non-sticky, then a package that was installed from this publisher could be updated from another publisher.
- Set publisher search order. The preferred publisher is first in the search order. A newly-added publisher is last in the search order by default.
- Add or remove a publisher mirror.
- Specify SSL keys and certificates for a publisher.
- Change a publisher URI.
- Set and unset a publisher property, and add and remove a publisher property value. See "Installing Signed Packages" on page 35.

Example 3–25 Adding a Publisher

Use the `-g` option to specify the publisher origin URI.

```
# pkg set-publisher -g http://pkg.example.com/release example.com
```

Example 3–26 Specifying the Preferred Publisher

Use the `-P` option to specify a publisher as the preferred publisher. The specified publisher moves to the top of the search order. You can specify the `-P` option when you add a publisher or you can modify an existing publisher.

```
# pkg set-publisher -P example.com
```

Example 3–27 Enabling or Disabling a Publisher

Use the `-d` option to disable a publisher. The preferred publisher cannot be disabled. A disabled publisher is not used in package operations such as list and install. You can modify the properties of a disabled publishers.

Use the `-e` option to enable a publisher.

```
# pkg set-publisher -d example2.com
```

Example 3–28 Specifying Publisher Stickiness

Use the `--non-sticky` option to make a publisher not sticky. A newly-added publisher is sticky by default. If a publisher is not sticky, then a package that was installed from this publisher could be updated from another publisher.

Use the `--sticky` option to make a publisher sticky.

```
# pkg set-publisher --non-sticky example.com
```

Example 3–29 Changing the Publisher Search Order

The preferred publisher is first in the search order. A newly-added publisher is last in the search order by default. The publisher search order is used to find packages to install. The publisher search order is used to find packages to update if the publisher that the package was originally installed from is non-sticky.

In the following example, the `example2.com` publisher is set to be searched before the `example1.com` publisher. If the `example1.com` publisher is the current preferred publisher, the `example2.com` publisher becomes the preferred publisher.

```
# pkg set-publisher --search-before example1.com example2.com
```

In the following example, the `example2.com` publisher is set to be searched after the `example1.com` publisher. If the `example2.com` publisher is the current preferred publisher, the `example1.com` publisher becomes the preferred publisher.

```
# pkg set-publisher --search-after example1.com example2.com
```

Example 3–30 Adding and Removing a Publisher Mirror

Use the `-m` option to add a URI as a mirror for the specified publisher. See "Repository Origins and Mirrors" on page 11 for an explanation of the difference between an origin and a mirror. Use the `-M` option to remove a URI as a mirror for the specified publisher.

```
# pkg set-publisher -m http://pkg.example.org/release example.com
$ pkg publisher
PUBLISHER                    TYPE    STATUS   URI
example.com     (preferred)  origin  online   http://pkg.example.com/release
example.com     (non-sticky) mirror  online   http://pkg.example.org/release
```

Example 3–31 Specifying SSL Keys and Certificates

Use the `-k` option to specify the client SSL key. Use the `-c` option to specify the client SSL certificate.

```
# pkg set-publisher -k /root/creds/example.key \
-c /root/creds/example.cert example.com
```

Use the `--approve-ca-cert` option to add the specified certificate as a CA certificate that is trusted. The hashes of the user approved CA certificates are listed in the output of the `publisher` subcommand for this publisher. See Example 3–24.

```
# pkg set-publisher --approve-ca-cert /tmp/example_file.pem example.com
```

Use the `--revoked-ca-cert` option to treat the specified certificate as revoked. The hashes of the user revoked CA certificates are listed in the output of the `publisher` subcommand for this publisher.

```
# pkg set-publisher --revoked-ca-cert a12345 example.com
```

Use the `--unset-ca-cert` option to remove the specified certificate from the list of approved and the list of revoked certificates.

```
# pkg set-publisher --unset-ca-cert a12345 example.com
```

Example 3–32 Changing a Publisher Origin URI

To change the origin URI for a publisher, add the new URI and remove the old URI. Use the `-g` option to add a new origin URI. Use the `-G` option to remove the old origin URI.

```
# pkg set-publisher -g http://pkg.example.com/support \
-G http://pkg.example.com/release example.com
```

Installing Signed Packages

If you are installing signed packages, set the image properties and publisher properties described in this section to verify package signatures.

Image Properties for Signed Packages

Configure the following image properties to use signed packages.

`signature-policy`
The value of this property determines what checks will be performed on manifests when installing a package into this image. The final policy applied to a package depends on the combination of image policy and publisher policy. The combination will be at least as strict as the stricter of the two policies taken individually. The following values are valid for this property.

`ignore`	Ignore signatures for all manifests.
`verify`	Verify that all manifests with signatures are validly signed, but do not require all installed packages to be signed.
`require-signatures`	Require that all newly installed packages have at least one valid signature. The `pkg fix` and `pkg verify` commands will also warn if an installed package does not have a valid signature.
`require-names`	Follow the same requirements as `require-signatures` but also require that the strings listed in the `signature-required-names` image property appear as a common name of the certificates used to verify the chains of trust of the signatures.

`signature-required-names`
The value of this property is a list of names that must be seen as common names of certificates while validating the signatures of a package.

`trust-anchor-directory`
The value of this property is the pathname of the directory that contains the trust anchors for the image. This path is relative to the image.

Publisher Properties for Signed Packages

Configure the following publisher properties to use signed packages from a particular publisher.

`signature-policy`	The function of this property is identical to the function of the `signature-policy` image property except that this property only applies to packages from the specified publisher.
`signature-required-names`	The function of this property is identical to the function of the `signature-required-names` image property except that this property only applies to packages from the specified publisher.

Configure Package Signature Properties

Use the `set-property`, `add-property-value`, `remove-property-value`, and `unset-property` subcommands to configure package signature properties for this image.

Use the `--set-property`, `--add-property-value`, `--remove-property-value`, and `--unset-property` options of the `set-publisher` subcommand to specify signature policy and required names for a particular publisher.

EXAMPLE 3–33 Requiring All Signed Packages To Be Verified

Configure this image to verify all signed packages.

```
# pkg set-property signature-policy verify
```

EXAMPLE 3–34 Requiring All Packages To Be Signed

Configure this image to require all packages to be signed. Also require the string "oracle.com" to be seen as a common name for one of the certificates in the chain of trust.

```
# pkg set-property signature-policy require-names oracle.com
```

EXAMPLE 3–35 Requiring All Packages From a Specified Publisher To Be Signed

Configure this image to require that all packages installed from the publisher `example.com` must be signed.

```
# pkg set-publisher --set-property signature-policy=require-signatures example.com
```

EXAMPLE 3–36 Adding a Required Signature Name

This example adds the string `trustedname` to the image's list of common names that must be seen in a signature's chain of trust to be considered valid.

```
# pkg add-property-value signature-require-names trustedname
```

EXAMPLE 3–37 Removing a Required Signature Name

This example removes the string `trustedname` from the image's list of common names that must be seen in a signature's chain of trust to be considered valid.

```
# pkg remove-property-value signature-require-names trustedname
```

EXAMPLE 3–38 Adding a Required Signature Name for a Specified Publisher

This example adds the string `trustedname` to the `example.com` publisher's list of common names that must be seen in a signature's chain of trust to be considered valid.

EXAMPLE 3–38 Adding a Required Signature Name for a Specified Publisher *(Continued)*

```
# pkg set-publisher --add-property-value signature-require-names=trustedname example.com
```

CHAPTER 4

Creating and Managing Images

IPS enables you to create and manage images on your Oracle Solaris system.

Why Create an Image?

An image is a location on your system where packages and their associated files, directories, links, and dependencies can be installed.

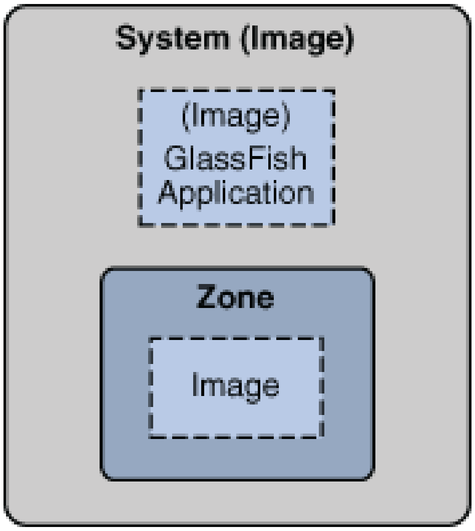
An image can be one of three types:

- Full images are capable of providing a complete system. An installed Oracle Solaris system is a full image.
- Partial images are linked to a full image (the parent image), but do not provide a complete system on their own.
- User images contain only relocatable packages.

After you have completed an installation of the Oracle Solaris OS, the root file system and its contents are contained in a full image. You might want to create new images to provide logical separation between different software applications.

The following figure shows partial images within a full system image.

FIGURE 4–1 Partial Images Within a System Image



See the Zones section in the *System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones* guide to learn about zones.

Best Practices for Creating Images

- **Where to create an image.** Do not create multiple images in the same directory. Do not create an image under root (/).
- **Which packages to use.** Publisher and package repository information must be set when creating an image. Use the `-p` option to provide this information as shown in "Create an Image" on page 41.
- **Which image to change.** When you have multiple images on your system, use the `PKG_IMAGE` environment variable or the `-R` option to specify which image to modify.
- **Multiple versions of a package.** IPS assumes that only one version of a package is installed in a particular image. To install multiple versions of the same package, the name of the package must include the version number, and dependencies must reflect the version number and bind to the appropriate package version.

Create an Image

Use the pkg image-create command to create an area where packages can be managed and to specify the publisher for that image. A user image is created by default.

- Ensure that you have write privileges to the directory where you are attempting to create an image.
- Use the -p option or the --preferred option to specify a preferred publisher URI. After an image has been created successfully, the preferred publisher's catalog is retrieved.

EXAMPLE 4–1 Creating a Full Image

This example specifies the -F option to create a full image. The -p option sets example.com as the preferred publisher, with repository located at http://pkg.example.com:10000. The new image is created in the directory /aux0/example_root.

```
# pkg image-create -F -p example.com=http://pkg.example.com:10000 \
/aux0/example_root
```

Display, Set, and Remove Image Properties

See "IMAGE PROPERTIES" in "pkg(1) Man Page" on page 46 for descriptions of image properties.

EXAMPLE 4–2 Displaying the Values of Image Properties

Use the pkg property command to view the properties of an image.

```
$ pkg property
PROPERTY                       VALUE
ca-path                        /etc/openssl/certs
flush-content-cache-on-success False
mirror-discovery               False
preferred-publisher            solaris
publisher-search-order         ['solaris']
send-uuid                      True
signature-policy               ignore
signature-required-names       []
trust-anchor-directory         etc/certs/CA
```

Most of these properties are managed with the set-property, unset-property, add-property-value, and remove-property-value subcommands of the pkg command. The preferred-publisher and publisher-search-order properties are set with the pkg set-publisher command. See Example 3–26 and Example 3–29.

EXAMPLE 4–3 Setting the Value of an Image Property

Use the pkg set-property command to set the value of an image property or add and set a property.

This example sets the value of the mirror-discovery property.

```
# pkg set-property mirror-discovery True
$ pkg property -H mirror-discovery
mirror-discovery True
```

EXAMPLE 4–4 Resetting the Value of an Image Property

Use the pkg unset-property command to reset the values of the specified properties to their default values.

This example unsets the mirror-discovery property.

```
# pkg unset-property mirror-discovery
$ pkg property -H mirror-discovery
mirror-discovery False
```

Update an Image

Use the pkg update command to update all installed packages to the latest version. The behavior of the pkg update command is similar to the behavior of Update Manager. See "Using Update Manager" on page 18.

EXAMPLE 4–5 Updating the Active Boot Environment

This command updates all installed packages in the current image to the newest version allowed by the constraints imposed by installed packages and publisher configuration.

```
# pkg update --accept
```

By default, each package is updated from the publisher that provided the current installed version. You can control the publisher that provides packages by specifying publisher stickiness and search order. See Example 3–28 and Example 3–29.

If you specify the --accept option, you accept the terms of the licenses of the packages that are updated. If you do not specify the --accept option, and any package licenses require acceptance, no packages are updated.

If particular packages are updated, such as certain kernel components, a new BE is created. See "Images and Boot Environments" on page 11.

EXAMPLE 4–6 Updating an Image in an Inactive Boot Environment

This example updates an image in a BE that is mounted at /mnt.

EXAMPLE 4–6 Updating an Image in an Inactive Boot Environment *(Continued)*

```
# beadm mount OracleSolaris-1 /mnt
# pkg -R /mnt update --accept --be-name OracleSolaris-3
```

◆ ◆ ◆ APPENDIX A

IPS Command Reference

This reference lists and briefly describes commands in the Image Packaging System. This appendix also shows the entire man page for the pkg(1) command.

IPS Commands

The pkg(5) man page describes the Image Packaging System. The pkg(5) man page defines an IPS package and discusses IPS package names, versions, and FMRIs. The pkg(5) man page defines an IPS action and describes many different kinds of actions. The pkg(5) man page describes actuators, properties, and image policies.

The Image Packaging System software provides the following commands:

`packagemanager(1)`	Opens the Package Manager tool, which is the IPS GUI. See "Using Package Manager" on page 15 for more information.
`pkg(1)`	Lists, searches, installs, updates, and uninstalls packages. Manages package publishers. See the following sections for more information: ■ Chapter 3, "Working With Packages" ■ Chapter 4, "Creating and Managing Images" ■ "pkg(1) Man Page" on page 46
`pkg.depotd(1M)`	Creates package repositories and manages or provides network access to existing package repositories.
`pkgdepend(1)`	Analyzes package dependencies. Used by package publishers to generate and resolve dependencies for packages.
`pkgdiff(1)`	Compares two package manifests and reports differences.
`pkgfmt(1)`	Formats a package manifest, sorting actions by type and placing key attributes first.

`pkglint(1)`	Performs verification operations on IPS package manifests, including tests for duplicate actions, missing attributes, and unusual file permissions.
`pkgmogrify(1)`	Provides for the automated editing of package manifests to simplify the typical transformations needed when automating software builds and package republication.
`pkgrecv(1)`	Retrieves the contents of a package from a package repository. The retrieved package can then be modified and republished using the `pkgsend` command.
`pkgrepo(1)`	Creates and manages package repositories.
`pkgsend(1)`	Publishes packages to a package repository.
`pkgsign(1)`	Adds a signature action, with specified key and certificates, to a package in a repository without changing the timestamp of the package.
`pm-updatemanager(1)`	Opens the Package Manager tool, which is the IPS GUI. See "Using Package Manager" on page 15 for more information.

pkg(1) Man Page

```
NAME
     pkg - image packaging retrieval client

SYNOPSIS
     /usr/bin/pkg [options] command [cmd_options] [operands]

     /usr/bin/pkg install [-nvq] [--accept] [--licenses] [--no-index]
         [--no-refresh] [--deny-new-be | --require-new-be] [--be-name name]
         pkg_fmri_pattern ...
     /usr/bin/pkg uninstall [-nrvq] [--no-index]
         [--deny-new-be | --require-new-be] [--be-name name]
         pkg_fmri_pattern ...

     /usr/bin/pkg update [-fnvq] [--accept] [--be-name name]
         [--deny-new-be | --require-new-be] [--licenses] [--no-index]
         [--no-refresh]

     /usr/bin/pkg refresh [--full] [publisher ...]

     /usr/bin/pkg contents [-Hmr] [-a attribute=pattern ...]
         [-o attribute ...] [-s sort_key] [-t action_type ...]
         [pkg_fmri_pattern ...]
     /usr/bin/pkg info [-lr] [--license] [pkg_fmri_pattern ...]
     /usr/bin/pkg list [-Hafnsuv] [--no-refresh] [pkg_fmri_pattern ...]
     /usr/bin/pkg search [-HIaflpr] [-o attribute ...] [-s repo_uri]
         query
     /usr/bin/pkg verify [-Hqv] [pkg_fmri_pattern ...]
     /usr/bin/pkg fix [--accept] [--licenses] [pkg_fmri_pattern ...]

     /usr/bin/pkg image-create [-FPUfz] [--force]
         [--full|--partial|--user] [--zone] [-k ssl_key] [-c ssl_cert]
         [--no-refresh] [--variant <variant_spec>=<instance> ...]
         [-g uri|--origin=uri ...] [-m uri|--mirror=uri ...]
         [--facet <facet_spec>=[True|False] ...]
         (-p|--publisher) [<name>=]<repo_uri> dir

     /usr/bin/pkg variant [-H] [<variant_spec>]
     /usr/bin/pkg change-variant [-nvq] [--accept]
         [--deny-new-be | --require-new-be] [--be-name name]
         [--licenses] <variant_spec>=<instance> ...

     /usr/bin/pkg facet [-H] [<facet_spec>]
     /usr/bin/pkg change-facet [-nvq] [--accept] [--be-name name]
         [--deny-new-be | --require-new-be]
         [--licenses] <facet_spec>=[True|False|None] ...

     /usr/bin/pkg set-property propname propvalue
     /usr/bin/pkg add-property-value propname propvalue
     /usr/bin/pkg remove-property-value propname propvalue
     /usr/bin/pkg unset-property propname ...
     /usr/bin/pkg property [-H] [propname ...]

     /usr/bin/pkg set-publisher [-Ped] [-k ssl_key] [-c ssl_cert]
         [-g origin_to_add|--add-origin=origin_to_add ...]
         [-G origin_to_remove|--remove-origin=origin_to_remove ...]
         [-m mirror_to_add|--add-mirror=mirror_to_add ...]
         [-M mirror_to_remove|--remove-mirror=mirror_to_remove ...]
         [-p repo_uri] [--enable] [--disable] [--no-refresh]
         [--reset-uuid] [--non-sticky] [--sticky]
         [--search-after=publisher] [--search-before=publisher]
         [--approve-ca-cert=path_to_CA]
         [--revoke-ca-cert=hash_of_CA_to_remove]
         [--unset-ca-cert=hash_of_CA_to_remove]
         [--set-property name_of_property=value]
         [--add-property-value name_of_property=value_to_add]
         [--remove-property-value name_of_property=value_to_remove]
         [--unset-property name_of_property_to_delete]
         [publisher]
     /usr/bin/pkg unset-publisher publisher ...
     /usr/bin/pkg publisher [-HPn] [publisher ...]

     /usr/bin/pkg history [-Hl] [-n number]
     /usr/bin/pkg purge-history

     /usr/bin/pkg rebuild-index

     /usr/bin/pkg version
     /usr/bin/pkg help

DESCRIPTION
     pkg is the retrieval client for the image packaging system.  With
     a valid configuration, pkg can be invoked to create locations for
     packages to be installed, called 'images', and install packages
     into those images.  Packages are published by publishers, who may
     make their packages available at one or more repositories.  pkg,
     then, retrieves packages from a publisher's repository and
     installs them into an image.

     A publisher is a forward domain name that can be used to identify a
     person, group of persons, or an organization as the source of one or
     more packages.  The name of a publisher does not have to be contained
     within the URIs that identify the locations of publisher repositories.
     For example, the name of a publisher might be "example.com", but its
     repositories may be hosted at "example.org" or "example.net".

     A repository is a location where clients can publish and retrieve
     package content (files contained within the package such as programs,
     documents, etc.) and metadata (information about the package such as
     its name, description, etc.).  As an example, a publisher named
     "example.org" may have their repository located at the URI
     "http://example.org/repository".

     pkg can also uninstall packages, refresh publisher metadata (such as
     catalogs), validate package installation in an image, and query the
     image for various tokens.  These queries can also be made of pkg(5)
     repositories.

     Images can be of three types: full images, capable of providing a
     complete system; partial images, which are linked to a full image
     (parent image), but do not provide a complete system on their own;
     and user images, which contain only relocatable packages.  (See
     NOTES on user images.)

     pkg(1) attempts to determine, based on its working directory, in
     what image it has been invoked.  If no image metadata can be found
     in the parent directories, the invocation will fail.

OPTIONS
     The following options are supported:

     -R dir
          Operate on the image rooted at dir, rather than the one discovered
          automatically.

     --help or -?
          Displays a usage message.

SUBCOMMANDS
     The following subcommands are supported:

     image-create [-FPUfz] [--force] [--full|--partial|--user] [--zone]
       [-k ssl_key] [-c ssl_cert] [--no-refresh]
       [--variant <variant_spec>=<instance> ...]
       [-g uri|--origin=uri ...] [-m uri|--mirror=uri ...]
       [--facet <facet_spec>=[True|False] ...]
       (-p|--publisher) [<name>=]<repo_uri> dir
          Create, at location given by dir, an image suitable for package
          operations.  The default image type is user, as given by the -U
          (--user) option.  The image type may be set to a full image (-F
          or --full) or to a partial image (-P or --partial) linked to the
          full image enclosing the given dir path.  Additional origins can
          be specified using -g or --origin, while additional mirrors can
          be specified using -m or --mirror.

          A package repository URI must be provided using the -p or
          --publisher option.  If a publisher name is also provided, then
          only that publisher will be added when the image is created.  If
          a publisher name is not provided, then all publishers known by the
          specified repository will be added to the image.  An attempt to
          retrieve the catalog associated with this publisher will be made
          following the initial creation operations.

          For publishers using client SSL authentication, a client key and
          client certificate may be registered via the -c and -k options,
          and will be used for all publishers added during image creation.

          If the image is to be run within nonglobal zone context, then
          the -z (--zone) option can be used to set an appropriate filter.

          With -f (--force), force the creation of an image over an existing
          image.  This option should be used with care.

          With --no-refresh, do not attempt to contact the repositories for
          the image's publishers to retrieve publisher metadata (e.g.
          catalogs).

          With --variant, set the specified variant to the indicated value.

          With --facet, set the specified facet to the indicated value.

     refresh [--full] [publisher ...]
          Retrieve updates to the metadata (e.g. catalogs) for each publisher
          specified.  When given no arguments, retrieves updates for each
          publisher registered within the image.

          With --full, retrieve all publisher metadata instead of attempting an
          incremental update.

     install [-nvq] [--accept] [--licenses] [--no-index] [--no-refresh]
       [--deny-new-be | --require-new-be] [--be-name] pkg_fmri_pattern ...

          Installs and updates packages to the newest version that match
          pkg_fmri_pattern allowed by the packages installed in the
          image.

          Some configuration files may be renamed or replaced during the
          install process.  For more information on how the package system
          determines which files to preserve, and how they are preserved
          during package operations, see "File Actions" in pkg(5).

          With the -n option, execute the requested operation but make
          no persistent changes to the image.

          With the -v option, issue verbose progress messages during
          the requested operation.  With the -q option, issue no
          progress messages during the requested operation.

          With --accept, you indicate that you agree to and accept the
          terms of the licenses of the packages that are updated or
          installed.  If you do not provide this option, and any
          package licenses require acceptance, the operation will
          fail.

          With --licenses, display all of the licenses for the
          packages that will be installed or updated as part of this
          operation.

          With --no-index, do not update the search indices after the
          operation has completed successfully.

          With --no-refresh, do not attempt to contact the
          repositories for the image's publishers to retrieve
          publisher metadata (e.g. catalogs).

          With --be-name, rename the newly created boot environment to
          be the argument given.  This option is only valid if a new
          boot environment is created during the operation.  See also
          beadm(1m).

          With --require-new-be, always create a new boot environment.
          Without this option, a boot environment is created
          automatically if needed.

          With --deny-new-be, disallow creation of a new boot
          environment; the operation will not be performed if
          a new boot environment is required.

     uninstall [-nrvq] [--no-index] [--deny-new-be | --require-new-be]
       [--be-name name] pkg_fmri_pattern ...

          Removes installed packages that match pkg_fmri_pattern.

          With -r, recursively uninstall any packages that contain
          'require' dependencies on the initial package.  (Packages
          containing 'optional' or 'incorporate' dependencies are
          not included in the removal.)

          For all other options, refer to the install command above
          for usage and their effects.

     update [-fnvq] [--accept] [--be-name name] [--licenses]
       [--no-index] [--no-refresh] [--deny-new-be | --require-new-be]
       [pkg_fmri_pattern ...]

          With no arguments, or if '*' is one of the patterns provided,
          update all installed packages in the current image to the newest
          version allowed by the constraints imposed on the system by
          installed packages and publisher configuration.

          If pkg_fmri_pattern is provided, update will replace packages
          that are installed, and that match pkg_fmri_pattern, with the
          newest version allowed by the pattern(s) and the constraints
          imposed on the system by installed packages and publisher
          configuration.  Versions older or newer than what is already
          installed may be specified to perform in place downgrades or
          upgrades of specific packages.  Please note that updating
          specific packages across package rename or obsolete boundaries
          is not supported.
          Any preserved configuration files that are part of packages to
          be downgraded by update and that have been changed since the
          original version was installed will be renamed using the
          extension '.update'.  For more information on how the package
          system determines which files to preserve, and how these files
          are preserved during package upgrades, see "File Actions" in
          pkg(5).

          With the -f option, don't execute the client up to date check
          when updating all installed packages.

          For all other options, refer to the install command above for
          usage and their effects.

     info [-lr] [--license] [pkg_fmri_pattern ...]
          Display information about packages in a human-readable form.
          Multiple FMRI patterns may be specified; with no patterns,
          display information on all installed packages in the image.

          With -l, use the data available from locally installed packages.
          This is the default.

          With -r, retrieve the data from the repositories of the image's
          configured publishers.  Note that you must specify one or more
          package patterns in this case.

          With --license, print out the license text(s) for the package.
          This may be combined with -l or -r.

     contents [-Hmr] [-a attribute=pattern ...] [-o attribute ...]
       [-s sort_key] [-t action_type ...] [pkg_fmri_pattern ...]
          Display the contents (action attributes) of packages in the
          current image.  By default, only the path attribute is displayed,
          but the attribute set may be determined with the -o option.  The
          -o option may be specified multiple times, or multiple attributes
          may be specified as the argument to one -o option by separating
          the attribute names with commas.  Only actions which have the
          requested attributes will be displayed.  The -m option may
          also be used, as a shorthand for '-Ho action.raw'.

          The -a option allows you to limit the output to those actions
          which have an attribute named in the option argument the value of
          which matches the (glob) pattern in the option argument
          (following the attribute name with an equals sign).  If multiple
          -a options are given, then actions matching any of them will be
          displayed.

          The -s option specifies the attribute by which the listing should
          be sorted.

          The -t option limits the action types which will be displayed.

          The -H option causes the headers to be omitted.

          The -r option retrieves the requested data from the repositories
          of the image's configured publishers.  This option is intended
          to be used when the named packages are not already installed.
          With no arguments, the output includes all installed packages.
          Alternatively, multiple FMRI patterns may be specified, which
          restricts the display to the contents of the matching packages.
          When using -r, one or more pkg_fmri_patterns must be specified.

          Several special "pseudo" attribute names are available for
          convenience:

          action.hash           Corresponds to the value of the action's
                                hash, if the action carries a payload.

          action.key            Corresponds to the value of the action's
                                key attribute.  For example, for a file
                                action, this is the path to the file.

          action.name           Corresponds to the name of the action.
                                For example, for a file action, this is
                                "file"

          action.raw            Corresponds to the complete contents of
                                the action as represented in the package
                                manifest.  This corresponds to the
                                lines of output of 'pkg contents -m'

          pkg.fmri              Corresponds to the full form FMRI of the
                                package containing the action, such as
                                pkg://extra/virtualbox@3.0.0,5.11-0.101:20090702T175410Z

          pkg.name              Corresponds to the name of the package
                                containing the action, such as "SUNWcs"

          pkg.publisher         Corresponds to the publisher of the
                                the package containing the action, such
                                as "opensolaris.org"

          pkg.shortfmri         Corresponds to the short form FMRI of the
                                package containing the action, such as
                                pkg://opensolaris.org/SUNWzone@0.5.11-0.79

          The contents and search subcommands are related: both are used to
          query the system for the contents of packages.  The contents
          subcommand displays actions in one or more packages, filtering
          the output based on the options chosen by the user.  The search
          subcommand approaches the query from the other direction, looking
          for packages which contain a user-supplied token.

          Each subcommand is capable of formulating some queries of which
          the other is capable.  Care should be taken in choosing the
          subcommand, as a given query may be more naturally formulated in
          one than in the other.

     search [-HIaflpr] [-o attribute ...] [-s repo_uri] query
          Search for matches to the query, and display the results.
          Which tokens are indexed are action-dependent, but may
          include content hashes and pathnames.  By default, queries are
          interpreted as a series of terms to be matched exactly.  The
          '?' and '*' characters can be used as glob(3C)-style
```

```
wildcards, allowing more flexible query matches.

With -H, omit the headers.

With -I, use a case-sensitive search.

By default, and with -a, perform the search and display information
about the matching actions.

By default, search prunes results from packages older than the
currently installed version and from package versions excluded by
current incorporations.  Use -f to show all results, regardless of
package version.

With -l, search the image’s installed packages.

With -o, the columns of the results may be controlled.  The
-o option may be specified multiple times, or multiple attributes
may be specified as the argument to one -o option by separating
the attribute names with commas.  In addition to the "pseudo"
attributes outlined above, more are defined for search results:

search.match          Corresponds to the string which matched the
                      search query.

search.match_type     Corresponds to the attribute which contained
                      the string that matched the search query.

With -p, display packages which have some actions that match each
query term.  Using this option is equivalent to putting ’<>’ around
each term  in the query.  (For a description of the ’<>’ operator,
please see below.)

By default, and with -r, search the repositories corresponding
to the image’s publishers.

With -s, search the pkg(5) repository located at the given URI.
This may be specified multiple times.

Both -l and -r (or -s) may be specified together, in which case both
local and remote searches will be performed.

In addition to simple token matching and wildcard search, a more
complicated query language is supported.  Phrases may be searched for
by using ’ or ".  Note: Please make sure to take your shell into
account so that pkg actually sees the ’ or ".

Boolean search using AND and OR is supported.  Field, or structured,
queries are supported.  The syntax for these is
pkg_name:action_type:key:token.  Missing fields are implicitly
wildcarded.  A search for :basename:pkg would match all actions
types in all packages with a key of basename and which matched
the token ’pkg’.  Explicit wildcards are supported in the pkg_name
and token fields, action_type and key must match exactly.

To convert actions to the packages which contain those actions,
use ’<>’.  With the -a option, Searching for ’token’ results in
information about the actions matching token, while searching for
```

```
          '<token>' results in a list of packages containing actions which
          matched token.

     list [-Hafnsuv] [--no-refresh] [pkg_fmri_pattern ...]
          Display a list of packages in the current image, including
          state and other information.  By default, package variants
          for a different architecture or zone type are excluded.
          The usual output is in four columns:

            NAME (PUBLISHER)                  VERSION          STATE      UFOXI
            SUNWcs                            0.5.11-0.126     installed  -----
            web/firefox/plugin/flash (extra)  10.0.32.18-0.111 installed  -----

          The first column contains the name of the package.  If the publisher
          from which it is installed (or available, if not installed) is not
          the preferred publisher, then the publisher name is listed in
          parentheses after the package name.  The second column contains the
          release and branch versions of the package (see pkg(5)).  The third
          column contains the state of the package as it exists on the system.
          Possible values are "installed" and "known".  The last column
          contains a set of flags that show how the package relates to other
          packages:

              - a "u" in the "U" column shows that a newer version is
                available, although it may not be possible to install
                this newer version due to package dependencies or
                constraints;

              - an "f" in the "F" column shows that this version has
                been frozen (not implemented);

              - an "o" in the "O" column shows that it is obsolete,
                while an "r" shows that it has been renamed (a form of
                obsoletion);

              - an "x" in the "X" column shows that it is prevented from
                being installed because some other package has excluded
                it (not implemented); and

              - an "i" in the "I" column shows that it has been
                constrained by an incorporation (not implemented).

          With -H, omit the headers from the listing.

          With -a, list installed packages and the newest version of
          packages that are available for installation.  Packages are
          considered to be available for installation if they are
          allowed by the installed incorporations and by the image's
          variants.  If one or more patterns are specified, then the
          newest version matching the specified pattern and is also
          allowed by any installed incorporations and the image's
          variants will be listed.  Without -a, list only installed
          packages.

          With -f and -a, list all versions of all packages for all
          variants regardless of incorporation constraints or installed
          state.
          With -n, display the newest versions of all known packages,
          regardless of installed state.

          With -s, display a one-line short-form giving the package name
          and description.  This option may be used with -a, -n, -u or
          -v.

          With -u, list only packages with newer versions available.

          With -v, show full package FMRIs, including publisher and
          complete version, all in the first column (the VERSION column
          disappears).  This option may be used with -a, -n, or -u.

          With --no-refresh, do not attempt to contact the repositories
          for the image's publishers to retrieve publisher metadata (e.g.
          catalogs).

     verify [-Hqv] [pkg_fmri_pattern ...]
          Validate the installation of packages in the current image.
          Please note that verification of installed package content is
          based on a custom content analysis that may return different
          results than those of other programs.

          With -H, omit the headers from the verification output.

          With -q, print nothing, but return failure if there are any
          fatal errors.

          With -v, include informational messages regarding packages.

     variant [-H] [<variant_spec> ...]
          Display the current values of all variants, or with arguments,
          only the variants specified

          With -H, omit the headers from the listing.

     change-variant [-nvq] [--accept] [--be-name name] [--licenses]
       <variant_spec>=<instance> ...
          Change the specified variants in the current image.

          With the -n option, plan the requested operation but make
          no actual changes.

          With the -v option, issue verbose progress messages during the
          requested operation.  With the -q option, be completely silent.

          With --accept, you indicate that you agree to and accept the
          terms of the licenses of the packages that are updated or
          installed.  If you do not provide this option, and any package
          licenses require acceptance, the operation will fail.

          With --licenses, display all of the licenses for the packages that
          will be installed or updated as part of this operation.

          With --be-name, rename the newly created boot environment to be the
          argument given.  This option is only valid if a new boot environment
          is created during image update.  See also beadm(1m).
          With --require-new-be, always create a new boot environment.  Without
          this option, a new boot environment is only created if needed.

          With --deny-new-be, disallow creation of a new boot environment;
          the operation will not be performed if a new boot environment is
          required.

     facet [-H] [<facet_spec> ...]
          Without arguments, displays the current values of all facets.  With
          argument(s), evaluate if each facet would be true or false and print
          the result.

          With -H, omit the headers from the listing.

     change-facet [-nvq] [--accept] [--be-name name] [--licenses]
       <facet_spec>=[True|False|None] ...

          Change the specified facets in the current image.

          With the -n option, plan the requested operation but make
          no actual changes.

          With the -v option, issue verbose progress messages during the
          requested operation.  With the -q option, be completely silent.

          With --accept, you indicate that you agree to and accept the
          terms of the licenses of the packages that are updated or
          installed.  If you do not provide this option, and any package
          licenses require acceptance, the operation will fail.

          With --licenses, display all of the licenses for the packages that
          will be installed or updated as part of this operation.

          With --be-name, rename the newly created boot environment to be the
          argument given.  This option is only valid if a new boot environment
          is created during the operation.  See also beadm(1m).

          With --require-new-be, always create a new boot environment.  Without
          this option, a new boot environment is only created if needed.

          With --deny-new-be, disallow creation of a new boot environment;
          the operation will not be performed if a new boot environment is
          required.

          Facets may be set to True or False.  Setting one to None removes
          that facet specification from the current image.

     fix [--accept] [--licenses] [pkg_fmri_pattern ...]
          Fix any errors reported by pkg verify.  Please note that
          verification of installed package content is based on a
          custom content analysis that may return different results
          than those of other programs.

          With --accept, you indicate that you agree to and accept the
          terms of the licenses of the packages that are updated or
          installed.  If you do not provide this option, and any package
          licenses require acceptance, the operation will fail.
          With --licenses, display all of the licenses for the packages that
          will be installed or updated as part of this operation.

     set-property propname propvalue
          Update an existing image property or add a new image property;
          except for preferred-publisher, which can only be changed using
          set-publisher.

     add-property-value propname propvalue
          Add a value to an existing image property or add a new image property;
          except for preferred-publisher, which can only be changed using
          set-publisher.

     remove-property-value propname propvalue
          Remove a value from an existing image property; except for
          preferred-publisher, which can only be changed using set-publisher.

     unset-property propname ...
          Remove an existing image property or properties; except for
          preferred-publisher, which can only be changed using
          set-publisher.

     property [-H] [propname ...]
          Display image property information.  With no argument, display the
          names and values for all image properties.  If a specific list of
          property names is requested, display the names and values for those
          properties.

          With -H, omit the headers from the listing.

     set-publisher [-Ped] [-k ssl_key] [-c ssl_cert]
       [-g origin_to_add|--add-origin=origin_to_add ...]
       [-G origin_to_remove|--remove-origin=origin_to_remove ...]
       [-m mirror_to_add|--add-mirror=mirror_to_add]
       [-M mirror_to_remove|--remove-mirror=mirror_to_remove]
       [-p repo_uri] [--enable] [--disable] [--no-refresh]
       [--reset-uuid] [--non-sticky] [--sticky]
       [--search-after=publisher] [--search-before=publisher]
       [--approve-ca-cert path_to_CA]
       [--revoke-ca-cert hash_of_CA_to_remove]
       [--unset-ca-cert hash_of_CA_to_remove]
       [--set-property name_of_property=value]
       [--add-property-value name_of_property=value_to_add]
       [--remove-property-value name_of_property=value_to_remove]
       [--unset-property name_of_property_to_delete]
       [publisher]

          Update an existing publisher or add an additional package
          publisher.  If no options affecting search order are specified,
          new publishers are appended to the search order and are thus
          searched last.

          With -P, set the specified publisher as the preferred
          publisher, i.e.  first in the search order.  When installing
          new packages, this publisher will be searched first.
          Updates to already installed packages will come from the
          same publisher that originally provided the package so long
          as that publisher remains sticky.
          With --non-sticky, specify that higher ranked publishers than
          this one may provide updates to packages originally installed
          from this publisher.

          With --sticky, return to the default behavior of always sourcing
          updates from the same publisher that provided the package originally.

          With --search-before, alter the publisher search order so that
          the publisher being modified is now searched before the specified
          publisher.

          With --search-after, alter the publisher search order so that
          the publisher being modified is now searched after the specified
          publisher.

          With --approve-ca-cert, add the given certificate as a CA certificate
          that is trusted.  The hashes of the user approved CA certificates are
          listed in the output of the detailed pkg publisher view for a
          publisher.

          With --revoked-ca-cert, treat the certificate with the given hash as
          revoked.  The hashes of the user revoked CA certificates are
          listed in the output of the detailed pkg publisher view for a
          publisher.

          With --unset-ca-cert, remove the certificate with the given hash from
          the list of approved and the list of revoked certificates.

          With --set-property, update an existing publisher property or add a
          new publisher property.

          With --add-property-value, add a value to an existing publisher
          property or add a new publisher property.

          With --remove-property-value, remove a value from an existing
          publisher property.

          With --unset-property, remove an existing publisher property.

          With -c and -k, specify client SSL certificate and key respectively.

          With -g (--add-origin), add the URI as an origin for the given
          publisher.  This should be the location of a package repository.

          With -G (--remove-origin), remove the URI from the list of origins
          for the given publisher.

          With --no-refresh, do not attempt to contact the publisher
          specified to retrieve its metadata (e.g. catalog).

          With --reset-uuid, choose a new unique identifier that identifies
          this image to its publisher.

          With -m (--add-mirror), add the URI as a mirror for the given
          publisher.

          With -M (--remove-mirror), remove the URI from the list of mirrors
          for the given publisher.

          With -p, retrieve publisher configuration information from the
          specified repository URI.  If a publisher is specified, then only
          the matching one will be added or updated.  If no publisher is
          specified, all will be added or updated as appropriate.  This option
          may not be combined with the -g, --add-origin, -G, --remove-origin,
          -m, --add-mirror, -M, --remove--mirror, --disable, --enable,
          --no-refresh, or --reset-uuid options.

          With -e (--enable), enable the publisher; with -d (--disable), disable
          the publisher.  A disabled publisher is not used when populating the
          package list or in certain package operations (install, uninstall, and
          update).  However, the properties for a disabled publisher can still
          be set and viewed.  If only one publisher exists, it cannot be
          disabled.

     unset-publisher publisher ...
          Remove the configuration associated with the given publisher
          or publishers.

     publisher [-HPn] [publisher ...]
          Display publisher information.  With no arguments, display
          the list of all publishers, their origin URIs, and mirrors
          in order of search preference.  If specific publishers are
          requested, display the configuration values, including
          mirrors, associated with those publishers.

          With -H, omit the headers from the listing.

          With -P, display only the preferred publisher.

          With -n, display only enabled publishers.

     history [-Hl] [-n number]
          Display the command history of the applicable image.

          With -H, omit the headers from the listing.

          With -l, display log records in long format, which, in addition to
          the standard format, includes the outcome of the command, the time
          the command completed, the version and name of the client used, the
          name of the user who performed the operation, and any errors that
          were encountered while executing the command.

          With -n, display only the specified number of most recent entries.

     purge-history
          Deletes all existing history information.

     rebuild-index
          Rebuilds the index used by 'pkg search'.  This is a recovery operation
          not intended for general use.

     version
          Display a unique string identifying the version of pkg(1).  This
          string is not guaranteed to be comparable in any fashion between
          versions.
```

```
IMAGE PROPERTIES
     The following properties are part of the image and may be set using
     the set-property subcommand.  The values of these properties are
     viewable with the property subcommand.

     ca-path
          (string) A pathname that points to a directory where CA certs are
          kept for SSL operations.  The format of this directory is specific
          to the underlying SSL implementation.  If the administrator
          would like to use an alternate location for trusted CA
          certificates, this value should be changed to point to a
          different directory.  Please see the 'CApath' portions of
          SSL_CTX_load_verify_locations(3openssl) for requirements
          about the CA directory.

          Default value: /etc/openssl/certs

     flush-content-cache-on-success
          (boolean) If this is set to True, the package client will remove
          the files in its content-cache when install or update operations
          complete.  For update operations, the content is removed only
          from the source BE.  When a packaging operation next occurs in
          the destination BE, it will flush its content cache, provided
          this option has not been changed.

          This property may be used to keep the content-cache small on
          systems with limited disk space, but it may cause operations
          to take longer to complete.

          Default value: False

     mirror-discovery
          (boolean)  Mirror-discovery tells the client to discover
          link-local content mirrors using mDNS and DNS-SD.  If this is
          set to True, the client will attempt to download package content
          from mirrors it dynamically discovers.  To run a mirror that
          advertises its content via mDNS, see pkg.depotd(1m).

          Default value: False

     send-uuid
          (boolean)  Send the image's Universally Unique Identifier
          (UUID) when performing network operations.  Although users may
          disable this option, some network repositories may refuse to talk
          to clients that do not supply a UUID.

          Default value: True

     signature-policy
          (string)  Determine what checks will be performed on manifests
          when installing a package into this image.  The final policy
          applied to a  package depends on the combination of image policy
          and publisher policy.  The combination will be at least as strict
          as the stricter of the two policies taken individually.  The
          following are the valid values for this property.

          ignore
```

```
               Ignore signatures for all manifests.
          verify
               Verify that all manifests with signatures are validly
               signed, but do not require all installed packages to be
               signed.
          require-signatures
               Require that all newly installed packages have at least
               one valid signature.  'pkg fix' and 'pkg verify' will also
               warn if an installed package does not have a valid
               signature.
          require-names
               Follow the same requirements as 'require-signatures' but
               also require that the strings listed in the
               'signature-required-names' property appear as a common
               name of the certificates used to verifiy the chains
               of trust of the signatures.

     signature-required-names
          (list of strings)  A list of names which must be seen as common
          names of certificates while validating the signatures of a
          package.

     trust-anchor-directory
          (string)  The pathname of the directory that contains the trust
          anchors for the image.  This path is relative to the image.

PUBLISHER PROPERTIES
     The following properties are part of the image and may be set using
     the set-property option of the set-publisher subcommand.

     signature-policy
          (string)  This property functions identically to the image
          property of the same name except it only applies to packages
          from the particular publisher.

     signature-required-names
          (list of strings)  This property functions identically to the
          image property of the same name except it only applies to
          packages from the particular publisher.

EXAMPLES
     Example 1:  Create a new, full image, with publisher example.com,
     stored at /aux0/example_root.

     $ pkg image-create -F -p example.com=http://pkg.example.com:10000 \
          /aux0/example_root

     Example 2:  Create a new, full image, with publisher example.com,
     that also has an additional mirror, two additional origins and is
     stored at /aux0/example_root.

     $ pkg image-create -F -p example.com=http://pkg.example.com:10000 \
          -g http://alternate1.example.com:10000/ \
          -g http://alternate2.example.com:10000/ \
          -m http://mirror.example.com:10000/ \
          /aux0/example_root

     Example 3:  Install the latest version of the widget package in the
```

```
current image.

$ pkg install application/widget

Example 4:  List the contents of the SUNWzfs package.  Display the
action name, the mode of the file (if defined), the size (if defined),
the path, and the target (if a link).  Limit the action to types dir,
file, link, and hardlink, since specifying the action.name attribute,
which is available for all actions, will display a line for all
actions, which is not desired here.

$ pkg contents -t dir,file,link,hardlink \
     -o action.name,mode,pkg.size,path,target SUNWzfs
NAME  MODE   SIZE PATH                                TARGET
dir   0755        etc
dir   0755        etc/fs
dir   0755        etc/fs/zfs
link              etc/fs/zfs/mount                    ../../../sbin/zfs
link              etc/fs/zfs/umount                   ../../../sbin/zfs
dir   0755        etc/zfs
dir   0755        lib
dir   0755        lib/amd64
link              lib/amd64/libzfs.so                 libzfs.so.1
file  0755 469616 lib/amd64/libzfs.so.1
file  0644  62057 lib/amd64/llib-lzfs.ln
link              lib/libzfs.so                       libzfs.so.1
....

Example 5:  List the contents of SUNWfirefox and SUNWthunderbird,
limiting the display to just the package name and path attributes of
actions whose "path" attribute ends in ".desktop" or ".png".

$ pkg contents contents -o pkg.name,path -a path=\*.desktop \
    -a path=\*.png SUNWfirefox SUNWthunderbird
PKG.NAME        PATH
SUNWfirefox     usr/lib/firefox/chrome/icons/default/default16.png
SUNWfirefox     usr/lib/firefox/chrome/icons/default/default32.png
SUNWfirefox     usr/lib/firefox/chrome/icons/default/default48.png
SUNWfirefox     usr/lib/firefox/icons/document.png
SUNWfirefox     usr/lib/firefox/icons/mozicon128.png
SUNWfirefox     usr/lib/firefox/res/html/folder.png
SUNWfirefox     usr/share/applications/firefox.desktop
SUNWthunderbird usr/share/applications/thunderbird.desktop
SUNWfirefox     usr/share/pixmaps/firefox-icon.png
SUNWthunderbird usr/share/pixmaps/thunderbird-icon.png

Example 6:  Search the package database for the token "bge".

$ pkg search bge
INDEX       ACTION VALUE                   PACKAGE
basename    file   kernel/drv/bge          pkg:/SUNWbge@0.5.11-0.79
driver_name driver bge                     pkg:/SUNWbge@0.5.11-0.79

The token shows up in the package SUNWbge both as the basename for the
file action representing /kernel/drv/bge and as a driver name.

Example 7: Search for installed packages which depend on SUNWipkg.
```

```
$ pkg search -l 'depend::SUNWipkg'
INDEX       ACTION VALUE                   PACKAGE
incorporate depend SUNWipkg@0.5.11-0.111   pkg:/entire@0.5.11-0.111
require     depend SUNWipkg@0.5.11-0.111   pkg:/slim_install@0.1-0.111
require     depend SUNWipkg@0.5.11-0.111   pkg:/SUNWipkg-brand@0.5.11-0.111

Example 8: Search for all incorporate dependencies in installed packages.

$ pkg search -l 'depend:incorporate:'
INDEX       ACTION VALUE                   PACKAGE
incorporate depend BRCMbnx@0.5.11-0.111    pkg:/entire@0.5.11-0.111
incorporate depend BRCMbnx@0.5.11-0.111    pkg:/entire@0.5.11-0.111
....

Example 9:  Add new publisher example.org, with a repository located at
http://www.example.org/repo:

$ pkg set-publisher -g http://www.example.org/repo example.org

Example 10:  Add new publisher example.com, with a secure repository
located at https://secure.example.com/repo, and a key and cert stored
in the directory /root/creds:

$ pkg set-publisher -k /root/creds/example.key \
     -c /root/creds/example.cert -g https://secure.example.com/repo \
     example.com

Example 11:  Add new publisher with a repository located at
/export/repo using automatic configuration:

$ pkg set-publisher -p file:/export/repo

Example 12:  Add new publisher example.org with a repository located
at /export/repo/example.com using manual configuration:

$ pkg set-publisher -g file:/export/repo example.com

Example 13:  Configure an image to verify all signed packages.

$ pkg set-property signature-policy verify

Example 14:  Configure an image to require all packages to be signed and
the string "opensolaris.org" has to be seen as a common name for one of
the certificates in the chain of trust.

$ pkg set-property signature-policy require-names opensolaris.org

Example 15:  Configure an image so that all packages installed from
publisher foo must be signed.

$ pkg set-publisher --set-property signature-policy=require-signatures

Example 16:  Add the string "foo" to the image's list of common names that
must be seen in a signature's chain of trust to be considered valid.

$ pkg add-property-value signature-require-names foo

Example 17:  Remove the string "foo" from publisher test's list of common
```

```
     names that must be seen to validate a signature.

     $ pkg set-publisher --remove-property-value signature-require-names=foo \
         test

     Example 18:  Add the certificate stored in /tmp/example_file.pem as a
     trusted CA certificate for the publisher test.

     $ pkg set-publisher --approve-ca-cert /tmp/example_file.pem

     Example 19:  Revoke the certificate with the hash a12345 for publisher
     test, preventing it from validating any signatures for packages from test.

     $ pkg set-publisher --revoke-ca-cert a12345

     Example 20:  Make pkg forget that the certificate a12345 was ever added or
     revoked by the user.

     $ pkg set-publisher --unset-ca-cert a12345

     Example 21:  Downgrade the installed package foo@1.1 to an older
     version:

     $ pkg update foo@1.0

EXIT STATUS
     The following exit values are returned:

     0     Command succeeded.

     1     An error occurred.

     2     Invalid command line options were specified.

     3     Multiple operations were requested, but only some of them
           succeeded.

     4     No changes were made - nothing to do.

     5     The requested operation cannot be performed on a live
           image.

     6     The requested operation cannot be completed as the licenses
           for the packages being installed or updated have not been
           accepted.

     7     The image is currently in use by another process and cannot
           be modified.

FILES
     A pkg(5) image can be located arbitrarily within a larger file
     system.  In the following, the token $IMAGE_ROOT is used to
     distinguish relative paths.  For a typical system installation,
     $IMAGE_ROOT is equivalent to "/".

     $IMAGE_ROOT/var/pkg        Metadata directory for a full or partial
                                image.
     $IMAGE_ROOT/.org.opensolaris,pkg
                           Metadata directory for a user image.

     Within a particular image's metadata, certain files and directories
     can contain information useful during repair and recovery.  We use
     the token $IMAGE_META to refer to the top-level directory
     containing the metadata.  $IMAGE_META is typically one of the two
     paths given above.

     $IMAGE_META/lost+found     Location of conflicting directories and
                                files moved during a package operation.

     $IMAGE_META/publisher      Contains a directory for each publisher.
                                Each directory stores publisher-specific
                                metadata.

     Other paths within the $IMAGE_META directory hierarchy are Private,
     and are subject to change.

ATTRIBUTES
     See attributes(5) for descriptions of the following attributes:
     ____________________________________________________________
    |       ATTRIBUTE TYPE        |       ATTRIBUTE VALUE       |
    |_____________________________|_____________________________|
    | Availability                | SUNWipkg                    |
    |                             | pkg:/package/pkg            |
    |_____________________________|_____________________________|
    | Interface Stability         | None / Under Development    |
    |_____________________________|_____________________________|

SEE ALSO
     pkgsend(1), pkg.depotd(1M), glob(3C), attributes(5), pkg(5)

NOTES
     The image packaging system is an under-development feature.
     Command names, invocation, formats, and operations are all subject
     to change.  Development is hosted in the OpenSolaris community
     at:

     http://hub.opensolaris.org/bin/view/Project+pkg/

     At present, user images are not restricted to relocatable
     packages--but they will be.

     The pkg(1) command recognizes use of the http_proxy and https_proxy
     environment variables to select a suitable HTTP or HTTPS proxy
     server.  At present, particular care is needed when using local
     repository URIs--such as http://localhost:10000/--with the
     http_proxy environment variable; this behavior may change in a
     future version of image packaging.

     At present, pkg(1), on directory removal, will move unpackaged
     contents of that directory to $IMAGE_META/lost+found.
```

Glossary

action	An installable object such as a file, directory, link, or dependency. Actions are described in the manifest of a package. Every action consists primarily of its name and a key attribute. Together, these refer to a unique object as it follows a version history.
attribute	A setting of a package or an action.
`beadm` utility	The user interface for managing BEs in the Oracle Solaris OS. The `beadm` command is the replacement for the Solaris Live Upgrade commands.
boot environment (BE)	A bootable instance of an Oracle Solaris environment consisting of a set of mount points, file systems, ZFS datasets, and possibly non-global zones. A BE is a collection of mandatory file systems that are critical to the operation of the Oracle Solaris OS. The active BE is the one that is currently booted. An inactive BE is available for booting on the next reboot.
catalog	All packages in an IPS package repository. The packages in a catalog are associated with a specific publisher.
clone	An exact copy. For installation, a clone could be an exact copy of an operating system, a file system, or a volume. This copy is 100 percent compatible with the original.
content	Package files.
dataset	A generic name for the following ZFS entities: clones, file systems, snapshots, or volumes. Each dataset is identified by a unique name in the ZFS namespace.
facet	An optional component of a package such as a locale. Facets appear as tags on IPS actions and affect whether that action is installable. If an action has any facet tags, at least one facet tag must match the selection criteria to install the action.
fault management resource identifier (FMRI)	The identifier for each package. The FMRI includes the package publisher, name, and version. The `pkg` command uses FMRIs, or portions of FMRIs, to operate on packages.
image	A location where packages can be installed.
ISO image	A collection of software that comprises an entire operating system in a single file. An ISO image can be used to create a bootable CD or DVD that can be used to install the system.
metadata	Package manifests and catalogs.

mirror	1. In IPS, a location of a package repository that contains only package content. A repository mirror does not include package metadata. 2. In ZFS, a virtual device that stores identical copies of data on two or more disks. If any disk in a mirror fails, any other disk in that mirror can provide the same data.
origin	The repository location that includes both package metadata and package content. IPS clients access the origin to obtain a publisher's catalog, even when the clients download package content from a mirror.
package	A collection of directories, files, links, drivers, dependencies, groups, users, and license information in a defined format.
package manifest	A file of information that defines a package.
pool	A logical group of devices describing the layout and physical characteristics of the available storage. Space for datasets is allocated from a pool.
preferred publisher	The publisher that is searched first for packages if multiple non-sticky publishers are configured. If more than one publisher is configured, you can use the `pkg` command to set the preferred publisher. You cannot disable or remove the preferred publisher.
publisher	A forward domain name that identifies a person, group of persons, or an organization that publishes one or more packages.
relocatable package	A package that enables a user to specify the installation path of the package.
repository	A location where clients publish and retrieve packages. The location is described by a Universal Resource Identifier (URI). A repository is also called a depot server.
rollback	Reversion to the BE that ran prior to a specific transaction. Use rollback when you are activating an environment and the BE that is designated for booting fails or shows some undesirable behavior. Rollback is known as fallback in Solaris Live Upgrade.
snapshot	A read-only image of a file system or BE at a given point in time. A snapshot is not bootable.
tag	In IPS, a representation of the settings of a file.
Universal Resource Identifier (URI)	The location of a machine or a resource on the internet.
upgrade	An installation that merges updated files with existing files and preserves modifications where possible. An upgrade of the Oracle Solaris OS merges the new version of the Oracle Solaris OS with the existing files on the system's disk or disks. An upgrade saves as many modifications as possible that you have made to the previous version of the OS.
USB image	A collection of software that comprises an entire operating system in a single file. A USB image can be used to create a bootable USB device that can be used to install the system.

variant	A mutually exclusive component of a package such as architecture. Variants appear as tags on IPS actions and affect whether that action is installable. If an action has any variant tags, all variant tags must match the selection criteria to install the action.
ZFS file system	A ZFS dataset of type `filesystem`, that is mounted within the standard system namespace and behaves like other file systems.