**Adding and Updating Software in Oracle®
Solaris 11.2**

ORACLE®

Part No: E36802
July 2014

Contents

Examples

Using This Documentation

- **Overview** – Describes the software installation functions of the Oracle Solaris Image Packaging System (IPS) feature. IPS commands enable you to list and search software packages, install and remove software, and upgrade to a new Oracle Solaris operating system release.
- **Audience** – System administrators who install and manage software and manage system images.
- **Required knowledge** – Experience administering Oracle Solaris systems.

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦   **C H A P T E R   1**

1

Introduction to the Image Packaging System

The Oracle Solaris Image Packaging System (IPS) is a framework that enables you to list and search software packages, install and remove software, and upgrade to a new Oracle Solaris operating system release. IPS commands enable you to restrict which packages can be installed or which versions of packages can be installed.

Image Packaging System

Oracle Solaris 11 software is distributed in IPS packages. IPS packages are stored in IPS package repositories, which are populated by IPS publishers. IPS packages are installed into Oracle Solaris 11 images. A subset of the capabilities that are available through the IPS command-line interface is available through the Package Manager graphical user interface.

IPS tools provide the following capabilities. See "IPS Concepts" on page 11 for definitions of terms such as publisher and repository.

- List, search, install, restrict installation, update, and remove software packages.
- List, add, and remove package publishers. Change publisher attributes such as search priority and stickiness. Set publisher properties such as signature policy.
- Upgrade an image to a new operating system release.
- Create copies of existing IPS package repositories. Create new package repositories. See "Copying and Creating Package Repositories in Oracle Solaris 11.2 ".
- Create and publish packages. See "Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 ".
- Create boot environments and other images.

To use IPS, you must be running the Oracle Solaris 11 OS. To install the Oracle Solaris 11 OS, see "Installing Oracle Solaris 11.2 Systems ".

IPS Concepts

This section defines terms and concepts related to IPS.

IPS Packages

An IPS *package* is defined by a text file called a *manifest*. A package manifest describes package *actions* in a defined format of key/value pairs and possibly a data payload. Package actions include files, directories, links, drivers, dependencies, groups, users, and license information. Package actions represent the installable objects of a package. Actions called `set` actions define package metadata such as classification, summary, and description.

You can search for packages by specifying package actions and action keys. See "Package Content: Actions" in "Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 " or the `pkg`(5) man page for descriptions of package actions.

Incorporation packages and group packages do not deliver content such as files. Incorporation and group packages specify dependencies that help you install sets of related packages.

Incorporation Packages

An *incorporation* package specifies the versions of other packages that can be installed. Installing an incorporation package does not install any other packages. An incorporation package ensures that if you install an `incorporate` dependency package of that incorporation package, only the prescribed version of the dependent package can be installed. For example, if a package specified as an `incorporate` dependency in an installed incorporation package has a version value of 1.4.3, then no version of that package can be installed that has a version value less than 1.4.3 or greater than or equal to 1.4.4. A version of the package with a version value of 1.4.3.7, for example, could be installed.

An incorporation package often specifies many `incorporate` dependencies to define a *surface* in the package version space that is compatible. Packages that contain such sets of `incorporate` dependencies are often called *incorporations*. Incorporations are typically used to define sets of software packages that are built together and are not separately versioned. The `incorporate` dependency is heavily used in Oracle Solaris to ensure that compatible versions of software are installed together.

Packages named as `incorporate` dependencies in the incorporation package might themselves be incorporation packages. In this way, many packages can be affected by an incorporation package even if they are not named in the manifest of the incorporation package. Packages whose installation is affected by an incorporation package are *constrained* by that incorporation package. Updating an incorporation package `B-incorporation` that is incorporated by `A-incorporation` results in also updating `A-incorporation` and all the other packages constrained by `A-incorporation`.

Incorporations force the constrained packages to upgrade synchronously to help maintain a working, supportable image. In general, you should not install or update a package that

is constrained by an incorporation package. Instead, you should update the incorporation package. A constrained package could be uninstalled, but if the constrained package is installed or updated, the version is constrained. See "Relaxing Version Constraints Specified by Incorporations" on page 84 for related information.

The `pkg://solaris/entire` package is a special incorporation that specifies `incorporate` dependencies on many other incorporation packages to constrain the versions of most of the system software installed in the image.

Caution - Do not remove the package named `entire`. The `entire` package constrains system package versions so that the resulting set of packages is a supportable image. Proper system update and correct package selection depend on this incorporation. Removing the `entire` package will result in an unsupported system.

Group Packages

A *group* package specifies the set of packages that constitute a feature or tool. Installing a group package installs all the `group` dependency packages in that group package. Packages specified as `group` dependencies in a group package do not specify the package version. The group package is a content management tool, not a version management tool.

A group package delivers the packages named in its `group` dependencies unless those packages are already installed or are on the avoid list. See "Avoiding Installing Some Packages in a Group Package" on page 88 for information about the avoid list of an image.

The `group/feature/storage-server` package, for example, delivers drivers, services, file systems, I/O components, libraries, and utilities related to storage if they are not already installed. The `group/system/solaris-minimal-server` package delivers the set of packages required for the minimum supported Oracle Solaris environment. See "Listing All Installable Packages in a Group Package" on page 26 for an example of how to list all of the packages that are delivered by a group package.

Uninstalling a group package does not necessarily uninstall all the packages named in its `group` dependencies. Packages that are required by other software that is still installed will not be uninstalled when you uninstall the group package.

Fault Management Resource Identifiers

Each package is represented by a Fault Management Resource Identifier (FMRI). The full FMRI for a package consists of the scheme, a publisher, the package name, and a version string in the following format:

scheme://*publisher*/*name*@*version*:*date*T*time*Z

The scheme, publisher, and version string are optional. In IPS command operands, you can use the smallest portion of the package name that uniquely identifies the package, and you can use the ? and * characters as `glob`(3C)-style wildcards to match one or more packages.

The following example package FMRI is for the `suri` storage library:

`pkg://solaris/system/library/storage/suri@0.5.11,5.11-0.175.2.0.0.34.0:20140303T145535Z`

Scheme

`pkg`

Publisher

`solaris`

If the publisher is specified, then the publisher name must be preceded by `pkg://` or `//`.

Package name

`system/library/storage/suri`

Package names are hierarchical with an arbitrary number of components separated by forward slash (/) characters. In IPS commands, leading components of package names can be omitted if the package name that is used in the command uniquely identifies the package. If you specify the full package name but omit the publisher, the full package name can be preceded by `pkg:/` or `/` but not by `pkg://` or `//`. If you specify an abbreviated package name, do not use any other characters to the left of the package name.

Version

The package version has four parts:

Component version: `0.5.11`

For components tightly bound to the operating system, the component version usually includes the value of `uname -r` for that version of the operating system. For a component with its own development lifecycle, the component version is a dotted release number, such as 2.4.10.

Build version: `5.11`

The build version must follow a comma (,). The build version specifies the version of the operating system on which the contents of the package were built.

Branch version: `0.175.2.0.0.34.0`

The branch version must follow a hyphen (-). The branch version provides vendor-specific information.

Oracle Solaris packages show the following information in the branch version portion of the version string of a package FMRI:

Major release number: `0.175`

>   The major or marketing development release build number. In this example, `0.175` indicates Oracle Solaris 11.

Update release number: `2`

>   The update release number for this Oracle Solaris release. The update value is 0 for the first customer shipment of an Oracle Solaris release, 1 for the first update of that release, 2 for the second update of that release, and so forth. In this example, 1 indicates Oracle Solaris 11.1.

SRU number: `0`

>   The Support Repository Update (SRU) number for this update release. SRUs include only bug fixes; they do not include new features. The Oracle Support Repository is available only to systems under a support contract.

Reserved: `0`

>   This field is not currently used for Oracle Solaris packages.

SRU build number: `34`

>   The build number of the SRU, or the respin number for the major release.

Nightly build number: `0`

>   The build number for the individual nightly builds.

Time stamp: `20140303T145535Z`

>   The time stamp must follow a colon (:). The time stamp is the time the package was published in ISO-8601 basic format: *YYYYMMDD*T*HHMMSS*Z.

Publishers, Repositories, and Package Archives

A *publisher* identifies a person or organization that provides one or more packages. Publishers can distribute their packages using package repositories or package archives. Publishers can be configured into a preferred search order. When a package installation command is given and the package specification does not include the publisher name, the first publisher in the search order is searched for that package. If a match of the specified package FMRI pattern is not found, the second publisher in the search order is searched, and so forth until the package is found or all publishers have been searched.

A *repository* is a location where packages are published and from where packages are retrieved. The location is specified by a Universal Resource Identifier (URI). A *catalog* is the list of all the packages in a repository.

A *package archive* is a file that contains publisher information and one or more packages provided by that publisher.

Repository Origins and Mirrors

An *origin* is a package repository that contains both package *metadata* (such as catalogs, manifests, and search indexes) and package *content* (files). If multiple origins are configured for a given publisher in an image, the IPS client attempts to choose the best origin from which to retrieve package data.

A *mirror* is a package repository that contains only package content. Clients that install and update packages from a mirror repository must still download metadata from an origin repository. IPS clients access the origin to obtain a publisher's catalog, even when the clients download package content from a mirror. If a mirror is configured for a publisher, the IPS client prefers the mirror for package content retrieval. If multiple mirrors are configured for a given publisher in an image, the IPS client attempts to choose the best mirror from which to retrieve package content. If all mirrors are unreachable, do not have the required content, or are slower, the IPS client retrieves the content from an origin. See "Publishers and Repositories" in the pkg(5) man page for more information.

Note - Even if a repository that is specified as a mirror repository is complete with both content and metadata, users cannot access the content in that mirror repository unless the same version of the same package also exists in an origin repository for that same publisher.

Images and Boot Environments

An *image* is a location where IPS packages can be installed and where other IPS operations can be performed.

A *boot environment* (BE) is a bootable instance of an image. You can maintain multiple BEs on a physical or virtual system, and each BE can have different software versions installed, including different operating system versions. When you boot your system, you have the option to boot into any of the BEs on the system. A new BE can be created automatically as a result of package operations. Whether a new BE is created automatically depends on image policy as described in "Boot Environment Policy Image Properties" on page 91. You can also explicitly create a new BE by specifying options described in "Boot Environment Options" on page 38. See "Creating and Administering Oracle Solaris 11.2 Boot Environments " for information about how to use the beadm command to create a new BE.

Packages can only be installed into file systems that are part of a BE. For example, on a default Oracle Solaris 11 installation, only datasets under `rpool/ROOT/`*BEname*`/` are supported for package operations.

An Oracle Solaris Zone is another example of an image. A non-global *zone* is a virtualized operating system environment created within an instance of the Oracle Solaris operating system called the global zone. The global zone is the parent image, and non-global zones within that global zone are child images of that global zone. In IPS command output, non-global zones are sometimes called linked images because they are linked to their parent global zone image.

IPS commands executed in a global zone can affect non-global zones as described in "Working with Non-Global Zones" on page 52. IPS commands executed in a global zone do not affect kernel zones (`solaris-kz` branded zones) or Oracle Solaris 10 zones (`solaris10` branded zones). In this guide, "non-global zone" means a `solaris` branded Oracle Solaris 11 non-global zone. See "Introduction to Oracle Solaris Zones " for information about zones.

Package Facets and Variants

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, an optional component is called a *facet* and a mutually exclusive component is called a *variant*.

Facets and variants are special properties set on the image. Facets and variants are also tags set on actions in a package manifest. The values of facet and variant tags on an action compared with the values of facets and variants set in the image determine whether that package action can be installed. For example, if you set a particular locale facet to `false` in the image, any file actions that specify that facet will not be installed, and currently installed file actions that specify that facet are uninstalled.

Most variants can have arbitrary values. Facet tags set on an action can only have the value `true`. Facet properties set on the image can only have the value `true` or `false`.

The following algorithm describes how the facets and variants set on the image affect whether a particular action is installed:

- Actions with no facet or variant tags are always installed.
- Actions with facet tags are installed unless all of the facets or facet patterns matching the tags are set to `false` on the image. If any facet is set to `true` or is not explicitly set (`true` is the default), then the action is installed.
- Actions with variant tags are installed only if the values of all the variant tags are the same as the values set in the image.
- Actions with both facet and variant tags are installed if both the facets and the variants allow the action to be installed.

To view or modify the values of the facets and variants set on the image, see "Controlling Installation of Optional Components" on page 79.

Installation Privileges

The commands discussed in Chapter 2, "Getting Information About Software Packages" do not require any special privilege to use. Tasks such as installing IPS packages, upgrading your operating system, and configuring publisher and image properties require more privilege. Use one of the following methods to gain the privilege you need. See "Securing Users and Processes in Oracle Solaris 11.2 " for more information about profiles and roles, including how to determine which profile or role you need.

Rights profiles

Use the `profiles` command to list the rights profiles that are assigned to you. For example, if you have the Software Installation rights profile, you can run the `pkg` and `beadm` commands to install and update packages and manage boot environments.

Roles

Use the `roles` command to list the roles that are assigned to you. For example, if you have the `root` role, you can use the `su` command with the `root` password to assume the `root` role.

sudo command

Depending on the security policy at your site, you might be able to use the `sudo` command with your user password to execute a privileged command.

2

Getting Information About Software Packages

This chapter describes commands that provide the following kinds of information about packages:

- Whether the package is installed or can be installed or updated
- The description, size, and version of the package
- Which packages are part of a group package
- Which package delivers a specified file
- Which package delivers a specified SMF service
- Which packages are dependents of the specified package
- Which packages are in a particular category

If the content of package repositories for your configured publishers might have changed, update your list of available packages at the beginning of your session to ensure that you receive the most current information. To update your list of packages execute the `pkg refresh` command. For example, if a repository has been updated with new packages, listing available packages might not show those new packages until you run `pkg refresh`.

For a complete list of all options for commands discussed in this chapter, see the `pkg`(1) man page.

Showing Package Install State Information

The `pkg list` command shows whether a package is installed in the current image and whether an update is available. With no options or operands, the `pkg list` command lists all packages that are installed in the current image. To narrow your results, provide one or more package names. You can use wildcards in the package names. Quote the wildcards so that the argument is passed directly to `pkg` and the shell does not expand it.

Installed Packages

The `pkg list` command displays one line of information for each matching package, as shown in the following example. The "i" in the I column indicates that these packages are installed in this image.

```
$ pkg list '*toolkit'
NAME (PUBLISHER)                VERSION             IFO
isvtoolkit (isvpub)             1.0                 i--
system/dtrace/dtrace-toolkit    0.99-0.175.2.0.0.34.0  i--
```

The publisher name in parentheses indicates that the `isvpub` publisher is not the first publisher in the publisher search order in this image. The `dtrace-toolkit` package that is installed in this image is published by the publisher that is the first publisher in the search order.

Installable Packages

To list packages that are installed and the newest versions of packages that are not installed but could be installed in this image, use the `-a` option.

```
$ pkg list -a '*toolkit'
NAME (PUBLISHER)                VERSION             IFO
image/nvidia/cg-toolkit         3.0.15-0.175.2.0.0.17.0  ---
isvtoolkit (isvpub)             1.0                 i--
system/dtrace/dtrace-toolkit    0.99-0.175.2.0.0.34.0  i--
```

This output indicates that the `image/nvidia/cg-toolkit` package is available and can be installed in this image.

Newest Packages

To list the newest versions of all matching packages, including packages that cannot be installed in this image, use the `-n` option.

```
$ pkg list -n '*toolkit'
NAME (PUBLISHER)                VERSION             IFO
developer/dtrace/toolkit        0.99-0.173.0.0.0.1.0  --r
image/nvidia/cg-toolkit         3.0.15-0.175.2.0.0.17.0  ---
isvtoolkit (isvpub)             1.0                 i--
system/dtrace/dtrace-toolkit    0.99-0.175.2.0.0.35.0  ---
```

The `developer/dtrace/toolkit` package and the `system/dtrace/dtrace-toolkit` package cannot be installed in this image. You know this because these packages were not listed by the `-a` option. The `developer/dtrace/toolkit` package has been renamed. See "Renamed and Obsolete Packages" on page 21 for more information.

A newer version of the `dtrace-toolkit` package than the version that is currently installed in this image exists from configured publishers. The newer version might be installable if you also update other packages of which the `dtrace-toolkit` package is a dependency, as shown by the "Reason" line in the following example. Time stamps have been omitted from the following command output for brevity. The `pkg update` and `pkg install` commands are discussed in .

```
$ pkg update -nv dtrace-toolkit
No updates are available for this image.
$ pkg install -nv dtrace-toolkit@0.99-0.175.2.0.0.35
pkg install: No matching version of system/dtrace/dtrace-toolkit can be installed:
  Reject:  pkg://solaris/system/dtrace/dtrace-toolkit@0.99,5.11-0.175.2.0.0.35.0
  Reason:  This version is excluded by installed incorporation
pkg://solaris/consolidation/osnet/osnet-incorporation@0.5.11,5.11-0.175.2.0.0.34.0
```

Packages for Which Updates Are Available

The `-u` option lists all installed matching packages that have newer versions available. The number of packages that have newer versions available might be larger than the number of packages that could be updated in this image, as demonstrated in . Packages can be updated only to versions allowed by the constraints imposed on the image by installed package dependencies and publisher configuration.

```
$ pkg list -u '*toolkit'
NAME (PUBLISHER)              VERSION              IFO
system/dtrace/dtrace-toolkit     0.99-0.175.2.0.0.34.0   i--
```

All Available Packages

To list all available versions of all matching packages, including packages that cannot be installed in this image, use the `-af` option. The `-f` option cannot be used without the `-a` option. You might want to specify a portion of the version string to narrow these results. Specifying the special version string `@latest` shows the same result as the `-n` option shows.

```
$ pkg list -af '*toolkit@0.99-0.175.2'
$ pkg list -af '*toolkit@latest'
```

Renamed and Obsolete Packages

In the following example, the "r" in the O column indicates that the package has been renamed, and the "o" in the O column indicates that the package is obsolete.

```
$ pkg list -n developer/dtrace/toolkit database/mysql-50 web/amp
NAME (PUBLISHER)                VERSION             IFO
database/mysql-50               5.0.91-0.171        --o
developer/dtrace/toolkit        0.99-0.173.0.0.0.1.0    --r
web/amp                         0.5.11-0.174.0.0.0.0.0  --r
```

None of these packages is installable. If you try to install an obsolete package, the installation fails with the message that no updates are necessary for this image. If you try to install a renamed package, the system attempts to install the package to which the renamed package has been renamed.

Use the pkg info command to determine the new name of a renamed package. Use the -r option to query the configured package repositories because the package is not installed. See the "Renamed to" line, as shown in the following example:

```
$ pkg info -r web/amp
          Name: web/amp
       Summary:
         State: Not installed (Renamed)
    Renamed to: group/feature/amp@0.5.11-0.174.0.0.0.0.0
                consolidation/ips/ips-incorporation
     Publisher: solaris
       Version: 0.5.11
 Build Release: 5.11
        Branch: 0.174.0.0.0.0.0
Packaging Date: September 21, 2011 07:15:02 PM
          Size: 5.45 kB
          FMRI: pkg://solaris/web/amp@0.5.11,5.11-0.174.0.0.0.0.0:20110921T191502Z
```

If you try to install the web/amp package, the group/feature/amp package will be installed instead if it is not already installed and if it can be installed in this image.

In the following example, the "Renamed to" package is already installed, so the packaging system reports that no updates are necessary.

```
$ pkg info -r developer/dtrace/toolkit
          Name: developer/dtrace/toolkit
       Summary:
         State: Not installed (Renamed)
    Renamed to: pkg:/system/dtrace/dtrace-toolkit@0.99,5.11-0.173.0.0.0.0.0
                consolidation/osnet/osnet-incorporation
     Publisher: solaris
       Version: 0.99
 Build Release: 5.11
        Branch: 0.173.0.0.0.1.0
Packaging Date: August 26, 2011 02:55:51 PM
          Size: 5.45 kB
          FMRI: pkg://solaris/developer/dtrace/toolkit@0.99,5.11-0.173.0.0.0.1.0:20110826T145551Z
$ pkg list dtrace-toolkit
NAME (PUBLISHER)                VERSION             IFO
system/dtrace/dtrace-toolkit    0.99-0.175.2.0.0.34.0   i--
$ pkg install developer/dtrace/toolkit
No updates necessary for this image.
```

Packages Frozen at a Specific Version

An "f" in the F column indicates the package is frozen. If a package is frozen, you can only install or update to packages that match the frozen version. See "Locking Packages to a Specified Version" on page 82 for information about freezing packages.

```
$ pkg list openssl
NAME (PUBLISHER)              VERSION                IFO
library/security/openssl      1.0.1.6-0.175.2.0.0.34.0 if-
```

Displaying Package Descriptions or Licenses

The `pkg info` command displays information about a package, including the name, description, installed state, version, packaging date, package size, and the full FMRI. With no options or operands, the `pkg info` command displays information about all packages that are installed in the current image. To narrow your results, provide one or more package names. You can use wildcards in the package names. Quote the wildcards so that the argument is passed directly to `pkg` and the shell does not expand it.

Both the `info` and `list` subcommands display the package name, publisher, and version information. 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, renamed, or frozen. The `pkg list` command can also show the package summary and the full FMRI. The `pkg info` command displays the package summary, description, category, and size, and can separately display the license information.

Displaying Package Description, Size, Full FMRI

You can use the `pkg list -s` command to show the package summary.

```
$ pkg list -s entire
NAME (PUBLISHER)    SUMMARY
entire              Incorporation to lock all system packages to the same build
```

The `pkg list -v` command lists the full package FMRI.

```
$ pkg list -v entire
FMRI                                                    IFO
pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.34.0:20140303T182643Z  ---
```

The `pkg info` command displays more information.

```
$ pkg info entire
          Name: entire
       Summary: Incorporation to lock all system packages to the same build
   Description: This package constrains system package versions to the same
                build.  WARNING: Proper system update and correct package
                selection depend on the presence of this incorporation.
                Removing this package will result in an unsupported system.
      Category: Meta Packages/Incorporations
         State: Installed
     Publisher: solaris
       Version: 0.5.11
        Branch: 0.175.2.0.0.34.0
Packaging Date: March  3, 2014 06:26:43 PM
          Size: 5.46 kB
          FMRI: pkg://solaris/entire@0.5.11-0.175.2.0.0.34.0:20140303T182643Z
```

As shown in , you can use the `pkg info`
command to find the new name of a renamed package.

Displaying Package Licenses

Use the `--license` option to display the license texts for the matching packages. This
information can be quite lengthy. Other information that the `pkg info` command shows when
the `--license` option is not specified is not displayed.

```
$ pkg info --license osnet-incorporation
You acknowledge that your use of this Oracle Solaris software product
is subject to (i) the license terms that you accepted when you obtained
the right to use Oracle Solaris software; or (ii) the license terms that
you agreed to when you placed your Oracle Solaris software order with
Oracle; or (iii) the Oracle Solaris software license terms included with
the hardware that you acquired from Oracle; or, if (i), (ii) or (iii)
are not applicable, then, (iv) the OTN License Agreement for Oracle
Solaris (which you acknowledge you have read and agree to) available at
http://www.oracle.com/technetwork/licenses/solaris-cluster-express-license-167852.html.
Note: Software downloaded for trial use or downloaded as replacement
media may not be used to update any unsupported software.
```

Showing Information From the Package Manifest

The `pkg contents` command displays the file system content of packages. With no options
or operands, this command displays path information for all packages that are installed in
the current image. Use command options to specify particular package content to display. To
narrow your results, provide one or more package names. You can use wildcards in the package
names. Quote the wildcards so that the argument is passed directly to `pkg` and the shell does not
expand it.

Both the `contents` and `search` subcommands query the content of packages. The `pkg contents` command displays actions and attributes of packages. The `pkg search` command lists the packages that match the query.

Listing Files Installed by a Package

The following example shows the default behavior of the `pkg contents` command: Display the value of the `path` attribute for each file system object that can be installed in this image.

```
$ pkg contents entire
pkg: This package delivers no filesystem content, but may contain metadata. Use
the -o option to specify fields other than 'path', or use the -m option to show
the raw package manifests.
$ pkg contents zip
PATH
usr/bin/zip
usr/bin/zipcloak
usr/bin/zipnote
usr/bin/zipsplit
usr/share/man/man1/zip.1
usr/share/man/man1/zipcloak.1
usr/share/man/man1/zipnote.1
usr/share/man/man1/zipsplit.1
```

The `pkg contents` command displays only content that can be installed in this image. If you view the package manifest (use the `-m` option), you see that the `zip` package has 12 file actions. The four files that are not shown in this output are files that cannot be installed in this image. This image is an x86 architecture. The files that are not shown are the four `/usr/bin` files for the SPARC architecture. See "Controlling Installation of Optional Components" on page 79 for information about variants and facets.

Displaying Attributes of Files Installed by a Package

To list more information about installed files than just the path, use the `-t` and `-o` options of the `pkg contents` command.

The `-t` option specifies the type of actions to display. You can specify multiple types in a comma-separated list, or you can specify the `-t` option multiple times.

The `-o` option specifies the attributes of the actions to display. You can specify multiple attributes in a comma-separated list, or you can specify the `-o` option multiple times. See the `pkg`(5) man page for a list of package actions and attributes.

```
$ pkg contents -t file -o owner,group,mode,pkg.size,path zip
```

```
OWNER GROUP MODE PKG.SIZE PATH
root  bin   0555   231260 usr/bin/zip
root  bin   0555   110852 usr/bin/zipcloak
root  bin   0555   104960 usr/bin/zipnote
root  bin   0555   109340 usr/bin/zipsplit
root  bin   0444    86192 usr/share/man/man1/zip.1
root  bin   0444     2705 usr/share/man/man1/zipcloak.1
root  bin   0444     2396 usr/share/man/man1/zipnote.1
root  bin   0444     1837 usr/share/man/man1/zipsplit.1
```

By default, output is sorted by path or by the first attribute specified by the `-o` option. You can use the `-s` option to specify a different attribute as the sort key. The `-s` option can be specified multiple times.

Displaying Other File System Objects and Attributes

The following example shows the path and target of links installed by the specified packages. In addition to the attributes shown in the `pkg`(5) man page, several pseudo attributes are available to use. See the pkg(1) man page for a list of pseudo attributes.

In the following example, the `pkg.name` pseudo attribute shows the name of the package that delivers the specified action. In this example, both Python 2.6.8 and Python 2.7.3 are installed, and the command shows the path to use to access the specific version if you do not want to rely on the `/usr/bin/python` link. See also "Specifying a Default Application Implementation" on page 86 for information about mediations of multiple versions.

```
$ pkg contents -t link -a path=usr/bin/python -o path,target,pkg.name
PATH            TARGET    PKG.NAME
usr/bin/python  python2.6 runtime/python-26
usr/bin/python  python2.7 runtime/python-27
```

Listing All Installable Packages in a Group Package

Oracle Solaris provides several system installation group packages. The Oracle Solaris 11 GUI installer installs the `solaris-desktop` group package. The text installer and the default AI manifest in an Automated Installer installation install the `solaris-large-server` group package. The default installation manifest for non-global zones installs the `solaris-small-server` group package. The `solaris-minimal-server` group package installs the minimal supported set of packages required to run Oracle Solaris.

You can use the following command to display the set of packages that is included in each group:

```
$ pkg contents -Hro fmri -t depend -a type=group solaris-minimal-server
network/ping
service/network/ssh
shell/tcsh
shell/zsh
system/network
```

The `-t` option matches `depend` actions in the package. The `-a` option matches the `depend` actions that are type `group`. The `-o` option displays only the `fmri` attribute of the group `depend` action. Recall that group packages do not specify content such as files; group packages specify other packages that are part of the group. See "Group Packages" on page 13 for more information about group packages.

To also show the summary description of each package, use the `pkg list -s` command:

```
$ pkg list -Has `pkg contents -Hro fmri -t depend -a type=group solaris-minimal-server`
network/ping                              Ping command
service/network/ssh                       Secure Shell protocol server
shell/tcsh                                Tenex C-shell (tcsh)
shell/zsh                                 Z Shell (zsh)
system/network                            Core Network Infrastructure
```

Displaying License Requirements

The following example displays all packages that require you to accept the package license:

```
$ pkg contents -rt license -a must-accept=true -o license,pkg.name '*'
LICENSE PKG.NAME
BCL     developer/java/jdk-7
BCL     runtime/java/jre-7
LICENSE developer/java/jdk-6
LICENSE library/java/java-demo-6
LICENSE runtime/java/jre-6
lic_OTN consolidation/osnet/osnet-incorporation
lic_OTN install-image/solaris-auto-install
```

See "Displaying Package Licenses" on page 24 for information about how to display the license text.

Searching for Packages

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

Comparing the `pkg search` and `pkg contents` Commands

Like the `pkg contents` command, the `pkg search` command examines the contents of packages. While the `pkg contents` command returns the contents, the `pkg search` command returns the names of packages that match the search query. The following table shows some of the similarities and differences between these two commands.

`pkg contents`	- Examines installed packages. Use the `-r` option to examine packages in repositories associated with all publishers configured for this image. - Use the `-g` option to specify the URI of a repository to examine. - Use the `-t` option to specify an action. - Use the `-a` option to specify an attribute and attribute value. - Use the `-o` option to specify columns of results. - Use the `-s` option to sort the results.
`pkg search`	- Searches packages in repositories associated with all publishers configured for this image. Use the `-l` option to search only installed packages. - Use the `-s` option to specify the URI of a repository to search. - Use the search query to specify an action. - Use the search query to specify an attribute and attribute value. - Use the `-o` option to specify columns of results.

Tip - Use the `pkg contents` command to show the contents of a specified package, and use the `pkg search` command to show packages that match a query. If you know which package delivers the content that you are interested in, use the `pkg contents` command.

Specifying the Search Query

By default, the search query is a series of terms to be matched exactly except for case. Use the `-I` option to specify a case-sensitive search.

You can use ? and * wildcards in query terms. You can use single or double quotation marks to search for phrases. Be sure to take your shell into account when you use wildcards or quotation marks.

You can specify more than one query term. By default, multiple terms are joined with AND. You can explicitly join two terms with OR.

Search queries can be expressed in the following structured form:

package:*action*:*index*:*token*

package	The name of the package to search or a pattern that might match multiple packages.
action	The name of an action listed in the "Actions" section in the pkg(5) man page.
index	The name of an attribute of *action* as listed in the "Actions" section of the pkg(5) man page.
token	The value of *index* or a pattern that might match the value of *index*.

Missing fields are implicitly wildcarded.

Not all attributes are searchable. For example, mode is an attribute of the file action, but mode is not a valid value for *index*.

Some values of *index* are values derived from other attributes. For example, *index* can be basename, which is the last component of the path attribute of a file or dir action. Examples of useful values for *index* include basename and path for file and dir actions, the dependency type (require or group, for example) for depend actions, and driver_name and alias for driver actions.

The value of *token* is compared with the value of the attribute named by *index*. For example, in the following partial driver action, alias is an attribute name that could be specified for *index*, and pci108e* could be specified for *token*.

```
driver alias=pci108e,1647 alias=pci108e,16a7
```

The syntax of a set action is slightly different. The two attributes of a set action are name and value. In this case, the value of *index* is the value of a name attribute, and the value of *token* is compared with the value of the matching value attribute. The following example show a partial set action for a driver package:

```
set name=pkg.summary value="Broadcom NetXtreme II 10GbE NIC Driver"
```

The following example specifies set for the *action*, pkg.summary for *index*, and Broadcom for *token*. The search.match and pkg.name column specifiers are pseudo attributes. See the pkg(1) man page.

```
$ pkg search -o search.match,pkg.name pkg.summary:Broadcom
SEARCH.MATCH                           PKG.NAME
Broadcom NetXtreme II 10GbE NIC Driver driver/network/ethernet/bnxe
Broadcom 57xx 1GbE NIC Driver          driver/network/ethernet/bge
Broadcom NetXtreme II 1GbE NIC Driver  driver/network/ethernet/bnx
Broadcom BCM4401 NIC Driver            driver/network/ethernet/bfe
```

```
Broadcom HT1000 SATA driver          driver/storage/bcm_sata
```

Some well-defined values of set action name attributes include pkg.fmri, info.classification, pkg.description, and pkg.summary. See "Set Actions" in the pkg(5) man page.

By default, matches are displayed only for currently installed or newer package versions. Use the -f option to display all matched versions.

By default, results are displayed for all matching actions, which can yield multiple lines of results for one package. Use the -p option to list each matching package only once.

Identifying Which Package Delivers a Specified File

The following examples show that the libpower library came from the system/kernel/power package.

```
$ pkg search -Hlo pkg.name /lib/libpower.so.1
system/kernel/power
$ pkg search -lo path,pkg.name libpower.so.1
PATH            PKG.NAME
lib/libpower.so.1 system/kernel/power
$ pkg search -Hlo path,pkg.name basename:libpower.so.1
lib/libpower.so.1 system/kernel/power
$ pkg search -Hlo path,pkg.name 'path:*libpower.so.1'
lib/libpower.so.1 system/kernel/power
```

Identifying Which Package Delivers a Specified SMF Service

To show which packages provide a particular SMF service, search for the name of the service as the value of the org.opensolaris.smf.fmri attribute.

```
$ pkg search -o pkg.name,search.match 'org.opensolaris.smf.fmri:*network/http*'
PKG.NAME                SEARCH.MATCH
web/java-servlet/tomcat svc:/network/http
web/proxy/squid         svc:/network/http
web/proxy/privoxy       svc:/network/http
web/server/lighttpd-14  svc:/network/http
web/server/apache-22    svc:/network/http
web/server/apache-22    svc:/network/http:apache22
web/server/lighttpd-14  svc:/network/http:lighttpd14
web/proxy/privoxy       svc:/network/http:privoxy
web/proxy/squid         svc:/network/http:squid
```

```
web/java-servlet/tomcat svc:/network/http:tomcat6
```

In this case, each attribute has two values: the service name with and without the instance name specified. The following example shows how this attribute is specified in the package manifest:

```
set name=org.opensolaris.smf.fmri value=svc:/network/http value=svc:/network/http:apache22
```

The following example shows this same information with each package listed only once. Escape the colon character so that it is interpreted as part of the *token* and not as another search query field.

```
$ pkg search -o pkg.name,search.match 'org.opensolaris.smf.fmri:*network/http\:*'
PKG.NAME                SEARCH.MATCH
web/server/apache-22    svc:/network/http:apache22
web/server/lighttpd-14  svc:/network/http:lighttpd14
web/proxy/privoxy       svc:/network/http:privoxy
web/proxy/squid         svc:/network/http:squid
web/java-servlet/tomcat svc:/network/http:tomcat6
```

Listing Packages by Classification or Category

The following example identifies all installed packages that have "Source Code Management" in the value of their info.classification attribute:

```
$ pkg search -Hlo pkg.shortfmri info.classification:'source code management'
pkg:/developer/versioning/sccs@0.5.11-0.175.2.0.0.8.0
pkg:/developer/versioning/git@1.7.9.2-0.175.2.0.0.34.0
pkg:/developer/versioning/mercurial-27@2.2.1-0.175.2.0.0.34.0
```

The following example shows the package metadata that is matched in this search:

```
set name=info.classification value="org.opensolaris.category.2008:Development/Source Code
 Management"
```

This information is displayed in the "Category" line in output from the pkg info command.

See "Classification Values" in "Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 " for other classifications that you can search for.

You can also use the pkg list command with a guess at one of the components of the package name, as in the following examples:

```
$ pkg list '*storage*'
$ pkg list -a '*database*'
```

Showing Dependent Packages

These examples show the packages that are dependencies of the specified package.

The following example shows packages that have a `require` dependency on the `system/kernel/power` package:

```
$ pkg search -Hlo pkg.name require:system/kernel/power
system/kernel/dynamic-reconfiguration/i86pc
system/hal
```

The following `pkg contents` command confirms the results of the search. The requested output `action.raw` is a pseudo attribute that displays the action exactly as it appears in the package manifest.

```
$ pkg contents -rt depend -a fmri='*power*' -o pkg.name,action.raw i86pc system/hal
PKG.NAME                                ACTION.RAW
system/hal                              depend fmri=pkg:/system/kernel/power
@0.5.11-0.175.2.0.0.34.0 type=require variant.opensolaris.zone=global
system/kernel/dynamic-reconfiguration/i86pc depend fmri=pkg:/system/kernel/power
 type=require
```

The following example shows that many packages have an `exclude` dependency on `pkg:/x11/server/xorg@1.14.99`:

```
$ pkg search -lo pkg.name,fmri 'depend:exclude:*xorg*'
PKG.NAME                                FMRI
x11/server/xorg/driver/xorg-video-ati   pkg:/x11/server/xorg@1.14.99
x11/server/xvnc                         pkg:/x11/server/xorg@1.14.99
x11/server/xserver-common               pkg:/x11/server/xorg@1.14.99
x11/server/xorg/driver/xorg-input-vmmouse  pkg:/x11/server/xorg@1.14.99
x11/server/xephyr                       pkg:/x11/server/xorg@1.14.99
...
```

Listing All Packages in a Group Package

The Oracle Solaris 11 GUI installer installs the `solaris-desktop` group package. The text installer and the default AI manifest in an Automated Installer installation install the `solaris-large-server` group package. The default installation manifest for non-global zones installs the `solaris-small-server` group package. The `solaris-minimal-server` group package installs the minimal supported set of packages required to run Oracle Solaris. You can use the following search form to display the set of packages that is included in each group:

```
$ pkg search -Ho fmri '*/solaris-minimal-server:depend:group:*'
network/ping
service/network/ssh
shell/tcsh
shell/zsh
system/network
```

In this example, `-o pkg.name` would return only the name of the package specified in the *package* field of the query:

```
group/system/solaris-minimal-server
```

The `-o fmri` option returns the value of the `fmri` attribute of the packages that are specified in the `solaris-minimal-server` package as `group` type dependencies.

The `pkg search` command is returning the value of an attribute of an action in a specified package. That attribute value happens to be a package name in this example. The number of results from this command can be larger than the number of the results from the similar `pkg contents` command shown in "Listing All Installable Packages in a Group Package" on page 26 because these search results include the names of all packages that are named in `depend` actions of type `group` in the specified package, not just installable packages. For example, package variants and facets might be included that are not installable in this image. To see this difference, try both examples with the `solaris-large-server` package.

♦ ♦ ♦   **C H A P T E R   3**

3

Installing and Updating Software Packages

Package installation and update are affected by image configuration such as constraining some packages to a particular version, configuring publisher search order, and setting package signing properties. Image configuration is discussed in Chapter 5, "Configuring Installed Images".

How to determine which packages are already installed, which packages are available to install, and which packages have updates available is covered in Chapter 2, "Getting Information About Software Packages".

This chapter shows how to perform the following tasks:

- Run a trial installation to see whether the installation would succeed and what would be installed
- Install, update, and uninstall packages
- Validate packages
- Fix problems with installed packages
- Restore an installed file to its original content
- Uninstall packages

"Working with Non-Global Zones" on page 52 discusses aspects of package operations that are unique to Oracle Solaris Zones.

Installing, updating, and uninstalling packages require increased privileges. See "Installation Privileges" on page 18 for more information.

For a complete list of all options for commands discussed in this chapter, see the pkg(1) man page.

Previewing an Operation

Many of the commands shown in this chapter and in Chapter 5, "Configuring Installed Images" have an -n option that enables you to see what the command will do without making any changes.

Tip - Best practice is to use the `-n` option whenever it is available. Use the `-n` option with one or more verbose options (`-nv`, `-nvv`) and review the effects of the command before you execute the command without the `-n` option.

The following example shows information about a package installation that is not actually performed:

```
$ pkg install -nv group/feature/amp
          Packages to install:        6
          Mediators to change:        1
           Services to change:        2
     Estimated space available:  22.70 GB
Estimated space to be consumed: 751.08 MB
        Create boot environment:       No
Create backup boot environment:       No
          Rebuild boot archive:       No

Changed mediators:
  mediator mysql:
          version: None -> 5.1 (system default)

Changed packages:
solaris
  database/mysql-51
    None -> 5.1.37,5.11-0.175.2.0.0.34.0:20140303T160611Z
  database/mysql-common
    None -> 5.11,5.11-0.175.2.0.0.34.0:20140303T161628Z
  group/feature/amp
    None -> 0.5.11,5.11-0.175.2.0.0.33.0:20140217T134747Z
  web/server/apache-22/module/apache-dtrace
    None -> 0.3.1,5.11-0.175.2.0.0.34.0:20140303T175456Z
  web/server/apache-22/module/apache-fcgid
    None -> 2.3.9,5.11-0.175.2.0.0.34.0:20140303T175502Z
  web/server/apache-22/module/apache-php5
    None -> 5.2.17,5.11-0.175.1.0.0.18:20120611T210317Z
Services:
  restart_fmri:
    svc:/system/manifest-import:default
    svc:/system/rbac:default
```

This output indicates that this installation operation will be done in the current BE and not in a new BE, and a backup of this the current BE will not be created. You could specify options or image properties to require a new BE or a backup BE. The "Changed packages" section shows that the `amp` group package would be installed, and its five group dependencies would be installed. The output shows which version of each package would be installed. The token `None` indicates that these packages are not currently installed and therefore are not being updated.

The following command produces a large amount of output because so many packages would be affected. Setting this facet would install all localized content for all packages. Running this preview command might change how you decide to schedule this operation or whether you

decide to add fewer new locales. This output shows that a new BE would not be created by default, but a backup BE would be created.

```
$ pkg change-facet -nv 'facet.locale.*=true'
            Packages to change:      130
     Variants/Facets to change:        1
      Estimated space available:  22.70 GB
Estimated space to be consumed:   3.45 GB
        Create boot environment:       No
Create backup boot environment:      Yes
          Rebuild boot archive:       No
Changed variants/facets:
    facet locale.* (local): False -> True
Changed packages:
solaris
  ...
```

Installing and Updating Packages

The following table shows similarities and differences between the `pkg install` and `pkg update` commands.

pkg install	- Requires one or more package names as operands. - Installs packages that are not currently installed. - Updates packages that are already installed. - Does not downgrade packages. If you specify an installed package at a lower version, the system does not install that package.
pkg update	- Takes zero or more names of packages that are already installed as operands. - Updates installed packages. - Specifying no package names or specifying '*' updates all packages that are installed in the image. - Downgrades installed packages to the version specified in the FMRI. - Does not install packages that are not already installed. If you specify a package that is not already installed, the system does not install that package.

See the `preserve` and `overlay` attributes of the `file` action in the [pkg(5)](#) man page to understand how files with these attributes are handled during installation and update.

Common Installation Options

This section discusses options that are common to multiple installation-related commands. Note that setting or unsetting a mediator, changing a variant or facet, fixing a package, or reverting a file can also involve installing, updating, or uninstalling packages.

Boot Environment Options

A new BE or a backup BE might be created automatically when you install, update, or uninstall a package. Within the constraints of the image policy regarding BEs, you can control the creation of new and backup BEs using the options described below. See "Boot Environment Policy Image Properties" on page 91 for information about new BEs and backup BEs and how to set image policy regarding BEs.

Use the following BE options to force a new BE or backup BE to be created or not created, to give the BE a custom name, and to specify that the new BE should not be activated. These options are available for the `install`, `exact-install`, `uninstall`, `update`, `revert`, `set-mediator`, `unset-mediator`, `change-variant`, and `change-facet` subcommands.

`--no-be-activate`

>   If a BE is created, do not set it as the active BE on the next boot.

>   In the command output, note any messages that say a new boot environment has been created. If a new boot environment has been created and activated, that BE is booted by default on the next reboot if you do not specify the `--no-be-activate` option.

>   Use the `beadm`(1M) command to show and change the active BE separate from any `pkg` command.

`--no-backup-be`

>   Do not create a backup BE.

`--require-backup-be`

>   Create a backup BE if a new BE will not be created. Without this option, a backup BE is created based on image policy. See "Boot Environment Policy Image Properties" on page 91 for an explanation of when backup BEs are created automatically.

`--backup-be-name` *name*

>   If a backup BE is created, name it *name* instead of a default name. Use of `--backup-be-name` implies `--require-backup-be`.

`--deny-new-be`

>   Do not create a new BE. The install, update, uninstall, or revert operation is not performed if a new BE is required.

`--require-new-be`

>   Create a new BE. Without this option, a BE is created based on image policy. See "Boot Environment Policy Image Properties" on page 91 for an explanation of when BEs are created automatically. This option cannot be combined with `--require-backup-be`.

--be-name *name*

>   If a BE is created, name it *name* instead of a default name. Use of `--be-name` implies `--require-new-be`. Using this option is the safest way to perform operations

Options That Operate on Non-Global Zones

As discussed in "Working with Non-Global Zones" on page 52, only some package installations, removals, and updates performed in the global zone automatically affect non-global zones. The `-r` option performs the same `pkg` operation in non-global zones that you entered in the global zone, possibly affecting many more packages than would be affected if you did not use `-r`. These options are available for the `install`, `uninstall`, `update`, `change-variant`, and `change-facet` subcommands.

-r

>   Run this operation in the global zone and also in all installed `solaris` branded non-global zones. The effect on the non-global zone is similar to logging into each non-global zone and running the command directly.
>
>   Without this option, when you run `pkg` commands in the global zone, non-global zones are modified only to the extent required to keep them compatible with the global zone as described in "Working with Non-Global Zones" on page 52. With this option, the `pkg` operation is applied to all installed non-global zones except as limited by the `-z` and `-Z` options. Zones that are excluded by the `-z` and `-Z` options might still be modified if updates are required to keep them in sync with the global zone.

-z *zone*

>   Run this operation only in the specified non-global zone. The `-z` option can be specified multiple times. The `-z` option can only be used with the `-r` option. The `-z` option cannot be used with the `-Z` option.

-Z *zone*

>   Run this operation in all non-global zones except for the specified zone. The `-Z` option can be specified multiple times. The `-Z` option can only be used with the `-r` option. The `-Z` option cannot be used with the `-z` option.

The following option specifies the number of non-global zones to update concurrently with the global zone. This option is available for the `install`, `exact-install`, `uninstall`, `update`, `change-variant`, and `change-facet` subcommands.

-C *n*

>   Update at most *n* installed `solaris` branded non-global zones in parallel with the global zone. If *n* is 0 or a negative number, all non-global zones are updated concurrently with the global zone.

The environment variable PKG_CONCURRENCY can also be set to the value *n*. The -C option overrides the PKG_CONCURRENCY setting. If the -C option is specified, PKG_CONCURRENCY is ignored.

Service Action Options

A package might specify SMF service actions such as restarting or refreshing a specified service when the package is installed or updated. If you are operating on a large number of packages, the pkg operation might finish before all the service actions finish. Then you might not be able to use the newly-installed software because an associated service is not yet available.

To avoid this problem, use one of the following options to run SMF actuators synchronously with the pkg command. These options are available for the install, uninstall, update, change-variant, and change-facet subcommands.

--sync-actuators

> When you specify this option, the pkg command will not return until all SMF actuators have finished in the zone in which pkg was invoked (the global zone or a non-global zone).

--sync-actuators-timeout *timeout*

> When you specify this option, the pkg command will not return until all SMF actuators have finished or the *timeout* period is reached, whichever is shorter. If the actuators do not finish within the given *timeout* in seconds, the pkg command continues operation and exits with return code 8.

License Options

You might be required to accept a license before you can install or update a package. Use the following options to view and accept required licenses. These options are available for the install, exact-install, update, fix, change-variant, and change-facet subcommands.

--licenses

> Use the --licenses option to display all of the licenses for the packages that are installed or updated as part of this operation. Licenses for all packages are displayed, not just licenses that must be accepted to enable this operation to proceed. If a license must be accepted to proceed, that license is displayed even if you do not specify the --licenses option. To view the license for a package without starting any other operation, use the pkg list command as shown in "Displaying Package Licenses" on page 24. To display a list

of licenses that must be accepted, use the `pkg info` command as shown in "Displaying License Requirements" on page 27.

`--accept`

Use the `--accept` option to 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 required license is displayed and the installation operation fails.

Other Installation Options

`--no-index`

By default, search indexes are updated when you install, update, or uninstall packages. Use the `--no-index` option to not update search indexes after successful completion of these operations. Specifying this option might save some time if you are installing a large number of packages. When you are finished with all install, update, and uninstall operations, you can use `pkg refresh` to update the list of available packages and publisher metadata for each publisher specified. If no publishers are specified, the refresh is performed for all publishers. This option is available for the `install`, `exact-install`, `uninstall`, and `update` subcommands.

`--no-refresh`

When you specify the `--no-refresh` option, the repositories for the image's publishers are not contacted to retrieve the newest list of available packages and other metadata. This option is available for the `install`, `exact-install`, and `update` subcommands.

Installing a New Package

By default, the newest version of a package that is compatible with the rest of the image is installed from the first publisher in the publisher search order that offers the package. To explicitly request the newest version, use `latest` for the version portion of the package FMRI.

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 more than one package is specified, and if any of the specified packages cannot be installed in this image, then none of the specified packages will be installed.

If a package is on the avoid list, installing it removes it from that list. See "Avoiding Installing Some Packages in a Group Package" on page 88 for information about the avoid list.

Identifying and Specifying an Installable Package

If the image has more than one publisher enabled, 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 13 for a description of a package FMRI. See "Configuring Publishers" on page 71 for information about setting publisher stickiness and search order.

If the package name does not specify the publisher, the first publisher that provides a matching package is used as the installation source. If that publisher does not provide a version of the package that can be installed in this image, then the installation operation fails. Use the `pkg list -a` command to see which publishers provide a version of the package that can be installed in this image.

The following commands show that an installable version of the package `atool` is available from a configured publisher, but the publisher that is first in the search order has a version that is not installable in this image. See "Showing Package Install State Information" on page 19 for information about options of the `pkg list` command.

```
$ pkg list -a atool
NAME (PUBLISHER)    VERSION    IFO
atool (isvpub)      2.0        ---
$ pkg list -af atool
NAME (PUBLISHER)    VERSION    IFO
atool               1.1        ---
atool (isvpub)      2.0        ---
```

In this case, the following install command fails. The packaging system finds a match of the package name `atool` from the publisher that is first in the search order, but that package cannot be installed.

```
$ pkg install atool
```

To install this package, make the package name more specific, as shown in the following examples:

```
$ pkg install //isvpub/atool
$ pkg install atool@2.0
```

Use the `-nv` option to see what will be installed before you perform the actual installation. If you receive an error message, see Appendix A, "Troubleshooting Package Installation and Update" for help.

Specifying the Source of the Package

Use the `-g` option to temporarily add the specified package repository or package archive to the list of sources in the image from which to retrieve package data. Repositories that require

a client SSL certificate cannot be used with this option. This option cannot be used in images that have child images (non-global zones). If non-global zones are installed in this image, use the `pkg set-publisher` command to add this publisher and origin. This option can be specified multiple times.

When you specify the `-g` option, publishers that are enabled in the image are preferred when retrieving packages.

- If a package that matches the specified package name or package name pattern is available from a publisher that is enabled in the image, and if that same publisher is not found in the location specified by the `-g` option, the packaging system attempts to install the package from the publisher that is enabled in the image. After `install` or `update`, any packages provided by publishers not configured in the image are added to the image configuration without an origin.
- If a package that matches the specified package name or package name pattern is available from a publisher that is enabled in the image, and if that same publisher publishes the package in the location specified by the `-g` option, the packaging system attempts to install the package from the location specified by the `-g` option.

In the following example, the `btool` package is available from the `solaris` publisher configured in the image. The `btool` package is also available from the `devtool` publisher with repository origin `http://pkg.example1.com/` but the `devtool` publisher is not configured in the image. The command attempts to install the package from the `solaris` publisher because the publisher configured in the image is preferred to the `-g` source when the package is available from the configured publisher.

`$` **`pkg install -g http://pkg.example1.com/ btool`**

To install the package from the `devtool` publisher, specify the publisher name in the package name.

`$` **`pkg install -g http://pkg.example1.com/ //devtool/btool`**

In the following example, `isvpub` is a publisher configured in the image with an origin of `/export/IPSpkgrepos/isvrepo`. The `isvpub` publisher also publishes packages to a repository at `http://pkg.example2.com/` but that origin is not specified for the publisher configured in the image. The following command attempts to install the package from the `http://pkg.example2.com/` location because the same publisher provides the package in both locations.

`$` **`pkg install -g http://pkg.example2.com/ atool`**

See also the description of publisher stickiness in "Adding, Modifying, or Removing Package Publishers" on page 73.

Installing a Package Into a New Boot Environment

Tip - Explicitly specifying a new BE is the safest way to install or update. See "Boot Environment Policy Image Properties" on page 91 for information about when BEs are created.

The new BE is a clone of the current BE with the specified install, uninstall, or update changes applied. The current BE is not modified. The system is not automatically restarted. The new BE is the default boot selection the next time you restart the system. The current BE is still available to be booted.

If you specify the `--no-be-activate` option, the new BE is not the default boot selection the next time you reboot.

Use the `--be-name` option to force a new BE to be created or to give the new BE a meaningful name if a new BE would be created by default.

The example in "Previewing an Operation" on page 35 showed that a new BE would not be created by default when you install the `group/feature/amp` package. In the following partial output, a new BE is created because the `--be-name` option is specified:

```
$ pkg install -v --be-name s11amp group/feature/amp
            Packages to install:         6
            Mediators to change:         1
       Estimated space available:  22.70 GB
Estimated space to be consumed: 751.08 MB
         Create boot environment:       Yes
       Activate boot environment:       Yes
Create backup boot environment:        No
           Rebuild boot archive:        No
```

The following message displays at the end of the installation operation:

```
A clone of s11 exists and has been updated and activated.
On the next boot the Boot Environment s11amp will be
mounted on '/'.  Reboot when ready to switch to this updated BE.
```

The `pkg list` command reports that the `group/feature/amp` package is not installed because the `group/feature/amp` package is not installed in the current BE. The `group/feature/amp` package is installed in the new `s11amp` BE.

```
$ pkg list group/feature/amp
pkg list: no packages matching 'group/feature/amp' installed
```

Use the `beadm list` command to check that the system has a new active BE named `s11amp`. The "N" BE is currently booted. The "R" BE is the default on reboot. Use the `beadm activate` command to change which BE is the default on reboot.

```
$ beadm list
```

```
BE          Active Mountpoint Space   Policy Created
--          ------ ---------- -----   ------ -------
s11         N      /          30.92M  static 2014-03-05 08:51
s11amp      R      -          25.75G  static 2014-03-26 10:45
```

Check that the `group/feature/amp` package is installed in the new BE. Mount the new BE, and use the `-R` option to operate on the mounted BE. The "i" in the I column indicates that the `group/feature/amp` package is installed.

```
$ beadm mount s11amp /mnt
$ beadm list
BE          Active Mountpoint Space   Policy Created
--          ------ ---------- -----   ------ -------
s11         N      /          30.92M  static 2014-03-05 08:51
s11amp      R      /mnt       25.75G  static 2014-03-26 10:45
$ pkg -R /mnt list group/feature/amp
NAME (PUBLISHER)     VERSION                 IFO
group/feature/amp    0.5.11-0.175.2.0.0.33.0   i--
```

Remember to unmount the `s11amp` BE.

```
$ beadm unmount s11amp
```

Rejecting a Package

Use the `--reject` option of the `pkg install` command to prevent the specified packages from being installed. If matching packages are already installed, they are removed as part of this operation.

Rejected packages that are group dependencies are placed on the avoid list. See "Avoiding Installing Some Packages in a Group Package" on page 88 for information about the avoid list.

The following example command installs all group dependencies of the `developer-gnu` package except for the `cvs` package:

```
$ pkg install --reject developer/versioning/cvs group/feature/developer-gnu
```

Updating a Package

You can use either the `install` or `update` subcommand to update an installed package to the newest version of the package that is compatible with the rest of the image from the publisher that provided the currently installed version. To avoid unintentionally installing a package that was not already installed, use the `pkg update` command to update packages.

If the image has more than one publisher enabled, 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. To explicitly request the newest version of a package, use the keyword `latest` for the version portion of package name. See "Fault Management Resource Identifiers" on page 13 for a description of a package FMRI. See "Configuring Publishers" on page 71 for information about setting publisher stickiness and search order.

Any preserved configuration files that are part of packages to be updated are installed, saved, or renamed according to the value of the `preserve` attribute on the file and whether the file has changed. For information about how files are preserved during package updates, see the `preserve` attribute in the "File Actions" section of the `pkg`(5) man page.

See "Installing a New Package" on page 41 for information about publisher stickiness and search order and about using the `-g` option.

If you attempt to update a package that is not currently installed, the `pkg update` operation exits without updating any packages. Use the `--ignore-missing` option to ignore packages that are not installed and prevent `pkg update` from failing if some packages to update are not currently installed.

See Chapter 4, "Updating or Upgrading an Oracle Solaris Image" for information about the special behavior of the `pkg update` command when no package FMRI or pattern is specified, or if the pattern specified is an asterisk character (*).

Downgrading a Package

You can use the `pkg update` command to downgrade as well as upgrade packages. To downgrade a package, specify the package FMRI with a version older than the version that is currently installed. See "Fault Management Resource Identifiers" on page 13 for a description of a package FMRI. Use the `pkg list` command to see which version of the package is installed and which versions are available from configured publishers.

Any preserved configuration files that are part of packages to be downgraded are installed or renamed according to the value of the `preserve` attribute on the file and whether the file has changed. For information about how files are preserved during package downgrades, see the `preserve` attribute in the "File Actions" section of the `pkg`(5) man page.

See "Installing a New Package" on page 41 for information about using the `-g` option.

Fixing Problems With Installed Packages

IPS provides operations to validate that an installed package is installed correctly, fix any validation issues, and restore installed files to their packaged state.

Comparing the `pkg fix` and `pkg revert` Commands

Both the `pkg fix` command and the `pkg revert` command reinstall components of installed packages. The following table shows some of the similarities and differences between these two commands.

`pkg fix`	- Operates on packages. Takes one or more package names or patterns that match package names as operands. - Operates only on packages that fail `pkg verify`. - Fixes only errors reported by `pkg verify`. Does not redeliver other content or metadata from the package.
`pkg revert`	- Operates on files. Takes one or more file names or tag names as operands. - Redelivers files identified by the operands. Does not redeliver other content or metadata from the package.

Verifying Packages and Fixing Verification Errors

Use the `pkg verify` command to validate the installation of packages in the image. If the current signature policy for related publishers is not `ignore`, the signatures of each package are validated based on policy. See "Image Properties for Signed Packages" on page 93 for an explanation of how signature policies are applied. Verification of installed package content is based on a custom content analysis that might return different results than those of other programs.

If you do not provide a package name, all installed packages are examined. The `-v` option provides informational messages, at least one line for each installed package. The following example shows only a small sample of output. The installation of the `pkg/depot` package has an error.

```
$ pkg verify -v
PACKAGE                                                          STATUS
pkg://solaris/archiver/gnu-tar                                       OK
pkg://solaris/audio/audio-utilities                                  OK
pkg://solaris/benchmark/x11perf                                      OK
...
pkg://solaris/package/pkg/depot                                   ERROR
        dir: var/cache/pkg/depot
                Group: 'pkg5srv (97)' should be 'bin (2)'
        file: var/log/pkg/depot/access_log
                editable file has been changed
        file: var/log/pkg/depot/error_log
                editable file has been changed
...
pkg://solaris/security/sudo                                          OK
```

```
          file: etc/sudoers
                  editable file has been changed
...
pkg://solaris/x11/xlock                                          OK
pkg://solaris/x11/xmag                                           OK
pkg://solaris/x11/xvidtune                                       OK
```

Use the pkg fix command to fix package errors that are reported by the pkg verify command.

The pkg verify output shows that components of the installed sudo package are different from the packaged components but these differences are not reported as validation errors. The pkg fix makes no changes. The /etc/sudoers file is not replaced.

```
$ pkg fix pkg://solaris/security/sudo
No repairs for this image.
```

If you remove the /etc/sudoers file, the package fails validation and pkg fix replaces the file.

```
$ pkg fix pkg://solaris/security/sudo
Verifying: pkg://solaris/security/sudo                           ERROR
        file: etc/sudoers
                Missing: regular file does not exist
Created ZFS snapshot: 2014-03-13-22:05:42
Repairing: pkg://solaris/security/sudo
Creating Plan (Evaluating mediators):

DOWNLOAD                              PKGS        FILES    XFER (MB)   SPEED
Completed                            1/1          1/1      0.0/0.0  990B/s

PHASE                                     ITEMS
Updating modified actions                  1/1
Updating package state database            Done
Updating package cache                     0/0
Updating image state                       Done
Creating fast lookup database              Done
```

Only the missing file is replaced, as noted by the one file downloaded and one action (the file action) modified. Other sudo package content was not touched. The operation saved a snapshot of the current installation before performing the repair. See the "Created ZFS snapshot" line in the pkg fix output. The repair was performed in the current image.

```
$ zfs list -r rpool/ROOT/s11
NAME                             USED  AVAIL  REFER  MOUNTPOINT
rpool/ROOT/s11                   16.3G  22.5G  26.1G  /
rpool/ROOT/s11@2014-03-13-23:52:19   249M     -  26.1G  -
```

The pkg verify output shows an error in ownership of a directory in the installed pkg/depot package. The pkg fix output shows only the error in the "Verifying" section. The other differences with the packaged components are not shown.

```
$ ls -ld /var/cache/pkg/depot
drwxr-xr-x   3 pkg5srv  pkg5srv        3 Dec  2 19:47 /var/cache/pkg/depot/
$ pkg fix pkg://solaris/package/pkg/depot
Verifying: pkg://solaris/package/pkg/depot                       ERROR
```

```
          dir: var/cache/pkg/depot
                    Group: 'pkg5srv (97)' should be 'bin (2)'
Created ZFS snapshot: 2014-03-13-22:18:52
Repairing: pkg://solaris/package/pkg/depot
Creating Plan (Evaluating mediators):

PHASE                                    ITEMS
Updating modified actions                 1/1
Updating package state database          Done
Updating package cache                    0/0
Updating image state                     Done
Creating fast lookup database            Done
```

The following output shows that only the error has been fixed. The other differences between installed and packaged components remain.

```
$ ls -ld /var/cache/pkg/depot
drwxr-xr-x   3 pkg5srv  bin            3 Dec  2 19:47 /var/cache/pkg/depot/
$ pkg verify -v pkg://solaris/package/pkg/depot
PACKAGE                                                      STATUS
pkg://solaris/package/pkg/depot                                  OK
        file: var/log/pkg/depot/access_log
                editable file has been changed
        file: var/log/pkg/depot/error_log
                editable file has been changed
```

Restoring a File

Use the pkg revert command to restore files to their packaged condition. File ownership and protections are also restored.

> ⚠ **Caution -** Reverting some editable files can make the system unbootable, or cause other malfunctions.

Use the --require-backup-be option when reverting a key editable file.

Reverting Named Files

The following example specifies one of the two installed files from the pkg/depot package that are different from their packaged versions.

```
$ pkg revert -v /var/log/pkg/depot/access_log
              Packages to fix:         1
    Estimated space available:  21.08 GB
Estimated space to be consumed: 460.87 MB
         Create boot environment:        No
Create backup boot environment:        No
```

```
          Rebuild boot archive:      No

Changed packages:
solaris
  package/pkg/depot
    0.5.11,5.11-0.175.2.0.0.33.0:20140217T134751Z
DOWNLOAD                              PKGS        FILES    XFER (MB)   SPEED
Completed                             1/1          1/1     0.0/0.0    50B/s

PHASE                                       ITEMS
Updating modified actions                    1/1
Updating package state database              Done
Updating package cache                       0/0
Updating image state                         Done
Creating fast lookup database                Done
```

The specified file was replaced by the packaged version. No other components of the
pkg.depot package were changed.

Reverting Tagged Files and Directories

Use the --tagged option to perform the following operations:

- Revert all files tagged with the specified tag name.
- Remove any unpackaged files or directories that are under directories with the specified
  tag name and that match the specified pattern.

See the description of the revert-tag attribute in "File Actions" in "Packaging and Delivering
Software With the Image Packaging System in Oracle Solaris 11.2 " and "Directory Actions" in
"Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 "
for more information.

The following example shows some packages that contain files that are tagged with the dev-
init tag name:

```
$ pkg contents -o pkg.name,path -s pkg.name -t file -a revert-tag=dev-init '*'
PKG.NAME                   PATH
system/device-administration etc/mpxio/devid_path.cache
system/device-administration etc/dev/chassis_aliases
system/device-administration etc/dev/.chassis_aliases
system/device-administration etc/dev/reserved_devnames
system/kernel              etc/path_to_inst
system/network             etc/dladm/datalink.conf
```

The following command shows a preview of an operation that would revert all files that have
the dev-init tag name. Notice that the boot archive would be rebuilt. Using an option to create
a new boot environment or a backup boot environment is a good practice.

```
$ pkg revert -nv --tagged dev-init
            Packages to fix:        6
    Estimated space available:  22.39 GB
Estimated space to be consumed: 468.60 MB
```

```
        Create boot environment:       No
Create backup boot environment:       No
           Rebuild boot archive:      Yes

Changed packages:
solaris
  system/core-os
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T144208Z
  system/device-administration
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T144459Z
  system/io/usb
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T145048Z
  system/kernel
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T145214Z
  system/kernel/platform
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T145112Z
  system/network
    0.5.11,5.11-0.175.2.0.0.34.0:20140303T150218Z
```

Uninstalling Packages

Use the `pkg uninstall` command to remove installed packages.

If you want to undo a package installation operation, you can use the `pkg history` command to find exactly what was installed. See for more information.

If you attempt to uninstall a package that is not currently installed, the `pkg uninstall` operation exits without uninstalling any packages. Use the `--ignore-missing` option to ignore packages that are not installed and prevent `pkg uninstall` from failing if some packages to uninstall are not currently installed.

If a package is a group dependency, uninstalling the package places it on the avoid list. See for information about the avoid list.

Reinstalling an Image

If you know exactly what end result you want, and achieving that result requires a large number of packaging changes, such as uninstalling a large number of packages, you might want to use the `pkg exact-install` command. The result of the `pkg exact-install` command is an image with only the specified packages and their dependencies installed. Any currently installed packages that are not specified on the `pkg exact-install` command line and are not a dependency of the specified packages are removed.

The `pkg exact-install` command ignores restrictions to not install packages that are on the avoid list. If a package is on the avoid list, installing it removes it from that list. See "Avoiding Installing Some Packages in a Group Package" on page 88 for information about the avoid list. The `pkg exact-install` command ignores restrictions to not update packages that are on the frozen list. See "Locking Packages to a Specified Version" on page 82 for information about frozen packages.

Otherwise, the `exact-install` subcommand behaves the same way that the `install` subcommand behaves. Image variant and facet settings, image property settings, and publisher settings are retained. If any package cannot be installed in this image, then none of the specified packages is installed. Non-global zones are affected by any resulting package updates or removals as described in "Working with Non-Global Zones" on page 52. Note that the `-r` option is not available for `exact-install`.

See "Installing a New Package" on page 41 for information about publisher stickiness and search order and about using the `-g` option.

The following practices are recommended for using the `pkg exact-install` command:

- Check the versions of packages that are available from configured publishers by using `pkg list -a` as shown in "Installable Packages" on page 20. If you want to use `exact-install` to reinstall current versions, and newer versions are available to be installed, you must specify the version portion of the package FMRI in the list of packages to install.
- Include the `entire` incorporation package in the list of packages to install.
- Include one of the system group packages such as the `solaris-minimal-server` package in the list of packages to install.
- Run the command first with the `-nv` or `-nvv` options to see exactly what will be installed and what will be removed.
- Use the `--be-name` option to install into a new BE that has a meaningful name.

The following example creates a new image with a minimal installation at the same version as the current image:

```
$ pkg list -Hv entire
pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.34.0:20140303T182643Z
$ pkg exact-install --be-name s11.2 entire@0.5.11,5.11-0.175.2.0.0.34 solaris-minimal-
server
```

Working with Non-Global Zones

You can use most IPS commands in a non-global zone the same way you use the commands in the global zone. See "Images and Boot Environments" on page 16 for introductory information about zones.

With regard to package installation, the global zone and non-global zones have a parent-child relationship as described in "Relationship Between Global and Non-Global Zones" on page 53 and "Updating Multiple Non-Global Zones Concurrently" on page 56.

An important difference between the global zone and non-global zones is the use of package publishers as described in "The System Repository and Proxy Services" on page 54.

Relationship Between Global and Non-Global Zones

Installed `solaris` branded non-global zones can be affected by installing, updating, and uninstalling packages in the global zone.

Changing facets and variants can cause package installations and removals and affect non-global zones.

Non-global zones do not need to be booted to be updated from the global zone. Non-global zones only need to be installed to be affected by package changes in the global zone.

When you run installation and update commands in the global zone, by default the global zone and each installed non-global zone is updated serially, and the non-global zones are modified only to the extent required to keep the non-global zone compatible with the global zone.

- To perform the same operation in non-global zones that you perform in the global zone instead of performing only the minimal required updates in the non-global zones, use the `-r` option as described in "Options That Operate on Non-Global Zones" on page 39.

- To update non-global zones concurrently with the global zone, use the `-C` option as described in "Options That Operate on Non-Global Zones" on page 39 and shown in "Updating Multiple Non-Global Zones Concurrently" on page 56.

Tip - Use the `-nv` options to review what changes will be made in non-global zones as well as in the global zone.

When you run package commands while logged into a non-global zone, only that non-global zone is affected. Non-global zones can be different from their parent global zone in the following ways, for example:

- Different packages can be installed.
- Different versions of the same package can be installed if the result is compatible with the global zone.
- Different packages can be on the avoid list.

- Different packages can be frozen and can be frozen at different versions.
- Mediators can be set to select different default implementations.
- Different facets can be set.

Versions of packages installed in a non-global zone can be restricted by the versions installed in the global zone. Some packages cannot be updated or downgraded in a non-global zone because those packages must be the same version in the non-global zone as they are in the global zone. For example, the package named `entire` must be the same version in each non-global zone as in the global zone.

If a package that is installed in a non-global zone has a `parent` dependency, then updating that package in the global zone causes that package to be updated in the non-global zone. Packages that are dependents of packages that have `parent` dependencies are also affected.

Packages that are not affected by `parent` dependencies can be installed at a different version in a non-global zone than the version that is installed in the global zone. To install a different version in the non-global zone, specify the version in the `pkg install` command or freeze the version at the version you want.

See "Sync Linked Package Cannot Be Installed" on page 120 and "Non-Global Zone Cannot Be Installed" on page 121 for some help related to installing packages in non-global zones.

The System Repository and Proxy Services

In a non-global zone, the *system repository* provides access to the package repositories configured in the global zone. Publisher configuration changes made to the global zone are seen immediately by all non-global zones via the system repository.

A publisher origin or mirror that is configured in a non-global zone must be accessible from the global zone even if that location is not configured in the global zone publisher list. For example, if you have the `localsw` publisher configured in a non-global zone but not in the global zone, all origins and mirrors for the `localsw` publisher still must accessible from the global zone.

The system repository can proxy `http`, `https`, and `v4` file repositories and `.p5p` archive repositories.

The zones proxy is a service that enables `pkg` commands running inside a zone to communicate with the system repository, which is running in the global zone. The zones proxy has two parts. The following service runs in the global zone:

```
svc:/application/pkg/zones-proxyd:default
```

The following service runs in the non-global zone:

```
svc:/application/pkg/zones-proxy-client:default
```

See the `pkg.sysrepo`(1M) man page for more information about the system repository and zones proxy services.

The following example shows publishers in a global zone:

```
global:~$ pkg publisher
PUBLISHER            TYPE     STATUS P LOCATION
solaris              origin   online F http://pkg.oracle.com/solaris/release/
solaris              origin   online F file:///export/IPSpkgrepos/Solaris/
devtool  (disabled)  origin   online F http://pkg.example1.com/
isvpub               origin   online F http://pkg.example2.com/
```

The following example shows how these same publishers appear when you are logged into a non-global zone:

```
z1:~$ pkg publisher
PUBLISHER            TYPE     STATUS P LOCATION
solaris  (syspub)    origin   online T <system-repository>
isvpub   (syspub)    origin   online T <system-repository>
z1:~$ pkg publisher -F tsv
PUBLISHER  STICKY  SYSPUB  ENABLED  TYPE    STATUS  URI                                     PROXY
solaris    true    true    true     origin  online  http://pkg.oracle.com/solaris/release/  http:
//localhost:1008
isvpub     true    true    true     origin  online  http://pkg.example2.com/                http:
//localhost:1008
```

In the non-global zone, the system repository always shows as a proxy. This is the proxy the non-global zone uses to communicate with the system repository in the global zone.

You cannot reconfigure the system repository from within a non-global zone. For example, you cannot change the origins or properties of publishers or the publisher search order of publishers whose location is `<system-repository>`. If a publisher is added or reconfigured in the global zone, those changes are seen immediately by non-global zones. If a publisher is unset in the global zone, that publisher is unset in non-global zones unless the non-global zone has a package installed from that publisher.

Tip - Before you unset a publisher in the global zone, uninstall packages from that publisher in non-global zones.

If you cannot reach a publisher, you can set a proxy in the global zone, as described in "Specifying a Proxy" on page 77. For more information about setting proxies when you have non-global zones, including instructions for when and how to use the `http_proxy` and `https_proxy` environment variables, see "Proxy Configuration on a System That Has Installed Zones" in "Creating and Using Oracle Solaris Zones ".

For a publisher that is already configured in the global zone, the following `pkg list` command gives the same result in the both the global zone and non-global zones:

```
z1:~$ pkg list -a isvtool
```

```
NAME (PUBLISHER)    VERSION    IFO
isvtool (isvpub)    2.0        ---
isvtool (isvpub)    1.0        ---
```

Repositories can be network or file system accessible to the non-global zone even if those repositories are not configured in the global zone. The non-global zone publisher configuration must match the global zone publisher configuration or must be a superset of the global zone publisher configuration. For example, the `localsw` publisher could be configured in a non-global zone with an origin of `file:///export/IPSpkgrepos/localrepo` because that location is accessible in the global zone even if the `localsw` publisher is not configured in the global zone.

Updating Multiple Non-Global Zones Concurrently

By default, when you use the `pkg update` command in the global zone, the packaging system updates the global zone and each non-global zone serially. To update multiple non-global zones concurrently, use the `-C` option or set the `PKG_CONCURRENCY` environment variable in the global zone. See for more information.

In the following example, both non-global zones are updated at the same time as the global zone. The output refers to the non-global zones as linked images because they are linked to their parent global zone image.

```
global:~$ pkg update -C 0 --be-name s11.2
 Startup: Linked image publisher check ... Done
 Startup: Refreshing catalog 'solaris' ... Done
 Startup: Refreshing catalog 'isvpub' ... Done
 Startup: Checking that pkg(5) is up to date ... Done
Planning: Solver setup ... Done
Planning: Running solver ... Done
Planning: Finding local manifests ... Done
Planning: Package planning ... Done
Planning: Merging actions ... Done
Planning: Checking for conflicting actions ... Done
Planning: Consolidating action changes ... Done
Planning: Evaluating mediators ... Done
Planning: Planning completed in 39.00 seconds
            Packages to remove:   2
           Packages to install:   1
            Packages to update: 640
       Create boot environment: Yes
Create backup boot environment:  No

Planning: Linked images: 0/2 done; 2 working: zone:z1 zone:z2
Planning: Linked image 'zone:z1' output:
| Packages to install:   1
|  Packages to update: 161
|  Services to change:   2
`
Planning: Linked images: 1/2 done; 1 working: zone:z2
```

```
Planning: Linked image 'zone:z2' output:
| Packages to install:   1
|  Packages to update: 161
|  Services to change:   2
`
Planning: Finished processing linked images.
Download:      0/12068 items    0.0/350.9MB  0% complete
...
Download: 11664/12068 items  336.1/350.9MB  95% complete
Download: Completed 350.91 MB in 187.08 seconds (0B/s)
Download: Linked images: 0/2 done; 2 working: zone:z1 zone:z2
Download: Linked images: 1/2 done; 1 working: zone:z1
Download: Finished processing linked images.
 Actions:      1/23382 actions (Removing old actions)
 Actions:  3867/23382 actions (Installing new actions)
 Actions:  8192/23382 actions (Updating modified actions)
...
 Actions: 23266/23382 actions (Updating modified actions)
 Actions: Completed 23382 actions in 96.16 seconds.
Finalize: Updating package state database ...  Done
Finalize: Updating package cache ...  Done
Finalize: Updating image state ...  Done
Finalize: Creating fast lookup database ...  Done
Finalize: Reading search index ...  Done
Finalize: Building new search index ...  Done
Finalize: Linked images: 0/2 done; 2 working: zone:z1 zone:z2
Finalize: Linked images: 1/2 done; 1 working: zone:z2
Finalize: Finished processing linked images.

A clone of s11 exists and has been updated and activated.
On the next boot the Boot Environment s11u1 will be
mounted on '/'.  Reboot when ready to switch to this updated BE.
```

♦♦♦ **C H A P T E R 4**

4

Updating or Upgrading an Oracle Solaris Image

Chapter 3, "Installing and Updating Software Packages", discussed installing, updating, fixing, and uninstalling one or a few packages that are named on the command line. This chapter describes how to upgrade an Oracle Solaris image to the next support update or to the next release. Upgrading your system means updating a bootable image. A physical or virtual machine can have multiple bootable images, as noted in "Images and Boot Environments" on page 16.

Updating requires increased privileges. See "Installation Privileges" on page 18 for more information.

For a complete list of all options for the `pkg update` command, see the `pkg(1)` man page.

Image Update Overview

When you use the `pkg update` command with no package FMRI or pattern specified, or with an asterisk character (*) as the pattern, all installed packages that have updates available are updated to the newest version allowed by the constraints imposed on the system by installed package dependencies and publisher configuration.

- No new packages are installed unless they are new dependencies required by an updated installed package.
- Installed packages are updated only if a configured publisher provides an updated version that fits the constraints of the image being updated. Constraints are imposed by package dependencies and by configuration such as the following that you can control as described in Chapter 5, "Configuring Installed Images":
  - Packages locked at a particular version
  - Facets and variants set in the image
  - Package signing properties configured
  - Publisher search order and stickiness
- If any required package cannot be installed, no packages are updated or installed. See Appendix A, "Troubleshooting Package Installation and Update".

If non-global zones are installed in the current image, these zones are also updated. See "Working with Non-Global Zones" on page 52.

Image Update Best Practices

Before you update, perform the following steps:

- Read the release notes.
- Check the package versions that are available from your configured publisher origin. You might need to run the `pkg refresh` command on the publisher or run the `pkgrepo refresh` command on the origin location.
- Use the `pkg update` command with the `-nv` options to display the list of packages that will be updated without actually performing the update.

When you update, use the `--be-name` or `--require-new-be` option to make the changes in a new boot environment, not in the current boot environment. For more information, see "Boot Environment Options" on page 38 and "Boot Environment Policy Image Properties" on page 91.

Check Available Versions

If you want to update your operating system release, check the available versions of the `pkg:/entire` incorporation package. The following command shows that Oracle Solaris 11 11/11 SRU 10 is installed, Oracle Solaris 11 11/11 SRUs 11, 12, and 13 are available, and Oracle Solaris 11.1 is available from the currently configured `solaris` publisher. For information about fields in the FMRI, see "Fault Management Resource Identifiers" on page 13.

```
$ pkg list -af entire
NAME (PUBLISHER)  VERSION                   IFO
entire            0.5.11,5.11-0.175.1.0.0.24.2  ---
entire            0.5.11,5.11-0.175.0.13.0.4.0  ---
entire            0.5.11,5.11-0.175.0.12.0.4.0  ---
entire            0.5.11,5.11-0.175.0.11.0.4.1  ---
entire            0.5.11,5.11-0.175.0.10.0.5.0  i--
```

If none of these versions is what you want, then you need to set your `solaris` publisher origin to a different package repository location.

By default, each package is updated from the publisher that provided the currently installed version. You can control the publisher that provides packages by specifying publisher stickiness and search order. See "Adding, Modifying, or Removing Package Publishers" on page 73.

Preview the Update Operation

The following command shows which packages, if any, would actually be installed by an
update. Because the `-v` option is specified, this command shows the full FMRIs, including
versions, of all 627 packages that would be updated, the three packages that would be removed,
and the one new package that would be installed. This example omits most of that output and
only shows the `entire` package. Because the `-n` option is specified, no update is actually done.
Review this output before you perform an update without the `-n` option.

```
$ pkg update -nv
            Packages to remove:        3
           Packages to install:        1
            Packages to update:      627
     Estimated space available: 48.43 GB
Estimated space to be consumed:  3.14 GB
        Create boot environment:      Yes
      Activate boot environment:      Yes
Create backup boot environment:       No
          Rebuild boot archive:      Yes

Changed packages:
solaris
...
  entire
    0.5.11,5.11-0.175.0.10.0.5.0:20120803T182627Z -> 0.5.11,5.11-0.175.1.0.0.24.2:20120919T190135Z
...
```

The preceding example shows that the `pkg:/entire` incorporation package for Oracle Solaris
11.1 would be installed.

- All installed packages that are constrained by the `entire` incorporation would be updated
  accordingly.
- Because no package FMRI was specified, installed packages that are not constrained by
  the `entire` incorporation would also be updated. All installed packages would be updated
  to the newest version allowed by the constraints imposed on the system by installed
  package dependencies and publisher configuration.
- Installed packages might be removed and new packages might be installed if updated
  installed packages specify new dependencies.

Specify a New Boot Environment

The example in shows that a new BE would be
created for this update if you ran this command without the `-n` option. If you run this command
without the `-n` option, you see the following message at the end of the update output:

```
A clone of currentBE exists and has been updated and activated.
On the next boot the Boot Environment newBE will be
mounted on '/'.  Reboot when ready to switch to this updated BE.
```

The current BE is not modified. All changes are made in the new BE.

Explicitly specifying a new BE is the safest way to install or update. See "Boot Environment Policy Image Properties" on page 91 for information about when BEs are created. You might want to use the `--be-name` option to give the new BE a meaningful name. The new BE is activated so this new environment is booted by default the next time you boot the system. If you do not want the new BE to be the default on the next reboot, use the `--no-be-activate` option with the `pkg update` command. You can change the default boot BE at any time by using the `beadm activate` command.

If you are satisfied with your new BE, you can destroy your old one.

Tip - Keep an early BE for each operating system release. If necessary, you can boot back to the older BE and use it to update to a version between that version and the next newer version that you have installed.

Specifying the Version to Install

If you do not want to update to the newest version allowed, you can specify the package name on the `pkg update` command, including a portion of the version string. The following example shows how to specify the version of the `entire` incorporation to update to Oracle Solaris 11 11/11 SRU 13, even though a newer version would be allowed. Be sure to use the `-nv` options again and check the output again.

```
$ pkg update -nv entire@0.5.11,5.11-0.175.0.13
            Packages to remove:        2
           Packages to install:        1
            Packages to update:      486
       Estimated space available: 48.39 GB
Estimated space to be consumed:  2.50 GB
         Create boot environment:      Yes
      Activate boot environment:      Yes
Create backup boot environment:       No
           Rebuild boot archive:      Yes

Changed packages:
solaris
...
  entire
    0.5.11,5.11-0.175.0.10.0.5.0:20120803T182627Z -> 0.5.11,5.11-0.175.0.13.0.4.0:20121106T194623Z
...
```

Some installed packages might not be dependent on any package that is constrained by the `entire` incorporation. Those packages will not be updated by updating just the `entire` incorporation. You can add those packages by name to the same `pkg update` command.

Specifying a Version Constraint Prior to Updating

If you want to allow updates to any Oracle Solaris 11 11/11 version but not allow update to Oracle Solaris 11.1, you can freeze the `entire` incorporation as shown in the following command. Specifying `0.175.0` means the `entire` package can be updated to `0.175.0.13`, for example, but not to `0.175.1`.

```
$ pkg freeze -c "Keep this image at 11 11/11." entire@0.5.11,5.11-0.175.0
entire was frozen at 0.5.11,5.11-0.175.0
$ pkg freeze
NAME    VERSION             DATE                    COMMENT
entire  0.5.11,5.11-0.175.0 30 Jan 2013 15:50:01 PST Keep this image at 11 11/11.
$ pkg list entire
NAME (PUBLISHER)  VERSION                   IFO
entire            0.5.11,5.11-0.175.0.10.0.5.0  if-
```

For more information about package freezing, see "Locking Packages to a Specified Version" on page 82.

Installing a Custom Incorporation

Similar to using the `pkg freeze` command as shown in the previous section, you can use your own custom incorporation package to specify the constraints you want. See "Incorporation Packages" on page 12 for more information about incorporation packages and their use in Oracle Solaris.

Create the incorporation package and install the package from a local IPS package repository or package archive file. To change the constraints, modify and redeliver the custom incorporation package, and use `pkg update` to install the new incorporation.

Using a custom incorporation to control the version of software that can be installed enables you to easily maintain different versions of Oracle Solaris on different machines without maintaining multiple package repositories. Each image can install a different version of the custom update control incorporation package. All systems share the same package repository that contains all versions of software needed by any of the systems.

Create a Custom Incorporation Package

The versions of core operating system packages that can be installed in an image are controlled by the `pkg:/entire` incorporation package. To control system upgrades, create a package that specifies a particular version of the `pkg:/entire` package as an `incorporate` dependency.

Create the Custom Incorporation Package Manifest

The following example shows a manifest named `upgradectrl.p5m` for a custom incorporation package that controls the version of the `pkg:/entire` package that can be installed. Some of the settings in this manifest are described below.

```
set name=pkg.fmri value=upgradectrl@1.0
set name=pkg.summary value="Incorporation to constrain the version of the OS"
set name=pkg.description value="This package controls the version of \
pkg://solaris/entire that can be installed."
set name=info.classification value="org.opensolaris.category.2008:Meta Packages/Incorporations"
set name=pkg.depend.install-hold value=core-os
set name=variant.opensolaris.zone value=global value=nonglobal
set name=variant.arch value=sparc value=i386
depend fmri=feature/package/dependency/self type=parent variant.opensolaris.zone=nonglobal
depend fmri=pkg://solaris/entire type=require
depend fmri=pkg://solaris/entire@0.5.11,5.11-0.175.1.0 type=incorporate
```

`pkg.depend.install-hold`

    If a user enters the `pkg update upgradectrl` command, the `pkg:/entire` package is automatically updated as well.

`variant.opensolaris.zone`

    This package can be installed in both global and non-global zones. See also the description of the `parent` dependency.

`variant.arch`

    This package can be installed on both SPARC and x86 systems.

`parent` dependency

    This package can be installed in a non-global zone only if it is already installed in the global zone.

`require` dependency

    The `upgradectrl` package can be installed only if the `pkg://solaris/entire` package is already installed or can be installed in this same operation.

`incorporate` dependency

    The `pkg://solaris/entire` package must be installed at the specified version. More than one version can satisfy an `incorporate` dependency, depending on how many places of accuracy are specified. In this example, `0.175.1.0` specifies Oracle Solaris 11.1 SRU 0. This upgrade control package will keep systems at Oracle Solaris 11.1 with no support updates. This upgrade control package will, however, allow packages that are not constrained by the `pkg:/entire` incorporation to be updated.

Publish the Upgrade Control Package

Publish the upgradectrl package to a local file-based repository. This repository is for developing and testing this new package. If you create a repository for general use, you should include additional steps such as creating a separate file system for the repository. For information about creating package repositories for general use, see "Copying and Creating Package Repositories in Oracle Solaris 11.2 ".

Create a package development repository on your system. See the pkgrepo(1) man page for more information about the pkgrepo command.

```
$ pkgrepo create myrepo
```

Set the default publisher for this repository. The default publisher is the value of the publisher/prefix property of the repository.

```
$ pkgrepo -s myrepo set publisher/prefix=site
```

Publish the upgradectrl package to the development repository.

```
$ pkgsend -s myrepo publish upgradectrl.p5m
pkg://site/upgradectrl@1.0,5.11:20131104T072336Z
PUBLISHED
```

Notice that the repository default publisher has been applied to the package FMRI.

Examine the repository to confirm that the package was published.

```
$ pkgrepo -s myrepo list
PUBLISHER NAME                    0 VERSION
site      upgradectrl               1.0,5.11:20131104T072336Z
$ pkg list -vg myrepo
FMRI                                            IFO
pkg://site/upgradectrl@1.0,5.11:20131104T072336Z      ---
```

Deliver the package to a local repository in a separate ZFS file system in a shared location.

```
$ pkgrecv -s myrepo -d /export/IPSpkgrepos/Solaris upgradectrl
Processing packages for publisher site ...
Retrieving and evaluating 1 package(s) ...
PROCESS      ITEMS    GET (MB)    SEND (MB)
Completed    1/1    0.0/0.0     0.0/0.0
```

Verify the package in the repository and the version of pkg:/entire that it incorporates.

```
$ pkg info -g /export/IPSpkgrepos/Solaris upgradectrl
         Name: upgradectrl
      Summary: Incorporation to constrain the version of the OS
  Description: This package controls the version of pkg://solaris/entire that
               can be installed.
     Category: Meta Packages/Incorporations
        State: Not installed
    Publisher: site
      Version: 1.0
```

```
 Build Release: 5.11
        Branch: None
Packaging Date: November 20, 2013 01:01:05 AM
          Size: 0.00 B
          FMRI: pkg://site/upgradectrl@1.0,5.11:20131120T010105Z
$ pkg contents -Hro fmri -t depend -a type=incorporate upgradectrl
pkg://solaris/entire@0.5.11,5.11-0.175.1.0
```

See "Creating and Publishing a Package" in "Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.2 " for more detailed information about creating and delivering IPS packages.

Set the Publisher Origin

Set the origin for the site publisher. The system repository is automatically updated with this information so that non-global zones can access packages from the site publisher.

```
$ pkg set-publisher -g /export/IPSpkgrepos/Solaris site
$ pkg publisher
PUBLISHER           TYPE     STATUS P LOCATION
solaris             origin   online F https://pkg.oracle.com/solaris/support/
site                origin   online F file:///export/IPSpkgrepos/Solaris/
```

Install the Upgrade Control Package

Install the package. In this case, few changes should be made because the installed version of pkg:/entire is the same as the version incorporated by the upgrade control package. Notice that the package is also installed in the non-global zone.

```
$ pkg list -v entire
FMRI                                                             IFO
pkg://solaris/entire@0.5.11,5.11-0.175.1.0.0.24.2:20120919T190135Z        i--
$ zoneadm list
global
z1
$ pkg install upgradectrl
          Packages to install:  1
       Create boot environment: No
Create backup boot environment: No

Planning linked: 0/1 done; 1 working: zone:z1
Planning linked: 1/1 done
Downloading linked: 0/1 done; 1 working: zone:z1
Downloading linked: 1/1 done
PHASE                                     ITEMS
Installing new actions                      9/9
Updating package state database            Done
Updating image state                       Done
Creating fast lookup database              Done
Reading search index                       Done
```

```
Updating search index                          1/1
Executing linked: 0/1 done; 1 working: zone:z1
Executing linked: 1/1 done
```

The following commands show that versions of pkg:/entire that are newer than the installed version are available from the configured solaris publisher, but an attempt to upgrade is controlled by the newly installed upgrade control package.

```
$ pkg list -af entire
NAME (PUBLISHER)                                VERSION              IFO
entire                                          0.5.11-0.175.1.13.0.6.0    ---
entire                                          0.5.11-0.175.1.12.0.5.0    ---
entire                                          0.5.11-0.175.1.11.0.4.0    ---
entire                                          0.5.11-0.175.1.10.0.6.0    ---
entire                                          0.5.11-0.175.1.10.0.5.0    ---
...
$ pkg update
pkg update: No solution was found to satisfy constraints
Plan Creation: Package solver has not found a solution to update to latest available versions.
This may indicate an overly constrained set of packages are installed.
latest incorporations:
...
Try specifying expected results to obtain more detailed error messages.
$ pkg update -nv entire@0.5.11-0.175.1.13.0.6.0
pkg update: No matching version of entire can be installed:
  Reject:  pkg://solaris/entire@0.5.11,5.11-0.175.1.13.0.6.0:20131108T211557Z
  Reason:  This version is excluded by installed incorporation pkg://site/
upgradectrl@1.0,5.11:20131120T010105Z
```

Update the Upgrade Control Package

When you are ready to allow users to update their systems to a new version, update the upgradectrl.p5m manifest, and republish and redeliver the new update control package. In the following manifest, the version of the update control package and the version of the entire incorporation are updated. As an aid for users, the version of the upgrade control package, 1.10, is set to match the updated version of the pkg:/entire package, 0.175.1.10.

```
set name=pkg.fmri value=upgradectrl@1.10
set name=pkg.summary value="Incorporation to constrain the version of the OS"
set name=pkg.description value="This package controls the version of \
pkg://solaris/entire that can be installed."
set name=info.classification value="org.opensolaris.category.2008:Meta Packages/Incorporations"
set name=pkg.depend.install-hold value=core-os
set name=variant.opensolaris.zone value=global value=nonglobal
set name=variant.arch value=sparc value=i386
depend fmri=feature/package/dependency/self type=parent variant.opensolaris.zone=nonglobal
depend fmri=pkg://solaris/entire type=require
depend fmri=pkg://solaris/entire@0.5.11,5.11-0.175.1.10 type=incorporate
```

The following commands republish and redeliver the update control package:

```
$ pkgsend -s myrepo publish upgradectrl.p5m
```

```
pkg://site/upgradectrl@1.10,5.11:20131120T021902Z
PUBLISHED
$ pkgrepo -s myrepo list
PUBLISHER NAME                              O VERSION
site     upgradectrl                          1.10,5.11:20131120T021902Z
site     upgradectrl                          1.0,5.11:20131120T010105Z
$ pkgrecv -s myrepo -d /export/IPSpkgrepos/Solaris upgradectrl
Processing packages for publisher site ...
Retrieving and evaluating 1 package(s)...
PROCESS                            ITEMS    GET (MB)   SEND (MB)
Completed                           1/1     0.0/0.0     0.0/0.0
$ pkg refresh site
$ pkg list -af pkg://site/upgradectrl
NAME (PUBLISHER)                          VERSION             IFO
upgradectrl (site)                        1.10                ---
upgradectrl (site)                        1.0                 i--
```

Upgrade the Image

The following pkg update command updates all packages to the newest available versions allowed because no packages are specified. The command updates to the newest available version of the upgrade control package, which upgrades the image because the pkg.depend.install-hold setting in the upgradectrl package causes the pkg:/entire package to be updated when the upgradectrl package is updated. The image is upgraded to the version of the pkg:/entire incorporation that is specified in the new upgradectrl incorporation.

```
$ pkg update --be-name s11u1_10
            Packages to remove:    1
            Packages to update: 186
          Mediators to change:    1
      Create boot environment: Yes
Create backup boot environment:  No

Planning linked: 0/1 done; 1 working: zone:z1
Linked image 'zone:z1' output:
| Packages to remove:  1
| Packages to install:  3
| Packages to update: 73
| Mediators to change:  1
|  Services to change:  3
`
Planning linked: 1/1 done
DOWNLOAD                          PKGS          FILES     XFER (MB)    SPEED
Completed                       187/187    16139/16139   507.9/507.9  562k/s

Downloading linked: 0/1 done; 1 working: zone:z1
Downloading linked: 1/1 done
PHASE                                  ITEMS
Removing old actions              1473/1473
Installing new actions            3451/3451
Updating modified actions       16378/16378
Updating package state database         Done
```

```
Updating package cache                      187/187
Updating image state                            Done
Creating fast lookup database                   Done
Reading search index                            Done
Building new search index                    851/851
Executing linked: 0/1 done; 1 working: zone:z1
Executing linked: 1/1 done

A clone of s11u1_0 exists and has been updated and activated.
On the next boot the Boot Environment s11u1_10 will be
mounted on '/'.  Reboot when ready to switch to this updated BE.
```

Verify that the current BE is not changed and the new BE contains the updated packages.

```
$ pkg list entire upgradectrl
NAME (PUBLISHER)                           VERSION               IFO
entire                                     0.5.11-0.175.1.0.0.24.2   i--
upgradectrl (site)                         1.0                   i--
$ beadm mount s11u1_10 /mnt
$ pkg -R /mnt list entire upgradectrl
NAME (PUBLISHER)                           VERSION               IFO
entire                                     0.5.11-0.175.1.10.0.6.0   i--
upgradectrl (site)                         1.10                  i--
$ beadm unmount s11u1_10
```

Downgrading an Image

To downgrade your operating system release, boot into a BE older than the version you want to downgrade to, and upgrade from there. For example, if you updated from Oracle Solaris 11 11/11 SRU 10 to Oracle Solaris 11 11/11 SRU 13 and then realized you need an SRU 12 image, reboot to your SRU 10 BE and update to SRU 12 from there.

5

Configuring Installed Images

This chapter describes how to configure characteristics that apply to an entire image, such as configuring package publishers, restricting which packages can be installed, setting package signing policy, and configuring boot environment (BE) policy.

- Configuring publishers, including setting origins, search order, keys and certificates, and proxies
- Controlling installation of optional components by setting variants and facets
- Locking packages to a specified version
- Relaxing version constraints specified by incorporations
- Specifying the default implementation for an application by using a mediation
- Avoiding installing some packages in a group package
- Configuring image and publisher properties, including BE creation policy and package signing policy
- Creating an image
- Viewing package operation history

Many of these operations require increased privileges. See "Installation Privileges" on page 18 for more information.

For a complete list of all options for commands discussed in this chapter, see the pkg(1) man page.

Configuring Publishers

To install and update software, the pkg client must be able to contact a package repository.

Displaying Publisher Information

Use the `pkg publisher` command to display information about package publishers configured for this image. The publishers are listed in the order in which they are searched to find packages when the publisher is not specified in the package FMRI.

By default, the `solaris` publisher is configured on a newly installed Oracle Solaris 11 system. Use the `pkg publisher` command to check the origins of your publishers.

```
$ pkg publisher
PUBLISHER                   TYPE     STATUS P LOCATION
solaris                     origin   online F http://pkg.oracle.com/solaris/release/
isvpub        (non-sticky)  origin   online F file:///export/IPSpkgrepos/isvrepo/
devtool       (disabled)    origin   online F http://pkg.example1.com/
```

The `TYPE` column indicates whether the `LOCATION` value is an origin or a mirror. See "Repository Origins and Mirrors" on page 16 for descriptions.

Between the `STATUS` and `LOCATION` columns, the `P` column specifies whether the location is proxied. Values in this column are true (`T`) or false (`F`). File repositories are not proxied. HTTP repositories with the value `T` are proxied using the proxy specified with the `--proxy` option when the origin was added with the `pkg set-publisher` command. When you specify the `-F tsv` option to `pkg publisher`, the `PROXY` column displays any proxy that is set for that location.

```
$ pkg publisher -F tsv
PUBLISHER STICKY SYSPUB ENABLED TYPE   STATUS URI                                    PROXY
solaris   true   false  true    origin online http://pkg.oracle.com/solaris/release/ -
isvpub    false  false  true    origin online file:///export/IPSpkgrepos/isvrepo/    -
devtool   true   false  false   origin online http://pkg.example1.com/               -
```

An `F` in the `P` column or `-` in the `PROXY` column indicates that the location was not proxied by using the `pkg set-publisher` command. If the location is proxied by setting an `http_proxy` environment variable, the `pkg publisher` output still shows `F` or `-`. See "Specifying a Proxy" on page 77 for information about different ways to set a proxy.

Specify publishers by name to display detailed configuration for those publishers.

```
$ pkg publisher solaris
       Publisher: solaris
           Alias:
      Origin URI: http://pkg.oracle.com/solaris/release/
         SSL Key: None
        SSL Cert: None
     Client UUID: e15e3228-eada-11df-80ab-8023183d954b
 Catalog Updated: March 4, 2014 11:48:02 PM
         Enabled: Yes
      Properties:
                  proxied-urls = []
```

Use the `-P` option to display only the first publisher in the publisher search order.

```
$ pkg publisher -P
PUBLISHER                 TYPE    STATUS P LOCATION
solaris                   origin  online F http://pkg.oracle.com/solaris/release/
```

Use the -n option to display only enabled publishers.

```
$ pkg publisher
PUBLISHER                 TYPE    STATUS P LOCATION
solaris                   origin  online F http://pkg.oracle.com/solaris/release/
isvpub        (non-sticky) origin  online F file:///export/IPSpkgrepos/isvrepo/
```

Adding, Modifying, or Removing Package Publishers

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

- Configure a new publisher
- Set publisher origins and mirrors
- Set publisher stickiness
- Set publisher search order
- Set and unset a publisher property, and add and remove a publisher property value
- Specify SSL keys and certificates for a publisher
- Set a publisher proxy
- Enable or disable a publisher
- Remove a publisher

The pkg set-publisher command has two forms. See the pkg(1) man page for details.

- In one form, the name of the publisher is a required operand.
- In the other form, a repository URI is provided as the argument to the -p option, and publisher information is retrieved from that specified repository. The publisher name is an optional operand so that you can configure only the named publisher if multiple publishers publish packages to that repository.

Adding Publishers

The examples below show both methods for adding a publisher.

EXAMPLE 5-1   Specify a New Publisher

The following command adds a new publisher named devtool with an origin URI specified with the -g option and sets this publisher to be first in the search order. Use the -P option or the --search-first option to set the specified publisher first in the search order.

```
$ pkg set-publisher -P -g http://pkg.example1.com/release/ devtool
```

EXAMPLE 5-2 Import Publisher Configuration

Use the `-p` option to retrieve publisher configuration information from the specified repository URI. If a publisher is specified, then only the matching publisher is added or updated. If no publisher is specified, all publishers are added or updated as appropriate.

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
$ pkg set-publisher -p /export/IPSpkgrepos/myrepo
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
site                      origin   online F file:///export/IPSpkgrepos/myrepo/
```

Adding and Changing Publisher Origins

The following commands show adding an origin to the solaris publisher. If multiple origins are configured for a given publisher in an image, the IPS client attempts to choose the best origin from which to retrieve package data.

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
$ pkg set-publisher -g /export/IPSpkgrepos/Solaris solaris
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
solaris                   origin   online F file:///export/IPSpkgrepos/Solaris/
```

Use the `-G` option to remove a URI as an origin for the specified publisher.

To change an origin URI for a publisher, add the new URI and remove the old URI.

```
$ pkg set-publisher -G '*' -g file:///export/IPSpkgrepos/isvrepo/ isvpub
```

Adding and Changing Publisher Mirrors

Use the `-m` option to add a URI as a mirror for the specified publisher. See "Repository Origins and Mirrors" on page 16 for an explanation of the difference between an origin and a mirror. You cannot access the content in a mirror repository unless the same version of the same package also exists in an origin repository for that same publisher.

```
$ pkg set-publisher -m http://pkg.example3.com/ devtool
$ pkg publisher
```

```
PUBLISHER               TYPE    STATUS P LOCATION
devtool                 origin  online F http://pkg.example1.com/
devtool                 mirror  online F http://pkg.example3.com/
```

Use the `-M` option to remove a URI as a mirror for the specified publisher.

To change a mirror URI for a publisher, add the new URI and remove the old URI.

Setting Publisher Search Order and 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. Use the `--sticky` and `--non-sticky` options to set publisher stickiness.

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. Use the `--search-before`, `--search-after`, and `--search-first` options for changing publisher search. The `-P` option is a synonym for the `--search-first` option.

The first publisher that provides a matching package is used as the installation source. If that publisher does not provide a version of the package that can be installed in this image, then the installation operation fails. To install from a publisher further down the search order, provide more information in the package FMRI, such as the publisher name or the package version string.

Configuring Publisher Properties

Use the following options to set and unset publisher properties and to add and remove values of publisher properties:

- `--set-property` *property=value*
- `--add-property-value` *property=value*
- `--remove-property-value` *property=value*
- `--unset-property` *property*

The `publisher-search-order` and `signature-required-names` properties can take multiple values.

See the `pkg set-publisher` examples in "Configuring Package Signature Properties" on page 94.

Configuring Publisher Keys and Certificates

EXAMPLE 5-3 Specify a Publisher Key and Certificate

Use the `-k` option to specify the client SSL key. Use the `-c` option to specify the client SSL certificate. 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 `pkg publisher` command for this publisher. See "Displaying Publisher Information" on page 72.

```
$ pkg set-publisher -k /root/creds/example.key -c /root/creds/example.cert \
--approve-ca-cert /tmp/example_file.pem isvpub
```

EXAMPLE 5-4 Revoke a Publisher Key and Certificate

Use the `--revoke-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 `pkg publisher` command for this publisher.

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

Configuring a Publisher Proxy

Use the `--proxy` option to specify a persistent proxy URI from which to retrieve content for the specified origin or mirror. The value of the proxy is *protocol*`://`*host*[`:`*port*], where *protocol* is `http` or `https` and `:`*port* is optional. See "Specifying a Proxy" on page 77 for information about different ways to set a proxy.

Enabling and Disabling Publishers

A newly added publisher is enabled by default. A disabled publisher is not used when populating the package list or in `install`, `uninstall`, or `update` package operations. The properties for a disabled publisher can still be set and viewed. If only one publisher is enabled, that publisher cannot be disabled.

The following command enables the `isvpub` publisher and sets it ahead of the `devtool` publisher in the search order.

```
$ pkg set-publisher --enable --search-before devtool isvpub
```

Use the `--disable` option to disable a publisher. You might want to disable a publisher if the publisher origin is temporarily unreachable, for example. If any publisher is unreachable, package installation and update operations fail.

Removing a Publisher

Use the pkg unset-publisher command to remove a publisher.

```
$ pkg unset-publisher devtool
```

Specifying a Proxy

The methods for setting a proxy have different effects and advantages. For example, the pkg set-publisher command stores the proxy as part of the publisher configuration, while the http_proxy environment variables enable you to set authenticated proxies.

Using the pkg set-publisher Command to Set a Proxy

The --proxy option of the pkg set-publisher command sets a persistent proxy URI for the specified publisher origin and mirror URIs. The proxy value is stored as part of the publisher configuration. Storing the proxy value as part of the publisher configuration automatically updates the system repository that is used by child images. Storing the proxy value as part of the publisher configuration also means you could use different proxies for different publishers.

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F file:///export/IPSpkgrepos/Solaris/
$ pkg publisher -F tsv
PUBLISHER STICKY SYSPUB ENABLED TYPE   STATUS URI                              PROXY
solaris   true   false  true    origin online file:///export/IPSpkgrepos/Solaris/ -
$ pkg set-publisher -g http://pkg.oracle.com/solaris/release/ --proxy proxyURI solaris
$ pkg publisher solaris
      Publisher: solaris
          Alias:
     Origin URI: file:///export/IPSpkgrepos/Solaris/
        SSL Key: None
       SSL Cert: None
     Origin URI: http://pkg.oracle.com/solaris/release/
          Proxy: proxyURI
        SSL Key: None
       SSL Cert: None
    Client UUID: e15e3228-eada-11df-80ab-8023183d954b
Catalog Updated: July 11, 2013 11:32:46 PM
        Enabled: Yes
     Properties:
                proxied-urls = []
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F file:///export/IPSpkgrepos/Solaris/
```

```
solaris                    origin  online T http://pkg.oracle.com/solaris/release/
$ pkg publisher -F tsv
PUBLISHER STICKY SYSPUB ENABLED TYPE   STATUS URI                                  PROXY
solaris   true   false  true    origin online file:///export/IPSpkgrepos/Solaris/ -
solaris   true   false  true    origin online http://pkg.oracle.com/solaris/release/ proxyURI
```

If this image has non-global zones, the system repository is automatically updated with this proxy information; you do not need to set properties in the `system-repository` service. If you examine the publisher proxy in the non-global zone, you do not see the same proxy URI that you see in the global zone. In the global zone, the system repository uses the proxy URI. In the non-global zone, the system repository acts as a proxy itself, enabling the non-global zone to communicate with the system repository in the global zone. "Relationship Between Global and Non-Global Zones" on page 53 shows an example of what the system repository publisher looks like in the non-global zone.

The `--proxy` option of the `pkg set-publisher` command cannot be used to set an authenticated proxy. The value of the `--proxy` option cannot have the form *protocol*://*user*:*password*@*host*.

Using Environment Variables to Set a Proxy

Proxy environment variable values apply to all URIs of that protocol. At runtime, values of `http_proxy` environment variables override values set with the `--proxy` option of the `pkg set-publisher` command. See the ENVIRONMENT section of the `curl`(1) man page for additional information about proxy environment variables.

If you set `http_proxy` environment variables in an image that has non-global zones, then in the global zone, set the proxy properties in the `svc:/application/pkg/system-repository` SMF service to these same values and refresh the service.

```
$ svccfg -s system-repository:default setprop config/http_proxy = astring: proxyURI
$ svccfg -s system-repository:default listprop config/*proxy
config/https_proxy astring
config/http_proxy  astring      proxyURI
$ svcprop system-repository:default | grep proxy
config/https_proxy astring ""
config/http_proxy astring ""
$ svcadm refresh system-repository:default
$ svcprop system-repository:default | grep proxy
config/https_proxy astring ""
config/http_proxy astring proxyURI
```

The `pkg publisher` command does not show proxies that were set by setting environment variables or SMF service properties.

If you change the value of an `http_proxy` environment variable, be sure to update the `system-repository` service properties and refresh the service.

Controlling Installation of Optional Components

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*.

You can display the values of variants and facets that are set on the current image, and you can change variants and facets in the current image. To view the current values of the facets and variants set on the image, use the `pkg facet` and `pkg variant` commands. To modify the values of the facets and variants set on the image, use the `pkg change-facet` and `pkg change-variant` commands. See the `pkg`(1) man page and the examples below. Changing variants and facets might update a large number of packages and might require a new BE. Use `-nv` to review what changes will be made before you make any changes.

How Variant and Facet Values Affect Package Installation

Facets and variants are special properties set on the image and are tags set on actions in a package manifest. The values of facet and variant tags on an action compared with the values of facets and variants set in the image determine whether that package action can be installed.

Each facet and variant tag has a name and a value. 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, or a component that is only for a SPARC global zone.

Most variant tags can have various values. The value of a variant tag must be set on the image in order for a package that specifies the variant value to be installed. The `arch` and `zone` variants are set by the program that creates the image and installs its initial content. The `debug.*` variants are `false` in the image by default.

Facet tags set on an action can only have the value `true`. If you set a particular facet value to `false` in the image, any files or other actions that specify that facet will not be installed, and currently installed files that specify that facet are uninstalled.

The following algorithm describes how the facets and variants set on the image affect whether a particular action is installed.

- Actions with no facet or variant tags are always installed.
- Actions with facet tags are installed unless all of the facets or facet patterns matching the tags are set to `false` on the image. If any facet is set to `true` or is not explicitly set (`true` is the default), then the action is installed.

- Actions with variant tags are installed only if the values of all the variant tags are the same as the values set in the image.
- Actions with both facet and variant tags are installed if both the facets and the variants allow the action to be installed.

Example Variant and Facet Values

Most variants can have any number of values. For example, the `arch` variant can be set to `i386`, `sparc`, `ppc`, `arm`, or whatever architectures the distribution supports. (Only `i386` and `sparc` are used in Oracle Solaris.) The exception are the `debug` variants. The `debug` variants can only be set to `true` or `false`; other values have undefined behavior. If a `file` action has both non-debug and debug versions, both versions must have the applicable `debug` variant explicitly set, as shown in the following example:

```
file group=sys mode=0644 overlay=allow owner=root \
  path=etc/motd pkg.csize=115 pkg.size=103 preserve=true \
  variant.debug.osnet=true

file group=sys mode=0644 overlay=allow owner=root \
  path=etc/motd pkg.csize=68 pkg.size=48 preserve=true \
  variant.debug.osnet=false
```

The following variant tags are in common use in Oracle Solaris.

Variant Name	Possible Values
`variant.arch`	`sparc`, `i386`
`variant.opensolaris.zone`	`global`, `nonglobal`
`variant.debug.*`	`true`, `false`

Facets are Boolean: They can be set only to `true` (enabled) or `false` (disabled). By default, all facets are considered to be set to `true` in the image. A facet tag on an action should only have the value `true`; other values have undefined behavior.

A facet set on the image can be a full facet such as `doc.man` or a pattern such as `locale.*`. This flexibility is useful when you want to disable a portion of the facet namespace and only enable individual facets within it. For example, you could disable all locales and then only enable one or two specific locales, as shown in the following example:

```
$ pkg change-facet 'locale.*=false'
[output about packages being updated]
$ pkg change-facet locale.en_US=true
[output about packages being updated]
```

The following list shows a small sample of the facet tags that are used in Oracle Solaris:

```
facet.devel            facet.doc
facet.doc.html         facet.doc.info
facet.doc.man          facet.doc.pdf
facet.locale.de        facet.locale.en_GB
facet.locale.en_US     facet.locale.fr
facet.locale.ja_JP     facet.locale.zh_CN
```

Showing and Changing Variant Values

Use the `pkg variant` command to display the values of variants that are set.

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

The `-v` option includes the possible variant values that can be set for installed packages.

```
$ pkg variant -v
VARIANT                 VALUE
variant.arch            i386
variant.arch            sparc
variant.debug.osnet     false
variant.debug.osnet     true
variant.opensolaris.zone  global
variant.opensolaris.zone  nonglobal
```

Use the `pkg change-variant` command to change the value of a variant. Use the `pkg variant -v` command to select a value to set.

The following command produces a large amount of output because so many packages would be affected. Notice that a new BE would not be created by default, but a backup BE would be created. See "Boot Environment Policy Image Properties" on page 91 for information about when BEs are created. The `-n` option shows what would change if you performed the operation without `-n`, but the command makes no actual changes.

```
$ pkg change-variant -nv --accept 'variant.debug.*=true'
            Packages to update:       851
    Variants/Facets to change:         3
     Estimated space available:  49.88 GB
Estimated space to be consumed: 270.57 MB
       Create boot environment:        No
Create backup boot environment:       Yes
          Rebuild boot archive:        No

Changed variants/facets:
  variant variant.debug.*: true
    facet facet.locale.en_US: None
    facet facet.locale.*: None
Changed packages:
solaris
```

```
...
```

Showing and Changing Facet Values

Use the `pkg facet` command to display the current values and source of all facets that either have been set locally in this image by using the `pkg change-facet` command or have been inherited from a parent image. For example, a non-global zone inherits facet settings from the global zone.

```
$ pkg facet
FACETS             VALUE
facet.locale.en_US True
facet.locale.en    True
facet.locale.*     False
```

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

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

The following command produces a large amount of output because so many packages would be affected. Notice that a new BE would not be created by default, but a backup BE would be created. See "Boot Environment Policy Image Properties" on page 91 for information about when BEs are created. The `-n` option shows what would change if you performed the operation without `-n`, but the command makes no actual changes.

```
$ pkg change-facet -nv 'facet.locale.*=true'
            Packages to update:     851
     Variants/Facets to change:       1
     Estimated space available: 49.88 GB
Estimated space to be consumed:  3.13 GB
        Create boot environment:      No
Create backup boot environment:     Yes
          Rebuild boot archive:      No

Changed variants/facets:
    facet facet.locale.*: True
Changed packages:
solaris
  ...
```

Locking Packages to a Specified Version

Use the `pkg freeze` command to constrain a package version.

If no version is provided in the package operand, the named package must be installed and is constrained to the version installed on the system. If a version is provided in the package

operand, then this constraint, or freeze, acts as if an `incorporate` dependency were installed where the `fmri` attribute had the value of the specified package version.

When a package that is frozen is installed or updated, it must end up at a version that matches the version at which it was frozen. For example, if a package was frozen at 1.2, then it could be updated to 1.2.1, 1.2.9, 1.2.0.0.1, and so on. That package could not end up at 1.3, or 1.1.

A publisher that is specified in the package operand is used to find matching packages. However, publisher information is not recorded as part of the freeze. A package is frozen with respect to its version only, not its publisher.

Freezing a package that is already frozen replaces the frozen version with the newly specified version.

If no packages are specified, the following information about currently frozen packages is displayed: package name, frozen version, when the package was frozen, and the reason the package was frozen.

Freezing a package does not prevent removal of the package. No warning is displayed if the package is removed.

In the following example, the package is frozen at the current installed version. The `-c` option argument is the reason the package is being frozen. The reason is shown if a freeze prevents an installation or update from succeeding. The "f" in the package listing indicates that the package is frozen.

```
$ pkg freeze -c "Downgrade to avoid bug" library/security/openssl
library/security/openssl was frozen at 1.0.0.10-0.175.1.0.0.18.0:20120611T201116Z
$ pkg freeze
NAME                      VERSION                                    DATE                  COMMENT
library/security/openssl 1.0.0.10-0.175.1.0.0.19.0:20120625T171753Z 29 Jul 2012 17:45:44 PDT
 Downgrade to
avoid bug
$ pkg list library/security/openssl
NAME (PUBLISHER)                 VERSION              IFO
library/security/openssl         1.0.0.10-0.175.1.0.0.18.0  if-
```

When you try to install a different version of the frozen package, you see a message about the freeze.

```
$ pkg update library/security/openssl@1.0.0.10-0.175.1.0.0.20.0
Creating Plan (Solver setup): -
pkg update: No matching version of library/security/openssl can be installed:
  Reject: pkg://solaris/library/security/openssl@1.0.0.10,5.11-0.175.1.0.0.20.0:20120709T180243Z
  Reason:  This version is excluded by a freeze on library/security/openssl at version
  1.0.0.10,5.11-0.175.1.0.0.18.0:20120611T201116Z.
  The reason for the freeze is: Downgrade to avoid bug
```

A freeze is never lifted automatically by the packaging system. Use the `pkg unfreeze` command to remove the constraints that freezing imposes from the specified packages. Any versions provided are ignored.

Relaxing Version Constraints Specified by Incorporations

An incorporation package specifies which versions of packages can be installed. These version constraints help keep the system in a supportable state across updates. See "Incorporation Packages" on page 12 for more information about incorporation packages and constraints.

Some incorporated packages might be safe to downgrade or upgrade at a version different from the version specified by the incorporation. The version constraint is expressed by a version-lock.*package* facet specified in the incorporation package. The default value of the version-lock.*package* facet is true. To relax the version constraint on a package, set the value of its version-lock. facet to false.

In the following example, you want to downgrade to an earlier version of a package. The pkg update command downgrades as well as upgrades packages.

```
$ pkg list -af library/security/openssl
NAME (PUBLISHER)                    VERSION                    IFO
library/security/openssl            1.0.1.5-0.175.2.0.0.24.0   i--
library/security/openssl            1.0.1.5-0.175.2.0.0.23.0   ---
$ pkg update library/security/openssl@1.0.1.5-0.175.2.0.0.23.0
Creating Plan (Solver setup): |
pkg update: No matching version of library/security/openssl can be installed:
  Reject: pkg://solaris/library/security/openssl@1.0.1.5,5.11-0.175.2.0.0.23.0:20130916T191702Z
  Reason:  This version is excluded by installed incorporation
  pkg://solaris/consolidation/userland/userland-
incorporation@0.5.11,5.11-0.175.2.0.0.24.0:20131001T160408Z
```

The pkg contents command shows how this version constraint is set. To relax the version constraint on this package, set its version-lock facet to false. Then try the downgrade again. Notice that a new BE is not created, but a backup BE is created. See "Boot Environment Policy Image Properties" on page 91 for information about when BEs are created.

```
$ pkg contents -m userland-incorporation | grep 'library/security/openssl'
depend facet.version-lock.library/security/openssl=true
fmri=pkg:/library/security/openssl@1.0.1.5-0.175.2.0.0.24.0 type=incorporate
$ pkg change-facet facet.version-lock.library/security/openssl=false
           Packages to update: 850
     Variants/Facets to change:   1
       Create boot environment:  No
Create backup boot environment: Yes

PHASE                                  ITEMS
Removing old actions                    1/1
Updating image state                   Done
Creating fast lookup database          Done
Reading search index                   Done
Building new search index           850/850
$ pkg update library/security/openssl@1.0.1.5-0.175.2.0.0.23.0
           Packages to update:   1
       Create boot environment:  No
```

```
Create backup boot environment: Yes

DOWNLOAD                            PKGS      FILES    XFER (MB)   SPEED
Completed                           1/1       10/10     1.6/1.6    0B/s

PHASE                                       ITEMS
Removing old actions                         3/3
Installing new actions                       3/3
Updating modified actions                  14/14
Updating package state database             Done
Updating package cache                       1/1
Updating image state                        Done
Creating fast lookup database               Done
Reading search index                        Done
Updating search index                        1/1
$ pkg list library/security/openssl
NAME (PUBLISHER)                VERSION                    IFO
library/security/openssl        1.0.1.5-0.175.2.0.0.23.0  i--
```

To prevent this package from being downgraded or upgraded, freeze the package at the current version. The "f" in the package listing indicates that the package is frozen.

```
$ pkg freeze -c "Downgrade to avoid bug" library/security/openssl
library/security/openssl was frozen at 1.0.1.5,5.11-0.175.2.0.0.23.0:20130916T191702Z
$ pkg list library/security/openssl
NAME (PUBLISHER)                VERSION                    IFO
library/security/openssl        1.0.1.5-0.175.2.0.0.23.0  if-
```

To re-enable downgrade or upgrade, use the pkg unfreeze command to remove the version freeze. If the package is installed at a version lower than the version specified in the incorporation package, setting the version-lock facet for this package to true installs the version specified in the incorporation package.

If other installed packages have require dependency relationships with the package that you want to downgrade or upgrade, you might need to also relax version constraints on those related packages. In the following example, version constraints have been lifted on the hexedit package but installation is rejected because of version constraints on the system/library package.

```
$ pkg install editor/hexedit@1.2.12-0.175.2.0.0.25.0
Creating Plan (Solver setup): -
pkg install: No matching version of editor/hexedit can be installed:
  Reject: pkg://solaris/editor/hexedit@1.2.12-0.175.2.0.0.25.0:20131014T170634Z
  Reason:  All versions matching 'require' dependency
  pkg:/system/library@0.5.11,5.11-0.175.2.0.0.24.0 are rejected
    Reject: pkg://solaris/system/library@0.5.11,5.11-0.175.2.0.0.24.0:20131001T152820Z

pkg://solaris/system/library@0.5.11,5.11-0.175.2.0.0.25.0:20131014T161136Z
    Reason:  This version is excluded by installed incorporation
    pkg://solaris/consolidation/osnet/osnet-
incorporation@0.5.11,5.11-0.175.2.0.0.24.0:20131001T150429Z
```

In addition to individual component packages, you can also relax version constraints on incorporations. In this case, setting the version-lock facet to false enables you to unlock the

incorporation from the rest of the system. Though the incorporation package is unlocked, the packages it incorporates continue to be synchronized.

Specifying a Default Application Implementation

Multiple versions of an application or tool might be available in the same image. Each version of the application is available to users by specifying the full path. One version, called the preferred version, is available from a common directory such as `/usr/bin` for ease of use. If all versions participate in the same mediation, as described below, you can easily reset the version that is the preferred version. This administrative selection is retained across package updates.

A *mediation* is a set of links in which the link path is the same for each link in the set, and the target of each link is different. For example, the link path might be `/usr/bin/myapp` and targets of the link might include `/usr/myapp/myapp1/bin/myapp` and `/usr/myapp/myapp2/bin/myapp`. Each link in a mediation is called a *participant* in the mediation. If `/usr/bin/myapp` currently invokes `myapp1`, you can easily change the selection so that `/usr/bin/myapp` invokes `myapp2`. The version of the software that is currently the target of the link is the *preferred* version.

Identifying Participants in a Mediation

Use the `pkg mediator` command to display the preferred versions of all mediated links in the image.

In the following output, `MEDIATOR` is the name of the set of links that share the same preferred link path. `VER. SRC.` and `IMPL. SRC.` show whether the preferred version was selected by the system, was selected according to an assigned priority (`vendor` or `site`) or was set by an administrator (`local`). `VERSION` is the version of the selected mediation participant, which should be similar to the version of the software that the link represents. The `VERSION` is set by the package developer. `IMPLEMENTATION` is a string that can be set by the package developer in addition to or instead of the version string.

```
$ pkg mediator
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
gcc-runtime system    4.7     system
java        system    1.7     system
php         system    5.2     system
python      vendor    2.6     vendor
ruby        system    1.9     system
```

The `-a` option shows all mediation participants. Use this option to show your choices if you want to select a different preferred version. The following example shows all participants in

the java mediation. The system keywords indicate that the preferred version in this mediation is not specified with a priority setting in the package and was not set by an administrator: The packaging system selected the version with the higher VERSION value as the preferred version.

```
$ pkg mediator -a java
MEDIATOR    VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java        system    1.7     system
java        system    1.6     system
```

The following output confirms that two different versions of the Java Runtime Environment are installed in this image and version 1.7.0_51 is the currently selected preferred version:

```
$ pkg list -s '*jre*'
NAME (PUBLISHER)        SUMMARY
runtime/java/jre-6      Java(TM) Platform Standard Edition Runtime Environment (1.6.0_71-b12)
runtime/java/jre-7      Java Platform Standard Edition Runtime Environment (1.7.0_51-b13)
$ java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Server VM (build 24.51-b03, mixed mode)
```

Both the jre-6 and jre-7 packages define a symbolic link whose path is /usr/bin/java. In the jre-6 package, the target of the link is jdk1.6.0. In the jre-7 package, the target of the link is jdk1.7.0. The pkg mediator and java -version commands above show that version 1.7 is currently the preferred version, the target of the /usr/bin/java link.

Changing the Preferred Application

Use the pkg set-mediator command to reset the version of a specified mediation that is the default or preferred version.

Use the output from pkg mediator -a to select a version for the -V argument or an implementation for the -I argument. If you make a typographical error or otherwise specify a mediator version or implementation that is not currently available, any links that use the specified mediator are removed.

Use the -n option with the set-mediator subcommand to see whether a backup BE will be created. If no backup BE will be created, you can specify the --require-backup-be option with the set-mediator subcommand. The mediator change is made in the current BE. When you have determined that your current BE has no problems after the mediator change, you can use beadm destroy to destroy the backup BE.

The previous output showed that version 1.7 is the currently selected preferred version of the java mediation. The following command shows setting version 1.6 as the preferred version, which means that invoking /usr/bin/java will invoke JRE version 1.6. JRE version 1.7 is

still available on the system when users specify the full path to that version. Compare the output of the two pkg mediator commands. When you changed the preferred version for the mediation, the VER. SRC. also changed to local, indicating that the selection was specified by an administrator. This selection will persist across reboots and package updates.

```
$ pkg mediator java
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java     system   1.7     system
$ pkg mediator -a java
MEDIATOR    VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java        system   1.7     system
java        system   1.6     system
$ pkg set-mediator -V 1.6 java
            Packages to update:  3
           Mediators to change:  1
        Create boot environment: No
Create backup boot environment: No

PHASE                                   ITEMS
Removing old actions                     2/2
Updating modified actions                3/3
Updating image state                    Done
Creating fast lookup database           Done
Reading search index                    Done
Updating search index                    3/3
$ pkg mediator java
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java     local    1.6     system
```

This administrative selection is retained across package updates, even if the selected implementation is no longer installed. If the selected implementation is no longer installed, the target of the mediated link does not exist. Use one of the following methods to reset the preferred implementation:

- Use the pkg set-mediator command again to select a different implementation from the updated list shown by pkg mediator -a.
- Use the pkg unset-mediator command to allow the system to select a new implementation.

  ```
  $ pkg unset-mediator java
  ```

Avoiding Installing Some Packages in a Group Package

Use the pkg avoid command to avoid installing specified packages if they are the target of a group dependency. You can always explicitly install a package that is compatible with the image even if that package is on the avoid list. Installing a package that is on the avoid list removes that package from the avoid list. The pkg avoid command enables you to avoid

installing specified packages that are part of a group package when you install that group package.

With no arguments, the `pkg avoid` command displays each avoided package along with any packages that have a group dependency on that package.

With packages specified, the `pkg avoid` command places the package names that currently match the specified patterns on the avoid list. Only packages that are not currently installed can be avoided. If a package is currently the target of a group dependency, uninstalling the package places it on the avoid list.

Packages that are on the avoid list are installed if needed to satisfy a `require` dependency and thus are removed from the avoid list. If that `require` dependency is removed, the package is uninstalled and placed back on the avoid list.

Use the `pkg unavoid` command to explicitly remove the specified packages from the avoid list.

Packages on the avoid list that match an installed package's group dependency cannot be removed from the avoid list using the `unavoid` subcommand. To remove a package from the avoid list that matches a group dependency, install the package.

EXAMPLE 5-5   Adding Packages To and Removing Packages From the Avoid List

The following command output shows that the `group/feature/amp` group package is not installed. Some of the packages that are part of that group package are installed because they were installed explicitly or as `require` dependencies of other packages. Use the `-r` option with the `pkg contents` command because the specified package is not installed.

```
$ pkg list -a group/feature/amp
NAME (PUBLISHER)                              VERSION              IFO
group/feature/amp                            0.5.11-0.175.2.0.0.33.0    ---
$ pkg list -a `pkg contents -o fmri -Hrt depend -a type=group group/feature/amp`
NAME (PUBLISHER)                              VERSION              IFO
database/mysql-51                            5.1.37-0.175.2.0.0.34.0    ---
web/php-52                                    5.2.17-0.175.2.0.0.34.0    i--
web/php-52/extension/php-apc                 3.0.19-0.175.2.0.0.34.0    i--
web/php-52/extension/php-mysql               5.2.17-0.175.2.0.0.34.0    i--
web/server/apache-22                         2.2.26-0.175.2.0.0.34.0    i--
web/server/apache-22/module/apache-dtrace    0.3.1-0.175.2.0.0.34.0     ---
web/server/apache-22/module/apache-fcgid     2.3.9-0.175.2.0.0.34.0     ---
web/server/apache-22/module/apache-php5      5.2.17-0.175.1.0.0.18      --r
```

The following command places one of the packages that is not installed and that belongs to this group package on the avoid list. The group package is not noted on the avoid list because the group package is not installed.

```
$ pkg avoid apache-fcgid
$ pkg avoid
    web/server/apache-22/module/apache-fcgid
```

The following commands show that the avoided package is not installed when the group package is installed. The `-r` option is not used with the `pkg contents` command because the specified package is installed.

```
$ pkg install group/feature/amp
$ pkg list -a `pkg contents -o fmri -Ht depend -a type=group group/feature/amp`
NAME (PUBLISHER)                             VERSION                 IFO
database/mysql-51                            5.1.37-0.175.2.0.0.34.0  i--
web/php-52                                   5.2.17-0.175.2.0.0.34.0  i--
web/php-52/extension/php-apc                 3.0.19-0.175.2.0.0.34.0  i--
web/php-52/extension/php-mysql               5.2.17-0.175.2.0.0.34.0  i--
web/server/apache-22                         2.2.26-0.175.2.0.0.34.0  i--
web/server/apache-22/module/apache-dtrace    0.3.1-0.175.2.0.0.34.0   i--
web/server/apache-22/module/apache-fcgid     2.3.9-0.175.2.0.0.34.0   ---
web/server/apache-22/module/apache-php5      5.2.17-0.175.1.0.0.18    i-r
```

After the group package is installed, the group package is noted on the avoid list.

```
$ pkg avoid
    web/server/apache-22/module/apache-fcgid (group dependency of 'group/feature/amp')
```

The `pkg unavoid` command does not remove a package from the avoid list if that package is part of an installed group package. To remove such a package from the avoid list, install the package.

```
$ pkg unavoid apache-fcgid
pkg unavoid: The following packages are a target of group dependencies; use install to unavoid
 these:
    web/server/apache-22/module/apache-fcgid
$ pkg install apache-fcgid
$ pkg avoid
$
```

You cannot place a package on the avoid list if that package is already installed. The package is placed on the avoid list if you uninstall the package.

```
$ pkg avoid apache-fcgid
pkg avoid: The following packages are already installed in this image; use uninstall to avoid these:
    web/server/apache-22/module/apache-fcgid
$ pkg uninstall apache-fcgid
$ pkg avoid
    web/server/apache-22/module/apache-fcgid (group dependency of 'group/feature/amp')
```

Uninstalling a package that is part of a group package automatically places that package on the avoid list.

```
$ pkg uninstall database/mysql-51
$ pkg avoid
    database/mysql-51 (group dependency of 'group/feature/amp')
    web/server/apache-22/module/apache-fcgid (group dependency of 'group/feature/amp')
```

If the group package is uninstalled, the avoided packages remain on the avoid list, but the avoid list no longer notes their association with the group package.

```
$ pkg uninstall group/feature/amp
```

```
$ pkg avoid
    database/mysql-51
    web/server/apache-22/module/apache-fcgid
$ pkg unavoid database/mysql-51 apache-fcgid
$ pkg avoid
$
```

Configuring Image and Publisher Properties

To implement image policies, set image properties. This section describes image and publisher properties and how to set these properties. See also "Image Properties" in the pkg(1) man page for descriptions of image properties.

Boot Environment Policy Image Properties

An image is a location where IPS packages can be installed and where other IPS operations can be performed. A boot environment (BE) is a 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 can be created automatically as a result of package operations. You can also explicitly create a new BE. Whether a new BE is created depends on image policy, as discussed in this section.

By default, a new BE is automatically created when you perform one of the following operations:

- Install or update particular key system packages such as some drivers and other kernel components. Key system components can be updated when you change a variant or facet as well as when you install, uninstall, and update packages.

- Specify any of the following options: `--be-name`, `--require-new-be`, `--backup-be-name`, `--require-backup-be`.

- Set the `be-policy` image policy to `always-new`. Under this policy, all package operations are performed in a new BE set as active on the next boot.

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

1. Creates a clone of the current BE.

   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 new BE accesses the original shared file systems.

2. Updates the packages in the clone BE. Does not update any packages in the current BE.

    If non-global zones are configured in the current BE, these existing zones are configured in the new BE.

3. Sets the new BE as the default boot choice the next time the system is booted unless `--no-be-activate` is specified. The current BE remains as an alternate boot choice.

When a backup BE is created, the system performs the following steps:

1. Creates a clone of the current BE.
2. Updates the packages in the current BE. Does not update any packages in the clone BE.

If a new BE is required but not enough space is available to create it, you might be able to delete existing unneeded BEs. For more information about BEs, see "Creating and Administering Oracle Solaris 11.2 Boot Environments ".

See "Setting Image Properties" on page 96 for information about how to set the following image properties.

`be-policy`

> Specifies when a BE is created during packaging operations. The following values are allowed:
>
> `default`
>
>> Apply the default BE creation policy: `create-backup`.
>
> `always-new`
>
>> Require a reboot for all package operations by performing them in a new BE set as active on the next boot. A backup BE is not created unless explicitly requested.
>>
>> This policy is the safest, but is more strict than most sites need because no packages can be added without a reboot.
>
> `create-backup`
>
>> For package operations that require a reboot, this policy creates a new BE that is set as active on the next boot. If packages are modified or content that could affect the kernel is installed and the operation affects the live BE, a backup BE is created but not set as active. A backup BE can also be explicitly requested.
>>
>> This policy is potentially risky only if newly installed software causes system instability, which is possible, but relatively rare.
>
> `when-required`
>
>> For package operations that require a reboot, this policy creates a new BE set as active on the next boot. A backup BE is not created unless explicitly requested.
>>
>> This policy carries the greatest risk because if a packaging change to the live BE makes further changes impossible, a recent fallback BE might not exist.

Properties for Signing 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 the checks that will be performed on manifests when installing, updating, modifying, or verifying packages in the 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. By default, the package client does not check whether certificates have been revoked. To enable those checks, which might require the client to contact external web sites, set the `check-certificate-revocation` image property to `true`. The following values are allowed:

> `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.

>> This is the default value.

> `require-signatures`

>> Require that all newly installed packages have at least one valid signature. The `pkg fix` and `pkg verify` commands 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.

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 applies only 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 applies only to packages from the specified publisher.

Configuring Package Signature Properties

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

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.

The following example configures this image to require all packages to be signed. This example also requires 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
```

The following example configures this image to require all signed packages to be verified.

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

The following example configures 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
```

The following example adds 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-required-names trustedname
```

The following example removes 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-required-names trustedname
```

The following example adds 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.

```
$ pkg set-publisher --add-property-value \
signature-required-names=trustedname example.com
```

Additional Image Properties

ca-path

> Specifies a path name that points to a directory where CA certificates are kept for SSL operations. The format of this directory is specific to the underlying SSL implementation. To use an alternate location for trusted CA certificates, change this value to point to a different directory. See the `CApath` portions of `SSL_CTX_load_verify_locations`(3openssl) for requirements for the CA directory.
>
> The default value is `/etc/openssl/certs`.

check-certificate-revocation

> If set to `true`, the package client attempts to contact any CRL distribution points in the certificates used for signature verification to determine whether the certificate has been revoked since being issued.
>
> The default value is `False`.

content-update-policy

> Specify when the package system will update non-editable files during packaging operations. The following values are allowed:
>
> default
>
> > Always apply the default content update policy.
>
> always
>
> > Always download and update non-editable files that have changed.
>
> when-required
>
> > Download and update non-editable files that have changed only if the package system has determined that an update is required.
>
> The default value is `always`.

`flush-content-cache-on-success`

If set to `true`, the package client removes the files in its content-cache when image-modifying operations complete successfully. For operations that create a BE, the content is removed from both the source and destination BE.

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

The default value is `True`.

`mirror-discovery`

This property tells the client to discover link-local content mirrors using mDNS and DNS-SD. If this property is set to `true`, the client attempts to download package content from mirrors it dynamically discovers. To run a mirror that advertises its content via mDNS, see `pkg.depotd`(1M).

The default value is `False`.

`send-uuid`

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

The default value is `True`.

`trust-anchor-directory`

The value of this property is the path name of the directory that contains the trust anchors for the image. This path is relative to the image.

The default value is `ignore`.

`use-system-repo`

This property indicates whether the image should use the system repository as a source for image and publisher configuration and as a proxy for communicating with the publishers provided. See `pkg.sysrepo`(1M) for information about system repositories.

The default value is `ignore`.

Setting Image Properties

Use the `pkg property` command to view image property settings. Use the `set-property`, `add-property-value`, `remove-property-value`, and `unset-property` subcommands to configure image properties.

Displaying the Values of Image Properties

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

```
$ pkg property
PROPERTY                         VALUE
be-policy                        default
ca-path                          /etc/openssl/certs
check-certificate-revocation     False
flush-content-cache-on-success   False
mirror-discovery                 False
preferred-authority              solaris
publisher-search-order           ['solaris', 'isvpub']
send-uuid                        True
signature-policy                 verify
signature-required-names         []
trust-anchor-directory           etc/certs/CA
use-system-repo                  False
```

You might want to use the search order options of the `pkg set-publisher` command to set the `publisher-search-order` property. See "Setting Publisher Search Order and Stickiness" on page 75.

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.

The following example sets the value of the `mirror-discovery` property.

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

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.

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

Creating an Image

An image is a location where IPS packages and their associated files, directories, links, and dependencies can be installed, and where other IPS operations can be performed.

Images created by using the `pkg image-create` command are not bootable. To create a bootable images use the `--be-name` or `--require-new-be` options with `pkg` commands, or use the `beadm` or `zonecfg` and `zoneadm` commands. The `pkg image-create` command is used for tasks such as maintaining packages and operating system distributions.

The `pkg image-create` command requires an operand that is the directory where the image will be created. The default type of image created is a user image. You can specify any of the following image types:

Full
: Full images are capable of providing a complete system. In a full image, all dependencies are resolved within the image itself and IPS maintains the dependencies in a consistent manner. After you have completed an installation of the Oracle Solaris OS, the root file system and its contents are contained in a full image. Use the `-F` or `--full` option to specify a full image.

Partial
: Partial images are linked to the full image that encloses the given *dir* path (the parent image). Partial images do not provide a complete system on their own. Use the `-P` or `--partial` option to specify a partial image.

: A non-global zone is a partial image. To use the image in non-global zone context, specify the `-z` or `--zone` option to set an appropriate variant. In a zone image, IPS maintains the non-global zone consistent with its global zone as defined by dependencies in the packages. See for more information about zones.

User
: User images contain only relocatable packages. This is the default type of image that is created if you do not specify an image type. Use the `-U` or `--user` option to specify a user image.

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

For publishers using client SSL authentication, use the `-c` and `-k` options to register a client key and client certificate. This key and certificate are used for all publishers added during image creation.

Use the `--variant`, `--facet`, and `--set-property` options to set variant values, facet values, and image property values.

Viewing Operation History

Use the `pkg history` command to view `pkg` command history. By default, the following information is displayed:

- The start time of the operation
- The name of the operation, for example, `install`
- The client, for example, `pkg`
- The outcome of the operation: `Succeeded` or `Failed`

Use options to display more information or more precise information.

`-l`

Display the following information in addition to the default information:
- The version of the client
- The name of the user who performed the operation
- Whether a new BE was created
- The time the operation completed
- The complete command that was issued
- Any errors that were encountered while executing the command
- Complete FMRIs of changed packages for operations such as `update`

`-n` *number*

Display only the specified number of most recent operations.

```
$ pkg history -n4
START                OPERATION          CLIENT         OUTCOME
2013-08-06T16:32:03  fix                pkg            Succeeded
2013-08-06T16:41:47  revert             pkg            Succeeded
2013-08-06T17:56:22  set-property       pkg            Succeeded
2013-08-06T17:56:53  unset-property     pkg            Succeeded
```

`-o` *column*[`,`*column*]`...`

Display output using the specified comma-separated list of column names. See the list of column names in pkg(1).

```
$ pkg history -o start,time,operation,outcome -n4
START               TIME     OPERATION          OUTCOME
2013-08-06T16:32:03  0:00:27  fix                Succeeded
2013-08-06T16:41:47  0:00:43  revert             Succeeded
```

```
    2013-08-06T17:56:22      0:00:00    set-property          Succeeded
    2013-08-06T17:56:53      0:00:00    unset-property        Succeeded
```

-t *time* | *time*-*time*[,*time* | *time*-*time*]...

Log records for a comma-separated list of time stamps, formatted with `%Y-%m-%dT%H:%M:%S` (see the `strftime`(3C) man page). To specify a range of times, use a hyphen (`-`) between a start and finish time stamp. The keyword `now` is an alias for the current time. If the time stamps specified contain duplicate time stamps or overlapping date ranges, only a single instance of each duplicate history event is displayed.

-N

Use the `-N` option to display any release note text for the operation. The `-N` option cannot be used with the `-o` option. If you specify the `-v` option in an install or update operation in which some of the packages being installed have release notes, the operation output displays the release notes. If the operation installs into a new BE, the operation output provides a path to a release notes file in `/tmp` in the current BE. When you boot into the new BE, the release notes are in `/usr/share/doc/release-notes`, or you can use the `-N` option to view the release notes as shown in the following command:

$ **pkg history -N -n 1**

If the operation that installed release notes is not the last `pkg` operation you performed in this BE, use a larger number for the `-n` argument or use the `-t` option to identify the `pkg` operation that installed the release notes as shown in the following command:

$ **pkg history -N -t 2013-07-17T08:31:23**

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

$ **pkg purge-history**

A

Troubleshooting Package Installation and Update

This appendix shows how to handle some errors that you might see when you install or update packages. This appendix also provides tips for increasing performance and minimizing stored metadata.

Initial Troubleshooting Steps

Check whether the package you want to install is available from configured publishers and can be installed in this image. The installed version of `pkg:/entire` and the publisher origins configured in this image are required pieces of information. You might need to update the `pkg:/entire` package. You might need to modify publisher origins.

The following two checks should be done first for almost any package installation problem:

- Check which version of the `pkg:/entire` incorporation package is installed.
- Check your package publisher origin.

When you have determined that the packages you need are available from configured publishers, use the following steps as you proceed with your installation:

- Use the `-nv` options whenever you install or update to see what changes will be made, such as which versions of which packages will be installed or updated and whether a new BE will be created. The `-v` option also shows any release notes that apply to this particular installation or update operation.
- To receive more detailed error messaging, specify more of the FMRI of the package you want to install, including the version and publisher.

Check the Installed Version of `pkg:/entire`

Use the `pkg list` command to check the version of the `pkg:/entire` incorporation package that is currently installed.

```
$ pkg list entire
NAME (PUBLISHER)                                VERSION               IFO
entire                                          0.5.11-0.175.2.0.0.34.0   i--
$ pkg list -Hv entire
pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.34.0:20140303T182643Z          i--
```

The pkg:/entire incorporation package constrains the versions of many other packages to help maintain a working, supportable image. See "Incorporation Packages" on page 12 for more information about packages constrained by incorporation packages.

You cannot directly install or update a package that is constrained by the pkg:/entire incorporation package. To install or update packages that are constrained by pkg:/entire, you must update the pkg:/entire package. For more information, see "Cannot Satisfy Constraints" on page 110. In some cases, you can remove the constraints as shown in "Updating a Package Constrained by an Incorporation" on page 111.

Check the Content of Your Configured Publisher Origins

Use the pkg publisher command to check your package publisher origins.

```
$ pkg publisher
PUBLISHER  TYPE    STATUS P LOCATION
solaris    origin  online F http://pkg.oracle.com/solaris/release/
```

For a secure URI, make sure you have the required key and certificate properly installed, and use the -k and -c options when you configure the publisher.

If your site requires a proxy for external locations, use the --proxy option of the pkg set-publisher command to set that proxy. See "Specifying a Proxy" on page 77 for instructions.

Use the pkg publisher *publisher* command to view more information about the publisher, such as key, certificate, and proxy.

If any origin URI of any of your enabled publishers is unreachable, the install or update operation fails, even if the locations you need are reachable. If you cannot fix the problem that makes the location unreachable, you can remove the unreachable origin with pkg set-publisher -G or disable that publisher with pkg set-publisher --disable. If you no longer need this publisher, use pkg unset-publisher to remove the publisher.

Check whether your package publisher origin contains the packages you need. For example, if your solaris publisher origin is set to the public release repository, then you cannot update a package to a version that is only available from a support repository.

Check Whether Required Installed Packages Are Available

To update installed packages, install packages that depend on installed packages, or install a non-global zone, the repository that you set as the publisher origin must contain at least the same software that is currently installed in the image. The repository can also contain older or newer software, but it must contain the same software that is installed in the image.

Use the `pkgrepo list` command, not the `pkg list` command, when checking for installed packages. The `pkg list` command always shows installed packages, even if the package is not available from any configured publisher origin.

The following command shows that the specified repository is a not suitable publisher origin for this image because the installed version of `pkg:/entire` is not available from that origin.

```
$ pkg list entire
NAME (PUBLISHER)                                 VERSION              IFO
entire                                           0.5.11-0.175.2.0.0.34.0   i--
$ pkg publisher
PUBLISHER  TYPE    STATUS P LOCATION
solaris    origin  online F http://pkg.oracle.com/solaris/release/
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release entire@0.5.11-0.175.2.0.0.34.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.34.0
```

If the package that you need is not listed, try running the `pkgrepo refresh` command and then retry the `pkgrepo list` command.

The following command shows that the installed version of `pkg:/entire` is available from the specified repository:

```
$ pkgrepo list -Hs /export/IPSpkgrepos/Solaris entire@0.5.11-0.175.2.0.0.34.0
solaris    entire              0.5.11,5.11-0.175.2.0.0.34.0:20140303T182643Z
```

If a needed package is not available from a configured publisher but is available from another repository origin, take one of the following actions:

- Use the `-g` option of the `pkg set-publisher` command to add this origin for the `solaris` publisher.
- Use the `-g` and `-G` options of the `pkg set-publisher` command to change the origin for the `solaris` publisher.
- If a different publisher provides the package you need, use the `pkg set-publisher` command to add that publisher.
- Use the `-g` option of the installation command (`install`, `uninstall`, `update`, `change-variant`, and `change-facet`) to temporarily add a repository to the end of the list of repositories to search.

Check Whether the Packages You Want to Install Are Available

Use the following command to check whether the package you want to install is available from your configured publishers. If you are updating a package, both the version of that package that is currently installed and the version to which you want to update should be available.

$ **pkg list -af** *package*

If the package that you need is not listed, try running the `pkg refresh` command and then retry the `pkg list` command.

If the package that you need still is not listed, add a new publisher or a new publisher origin.

Check Whether the Packages You Want to Install Are Installable in This Image

If the package version you want is listed when you use the `-af` options, then use the same command again without the `-f` option:

$ **pkg list -a** *package*

If the version you want is still listed, then this package is not constrained and you should be able to install it without installing or updating any other packages.

If the version you want is not listed, then this version is available from configured publishers but not installable in this image. Reasons that a package is not installable can include the following:

- The package is constrained by variant or facet settings.
- The version of the package is constrained by an incorporation package. You can update the constraining package or, in some cases, you can relax the constraint. For more information, see "Cannot Satisfy Constraints" on page 110.
- The version of the package is constrained by a freeze operation. Run the `pkg freeze` command. The freeze could be on a package that has a `require` dependency on a different version of the package you want to install and both versions cannot be installed at the same time.

Retry Your Installation

If you specify a package to install or update and you do not specify the publisher, the first publisher in the publisher search order that provides a package that matches that package FMRI or pattern is used as the installation source. If that publisher does not provide a version of the package that can be installed in this image, then the installation operation fails, even if another

enabled publisher provides a version of the package that can be installed in this image. Take one of the following actions to address this issue:

- Specify the publisher in the package FMRI. For example, specify `pkg://solaris/` in front of the full package name.
- Use the `-P` option of the `pkg set-publisher` command to set the publisher that provides the package version you want as the first publisher in the search order.

Use the `-nv` options whenever you install or update to see what changes will be made, such as which versions of which packages will be installed or updated and whether a new BE will be created. The `-v` option also shows any release notes that apply to this particular install or update operation.

- If you do not receive any error messages when you use the `-nv` options, run the command again without the `-n` option to actually perform the installation or update. Consider whether you should specify options to do the installation in a new BE or to create a backup BE if no new BE or backup BE will be created by default.
- If you do receive error messages, take the following actions:
  - Specify more of the version you want in the package FMRI to get more information to help you diagnose and fix the problem.
  - Specify more `-v` options (for example, `-nvv`).
  - Use the `pkg history` command. The `-l` option provides complete FMRIs of changed packages. See "Viewing Operation History" on page 99.

If you specify multiple packages to install or update, or if you omit the package specification for an update operation, the installation or update operation fails if any of the packages cannot be installed in this image. If one package cannot be installed, no packages are installed. For more information, invoke the command again, specifying only the package that cannot be installed, specifying the full FMRI of that package, and providing one or more `-v` options.

Cannot Access Publisher or Repository

The errors discussed in this section are related to an inability to access the URI of the publisher.

Cannot Access Package Repository

Error messages:

- `Couldn't resolve host`
- `Unable to contact any configured publishers`
- `Unable to contact valid package repository`

- Origin URIs do not appear to point to a valid pkg repository

Use the pkg publisher command to display the URIs of your publishers. See the LOCATION column or the URI row in the following examples:

```
$ pkg publisher
PUBLISHER                     TYPE     STATUS P LOCATION
solaris                       origin   online F http://pkg.oracle.com/solaris/release/
$ pkg publisher solaris
          Publisher: solaris
              Alias:
          Origin URI: http://pkg.oracle.com/solaris/release/
```

If publishers are listed that you are no longer using, either disable or remove those publishers:

```
$ pkg set-publisher --disable publisher
$ pkg unset-publisher publisher
```

For publishers that you are using, make sure the URIs are correct.

- Try to view each publisher origin location in a browser or ping that location.
- Try to list packages at that origin by using the pkgrepo list command.

If an origin location is not correct, use the -G and -g options of the pkg set-publisher command together to change the URI.

If a publisher has more than one origin, all origin locations must be accessible. If any origin location is not accessible, use the -G option of the pkg set-publisher command to remove that origin.

If a publisher is configured in a non-global zone, all locations for that publisher must be accessible from the global zone even if that publisher is not configured in the global zone.

SSL Certificate Problem

Error message: SSL certificate problem, verify that the CA cert is OK

Take one or more of the following actions if the pkg command displays a message about an SSL certificate problem:

- Make sure the required key and certificate are installed. Information about how to obtain a key and certificate should be included with other information about the secure repository.
- Use the -k and -c options with the pkg set-publisher command to specify the locations where the key and certificate are installed. The pkg publisher command for the publisher shows where the publisher expects to find the key and certificate. If the key and certificate are installed elsewhere, reconfigure the publisher, specifying the correct values for the -k and -c options.

In the following example, the output shows the key and certificate file paths and shows that the certificate for the configured origin for the abc publisher is expired:

```
$ pkg publisher abc
 Publisher: abc
Origin URI: https://pkg.oracle.com/abc/release/
   SSL Key: /var/pkg/ssl/keyfile
  SSL Cert: /var/pkg/ssl/certfile
Certificate '/var/pkg/ssl/certfile' has expired.
  Please install a valid certificate.
```

- Make sure the time and date are correct on the system.
- Refresh the ca-certificates SMF service:

  ```
  $ svcadm refresh svc:/system/ca-certificates:default
  ```

- If the origin URI contains oc-mgmt, contact your Ops Center support representative.

Note - A repository that requires a client SSL certificate cannot be specified by using a -g option with a pkg installation command.

Location Not Found

Error message: http protocol error: code: 404 reason: Not Found

Check your publisher URIs as described in "Cannot Access Package Repository" on page 105. If you can view or ping the location successfully, use the pkgrepo list command to try to show one of the packages in the repository.

If the URI is a file-based repository, make sure the files and directories are readable by the pkg5srv user. You can use the pkgrepo verify command to check whether the repository is readable by the pkg5srv user.

Check your web server configuration. See Chapter 5, "Running the Depot Server Behind a Web Server," in "Copying and Creating Package Repositories in Oracle Solaris 11.2 " for more information.

- If you run the package depot server behind an Apache web server instance, include the following setting in your httpd.conf file to not decode encoded forward slashes:

  ```
  AllowEncodedSlashes NoDecode
  ```

- Set the depot server pkg/proxy_base to the URL of the repository on the Apache server:

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

```
$ svcadm refresh pkg/server:repo
```

If the problem occurs in a non-global zone, take the following troubleshooting steps. Remember that non-global zones use a special package repository called the system repository. See the pkg.sysrepo(1M) man page for more information about the system repository.

- You cannot use the -g option in an image that has non-global zones. Instead, use the pkg set-publisher command to explicitly add that publisher and origin.

- Make sure the files and directories in file-based repositories are readable by the pkg5srv user. The pkg5srv user runs the system-repository Apache instance. See "Relationship Between Global and Non-Global Zones" on page 53 for examples that show how to find the location of a system repository.

- If your site requires a proxy to access external locations, make sure the proxy has been specified correctly for publishers in the global zone. Use the --proxy option of the pkg set-publisher command to specify the proxy. See "Specifying a Proxy" on page 77 for instructions. One way to check your proxy is to make sure you get no access error messages from the pkg refresh --full command.

- Make sure the service svc:/application/pkg/system-repository:default is online in the global zone.

- Make sure the service svc:/application/pkg/zones-proxyd:default is online in the global zone and the service svc:/application/pkg/zones-proxy-client:default is online in the non-global zone.

- In the global zone, check the log files in /var/log/pkg/sysrepo/* for any permissions errors reported when trying to read files. Check for 404 or 503 errors reported in /var/log/pkg/sysrepo/access_log. Check for errors reported in /var/log/pkg/sysrepo/error_log.

- In the global zone, verify that localhost is set to 127.0.0.1 in the /etc/hosts file. Verify that Listen is set to 127.0.0.1:1008 and ServerName is set to 127.0.0.1 in the /system/volatile/pkg/sysrepo/sysrepo_httpd.conf file.

- In the global zone, check whether the file /system/volatile/pkg/sysrepo/sysrepo_httpd.conf contains Alias lines of the following form:

```
$ grep Alias /system/volatile/pkg/sysrepo/sysrepo_httpd.conf
WSGIScriptAlias /wsgi_p5p /etc/pkg/sysrepo/sysrepo_p5p.py
```

If the sysrepo_httpd.conf file has no Alias lines, restart the sysrepo service:

```
$ svcadm restart svc:/application/pkg/system-repository:default
```

Service Is Not Available

Error message: http protocol error: code: 503 reason: Service Unavailable

Use the `pkg publisher` command to find the location of the package repository that you are trying to use, and examine SMF services on that system. Use the following command to identify any package repository SMF service instances that are enabled but not running and any instances that are preventing another enabled instance from running:

```
$ svcs -xv pkg/server
svc:/application/pkg/server: default (image packaging repository)
 State: online since July 25, 2013 07:53:50 AM PDT
   See: /var/svc/log/application-pkg-server:default.log
Impact: None.
```

If any service is reporting a problem, check the log file listed in the `svcs` output to determine the specific problem.

Make sure that the `inst_root` property, the `port` property, and other properties are set correctly.

```
$ svcprop -p pkg pkg/server:default
$ svcprop -p pkg/inst_root -p pkg/port pkg/server:default
/export/IPSpkgrepos/Solaris
80
```

If necessary, use the `svccfg` command to reset property values, as shown in the following example:

```
$ svccfg -s pkg/server:default setprop pkg/port=1008
```

Use the `svcadm` command to clear, refresh, restart, and enable the service instance as necessary.

No Updates Are Available

Error message: `No updates available for this image`

If you are updating a particular package, use the following command to show what version of that package is currently installed in this image. If you are updating all installed packages (`pkg update` with no packages specified or with `'*'` specified for the package name), use `pkg:/entire` for *package* in these commands.

```
$ pkg list -v package
```

If no version is currently installed, use the `pkg install` command, not `pkg update`.

If a version of *package* is currently installed, use the following command to show which versions of *package* are available from your configured publishers:

```
$ pkg list -afv package
```

If the package with the highest version number is already installed, then perhaps no newer version exists.

If a newer version does exist, determine the package repository location where the newer version is available, and use the `pkg set-publisher` command to reset the origin URI or add an origin URI for the appropriate publisher. If necessary, install any required key and certificate and use the `-k` and `-c` options to specify them. Use the `pkgrepo list` command to verify that the currently installed version of the package also is available from configured publishers.

Execute the `pkg update` command again, specifying the `-nv` options and including the version (or the keyword `latest`) in the FMRI of the packages you want to install, as shown in the following example. Providing more information in the package names usually gives you more information in any error output.

$ **pkg update -nv** *package***@latest**

Package Cannot Be Installed

Error message: `No matching version of` *package* `can be installed`

Use the following command to show which versions of *package* are available from your configured package publishers:

$ **pkg list -afv** *package*

Specify more of the FMRI of the package you want to install. The first match found might not be installable in this image, but the specific version you want to install might be installable. If the more specific FMRI is still not installable, specifying more of the FMRI should display more information about why the package cannot be installed.

Verify that the version of the package you want to install or update is not frozen. Use the `pkg freeze` command with no arguments to display a list of all packages whose versions are frozen.

Use the following command to show what version of the `pkg:/entire` package is installed:

$ **pkg list -v entire**

You cannot install or update a package that is constrained by the `pkg:/entire` incorporation package or by some other incorporation package. You must update the incorporation package. For more information, see .

Cannot Satisfy Constraints

Error message: `No solution was found to satisfy constraints`

This message indicates that you attempted to install a version of a package that does not match the version to which the package is constrained by an incorporation package. See

"Incorporation Packages" on page 12 for information about incorporation packages and constraints.

Incorporations constrain a set of packages to versions that work together to help maintain a supportable image. For this reason, you should not update one package that is constrained by an incorporation. Instead, you should update the incorporation package, which results in updating all the constrained packages to a new tested-together set of versions.

When packages are being modified (for example, being updated), the `pkg` client examines related packages and their dependencies. If any dependent package cannot be installed or updated, a separate error message is produced for each package that has a dependency on the package that cannot be installed or updated. The most effective way to handle a large number of error messages is to examine the error messages that are the most indented first.

Updating a Package Constrained by an Incorporation

Error messages:

- `No suitable version of installed package` *package* `found`
- `All versions matching 'incorporate' dependency` *package* `are rejected`
- `This version excluded by specified installation version`
- `This version is excluded by installed incorporation`

For packages constrained by an incorporation, best practice is to update the incorporation package, keeping all of the constrained packages as a tested-together set.

If you still want to update just one package from an incorporation, check whether that package has a `version-lock` facet set to `true`. If a package has an associated `version-lock` facet, then you can unlock that package from its incorporation. Set the `version-lock` facet to `false` to remove the constraint, and then try again to install or update the package. Specify the `-nv` options and the version of the package you want in the FMRI of the package name. See also "Relaxing Version Constraints Specified by Incorporations" on page 84.

EXAMPLE A-1   Unlocking and Updating the Java Runtime Environment

The following example shows how to update the `runtime/java/jre-7` package. The `jre-7` package is constrained by the `consolidation/java/java-incorporation` package, and the `java-incorporation` package is in turn constrained by the `pkg:/entire` incorporation package.

The following command shows that the `0.175.2.0.0.9.0` version of `jre-7` is currently installed and shows that a newer version is available from configured package repositories:

```
$ pkg list -af runtime/java/jre-7
```

```
NAME (PUBLISHER)                                    VERSION                 IFO
runtime/java/jre-7                                  1.7.0.21-0.175.2.0.0.13.0  ---
runtime/java/jre-7                                  1.7.0.17-0.175.2.0.0.9.0   i--
```

Removing the `-f` option shows which versions are available to update to. The following `pkg list` output shows that no newer version can be installed in this image, and the `pkg update` command output confirms this state. The `-n` option shows what changes would be made but does not make any changes.

```
$ pkg list -a runtime/java/jre-7
NAME (PUBLISHER)                                    VERSION                 IFO
runtime/java/jre-7                                  1.7.0.17-0.175.2.0.0.9.0   i--
$ pkg update -nv runtime/java/jre-7
No updates available for this image.
```

To show more information about why this package cannot be updated, specify the version to which you want to update. The output shown in the following example indicates that the installed `java-incorporation@0.5.11,5.11-0.175.2.0.0.9.0` package does not allow the installation of the `jre-7@1.7.0.21-0.175.2.0.0.13.0` package. The `java-incorporation@0.5.11,5.11-0.175.2.0.0.13.0` package would allow the installation of the `jre-7@1.7.0.21-0.175.2.0.0.13.0` package, but the installed `entire@0.5.11,5.11-0.175.2.0.0.12.0` incorporation package does not allow the installation of the `jre-7@1.7.0.21-0.175.2.0.0.13.0` package.

```
$ pkg update -nv runtime/java/jre-7@1.7.0.21-0.175.2.0.0.13.0
pkg update: No solution was found to satisfy constraints

maintained incorporations:
  [output omitted]
  pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.12.0:20130415T172730Z

Plan Creation: dependency error(s) in proposed packages:
  [output omitted]
  No suitable version of required package pkg://solaris/consolidation/java/java-
incorporation@0.5.11,5.11-0.175.2.0.0.9.0:20130304T213946Z found:
    Reject:  pkg://solaris/consolidation/java/java-incorporation@0.5.11,5.11-0.1
75.2.0.0.9.0:20130304T213946Z
    Reason:  All versions matching 'incorporate' dependency pkg:/runtime/java/jr
e-7@1.7.0.17,5.11-0.175.2.0.0.9.0 are rejected
      Reject:  pkg://solaris/runtime/java/jre-7@1.7.0.17,5.11-0.175.2.0.0.9.0:20
130304T214022Z
      Reason:  This version excluded by specified installation version
        Reject:  pkg://solaris/runtime/java/jre-7@1.7.0.17,5.11-0.175.2.0.0.9.0:
20130304T214022Z
        Reason:  This version excluded by specified installation version
      Reject:  pkg://solaris/consolidation/java/java-incorporation@0.5.11,5.11-0
.175.2.0.0.13.0:20130429T145534Z
      Reason:  This version is excluded by installed incorporation pkg://solaris
/entire@0.5.11,5.11-0.175.2.0.0.12.0:20130415T172730Z

Plan Creation: Errors in installed packages due to proposed changes:
  [output omitted]
  No suitable version of installed package pkg://solaris/consolidation/java/java
-incorporation@0.5.11,5.11-0.175.2.0.0.9.0:20130304T213946Z found
```

```
     Reject:  pkg://solaris/consolidation/java/java-incorporation@0.5.11,5.11-0.1
75.2.0.0.9.0:20130304T213946Z
     Reason:  All versions matching 'incorporate' dependency pkg:/runtime/java/jr
e-7@1.7.0.17,5.11-0.175.2.0.0.9.0 are rejected
      Reject:  pkg://solaris/runtime/java/jre-7@1.7.0.17,5.11-0.175.2.0.0.9.0:20
130304T214022Z
       Reason:  This version excluded by specified installation version
     Reject:  pkg://solaris/consolidation/java/java-incorporation@0.5.11,5.11-0.1
75.2.0.0.13.0:20130429T145534Z
     Reason:  This version is excluded by installed incorporation pkg://solaris/e
ntire@0.5.11,5.11-0.175.2.0.0.12.0:20130415T172730Z
```

Best practice is to update the `entire` package. Updating the `entire` package would update the `java-incorporation` package, which would update the `jre-7` package. In this example, you need to update the Java packages and cannot move your image forward to an updated version of `entire`.

The version of the Java software that can be installed is constrained by setting the `version-lock` facet for the Java incorporation package. To update the Java software without updating other software, unlock the `version-lock` facet of the Java incorporation package and then update the Java incorporation package. For more information about version-lock facets, see "Relaxing Version Constraints Specified by Incorporations" on page 84.

The following command changes the value of the `version-lock` facet of the installed `java-incorporation` package to `false`. The number of packages to update is the number of packages installed in this image, because each package in the image is checked for this facet.

```
$ pkg change-facet \
facet.version-lock.consolidation/java/java-incorporation=false
          Packages to update: 856
    Variants/Facets to change:   1
      Create boot environment:  No
Create backup boot environment: Yes

Planning linked: 1/1 done
PHASE                                ITEMS
Removing old actions                  1/1
Updating image state                 Done
Creating fast lookup database        Done
Reading search index                 Done
Building new search index         856/856
```

The following command shows that the facet value has been changed:

```
$ pkg facet
FACETS                                               VALUE
facet.version-lock.consolidation/java/java-incorporation False
```

Because the following command specifies the `-n` option, this command shows what would be change but does not actually make any changes to your image.

```
$ pkg update -nv java-incorporation
          Packages to update:        2
```

```
      Estimated space available:  80.91 GB
Estimated space to be consumed: 687.28 MB
        Create boot environment:      No
Create backup boot environment:      Yes
          Rebuild boot archive:      No

Changed packages:
solaris
  consolidation/java/java-incorporation
    0.5.11,5.11-0.175.2.0.0.9.0:20130304T213946Z -> 0.5.11,5.11-0.175.2.0.0.13.0
:20130429T145534Z
  runtime/java/jre-7
    1.7.0.17,5.11-0.175.2.0.0.9.0:20130304T214022Z -> 1.7.0.21,5.11-0.175.2.0.0.
13.0:20130429T145626Z
```

The following command performs the actual update. This command performs the update in the current image. You might want to use the `--be-name` option to perform the update in a new boot environment.

```
$ pkg update -v java-incorporation
            Packages to update:      2
      Estimated space available:  80.91 GB
Estimated space to be consumed: 687.28 MB
        Create boot environment:      No
Create backup boot environment:      Yes
          Rebuild boot archive:      No

Changed packages:
solaris
  consolidation/java/java-incorporation
    0.5.11,5.11-0.175.2.0.0.9.0:20130304T213946Z -> 0.5.11,5.11-0.175.2.0.0.13.0
:20130429T145534Z
  runtime/java/jre-7
    1.7.0.17,5.11-0.175.2.0.0.9.0:20130304T214022Z -> 1.7.0.21,5.11-0.175.2.0.0.
13.0:20130429T145626Z
DOWNLOAD                          PKGS         FILES    XFER (MB)    SPEED
Completed                         2/2        171/171    61.9/61.9    0B/s

PHASE                                    ITEMS
Removing old actions                      7/7
Installing new actions                    6/6
Updating modified actions               170/170
Updating package state database           Done
Updating package cache                    2/2
Updating image state                      Done
Creating fast lookup database             Done
Reading search index                      Done
Updating search index                     2/2
```

The following command verifies that the `jre-7` package is updated in this image. If you perform the update in a new boot environment, use `beadm mount` and the `pkg -R` to do the following check in that new boot environment.

```
$ pkg list jre-7
NAME (PUBLISHER)                              VERSION                IFO
runtime/java/jre-7                            1.7.0.21-0.175.2.0.0.13.0  i--
```

Updating an Incorporation When a Suitable Dependency Cannot Be Found

Error message: `A version for 'incorporate' dependency cannot be found`

See "Incorporation Packages" on page 12 for information about incorporation packages and their `incorporate` dependencies.

Examples of reasons an incorporation package might fail to update include the following installation states of an `incorporate` dependency of the incorporation package:

- The dependent package is frozen at a different version.
- The dependent package is already installed at a higher version.
- The dependent package is installed from a different publisher, and that publisher is sticky.

EXAMPLE A-2   Updating `pkg:/entire` When a Dependency is Unlocked and Separately Updated

The following example attempts to update all installed packages because no package names are specified. One of the installed packages that this operation attempts to update is the `pkg:/entire` incorporation package. This example shows an `incorporate` dependency of `pkg:/entire` that is already installed at a higher version.

```
$ pkg update --be-name s11.2
Creating Plan (Solver setup): /
pkg update: No solution was found to satisfy constraints
Plan Creation: Package solver has not found a solution to update to latest available versions.
This may indicate an overly constrained set of packages are installed.

latest incorporations:
[output omitted]
pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.10.0:20130318T181506Z

The following indicates why the system cannot update to the latest version:

No suitable version of required package pkg://solaris/entire@0.5.11,5.11-0.175.2
.0.0.10.0:20130318T181506Z found:
 Reject:  pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.10.0:20130318T181506Z
 Reason:  A version for 'incorporate' dependency on pkg:/consolidation/ub_javavm
/ub_javavm-incorporation@0.5.11,5.11-0.175.2.0.0.9.0 cannot be found
```

This message says that the version of the `pkg:/entire` incorporation package to which the system attempted to update specifies a version of the `ub_javavm-incorporation` package that cannot be installed. Because one package cannot be installed, no packages are installed and the update fails.

The following techniques can provide more information about why the `ub_javavm-incorporation` package cannot be installed:

- Use -v options. For example, use -v or -vv to receive more verbose output.
- Specify a package to update. Providing more detailed input often results in more detailed messaging. For example, in addition to the name of the package, include the version in the package FMRI.

The following command includes a -v option and specifies to update to the entire@0.5.11,5.11-0.175.2.0.0.10.0 package, copied from the message above. This command also specifies the -n option instead of the --be-name option. The -n option shows what will be done but does not actually make any changes to this image.

```
$ pkg update -nv entire@0.5.11,5.11-0.175.2.0.0.10.0
Creating Plan (Solver setup): /
pkg update: No matching version of entire can be installed:
 Reject:  pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.10.0:20130318T181506Z
 Reason:  All versions matching 'require' dependency pkg:/consolidation/ub_javav
m/ub_javavm-incorporation are rejected
  Reject:  pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,
5.11-0.151.0.1:20101105T053418Z
  pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5.11-0.17
5.0.0.2.0:20111019T144756Z
  pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5.11-0.17
5.0.10.1.0.0:20120920T143020Z
  Reason:  Excluded by proposed incorporation 'entire'
  Newer version pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.
5.11,5.11-0.175.2.0.0.13.0:20130429T145201Z is already installed
  Reject:  pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,
5.11-0.175.2.0.0.9.0:20130304T213739Z
  Reason:  Newer version pkg://solaris/consolidation/ub_javavm/ub_javavm-incorpo
ration@0.5.11,5.11-0.175.2.0.0.13.0:20130429T145201Z is already installed
 Reject:  pkg://solaris/consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5
.11-0.175.2.0.0.13.0:20130429T145201Z
 Reason:  Excluded by proposed incorporation 'entire'
```

These messages say that the version of the ub_javavm-incorporation package that is currently installed in this image is newer than the version specified by the pkg:/entire incorporation package that would be installed by the update operation.

The following command shows the version of the ub_javavm-incorporation package that is currently installed.

```
$ pkg list ub_javavm-incorporation
NAME (PUBLISHER)                                 VERSION              IFO
consolidation/ub_javavm/ub_javavm-incorporation  0.5.11-0.175.2.0.0.13.0   i--
```

The following command shows the version of the ub_javavm-incorporation package that is specified by the pkg:/entire incorporation package to which the system attempted to update. The version of the pkg:/entire incorporation package to which the system attempted to update is copied from the first "Reject" message in the above output. The -r option looks for this package in the configured package repositories, not in the installed image.

```
$ pkg contents -Hrt depend \
-a facet.version-lock.consolidation/ub_javavm/ub_javavm-incorporation=true \
```

```
-o fmri entire@0.5.11,5.11-0.175.2.0.0.10.0
consolidation/ub_javavm/ub_javavm-incorporation@0.5.11-0.175.2.0.0.9.0
```

To fix this problem, you can instruct the update operation to update all installed packages except for particular specified packages. Use one or more `--reject` options in the `pkg update` command to perform the update without attempting to update the packages specified in the `--reject` options. Wildcards can be used in `--reject` arguments. In the following command, packages to reject are copied from the "Reject" messages above that precede the "Reason: Newer version is already installed" message.

```
$ pkg update -v --be-name s11.2 \
--reject 'consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5.11-0.151*' \
--reject 'consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5.11-0.175.0*' \
--reject 'consolidation/ub_javavm/ub_javavm-incorporation@0.5.11,5.11-0.175.2.0.0.9.0*'
```

Tip - Be careful when you specify `--reject` arguments: Any packages that match the `--reject` argument that are already installed will be uninstalled.

Updating an Incorporation When an Installed Dependency is Not Permissible

Error messages:

- `The installed package` *package* `is not permissible`
- `Excluded by proposed incorporation`

If you receive these error messages, *package* probably is a core operating system package that must be kept in sync with other core operating system packages. Use the `pkg facet` command to check the value of the `facet.version-lock.`*package* facet. If the value of this `version-lock` facet is `false`, use the `pkg change-facet` command to change the value of this facet to `true`, and then try the update operation again.

Required Package Cannot Be Found

Error message: `A version for 'require' dependency cannot be found`

If you receive a message such as the following message that a required package cannot be found, check whether the package is available from your configured publishers.

```
pkg update: No solution was found to satisfy constraints
Plan Creation: Package solver has not found a solution to update to
latest available versions.
This may indicate an overly constrained set of packages are installed.
```

```
[output omitted]
No suitable version of required package package1 found:
Reject: package1
Reason: A version for 'require' dependency package2 cannot be found
```

Use the following command to show whether *package2* is available from your configured package publishers:

$ **pkg list -afv** *package2*

Use the pkg publisher command to check your publisher origin location. The publisher of this package is given after pkg:// in the full FMRI. You might need to change your publisher origin location. If the location is a local package repository, you might need to update the repository.

Required Package is Rejected

Error messages:

- No solution was found to satisfy constraints
- All versions matching 'require' dependency *package* are rejected

In the following output from the pkg update command, the initial error message states that a suitable version of the required package desktop-incorporation cannot be found. The reason the desktop-incorporation package is not suitable is because one of its dependent packages is not found. Because no suitable version is found, the desktop-incorporation package is rejected. The pkg update fails because the desktop-incorporation package is required by the pkg:/entire incorporation. The following command shows that the desktop-incorporation package is required by the pkg:/entire incorporation:

$ **pkg search -Hlo pkg.name require:consolidation/desktop/desktop-incorporation**

The next error message states that a suitable version of the required package python-extra-26 cannot be found. The reason the python-extra-26 package is not suitable is because the python-extra-26 package requires the desktop-incorporation package, and no suitable version of desktop-incorporation was found.

This information shows that the solution to this update failure is to find a way to install the required version of the desktop-incorporation package.

```
pkg update: No solution was found to satisfy constraints

maintained incorporations:
  [output omitted]

Plan Creation: dependency error(s) in proposed packages:
  [output omitted]
  No suitable version of required package pkg://solaris/consolidation/desktop
/desktop-incorporation@0.5.11,5.11-0.175.2.0.0.26.0:20131028T145233Z found:
    Reject: pkg://solaris/consolidation/desktop/desktop-incorporation
```

```
@0.5.11,5.11-0.175.2.0.0.26.0:20131028T145233Z
    Reason: A version for 'incorporate' dependency on pkg:/library/python-2
/python-sexy-26@0.1.9-0.175.0.0.0.1.0 cannot be found
  No suitable version of required package pkg://solaris/library/python-2
/python-extra-26@2.6.4-0.175.1.0.0.15.0:201205014T200156Z found:
    Reject:  pkg://solaris/library/python-2/python-extra-26@
2.6.4-0.175.1.0.0.15.0:201205014T200156Z
    Reason: All versions matching 'require' dependency pkg:/consolidation
/desktop/desktop-incorporation are rejected
      Reject: pkg://solaris/consolidation/desktop/desktop-incorporation
@0.5.11,5.11-0.175.0.0.0.2.0:20111019T132128Z
  [output omitted]
pkg://solaris/consolidation/desktop/desktop-incorporation
@0.5.11,5.11-0.175.2.0.0.26.0:20131028T145233Z
```

The following commands show that the package that cannot be found for the required `desktop-incorporation` package is not required. The `python-sexy-26` package is incorporated by the `desktop-incorporation` package, but it is not required.

```
$ pkg search -Hlo pkg.name require:library/python-2/python-sexy-26
$ pkg search -Hlo pkg.name incorporate:library/python-2/python-sexy-26
consolidation/desktop/desktop-incorporation
```

The `incorporate` dependency says that if the `python-sexy-26` package is installed, it must be installed at the specified version. However, because no package requires the `python-sexy-26` package, the `python-sexy-26` package does not need to be installed. Therefore, one solution to this update failure is to uninstall the `python-sexy-26` package. A different version of this package is currently installed, and the update could not locate the version specified by the `desktop-incorporation` incorporation. If you uninstall the package, the update process will not need to find the updated package.

If you need the `python-sexy-26` package and do not want to uninstall it, find a package repository that provides `pkg:/library/python-2 /python-sexy-26@0.1.9-0.175.0.0.0.1.0`. Either use the `pkg set-publisher` command to add that repository to your publisher origin locations, or use the `pkgrecv` command to add that package to your currently set publisher origin.

Packages Are Not Updated as Expected

Error message: `pkg update: The installed package` *package* `is not permissible.`

If you use wildcards with an update operation, you might not see any error messages even though some packages that should have been updated were not updated. You might see an error message if you specify a package name without wildcards.

For example, the following operation might complete without error and yet not update any packages, even though you verified that newer packages are available from your configured publishers:

```
$ pkg update '*'
```

To get more information, instead of using the wildcard, specify the version of `pkg:/entire` to which you want to update:

```
$ pkg list -Hafv entire
pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.9.0:20130304T214506Z  ---
$ pkg update -nv pkg://solaris/entire@0.5.11,5.11-0.175.2.0.0.9.0:20130304T214506Z
Creating Plan (Solver setup): -
pkg update: The installed package compress/zip is not permissible.
  Reject:  pkg://solaris/compress/zip@3.0,5.11-0.175.2.0.0.7.0:20121119T070339Z
  Reason:  Excluded by proposed incorporation 'consolidation/userland/userland-incorporation'
```

In this example, the package `compress/zip` is preventing the update of the `userland-incorporation` incorporation package, which is preventing the update of the `pkg:/entire` incorporation. The following command displays more information about the `compress/zip` package:

```
$ pkg list compress/zip
NAME (PUBLISHER)  VERSION                IFO
compress/zip      3.0-5.11-0.175.2.0.0.7.0  if-
```

This output shows that the `compress/zip` package is frozen. Because the package is frozen, it cannot be updated.

```
$ pkg unfreeze compress/zip
compress/zip was unfrozen.
```

With the `compress/zip` package unfrozen, the original `pkg update '*'` operation should update all packages in the image that have updates available. See "Locking Packages to a Specified Version" on page 82 for more information about the `pkg freeze` and `pkg unfreeze` commands.

The `pkg sync-linked` error shown in "Sync Linked Package Cannot Be Installed" on page 120 is similar but indicates a package in a non-global zone is preventing the update.

Sync Linked Package Cannot Be Installed

Error message: pkg sync-linked: The installed package *package* is not permissible.

A `sync-linked` package is a package in a non-global zone. You might see a package rejected with the reason being a version mismatch with the parent image.

```
Linked progress: -pkg: update failed (linked image exception(s)):

A 'sync-linked' operation failed for child 'zone:z1' with an unexpected
return value of 1 and generated the following output:
```

```
pkg sync-linked: The installed package package is not permissible.
  Reject: package
  Reason:  Parent image has a incompatible newer version: package
```

You might receive this incompatibility message for the following reasons:

■ Core operating system packages must be the same version in non-global zones as in the global zone. You cannot update these packages separately in a non-global zone. Similarly, freezing any of these packages in a non-global zone causes the update to fail for the global zone and all non-global zones.

The following command displays the list of packages that must remain in sync between the global zone and non-global zones:

```
$ pkg search -o pkg.name :depend:parent:
```

■ If you are updating a BE that has zones configured, you cannot mount that BE and then use the pkg -R command to update that alternate BE (ABE) if the publisher configuration in the ABE is different from the publisher configuration in the currently booted BE. Non-global zones in the ABE use the publisher configuration from the currently active BE.

Non-Global Zone Cannot Be Installed

Error message: `The following pattern(s) did not match any allowable packages. Try using a different matching pattern, or refreshing publisher information`

Core operating system packages must be the same version in non-global zones as in the global zone. If the solaris publisher origin in this image is set to a package repository that does not contain the same versions of the system packages that are installed in the global zone, attempting to install a non-global zone results in the following error:

```
$ zoneadm -z myzone install
The following ZFS file system(s) have been created:
    rpool/export/zones/myzone
Progress being logged to /var/log/zones/zoneadm.20131104T181301Z.myzone.install
       Image: Preparing at /export/zones/myzone/root.

AI Manifest: /tmp/manifest.xml.9daq.i
SC Profile: /usr/share/auto_install/sc_profiles/enable_sci.xml
  Zonename: myzone
Installation: Starting ...

            Creating IPS image
Startup linked: 1/1 done
            Installing packages from:
                solaris
                    origin:  http://pkg.oracle.com/solaris/release/
            Error occurred during execution of 'generated-transfer-4606-1' checkpoint.
            Failed Checkpoints:

            Checkpoint execution error:
```

```
                        The following pattern(s) did not match any allowable packages. Try
                        using a different matching pattern, or refreshing publisher information:

Installation: Failed.  See install log at /system/volatile/install.4606/install_log
ERROR: auto-install failed.
```

To install a non-global zone, the repository that you set as the `solaris` publisher origin must contain at least the same system software that is installed in the global zone where you are installing the non-global zone. The repository can also contain older or newer software, but it must contain the same software that is installed in the global zone. The following command shows that the `/export/IPSpkgrepos/Solaris` repository is a suitable publisher origin for this global zone because this repository contains the same version of the `pkg:/entire` package that is installed in the global zone:

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                 IFO
entire                0.5.11-0.175.2.0.0.26.0  i--
$ pkgrepo list -H -s /export/IPSpkgrepos/Solaris \
entire@0.5.11-0.175.2.0.0.26.0
solaris      entire     0.5.11-0.175.2.0.0.26.0:20131028T190148Z
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

Image Cannot Be Modified

Error message: `pkg: The image cannot be modified as it is currently in use by another package client`

The error message should include the name and the pid of the client that has the image locked, as shown in the following example:

```
pkg: The image cannot be modified as it is currently in use by another package
client: pkg on cbusl0406l, pid 26604.
```

Try your `pkg` command again after the current package process exits (process 26604 in this example). If the process does not exit quickly, use `ptree -a`, for example, to begin to examine the process.

The package process that is currently operating on this image might be Update Manager checking whether updates are available. See "Using Update Manager" on page 128 for more information about the Update Manager `cron` job and `pkg/update` service.

Files Were Salvaged

The following informational message is followed by the path of the files that were salvaged and the temporary location where the files were moved:

```
The following unexpected or editable files and directories were
salvaged while executing the requested package operation; they
have been moved to the displayed location in the image:
```

This is an informational message and can be ignored or acted upon as necessary.

Directories are reference-counted in IPS. When the last package installed in the image that either explicitly or implicitly references a directory no longer references that directory, that directory is removed. If that directory contains unpackaged file system objects, those items are moved into `$IMAGE_META/lost+found`. Unpackaged file system objects are files and directories that were not delivered by an IPS package. The value of `IMAGE_META` is typically `/var/pkg`. See the "Files" section of the `pkg`(5) man page for information about the `IMAGE_META` directory.

Minimize Stored Image Metadata

The `/var/pkg` directory holds metadata for the image. This directory can get quite large. Do not manually remove anything from the `/var/pkg` directory.

Make sure the `flush-content-cache-on-success` image property value is set to `true`. The value of the `flush-content-cache-on-success` property is `true` by default. When the value of `flush-content-cache-on-success` is `true`, cached files are removed when `pkg install` and `pkg update` operations complete successfully. If the `flush-content-cache-on-success` property is set to `false`, you can use the command shown in the following example to reset the value to `true`:

```
$ pkg property flush-content-cache-on-success
PROPERTY                         VALUE
flush-content-cache-on-success False
$ pkg set-property flush-content-cache-on-success true
$ pkg property -H flush-content-cache-on-success
flush-content-cache-on-success True
```

Flushing the content cache (setting `flush-content-cache-on-success` to `true`) can cause some `pkg` operations to take longer to complete.

Non-global zones have a different cache, which you can set with the `-c` option of `/usr/lib/pkg.sysrepo`. You can set the maximum size of this cache with the `-s` option. See the `pkg.sysrepo`(1M) man page.

Increase Package Installation Performance

The following steps can help increase package installation and update performance:

- Make sure your ZFS storage pool capacity is less than 80%.

```
$ zpool list
NAME  SIZE ALLOC FREE CAP DEDUP HEALTH ALTROOT
rpool 186G 75.2G 111G 40% 1.00x ONLINE -
```

■ Use a local package repository. See "Copying and Creating Package Repositories in Oracle Solaris 11.2 ".

■ If `http_proxy` is set, check the performance of the proxy.

B

♦ ♦ ♦   **A P P E N D I X   B**

IPS Graphical User Interfaces

IPS includes two Graphical User Interface (GUI) tools.

■   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, you can use Package Manager to quickly identify and install packages.

■   Update Manager updates all packages in the image 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:

Toolbar	Click the Package Manager icon in the toolbar. The Package Manager icon is a box with a circling arrow.
Desktop icon	Double-click the Package Manager icon on the desktop.
Menu bar	Select System → Administration → Package Manager.
Command line	`$ packagemanager &`

For complete Package Manager documentation, select Help → Contents from the Package Manager menu bar.

Package Manager Command Line Options

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

TABLE B-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`: $ **packagemanager -R /aux0/example_root**
`--update-all` or `-U`	Update all installed packages that have updates available. Specifying this option is the same as selecting the Updates option in the Package Manager GUI. See "Using Update Manager" on page 128 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 "Using Web Install" on page 126 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.

▼ How to Create a Web Install File

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.

1. **Select the publisher.**

    From the Package Manager Publisher drop-down menu, select the publisher from which you want to include the packages in the `.p5i` file.

2. **Select the package.**

In the Package Manager package list pane, select the package whose installation instructions you want to distribute.

3. **Export your selections.**

   Select File → Export Selections to display the Export Selections Confirmation window.

4. **Confirm the selections.**

   Click the OK button to confirm the selections. The Export Selections window is displayed.

5. **(Optional) Change the file name.**

   A default name for the `.p5i` file is provided. You can change this file name.

   Do not change the `.p5i` extension.

6. **(Optional) Change the file location.**

   A default location for the `.p5i` file is provided. You can change the location.

7. **Save the Web Install file.**

   Click the Save button to save the file name and location.

▼ How to Use 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. **Start Package Manager in Web Install mode.**

   - **Select a `.p5i` file on your desktop.**

   - **Start Package Manager from the command line and specify a `.p5i` file.**

     ```
     $ 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 select it.

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.

2. **Click the Proceed button to continue with the installation.**

   - **If necessary, add publishers.**

   - **If necessary,**

3. **If necessary, add publishers.**

   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.

   If the publisher is added successfully, the Adding Publisher Complete dialog displays.

4. **Click the OK button to continue with the installation.**

5. **Enable any disabled publishers.**

   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 package dependencies and publisher configuration. This function is the same as the following functions:

- In the Package Manager GUI, selecting the Updates button or the Package→Updates menu option.
- Using the `packagemanager` command.

  ```
  $ packagemanager --update-all
  ```
- Using 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	Select System → Administration → Update Manager.
Command line	`$ pm-updatemanager`
Automated	The Update Manager package, `package/pkg/update-manager`, delivers the `cron` job `/usr/lib/update-manager/update-refresh.sh`. `30 0,9,12,18,21 * * * /usr/lib/update-manager/update-refresh.sh` When the SMF service `svc:/application/pkg/update` is online, this `cron` job checks periodically for updated packages available from configured publishers (the first two steps of the following process). If updated packages are available, you receive a notification in your desktop toolbar. Select the notification icon to open the Update Manager GUI.

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.

   Click the Proceed button to continue with the update.
3. 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/package-manager
   package/pkg/update-manager
   ```

   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.

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

4. A new BE might be created, depending on which packages are updated and depending on your image policy.

   If the system created a new BE for the update, you can edit the default BE name.

   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.

   - Click the Restart Now button to restart your system immediately.
   - Click the Restart Later button to restart your system at a later time.

Update Manager Command Line Options

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

TABLE B-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`:
	`$` **`pm-updatemanager -R /aux0/example_root`**
`--help` or `-h`	Display command usage information.

Index

A

application version mediation, 86
avoid list, 88
   adding to, 45, 51
   removing from, 41, 52

B

BE, 16
   activating, 38
   creating during package installation, 44
   image policy property settings, 91
   naming, 38
   `pkg` command options, 38
   requiring, 38
`beadm` command, 44
boot environment (BE), 16

C

child image, 16
constraints, 12, 110
   relaxing, 84

E

`entire` incorporation package, 13

F

facet, 17, 79, 84
Fault Management Resource Identifier (FMRI), 13
files
   preserve or overlay editable, 37
FMRI, 13

G

global zone, 16, 39, 52
group, 13, 26, 32, 88
group package, 13, 26, 32, 88

I

image, 16
   creating, 98
   policies, 91
   properties, 91
      adding and removing, 94
      BE creation policy, 91
      using signed packages, 93
   reinstalling, 51
   upgrading, 59
incorporation, 12, 84, 110
incorporation package, 12, 84, 101, 110
   custom, 63
installation constraints, 12, 110

K

kernel zone, 16

L

linked image, 16, 39, 56
   `pkg uninstall`, 51
   `pkg update`, 46

M

mediation, 86
mediator, 86
mirror repository, 16

W
Web Install, 126

Z
zone, 39, 52
   global zone, 16
   kernel zone, 16
   non-global zone, 16
   Oracle Solaris 10 zone, 16
zones proxy services
   `zones-proxy-client`, 54
   `zones-proxyd`, 54