Updating Systems and Adding Software in Oracle Solaris 11.4

E60979-02
February 2024

ORACLE

Updating Systems and Adding Software in Oracle Solaris 11.4,

E60979-02

Primary Author: Cathleen Reiher

Contents

4 Updating or Upgrading an Oracle Solaris Image

5 Configuring Installed Images

A Troubleshooting Package Installation and Update

Index

Using This Documentation

- **Overview –** Describes the software installation functions of the Oracle Solaris 11.4 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

Documentation and resources for this product and related products are available at http://www.oracle.com/pls/topic/lookup?ctx=E37838-01.

Feedback

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

1

Introduction to the Image Packaging System

The Oracle Solaris Image Packaging System (IPS) is a framework that enables you to do the following tasks:

- List and search software packages
- Install, update, and remove software
- Upgrade to a new Oracle Solaris operating system release

IPS interfaces enable you to restrict which packages can be installed, which versions of packages can be installed, and how installed software needs to be signed.

This chapter describes IPS concepts and describes how to gain the privilege you need to do tasks such as install and update software.

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.

IPS tools provide the following capabilities. See IPS Concepts 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 *Creating Package Repositories in Oracle Solaris 11.4*.
- Create and publish packages. See *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4*.
- 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 *Automatically Installing Oracle Solaris 11.4 Systems* or *Manually Installing an Oracle Solaris 11.4 System*.

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 11.4* or the `pkg`(7) man page for descriptions of package actions.

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

Constraint Packages

A *constraint* package specifies `incorporate` dependencies to define a *surface* in the package version space that is compatible. Constraint packages are 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.

An `incorporate` dependency constrains the versions of that package that can be installed in this image. A package is not installed just by being named as an `incorporate` dependency. If the package is installed by some other means (for example, it is also a `require` dependency or you explicitly install the package), then only a version prescribed by the `incorporate` dependency can be installed. For example, if a package specified as an `incorporate` dependency in an installed constraint 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.

Packages named as `incorporate` dependencies in the constraint package might themselves be constraint packages. In this way, many packages can be affected by a constraint package even if they are not named in the manifest of the constraint package. Packages whose installation is affected by a constraint package are *constrained* by that constraint package. Updating a constraint package `A-constraint` that has an `incorporate` dependency on the constraint package `B-constraint` results in also updating `B-constraint` and all the other packages that are constrained by `B-constraint`. An attempt to update `B-constraint` separately from `A-constraint` could result in an error.

Constraint packages 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 an `incorporate` dependency of a constraint package. Instead, you should update the constraint 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 Constraint Packages for related information.

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

> **Caution:**
>
> Do not remove the `pkg:/entire` package. The `pkg:/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 constraint package. Removing the `pkg:/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 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 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

The *scheme*, *publisher*, and *version* parts of the FMRI string are optional.

The following rules apply to matching package names and versions in IPS command operands:

- Package names. You can use the smallest portion of the package name that uniquely identifies the package. You can use the `?` and `*` characters as `glob`(3C)-style wildcards in the package name to match one or more packages.
- Package versions. If you specify a version, you can omit components from the right hand side of the version string. You can use the `*` character as a wildcard to match a whole component of the version string. A `*` character cannot be used to match a partial component. A single `*` character cannot match more than one component. The `?` cannot be used in a version string.

The following examples illustrate package matching:

- Match any installed package that has `jre` anywhere in the name:

```
$ pkg list '*jre*'
NAME (PUBLISHER)                        VERSION                    IFO
runtime/java/jre-8                      1.8.0.181.12               i--
```

- Match all installed packages with `java` anywhere in the name, any component version, and `11.4` branch version:

```
$ pkg list '*java*@*-11.4'
NAME (PUBLISHER)                        VERSION                    IFO
library/javascript/jjv                  1.0.2-11.4.0.0.1.10.0      i--
system/management/rad/client/rad-java   11.4-11.4.0.0.1.10.1       i--
```

- Notice the one-character difference between the following command and the previous command:

```
$ pkg list '*java*@*11.4'
pkg list: Illegal FMRI '*java*@*11.4': Bad Version: *11.4
```

- An asterisk `*` cannot match a partial version component or more than one version component:

```
$ pkg list '*java*@11.4-11.4.*.10.0'
pkg list: no packages matching the following patterns are installed:
  *java*@11.4-11.4.*.10.0
$ pkg list '*java*@*.10.0'
pkg list: no packages matching the following patterns are installed:
*java*@*.10.0
```

The scheme for every IPS package FMRI is `pkg`. In the following example package FMRI for the `compliance` security compliance framework, `solaris` is the publisher, `security/compliance` is the package name, and `11.4-11.4.0.0.1.10.1:20180702T144054Z` is the version:

```
pkg://solaris/security/compliance@11.4-11.4.0.0.1.10.1:20180702T144054Z
```

Scheme
`pkg`

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

Package name
`security/compliance`
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 version string consists of the following three parts, and the format of the time stamp is *date*`T`*time*`Z`:

```
component_version-branch_version:time_stamp
```

Component version number: `11.4`
For components that are tightly bound to the operating system, the component version number is *minor*`.`*update*. Other components, such as FOSS components, have their own version numbers. In the following example, the version number of the `web/server/apache-24` package is 2.4.25:

```
pkg://solaris/web/server/apache-24@2.4.33-11.4.0.0.1.10.0:20180702T172601Z
```

The component version number is the same as the Version in `pkg info` output.

Branch version: `11.4.0.0.1.10.1`
The branch version, if present, must follow a hyphen (-). The branch version string is the same as the Branch in `pkg info` output.
Oracle Solaris packages show the following information in the branch version portion of the version string of a package FMRI:

`minor.update.sru.order.platform.build.rev`

Minor release number: `11`
The *minor* portion of *major*`.`*minor* that is output by the `uname -r` command.

Update release number: `4`
The update release number for this Oracle Solaris release.

SRU number: `0`
The Support Repository Update (SRU) number for this update release. SRUs are approximately monthly updates that fix bugs, fix security issues, or provide support for new hardware. The Oracle Support Repository is available only to systems under a support contract.

Order: `0`
This value is used internally.

Platform: `1`
This value is used internally.

Release or SRU build number: `10`
The build number of the SRU, or the respin number for the release.

Revision or nightly build number: `1`
The build number for the individual nightly builds.

If the package is an Interim Diagnostic or Relief (IDR) update, then the *component_version-branch_version* of the package FMRI is just a single field. For example, the FMRI for `idr1929` is `pkg://solaris/idr1929@4:20160216T222617Z`; the complete *component_version-branch_version* part of the version is simply `4`. IDRs are package updates that help diagnose customer issues or provide temporary relief for a problem until a formal package update is issued. See Installing an IDR Custom Software Update for more information about IDRs.

Time stamp: `20180702T144054Z`
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. IPS 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`(7) 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. You can also explicitly create a new BE by specifying options described in Boot Environment Options. See *Creating and Administering Oracle Solaris 11.4 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. 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.

For more information about facets and variants, including how to view or modify the values of the facets and variants set on the image, see Controlling Installation of Optional Components.

Installation Privileges

The commands discussed in 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 11.4* for more information about roles and profiles, including how to determine which role or profile you need.

Roles
Use the `roles` command to list the roles that are assigned to you. Use the `su` command with the name of the role to assume that role. As this role, you can execute any commands that are permitted by the rights profiles that are assigned to that role. For example, if the role

is assigned the Software Installation rights profile, you can execute the `pkg` and `beadm` commands to install and update packages and manage boot environments.

Rights profiles
Use the `profiles` command to list the rights profiles that are assigned to you. Use one of the following methods to execute commands that your rights profiles permit you to execute:

- Use a profile shell such as `pfbash` or `pfksh`.
- Use the `pfexec` command in front of the command that you want to execute. In general, you must specify the `pfexec` command with each privileged command that you execute.

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. Escape 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 '*java*8*'
NAME (PUBLISHER)                                  VERSION            IFO
consolidation/java-8/java-8-incorporation         1.8.0.172.11-0     i--
runtime/java/jre-8                                1.8.0.172.11       i--
```

If a publisher name is shown in parentheses after the package name, that publisher is not the first publisher in the publisher search order in this image. Both of the packages shown in this example are published by the publisher that is the first publisher in the search order. See Setting Publisher Search Order and Stickiness.

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 '*java*8*'
NAME (PUBLISHER)                                VERSION                    IFO
consolidation/java-8/java-8-incorporation       1.8.0.172.11-0             i--
developer/java/jdk-8                            1.8.0.172.11               ---
library/java/java-demo-8                        1.8.0.172.11               ---
runtime/java/jre-8                              1.8.0.172.11               i--
web/java-servlet/tomcat-8                       8.5.28-11.4.0.0.1.10.0     ---
web/java-servlet/tomcat-8/tomcat-admin          8.5.28-11.4.0.0.1.10.0     ---
web/java-servlet/tomcat-8/tomcat-examples       8.5.28-11.4.0.0.1.10.0     ---
```

This output indicates that a developer kit, a demo library, and `tomcat` packages are available and can be installed in this image.

The `-a` option also shows any matching obsolete packages. An obsolete package shows an `o` in the O column and cannot be installed. See Renamed, Legacy, and Obsolete Packages for more information.

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 '*java*8*'
NAME (PUBLISHER)                                VERSION                    IFO
consolidation/java-8/java-8-incorporation       1.8.0.181.12-0             ---
developer/java/jdk-8                            1.8.0.181.12               ---
library/java/java-demo-8                        1.8.0.181.12               ---
runtime/java/jre-8                              1.8.0.181.12               ---
web/java-servlet/tomcat-8                       8.5.28-11.4.0.0.1.10.0     ---
web/java-servlet/tomcat-8/tomcat-admin          8.5.28-11.4.0.0.1.10.0     ---
web/java-servlet/tomcat-8/tomcat-examples       8.5.28-11.4.0.0.1.10.0     ---
```

The `tomcat` packages are the same version as the packages listed with the `-a` option; these `tomcat` packages could be installed. Other packages on this list are a newer version than the same packages that are listed with the `-a` option. You know these packages cannot be installed into this image because they are not listed by the `-a` option. These packages cannot be installed because the version that can be installed is constrained as described in Constraint Packages. To get these packages on your system, you must `pkg update` the system, which will create a new image.

The newest version of a package could be a renamed, legacy, or obsolete package (shows an `r`, `l`, or `o` in the `O` column). See Renamed, Legacy, and Obsolete Packages for more information.

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 discussed in Newest Packages. The installed packages shown below have updates available, but those

updates cannot be installed into the current image. 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 '*java*8*'
NAME (PUBLISHER)                                  VERSION                    IFO
consolidation/java-8/java-8-incorporation         1.8.0.172.11-0             i--
untime/java/jre-8                                 1.8.0.172.11               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 jre-8
NAME (PUBLISHER)                                  VERSION                    IFO
runtime/java/jre-8                                1.8.0.181.12               ---
runtime/java/jre-8                                1.8.0.172.11               i--
runtime/java/jre-8                                1.8.0.162.12               ---
...
```

Renamed, Legacy, and Obsolete Packages

Packages that are designated as legacy deliver software that is in the end-of-feature (EOF) process. This software is listed in End of Features (EOF) Planned for Future Updates of Oracle Solaris. Packages that are designated as obsolete deliver software that is EOF. This software is listed in that same EOF document, under the release in which the software was removed.

In the following example, `l` in the `O` column indicates that the package is in the legacy state, and `o` indicates that the package is obsolete.

```
NAME (PUBLISHER)      VERSION                    IFO
runtime/python-27          2.7.18-11.4.27.0.1.82.0    i-l
runtime/python-26          2.6.8-11.4.0.0.1.9.0       --o
```

Obsolete packages cannot be installed. If you try to install an obsolete package, the installation fails with the message that no updates are necessary for this image.

```
$ pkg install runtime/python-26
No updates necessary for this image.
```

The following command lists all installed legacy packages:

```
$ pkg search -l -o pkg.fmri :pkg.legacy:true
```

See End of Features (EOF) Planned for Future Updates of Oracle Solaris and the SRU Readme files for suggestions for alternatives for some legacy software.

An `r` in the `O` column indicates that the package has been renamed. If you try to install a renamed package, the system attempts to install the package to which the renamed package has been renamed.

The following example shows a package that was renamed:

```
NAME (PUBLISHER)         VERSION                    IFO
print/filter/ghostscript      9.27-11.4.21.0.1.69.0      --r
```

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 print/filter/ghostscript
          Name: print/filter/ghostscript
         State: Not installed (Renamed)
    Renamed to: desktop/pdf-viewer/gsx
     Publisher: solaris
       Version: 9.27
        Branch: 11.4.21.0.1.69.0
Packaging Date: April 24, 2020 at  8:20:39 AM
          Size: 2.52 kB
          FMRI: pkg://solaris/print/filter/
ghostscript@9.27-11.4.21.0.1.69.0:20200424T082039Z
```

If you try to install the `print/filter/ghostscript` package, the `desktop/pdf-viewer/gsx` package will be installed instead if it is not already installed and if it can be installed in this image.

If you try to install the `print/filter/ghostscript` package and the Renamed To package is already installed, the packaging system reports that no updates are necessary.

```
$ pkg info desktop/pdf-viewer/gsx
             Name: desktop/pdf-viewer/gsx
          Summary: GPL Ghostscript desktop utilities
         Category: System/Printing
            State: Installed
        Publisher: solaris
          Version: 9.52
           Branch: 11.4.27.0.1.82.0
   Packaging Date: October  9, 2020 at  7:42:10 PM
Last Install Time: July  8, 2020 at  7:55:47 PM
 Last Update Time: October 15, 2020 at  2:43:28 AM
             Size: 104.45 kB
             FMRI: pkg://solaris/desktop/pdf-viewer/
gsx@9.52-11.4.27.0.1.82.0:20201009T194210Z
      Project URL: http://ghostscript.com/
       Source URL: https://github.com/ArtifexSoftware/ghostpdl-downloads/
releases/download/
gs952/ghostscript-9.52.tar.gz
```

The `pkgrepo list` command also shows the `r` flag for renamed packages, the `l` flag for legacy packages, and the `o` flag for obsolete packages. The output of the following example is reduced for brevity:

```
$ pkgrepo -s repository-uri list
PUBLISHER NAME                          O VERSION
solaris   print/filter/ghostscript      r 9.27-11.4.21.0.1.69.0:20200424T082039Z
solaris   runtime/python-27             l 2.7.18-11.4.27.0.1.82.0:20201009T225207Z
solaris   runtime/python-26             o 2.6.8-11.4.0.0.1.9.0:20200305T223006Z
```

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 for information about freezing packages.

```
$ pkg list -v entire
FMRI                                                                    IFO
pkg://solaris/entire@11.4-11.4.0.0.1.10.0:20180702T173343Z              i--
$ pkg freeze -c "Prevent update to SRU 1 until ready." entire@11.4-11.4.0
entire was frozen at 11.4-11.4.0
$ pkg list -v entire
FMRI                                                                    IFO
pkg://solaris/entire@11.4-11.4.0.0.1.10.0:20180702T173343Z              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, size, whether the package is frozen, the last install or update time, and the source location for open source software. The `pkg info` command can separately display license information.

Displaying Package Description, When Packaged and Installed, and 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@11.4-11.4.0.0.1.10.0:20180702T173343Z              i--
```

The `pkg info` command displays more information, including the time the package was packaged and the size of the package. In this example, the State of the `pkg info` output shows that this package is frozen:

```
$ 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 (Frozen)
```

```
     Publisher: solaris
       Version: 11.4 (Oracle Solaris 11.4.0.0.1.10.0)
        Branch: 11.4.0.0.1.10.0
Packaging Date: Mon Jul 02 17:33:43 2018
          Size: 2.53 kB
          FMRI: pkg://solaris/entire@11.4-11.4.0.0.1.10.0:20180702T173343Z
```

If the information is available, the `pkg info` command displays the last time the package was installed or updated. If the last install or update time is not available from `pkg info`, the package might have been installed when this BE was initially created. In that case, use the BE creation date shown by the `beadm list` command. You can also use the `pkg history` command as described in Viewing Operation History to find all package install, update, and uninstall times.

The following output shows the last install time for the package:

```
$ pkg info mercurial-27
             Name: developer/versioning/mercurial-27
          Summary: The Mercurial Source Control Management System
      Description: A fast, lightweight source control management system designed
                   for efficient handling of very large distributed projects.
         Category: Development/Source Code Management
            State: Installed
        Publisher: solaris
          Version: 4.1.3
           Branch: 11.4.0.0.1.10.0
   Packaging Date: Mon Jul 02 16:32:13 2018
Last Install Time: Mon Aug 06 15:22:47 2018
             Size: 9.90 MB
             FMRI: pkg://solaris/developer/versioning/
mercurial-27@4.1.3-11.4.0.0.1.10.0:20180702T163213Z
      Project URL: http://mercurial-scm.org/
       Source URL: https://www.mercurial-scm.org/release/mercurial-4.1.3.tar.gz
```

As shown in Renamed, Legacy, and Obsolete Packages, you can use the `pkg info` command to find the new name of a renamed package. The State line of the `pkg info` output also shows whether the package is frozen.

Displaying Project and Source Locations for Open Source

For FOSS packages, the Project URL and Source URL are displayed:

```
$ pkg info openssl
          Name: library/security/openssl
       Summary: OpenSSL - a Toolkit for Secure Sockets Layer (SSL v2/v3) and
                Transport Layer (TLS v1) protocols and general purpose
                cryptographic library
   Description: OpenSSL is a full-featured toolkit implementing the Secure
                Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1)
                protocols as well as a full-strength general purpose
                cryptography library.
      Category: System/Security
         State: Installed
     Publisher: solaris
       Version: 1.0.2.15 (1.0.2o)
        Branch: 11.4.0.0.1.10.0
Packaging Date: Mon Jul 02 17:04:44 2018
          Size: 16.05 MB
          FMRI: pkg://solaris/library/security/
```

```
openssl@1.0.2.15-11.4.0.0.1.10.0:20180702T170444Z
   Project URL: https://www.openssl.org/
    Source URL: https://www.openssl.org/source/openssl-1.0.2o.tar.gz
```

Displaying Package Licenses

Use the `--license` option to display the license texts for the matching packages. Use the `-r` option to display the license text for a package that is not installed.

```
$ pkg info -r --license install-image/solaris-auto-install
Copyright (c) 1983, 2020, Oracle and/or its affiliates.

You acknowledge that your use of this Oracle Solaris software product
is subject to, and may not exceed the use for which you are authorized,
(i) the license or cloud services 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
https://www.oracle.com/downloads/licenses/solaris-cluster-express-license.html.
Note: Software downloaded for trial use or downloaded as replacement
media may not be used to update any unsupported software.
```

License information can be quite lengthy. Other information that the `pkg info` command shows when the `--license` option is not specified is not displayed. See Displaying License Requirements to list packages that require you to accept their license.

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. See also Comparing the pkg search and pkg contents Commands.

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 amp
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 for information about variants and facets.

Displaying Information that is not a Path

To display information that is not a path or to display a subset of path information, use the `-t`, `-a`, and `-o` options of the `pkg contents` command.

The `-t` option specifies the type of action to select, such as `file`, `link`, or `depend`. See the `pkg`(7) man page for a list of package actions. You can specify multiple action types in a comma-separated list, or you can specify the `-t` option multiple times. When you use the `-t` option, the default output are the values of the key attributes for that action.

The `-a` option specifies which attribute values of the action to select. See the `pkg`(7) man page for a list of attributes of each action type. You can specify the `-a` option multiple times.

The `-o` option specifies the output to display. You can specify the name of an attribute as described for the `-a` option or any of several pseudo attribute names listed in the `pkg`(1) man page. You can specify multiple attribute arguments in a comma-separated list, or you can specify the `-o` option multiple times. Each `-o` argument is one column of output.

Example 2-1 Displaying Attributes of Files Installed by a Package

This example displays additional attributes of the paths listed in the example in Listing Files Installed by a Package.

```
$ pkg contents -o owner,group,mode,path zip
OWNER GROUP MODE PATH
root  bin   0555 usr/bin/zip
root  bin   0555 usr/bin/zipcloak
root  bin   0555 usr/bin/zipnote
root  bin   0555 usr/bin/zipsplit
root  bin   0444 usr/share/man/man1/zip.1
root  bin   0444 usr/share/man/man1/zipcloak.1
root  bin   0444 usr/share/man/man1/zipnote.1
root  bin   0444 usr/share/man/man1/zipsplit.1
```

The default output displayed if the `-o` option is not specified is values of `path` attributes. The following output shows that the `oracle-rdbms-server-18c-preinstall` package does not deliver any files, links, or directories directly.

```
$ pkg contents -r oracle-rdbms-server-18c-preinstall
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.
```

All of the following commands show that the `oracle-rdbms-server-12-1-preinstall` package has `set`, `signature`, and `depend` actions:

```
$ pkg contents -rm oracle-rdbms-server-18c-preinstall
$ pkg contents -ro action.name oracle-rdbms-server-18c-preinstall
$ pkg contents -ro action.raw oracle-rdbms-server-18c-preinstall
```

Example 2-2 Specifying an Action Type

The following command displays the dependency type and package name of each dependency in the `oracle-rdbms-server-18c-preinstall` package:

```
$ pkg contents -rt depend oracle-rdbms-server-18c-preinstall
TYPE    FMRI
group   system/header
group   system/kernel/oracka
group   system/picl
group   x11/diagnostic/x11-info-clients
group   x11/library/libxi
group   x11/library/libxtst
group   x11/session/xauth
require compress/unzip
require developer/assembler
require developer/build/make
require system/dtrace
require system/library/openmp
```

Example 2-3 Specifying Action Attributes

If you are only interested in some of the dependencies, use the `-a` option to narrow the selection.

```
$ pkg contents -rt depend -a fmri='*lib*' oracle-rdbms-server-18c-preinstall
TYPE    FMRI
group   x11/library/libxi
group   x11/library/libxtst
require system/library/openmp
```

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.

In the following command, the specified attributes are unique to the `set` action, and you do not need to specify the `set` action type:

```
$ pkg contents -ra name=pkg.summary -a name=pkg.description -o name,value -s value
oracle-rdbms-server-18c-preinstall
NAME            VALUE
pkg.summary     Prerequisite package for Oracle Database 18c
pkg.description Provides the set of Oracle Solaris packages required for installation
and operation of Oracle Database 18c.
```

If you do specify the `set` action type, then you do not need to specify the output columns because `name` and `value` are the default output for `set` actions:

```
$ pkg contents -rt set -a name=pkg.summary -a name=pkg.description -s value oracle-
rdbms-server-18c-preinstall
```

Example 2-4 Displaying All Links Involved in a Mediation

The following command shows the `path` and `target` of links that participate in the `python` mediation delivered by the `python-35` package. See Specifying a Default Application Implementation for information about mediations of multiple versions.

```
$ pkg contents -a mediator=python -o path,target python-35
PATH                                TARGET
usr/bin/2to3                        2to3-3.5
usr/bin/idle                        idle3.5
usr/bin/pydoc                       pydoc3.5
usr/bin/python                      python3.5
usr/bin/python-config               python3.5-config
usr/bin/pyvenv                      pyvenv-3.5
usr/lib/pkgconfig/python3.pc        python-3.5.pc
usr/lib/sparcv9/pkgconfig/python3.pc python-3.5.pc
usr/share/man/man1/python3.1        python3.5.1
```

Example 2-5 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`(7) 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, Python 2.7.14, Python 3.4.6, and Python 3.5.3 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 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 python2.7 runtime/python-27
usr/bin/python python3.4 runtime/python-34
usr/bin/python python3.5 runtime/python-35
```

Listing All Installable Packages in a Group Package

Oracle Solaris provides several system installation group packages. 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 the specified group package.

> **Note:**
>
> This package list does not include every package that is installed when you install the group package. Dependencies of the packages in this list will also be installed, and dependencies of those dependencies.

```
$ pkg contents -rt depend solaris-minimal-server
TYPE    FMRI
group   network/ping
group   network/ssh
group   service/network/ssh
group   shell/tcsh
group   shell/zsh
group   system/network
group   system/rsyslog
require developer/debug/mdb
require editor/vim/vim-core
require group/system/solaris-core-platform
require package/pkg
require release/name
require release/notices
require shell/bash
require shell/ksh93
require system/core-os
require system/library/platform
```

The `-t` option matches `depend` actions in the package. You can avoid installing packages that are `group` type dependencies. See Avoiding Installing Some Packages in a Group Package. Recall that group packages do not specify file system content; group packages specify other packages that are part of the group. See Group Packages for more information about group packages.

To also show the summary description of each package, use the `pkg list -s` command. The `-o fmri` option of the `pkg contents` command limits the output to what is shown in the second column in the previous example.

```
$ pkg list -Has `pkg contents -Hro fmri -t depend solaris-minimal-server`
developer/debug/mdb                 Modular Debugger (MDB)
editor/vim/vim-core                 Vi IMproved (core executables)
group/system/solaris-core-platform  Oracle Solaris Core Platform
network/ping                        Ping command
network/ssh                         OpenSSH client and associated utilities
package/pkg                         Image Packaging System
release/name                        Solaris Naming Enabler
release/notices                     Oracle Solaris notices
service/network/ssh                 OpenSSH servers and SSH (Secure Shell) services
shell/bash                          GNU Bourne-Again shell (bash)
shell/ksh93                         Ksh93 - The AT&T Korn Shell
shell/tcsh                          Tenex C-shell (tcsh)
shell/zsh                           Z Shell (zsh)
system/core-os                      Core Solaris
system/library/platform             Core Architecture, (Kvm)
system/network                      Core Network Infrastructure
system/rsyslog                      reliable and extended syslogd
```

Man pages are installed by default when the software they document is installed. However, the `solaris-minimal-server` installation group does not install the `man` command. If you install the `solaris-minimal-server` group, you might want to do one of the following:

- Set the `doc.man` facet to `false` to avoid installing the man pages. See Showing and Changing Facet Values.
- Install the `text/doctools` package to install the `man` command.

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
lic_OTN install-image/solaris-auto-install
lic_OTN release/notices
```

You might need to specify the `--accept` option to install or update these packages.

Use a command such as the following example to display the license text, as described in Displaying Package Licenses. You can list multiple FMRIs.

```
$ pkg info -r --license install-image/solaris-auto-install
```

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.

Table 2-1 `pkg search` and `pkg contents` Comparison

`pkg search`	`pkg contents`
• 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.	• 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.

> **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`(7) man page.

index
Usually selects an attribute of *action* as listed in the Actions section of the `pkg`(7) man page. See below for more information.

token
The value of *index* or a pattern that might match the value of *index*.

Missing fields are implicitly wildcarded. The *token* string can be explicitly wildcarded. Names of actions and indexes cannot be explicitly wildcarded.

The *index* is not necessarily the same as the name of an action attribute. For example, the *index* for the `user` action is `name`, while the name of that attribute is `username`.

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

The following example shows one way to find all possible indexes for an action:

```
$ pkg search :user::
```

This command shows that the only searchable attribute of a `user` action is `name`.

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 *index* can also be a special pseudo attribute name such as `action.key` (the key attribute of the action) or `pkg.name`. See the `pkg`(1) man page for a complete list.

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 `pci14e*` could be specified for *token*.

```
driver alias=pci14e4,1000 alias=pci14e4,1200 alias=pci14e4,1203
```

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.

```
$ pkg search -Hpo pkg.name driver:alias:pci14e\*
driver/network/ethernet/bge
driver/network/ethernet/bnx
driver/network/ethernet/bnxe
```

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 shows a `set` action for a driver package:

```
set name=pkg.summary value="QLogic 57xxx 10/20GbE NIC Driver"
```

The following example specifies `*driver*` for package name, `pkg.summary` for *index*, and `qlogic` for *token*. The `set` action is implied because the `pkg.summary` attribute only applies to a `set` action. The `search.match` and `pkg.name` column specifiers are pseudo attributes. See the `pkg`(1) man page.

```
$ pkg search -o search.match,pkg.name \*driver\*::pkg.summary:qlogic
SEARCH.MATCH                                          PKG.NAME
QLogic 570x/571x Gigabit Ethernet Driver              driver/network/ethernet/bnx
QLogic 57xxx 10/20GbE NIC Driver                      driver/network/ethernet/bnxe
QLogic ISP Fibre Channel device storage and NIC driver driver/fc/qlc
QLogic P3+ 10GbE NIC Driver                           driver/network/ethernet/qlcnic
```

The following command gives the same result as the preceding command:

```
$ pkg search -o search.match,pkg.name pkg.summary:'qlogic * driver'
```

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`(7) man page.

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

Identifying Which Package Delivers a Specified File

In the following example, the search token is the full path of a file system object:

```
$ pkg search /lib/libinetutil.so.1
INDEX      ACTION VALUE                PACKAGE
path       file   lib/libinetutil.so.1 pkg:/system/library@11.4-11.4.0.0.1.10.1
```

The `pkg search` command searches package actions. The preceding command found the search token in the `path` attribute of a `file` action and shows that this file is delivered by the `system/library` package.

The following example specifies that the desired output is only the name of the package. Specifying the `-l` option can quicken the search because only installed packages are searched:

```
$ pkg search -Hlo pkg.name '*libinetutil*'
system/library
```

If you do not specify the output you want, then using just the base name of the file as the search token, or using wild cards as shown above, shows that this file is delivered to six different directories and is the target of three links. Another way to show each package name only one time, instead of showing many lines of output for one package, is to use the `-p` option or enclose the search token in angle brackets. Depending on your shell, you might need to escape the angle brackets. The following commands have the same output:

```
$ pkg search -p '*libinetutil*'
$ pkg search \<'*libinetutil*'\>
PACKAGE                                   PUBLISHER
pkg:/system/library@11.4-11.4.0.0.1.10.1 solaris
```

The `-p` option of the `pkg verify` command also tells you which package delivers a named file. In this case, you must specify the full path to the file relative to `/`:

```
$ pkg verify -vp lib/libinetutil.so.1
PACKAGE                                                                 STATUS
pkg://solaris/system/library                                               OK
```

If you cannot find the package that delivers a file on your system, perhaps that file is not packaged. See the `-p` and `--unpackaged` options of the `pkg verify` command in Identifying Unpackaged File System Content.

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/proxy/privoxy         svc:/network/http
web/proxy/squid           svc:/network/http
web/server/lighttpd-14    svc:/network/http
web/java-servlet/tomcat-8 svc:/network/http
web/server/apache-24      svc:/network/http
web/server/apache-24      svc:/network/http:apache24
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-8 svc:/network/http:tomcat8
```

In this case, each attribute has two values: the service name with and without the service 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:apache24 value=svc:/network/
http
```

The following example shows this same information with each package listed only once. The `-p` option cannot be used when requesting action-level output such as `search.match`. 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-24      svc:/network/http:apache24
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-8 svc:/network/http:tomcat8
```

Identifying Which Package Delivers a Specified User

IPS packages deliver users for daemons or other software to use.

The following command shows users that are delivered by installed packages:

```
$ pkg search -lo action.key user::
```

The following command shows which package delivers a particular user definition:

```
$ pkg search -o action.key,pkg.name user::openldap
ACTION.KEY PKG.NAME
openldap   system/network/ldap/openldap
```

The following command shows an example of using the OR keyword:

```
$ pkg search -o action.key,pkg.name user::openldap OR user::sshd
ACTION.KEY PKG.NAME
openldap   system/network/ldap/openldap
sshd       service/network/ssh
```

Identifying Which Packages Deliver a Specified Fix

The following command shows which package delivers a particular bug fix:

```
$ pkg search -Ho pkg.shortfmri bugid
```

All packages that were modified to fix the bug are listed.

See Critical Patch Update Packages to learn how to identify which SRU delivered a particular CVE fix.

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.name info.classification:'source code management'
pkg:/developer/versioning/sccs
pkg:/developer/versioning/git
pkg:/developer/versioning/mercurial-27
```

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.

```
$ pkg info mercurial-27
             Name: developer/versioning/mercurial-27
          Summary: The Mercurial Source Control Management System
      Description: A fast, lightweight source control management system designed
                   for efficient handling of very large distributed projects.
         Category: Development/Source Code Management
            State: Installed
        Publisher: solaris
          Version: 4.1.3
           Branch: 11.4.0.0.1.10.0
   Packaging Date: Mon Jul 02 16:32:13 2018
Last Install Time: Mon Aug 06 15:22:47 2018
             Size: 9.90 MB
             FMRI: pkg://solaris/developer/versioning/
mercurial-27@4.1.3-11.4.0.0.1.10.0:20180702T163213Z
      Project URL: http://mercurial-scm.org/
       Source URL: https://www.mercurial-scm.org/release/mercurial-4.1.3.tar.gz
```

See Classification Values in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* 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 -a '*versioning*'
$ pkg list '*storage*'
$ pkg list -n '*database*'
```

Showing Dependent Packages

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

The following command shows that many packages have a `require` dependency on the `system/network/ldap/openldap` package:

```
$ pkg search -Hlo pkg.name require:system/network/ldap/openldap
```

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

```
$ pkg contents -t depend -a fmri='*openldap*' -o pkg.name,action.raw openscap
PKG.NAME                    ACTION.RAW
security/compliance/openscap depend fmri=pkg:/system/network/ldap/
openldap@2.4.45-11.4.0.0.1.9.0 type=require
```

The following example lists packages that have an `exclude` dependency on `pkg:/driver/graphics/nvidiaR340`:

```
$ pkg contents -rt depend -a type=exclude nvidiaR340
TYPE    FMRI
exclude pkg:/driver/graphics/nvidia
exclude pkg:/driver/graphics/nvidiaR304
$ pkg search -o pkg.name,fmri 'depend:exclude:*nvidiaR340*'
PKG.NAME                  FMRI
driver/graphics/nvidia    pkg:/driver/graphics/nvidiaR340
driver/graphics/nvidiaR304 pkg:/driver/graphics/nvidiaR340
```

Listing All Packages in a 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 the specified group package.

> **Note:**
>
> This package list does not include every package that is installed when you install the group package. Dependencies of the packages in this list will also be installed, and dependencies of those dependencies.

```
$ pkg search -o type,fmri \*/solaris-minimal-server:depend::
TYPE    FMRI
require developer/debug/mdb
require editor/vim/vim-core
require group/system/solaris-core-platform
group   network/ping
group   network/ssh
require package/pkg
require release/name
require release/notices
group   service/network/ssh
require shell/bash
require shell/ksh93
group   shell/tcsh
group   shell/zsh
require system/core-os
require system/library/platform
group   system/network
group   system/rsyslog
```

The `depend` query field matches `depend` actions in the package. The `-o` option displays the values of the `type` and `fmri` attributes of the `depend` action. Recall that group packages do not specify file system content; group packages specify other packages that are part of the group. See Group Packages for more information about group packages.

The `pkg search` command is returning the value of an attribute of an action in a specified package. In this example, that attribute value is a package name. 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 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.

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 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 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 discusses aspects of package operations that are unique to Oracle Solaris Zones.

Installing, updating, and uninstalling packages require increased privileges. See Installation Privileges 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 Configuring Installed Images have an `-n` option that enables you to see what the command will do without making any changes to the image.

> **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.

Previewing Package Installation

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

```
$ pkg install -nv oracle-rdbms-server-18c-preinstall
           Packages to install:         4
            Services to change:         1
     Estimated space available: 852.48 GB
Estimated space to be consumed: 456.23 MB
       Create boot environment:        No
Create backup boot environment:        No
          Rebuild boot archive:        No

Changed packages:
solaris
  group/prerequisite/oracle/oracle-rdbms-server-18c-preinstall
    None -> 11.4-11.4.0.0.1.10.0
  system/kernel/oracka
    None -> 11.4-11.4.0.0.1.10.1
  x11/diagnostic/x11-info-clients
    None -> 7.7-11.4.0.0.1.10.0
  x11/library/libdmx
    None -> 1.1.3-11.4.0.0.1.10.0

Services:
  refresh_fmri:
    svc:/network/socket-config: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 group package listed on the command line would be installed, and three other packages 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 `oracle-rdbms-server-18c-preinstall` package has more than three dependencies:

```
$ pkg contents -rt depend oracle-rdbms-server-18c-preinstall
TYPE    FMRI
group   system/header
group   system/kernel/oracka
group   system/picl
group   x11/diagnostic/x11-info-clients
group   x11/library/libxi
group   x11/library/libxtst
group   x11/session/xauth
require compress/unzip
require developer/assembler
require developer/build/make
require system/dtrace
require system/library/openmp
```

The following command shows that ten of the twelve dependencies are already installed in this image:

```
$ pkg list `pkg contents -Hro fmri -t depend oracle-rdbms-server-18c-preinstall`
NAME (PUBLISHER)                                VERSION                    IFO
compress/unzip                                  6.0.3.23-11.4.0.0.1.10.0   i--
developer/assembler                             11.4-11.4.0.0.1.4.0        i--
developer/build/make                            11.4-11.4.0.0.1.4.0        i--
system/dtrace                                   11.4-11.4.0.0.1.10.1       i--
system/header                                   11.4-11.4.0.0.1.10.1       i--
```

```
system/library/openmp                                 11.4-11.4.0.0.1.4.0       i--
system/picl                                           11.4-11.4.0.0.1.10.1      i--
x11/library/libxi                                     1.7.9-11.4.0.0.1.10.0     i--
x11/library/libxtst                                   1.2.3-11.4.0.0.1.10.0     i--
x11/session/xauth                                     1.0.10-11.4.0.0.1.10.0    i--

pkg list: no packages matching the following patterns are installed:
  system/kernel/oracka
  x11/diagnostic/x11-info-clients
```

In addition to the two direct dependencies of `oracle-rdbms-server-18c-preinstall` that must be installed, the `x11/library/libdmx` package is installed because it is a required dependency of the `x11-info-clients` package, as shown by the following command:

```
$ pkg search -o type,pkg.name :depend:group:x11/library/libdmx OR :depend:require:x11/
library/libdmx
TYPE    PKG.NAME
require x11/diagnostic/x11-info-clients
require x11/server/xdmx
require developer/opensolaris/userland
```

Previewing Facet Change

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.

Table 3-1 `pkg install` and `pkg update` Comparison

`pkg install`	`pkg update`
• 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.	• 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`(7) man page to understand how files with these attributes are handled during installation and update.

After installing or updating packages, verify installed packages. See Verifying Packages and Fixing Verification Errors.

For information about how a file action with certain attributes is treated when the package that delivers the file is installed or updated, see File Actions in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* or the `pkg`(7) man page.

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 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.

> **Note:**
>
> Options to create a new or backup BE are ignored if the BE on which you are operating is not the currently active BE. When the BE on which you are operating is not the currently active BE, changes are made directly to that image. See the description of the `-R` option and the `PKG_IMAGE` environment variable in the `pkg`(1) man page.

`--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` command as described in the `beadm`(8) man page to show and change the active BE.

`--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 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 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, 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. 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 info` command as shown in Displaying Package Licenses. To display a list of licenses that must be accepted, use the `pkg contents` command as shown in Displaying License Requirements.

`--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 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 for a description of a package FMRI. See Configuring Publishers 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 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 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 `/var/share/pkgrepos/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.

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 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 Package Installation showed that a new BE would not be created by default when you install the `oracle-rdbms-server-18c-preinstall` package. In the following partial output, a new BE is created because the `--be-name` option is specified:

```
$ pkg install --be-name rdbms oracle-rdbms-server-18c-preinstall
           Packages to install:   4
       Create boot environment: Yes
Create backup boot environment:  No
```

The number of packages to install is still only four because the new BE starts as a copy of the current BE.

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

```
A clone of 11.4.0 exists and has been updated and activated.
On the next boot the Boot Environment be://rpool/rdbms will be
mounted on '/'.  Reboot when ready to switch to this updated BE.
```

The `pkg list` command reports that the `oracle-rdbms-server-18c-preinstall` package is not installed because the `oracle-rdbms-server-18c-preinstall` package is not installed in the current BE. The `oracle-rdbms-server-18c-preinstall` package is installed in the new `rdbms` BE.

```
$ pkg list oracle-rdbms-server-18c-preinstall
pkg list: no packages matching the following patterns are installed:
  oracle-rdbms-server-18c-preinstall
```

Use the `beadm list` command to check that the system has a new active BE named `rdbms`. 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 Name          Flags Mountpoint Space   Policy Created
---------------- ----- ---------- ------- ------ ----------------
11.4.0           N     /          221.09M static 2018-07-24 15:18
rdbms            R     -          18.93G  static 2018-08-06 12:37
```

Check that the `oracle-rdbms-server-18c-preinstall` 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 `oracle-rdbms-server-18c-preinstall` package is installed.

```
$ beadm mount rdbms /mnt
$ beadm list
BE Name          Flags Mountpoint Space   Policy Created
---------------- ----- ---------- ------- ------ ----------------
11.4.0           N     /          221.09M static 2018-07-24 15:18
rdbms            R     /mnt       18.93G  static 2018-08-06 12:37
$ pkg -R /mnt list oracle-rdbms-server-18c-preinstall
NAME (PUBLISHER)
VERSION              IFO
group/prerequisite/oracle/oracle-rdbms-server-18c-preinstall
11.4-11.4.0.0.1.10.0 i--
```

Remember to unmount the `rdbms` BE.

```
$ beadm unmount rdbms
```

If you continue to work in the current BE and make changes to the image configuration, you will see reminders such as the following because the current BE is not the activated BE:

```
WARNING: The boot environment being modified is not the active one.  Changes
made in the active BE will not be reflected on the next boot.
```

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 for information about the avoid list.

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

```
$ 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 for a description of a package FMRI. See Configuring Publishers 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`(7) man page.

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

To be sure which versions of which packages will be installed and which files preserved, preview the operation as described in Previewing Package Installation.

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 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 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`(7) man page.

See Installing a New Package for information about using the `-g` option.

Dependencies might prevent you from downgrading a package.

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.

> **Note:**
>
> Security best practice recommends that you periodically run `pkg verify -v` to help ensure that packaged file system objects have not been changed insecurely.

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.

Table 3-2 `pkg fix` and `pkg revert` Comparison

`pkg fix`	`pkg revert`
• 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.	• 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 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. If the fix affects files that cannot be modified in the live image, the fix will be done in a new BE. You can specify `-nv` options to see what changes will be made, and you can specify BE options as described in Boot Environment Options.

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
```

To evaluate `pkg verify` output programmatically, specify the `--parsable 0` option. Do not use the `-v` option if you use the `--parsable` option.

Verifying File System Content

In addition to verifying installed packages, the `pkg verify` command can verify installed directories, files, and links.

The following example uses the `-p` option to show the same information that was shown by doing a full verify of all installed packages in the previous section:

```
$ pkg verify -p var/cache/pkg/depot
PACKAGE                                                                 STATUS
pkg://nightly/package/pkg/depot                                          ERROR
        dir: var/cache/pkg/depot
                ERROR: Group: 'pkg5srv (97)' should be 'bin (2)'
```

In the following example, because the file verification status is OK, no output is shown unless the `-v` option is added:

```
$ pkg verify -vp etc/sudoers
PACKAGE                                                                 STATUS
pkg://nightly/security/sudo                                                 OK
        file: etc/sudoers
                editable file has been changed
```

Identifying Unpackaged File System Content

The message in the following example indicates that the file is not delivered by any package:

```
$ pkg verify -p etc/resolv.conf
PACKAGE                                                                 STATUS
etc/resolv.conf is not found in the image
```

The file `/etc/resolv.conf` exists on the system but is not packaged. The file contains the following comments, indicating the file is generated from SMF data:

```
# _AUTOGENERATED_FROM_SMF_V1_
#
# WARNING: THIS FILE GENERATED FROM SMF DATA.
#   DO NOT EDIT THIS FILE.  EDITS WILL BE LOST.
# See resolv.conf(5) for details.
```

To include file system content that is not delivered by any package in the report of installed package verification, use the `--unpackaged` option:

```
$ pkg verify -v --unpackaged
```

To report only file system content that is not delivered by any package, use the `--unpackaged-only` option:

```
$ pkg verify --unpackaged-only
```

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 11.4* and Directory Actions in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* for more information.

The following example shows directories that are tagged with the `system:sysconfig-profile` tag name. Unpackaged files will be removed from these directories when you use the `--remove-profiles` option with the `sysconfig unconfigure` command, as described in the `sysconfig`(8) man page.

```
$ pkg contents -H -a revert-tag='system:sysconfig-profile*' '*'
etc/svc/profile/enterprise
```

```
etc/svc/profile/incoming
etc/svc/profile/node
etc/svc/profile/site
etc/svc/profile/sysconfig
etc/svc/profile/system
```

The following command shows files that are tagged with the `system:dev-init` tag name. These files are reverted to their packaged state during recovery archive creation because these files contain configuration that is specific to that system and should not be included in a recovery archive. See the `archiveadm`(8) man page for more information.

```
$ pkg contents -Ha revert-tag='system:dev-init*' '*'
```

The following files are reverted to their packaged state during clone archive creation. In addition to the instance-specific information described in the previous example, information such as log file content and some configuration files also is reverted in a clone archive.

```
$ pkg contents -H -a revert-tag='system:dev-init*' -a revert-tag='system:clone*' '*'
```

The following command shows a preview of an operation that would revert all files that have the `system:dev-init` tag name. The files to be reverted would be listed by the `-v` option but are not shown in this example. Notice that the boot archive would be rebuilt. Using the `--be-name` option to create a new boot environment with a meaningful name is a good practice.

```
$ pkg revert -nv --tagged system:dev-init
               Packages to fix:         5
     Estimated space available: 852.20 GB
Estimated space to be consumed: 470.42 MB
       Create boot environment:       Yes
     Activate boot environment:       Yes
Create backup boot environment:        No
          Rebuild boot archive:       Yes
```

Uninstalling Packages

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

When a package is uninstalled, the command output is very similar to the output when a package is installed, with “Packages to remove” instead of “Packages to install”, for example.

The `--reject` option of the `pkg install` command can also remove installed packages, as described in Rejecting a Package.

Uninstalling a package can cause a mediated link to be removed as described in Setting a Preferred Path that is Not Available: Consequences and Recovery. Use the `-nv` or `-nvv` options to check whether a mediated link will be removed.

For information about how a file action with certain attributes is treated when the package that delivers the file is uninstalled, see File Actions in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* or the `pkg`(7) man page.

Example 3-1 Undoing a Package Installation

Uninstalling a package does not uninstall dependencies of that package. A user might want to continue to use the content delivered by a dependency package even if the only package that requires the dependency is uninstalled.

If you want to undo a package installation operation so that you uninstall all packages that the installation operation installed, use the `pkg history` command to determine exactly what was installed. See Viewing Operation History for more information.

In the example shown in Previewing Package Installation, four packages were installed. However, the following command shows that only the package named on the command line is removed by `uninstall`:

```
$ pkg uninstall -v oracle-rdbms-server-18c-preinstall
            Packages to remove:         1
     Estimated space available: 852.20 GB
Estimated space to be consumed: 455.25 MB
          Rebuild boot archive:        No

Changed packages:
solaris
  group/prerequisite/oracle/oracle-rdbms-server-18c-preinstall
    11.4-11.4.0.0.1.10.0 -> None
...
```

The `pkg list` command shows that the other three packages that were installed are still installed:

```
$ pkg -R /mnt list oracle-rdbms-server-18c-preinstall system/kernel/oracka \
x11/diagnostic/x11-info-clients x11/library/libdmx
NAME (PUBLISHER)                                VERSION                    IFO
system/kernel/oracka                            11.4-11.4.0.0.1.10.1       i--
x11/diagnostic/x11-info-clients                 7.7-11.4.0.0.1.10.0        i--
x11/library/libdmx                              1.1.3-11.4.0.0.1.10.0      i--

pkg list: no packages matching the following patterns are installed:
  oracle-rdbms-server-18c-preinstall
```

To undo this installation if you do not have the original `pkg install` output, find the list of packages that were installed in the history. Use a short listing to find the installation command that you want, and then use the `-l` option to list exactly what was installed at that time (`-t`), as shown in Viewing Operation History. The list is at the end of the output under the heading End State.

```
$ pkg history -n 1
START                   OPERATION               CLIENT              OUTCOME
2018-08-06T12:36:43     install                 pkg                 Succeeded
$ pkg history -lt 2018-08-06T12:36:43
         Operation: install
...
           Command: /usr/bin/pkg install --be-name rdbms oracle-rdbms-server-18c-
preinstall
...
         End State:
None -> pkg://solaris/x11/diagnostic/x11-info-
clients@7.7,5.11-11.4.0.0.1.10.0:20180702T172622Z
None -> pkg://solaris/group/prerequisite/oracle/oracle-rdbms-server-18c-
preinstall@11.4,5.11-11.4.0.0.1.10.0:20180702T172914Z
None -> pkg://solaris/system/kernel/
oracka@11.4,5.11-11.4.0.0.1.10.1:20180702T144645Z
None -> pkg://solaris/x11/library/
libdmx@1.1.3,5.11-11.4.0.0.1.10.0:20180702T172638Z
```

Example 3-2 Specifying Multiple Packages to Uninstall

If you specify multiple packages to the `pkg uninstall` command, or if you specify options such as `-r` to perform this operation in non-global zones, you might be attempting to uninstall a package that is not installed in that image.

If you attempt to uninstall a package that is not currently installed, the `pkg uninstall` operation exits without uninstalling any packages.

When a package to be uninstalled does not exist in that image, a message about no matching packages is displayed for that package name. If additional packages to be uninstalled do exist in the image, no message is displayed to tell you that those packages were not uninstalled.

To uninstall all specified packages that are installed and skip the operation for any specified packages that are not installed, use the `--ignore-missing` option. The `--ignore-missing` option ignores packages that are not installed and prevents `pkg uninstall` from failing if some packages to uninstall are not currently installed.

In the following command, if one zone does not have the `git` package installed, no packages are uninstalled, even if some zones do have `git` installed:

```
$ pkg uninstall -r git
```

In the following command, the `git` package is uninstalled from any zone where it is currently installed:

```
$ pkg uninstall -r --ignore-missing git
```

Example 3-3 Uninstalling a Package that is Required by Another Package

Attempting to uninstall a package that is required by another installed package fails.

For example, if you try to uninstall the `libdmx` package while the `x11-info-clients` package is still installed, you receive the following error message:

```
$ pkg uninstall libdmx
pkg uninstall: Unable to remove 'x11/library/libdmx@1.1.3-11.4.0.0.1.10.0' due to the
following packages that depend on it:
  x11/diagnostic/x11-info-clients@7.7-11.4.0.0.1.10.0
```

To uninstall `libdmx`, you must uninstall both `libdmx` and `x11-info-clients`. Recall that simply uninstalling `x11-info-clients` does not uninstall the dependent package `libdmx`.

```
$ pkg uninstall x11-info-clients libdmx
```

Example 3-4 Uninstalling a Package that is a `group` Dependency

When you uninstall a package that is a `group` dependency, that package is placed on the avoid list. See Avoiding Installing Some Packages in a Group Package for information about the avoid list.

```
$ pkg avoid
$ pkg uninstall mysql-56
...
$ pkg list mysql-56
pkg list: no packages matching the following patterns are installed:
  mysql-56
$ pkg avoid
    database/mysql-56 (group dependency of 'group/feature/amp')
```

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.

> **Note:**
>
> A version of the `pkg:/entire` constraint package must be included in the `exact-install` operands. An image with no `pkg:/entire` installed is not supported.

The `pkg exact-install` command ignores restrictions to not install packages that are on the avoid list. If a package that is on the avoid list in the current BE is named as an operand on the `exact-install` command or is in the dependency chain, that package will be installed in the new BE; the new BE will have an empty avoid list. See Avoiding Installing Some Packages in a Group Package for information about the avoid list.

The `pkg exact-install` command ignores restrictions to not update packages that are on the frozen list. If a package that is frozen in the current BE is named as an operand on the `exact-install` command or is in the dependency chain, that package will be installed in the new BE at the version specified by the `exact-install` command; the new BE will not have any frozen packages. See Locking Packages to a Specified Version 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. Note that the `-r` option is not available for `exact-install`.

See Installing a New Package 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. 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` constraint package in the list of packages to install at a version equal to or greater than the version of the currently installed `pkg:/entire` package.

- 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.
- Always keep the current working BE as a backup until the new BE is thoroughly tested.

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@11.4-11.4.0.0.1.10.0:20180702T173343Z
$ pkg exact-install --be-name 11.4.0min entire@11.4-11.4.0 solaris-minimal-server
```

> **Note:**
>
> Specifying a version of the `pkg:/entire` constraint package that is older than the version of the currently installed `pkg:/entire` package is not guaranteed to work. Some changes, such as a change in `zpool` version, could cause a downgraded `exact-install` BE to fail to boot or to behave in unexpected ways. Smaller version deltas are more likely to be successful than larger version deltas.

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 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 and Updating Multiple Non-Global Zones Concurrently.

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.

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

> **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 and Non-Global Zone Cannot Be Installed 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 be accessible from the global zone.

The system repository can proxy `http`, `https`, `file`, and `.p5p` archive repositories. Only version 4 file system repositories are supported, which is the default format for the `pkgrepo create` command. See the `pkgrepo`(1) man page for more information about repository versions.

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`(8) 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:///var/share/pkgrepos/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>
solaris   (syspub)    origin   online F <system-repository>
isvpub    (syspub)    origin   online T <system-repository>
```

Notice that the disabled repository is not available in the non-global zone.

Use the `-F` option to display the URI and proxy values for `system-repository` locations:

```
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
solaris    true    true    true     origin  online  http://localhost:1008/solaris/
35024e7d1859bedee9af156d22a591c433adc0ee/ -
isvpub     true    true    true     origin  online  http://pkg.example2.com/                http://
localhost:1008
```

Notice that the `file://` repository in the global zone has been assigned an `http://` location in the non-global zone.

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. 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 Configuring Proxies to the Package Repository for Non-Global 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:///var/share/pkgrepos/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 Options That Operate on Non-Global Zones 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 s12.0
 Startup: Linked image publisher check ... Done
 Startup: Refreshing catalog 'solaris' ... Done
 Startup: Refreshing catalog 'isvpub' ... Done
 Startup: Checking that pkg(7) 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.
```

4

Updating or Upgrading an Oracle Solaris Image

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. This chapter discusses the following topics:

- Best practices
- Controlling how far to update
  - Specify a version number
  - Freeze the image at a specified version number
  - Use an Oracle Solaris constraint package
  - Use a custom constraint package
- Applying support updates
  - Support Repository Updates (SRUs)
  - Critical Patch Updates (CPUs)
- Applying platform firmware updates
- Applying Interim Diagnostic or Relief (IDR) updates

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.

If you do not want to update to the newest version allowed, see the options discussed in Updating to a Version Older Than the Newest Version Allowed.

- 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 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 Troubleshooting Package Installation and Update.

Upgrading your system means updating a bootable image. A system can have multiple bootable images, as noted in Images and Boot Environments.

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

Updating requires increased privileges. See Installation Privileges for more information.

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

How to Update an Oracle Solaris 11 System

Upgrading a system is just one step: `pkg update`.

As a prerequisite, make sure the system can access all required software packages. As a followup, you might in some cases need to do more than one `pkg update`.

1. **Make sure the system has access to the necessary software packages.**

   See Check Available Versions.

2. **Perform a test update.**

   See Preview the Update Operation.

3. **Perform the actual update.**

   Use the `--be-name` option to give the new BE a meaningful name.

   Specify the result you want: Specify part of the version of the packages that you are updating.

   ```
   $ pkg update --be-name 11.4.0 entire@11.4-11.4.0
   ```

   In some cases, image upgrade requires multiple steps because a subset of software must be updated before other software can be updated. If you are not able to upgrade to your desired end state in one step, update as far as you can, reboot to the new BE, and update again. Release notes provide information about whether a multi-step upgrade is required.

Image Update Best Practices

- Before you update, perform the following steps:
  - Review the license. Displaying License Requirements describes how to list packages that require you to accept their license. You might need to specify the `--accept` option to install or update these packages. Displaying Package Licenses describes how to show the license text.
  - Read the release notes. Release notes for an Oracle Solaris release are provided on `docs.oracle.com`. Release notes for an SRU are provided on `support.oracle.com`.
  - Check the package versions that are available from your configured publisher origin. See Check Available Versions. You might need to run the `pkg refresh` command on the publisher or run the `pkgrepo refresh` command

on a particular location of the publisher. If you create your own repository, do not create a partial repository; create a complete repository as described in *Creating Package Repositories in Oracle Solaris 11.4*.

  - Use the `pkg update` command with the `-nv` options to display the list of packages that will be updated without actually performing the update. See Preview the Update Operation.

- 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, as described in Specify a New Boot Environment.
- If the update fails, check Troubleshooting Package Installation and Update.
- After you update, verify the installed packages.

  ```
  $ beadm mount name-of-new-BE /mnt
  ```

  Make sure you got what you wanted in the update:

  ```
  $ pkg -R /mnt list -v entire
  ```

  Verify the packages:

  ```
  $ pkg -R /mnt verify -v
  ```

  If errors are reported, use the `pkg fix` command and then verify that the errors are fixed.

  ```
  $ pkg -R /mnt fix -v
  $ pkg -R /mnt verify -v
  ```

  Remember to unmount the BE.

  ```
  $ beadm unmount name-of-new-BE
  ```

  See also Verifying Packages and Fixing Verification Errors.
- Use the `-N` option with the `pkg history` command to see whether release notes are reported in any updated or installed package.
- Keep your systems updated with all support updates as described in Applying Support Updates. The “Oracle Solaris Binary and Source Guarantee Program” (Doc ID 1391762.1) ensures that updating across release boundaries is low risk.

Check Available Versions

The cause of most update errors is an incomplete package repository. See Best Practices for Creating and Using Local IPS Package Repositories in *Creating Package Repositories in Oracle Solaris 11.4*. The system to be updated must have access to at least the package versions described in Minimal Required Repository in *Creating Package Repositories in Oracle Solaris 11.4*.

The system must have access to a package repository that provides the packages that are currently installed on the system. For example, if you are updating from Oracle Solaris 11.3 to Oracle Solaris 11.4, the `solaris` publisher must be configured with access to both the installed Oracle Solaris 11.3 packages and the desired Oracle Solaris 11.4 packages. If packages are installed from another publisher, such as `ha-cluster` or `solarisstudio`, those publishers also must be configured with access to currently installed packages as well as desired newer packages.

If you want to update your operating system release, check the available versions of the `pkg:/entire` constraint package.

Use the following command to determine what version is installed on the system:

```
$ pkg list entire
```

Use the following command to determine what versions are available from configured publishers:

```
$ pkg list -af entire
```

The output from this command needs to show the currently installed release, the release to which you want to update, and required intermediate releases as described in Minimal Required Repository in *Creating Package Repositories in Oracle Solaris 11.4*.

To ensure that the currently installed version of `pkg:/entire` is still available from configured publishers, check for that specific version in specific enabled publisher locations, as shown in the following example:

```
$ pkgrepo -s /var/share/pkgrepos/solaris list entire@0.5.11-0.175.3.22.0.3.0
```

The following command shows that Oracle Solaris 11.3 SRU 22 is installed, and Oracle Solaris 11.3 SRUs 25 and 28 are available from the currently configured `solaris` publisher. For information about fields in the FMRI, see Fault Management Resource Identifiers.

```
$ pkg list -af entire@0.5.11-0.175.3
NAME (PUBLISHER)                  VERSION                    IFO
entire                            0.5.11-0.175.3.28.0.4.0    ---
entire                            0.5.11-0.175.3.25.0.3.0    ---
entire                            0.5.11-0.175.3.22.0.3.0    i--
```

If none of these versions is what you want, or if some releases required to accomplish the update are missing, then set your `solaris` publisher origin to a different package repository location or add a location. Use the following command to check the publishers that are configured on the system. The `-n` option says only show enabled publishers:

```
$ pkg publisher -n
```

A single publisher, such as `solaris`, can have multiple repository locations. If a publisher has more than one repository location configured, use the `-F` option to check whether all locations are enabled, as shown in Enabling and Disabling Publisher Origins. If some of the package versions required to complete the update are in different package repositories, add those package repository locations to the publisher by using additional `-g` options with the `pkg set-publisher` command as shown in Adding and Changing Publisher Origins.

If any publisher repository locations are local to your site, you might need to add more content to those repositories. See *Creating Package Repositories in Oracle Solaris 11.4* for instructions.

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.

If one of the publisher repository locations is the Oracle Solaris support repository or other secured repository, make sure the key and certificate are valid.

Make sure the system's access to a repository is not blocked by a firewall or proxy.

For more information, see Configuring Publishers.

Available Versions of Open Source Software

You might find that an Oracle Solaris package repository contains versions of open source software that are newer than the newest version of `pkg:/entire` in that repository. You might be able to install these newer versions if you unlock the version-lock facets as described in Relaxing Version Constraints Specified by Constraint Packages. You might not be able to uninstall older versions because other software might have a dependency on the older version. In many cases, multiple versions of software are allowed to be installed on the system at the same time. In those cases, the default version is determined by the mediator setting, as described in Specifying a Default Application Implementation. If you require a particular version, specify the full path, not the mediated link, in your script or other software.

For specific versions of FOSS that are available in this release of Oracle Solaris, see Oracle Solaris 11.4 Bundled Software Updates in *Freeware Available in Oracle Solaris 11.4*.

Preview the Update Operation

Use the following options with the `pkg update` command:

- Use the preview option (`-n`) to see the result of the operation without actually doing the operation. Review this output before you perform an update without the `-n` option.
- Use the verbose option (`-v`) to see what packages will be updated, removed, and installed and at what versions; what mediators will be changed; what services will be restarted; what editable files will be changed; whether a new BE will be created. If the update does not succeed, the verbose option shows more information to help you diagnose the problem. To get even more information, use `-vv`.

Specify the result you want. If the update does not succeed, you will get more information to help you diagnose the problem if you specify more of the version of the packages that you are updating.

The following example command shows using these options:

```
$ pkg update -nv entire@11.4-11.4.0.0.1.12.0
```

If the update operation does not succeed, consider some of the following steps, depending on the error messages that you received. See also Troubleshooting Package Installation and Update.

- Use the `pkg freeze` command to check whether the system has any packages frozen that could be preventing update.
- Use the `pkg facet` command to check whether the version constraint is unlocked on any package. You might need to re-lock that constraint to accomplish the update you want. Note that the act of setting the version lock can result in a system update.
- Check whether an IDR is installed that is constraining the update.
- Run `pkgrepo verify` on the publisher repository locations. Perhaps required dependencies are missing or perhaps permissions are set incorrectly.

- Run `pkg verify` on the system that you are trying to update. Perhaps packaged content was changed on the system in an unsupported way. For example, if someone changed a link that was delivered by an IPS package to be a directory or file, that could prevent the system update.

Software in a non-global zone that is required to be the same release as that software in the parent global zone is updated automatically when you update the global zone. Software in a non-global zone that is not required to be the same release as that software in the parent global zone must be updated separately.

Specify a New Boot Environment

If you request a new BE or if a new BE is required to accomplish the update, 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 for information about when BEs are created. You should use the `--be-name` option to give the new BE a meaningful name.

The new BE is activated so that 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. For more information about boot environment options, see Boot Environment Options.

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.

Updating to a Version Older Than the Newest Version Allowed

Sometimes you do not want to update to the newest version allowed. This section describes methods to update to a version that is older than the newest version allowed:

- Specify the version in the update command. This is the simplest method with no lasting side effects.
- Specify a version limit prior to giving the update command. This method uses `pkg freeze`, and therefore you need to remember to unfreeze later when you want to update to a newer version.

- Use an Oracle Solaris constraint package. Using a constraint package is the most scalable, controllable, and trackable method.
- Use a custom constraint package. Using a constraint package is the most scalable, controllable, and trackable method.

Specifying the Version to Install

A simple way to update to a version that is older than the newest version allowed is to 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 `pkg:/entire` constraint package to update to Oracle Solaris 11.3 SRU 13, even though a newer version would be allowed:

```
$ pkg update -nv entire@0.5.11,5.11-0.175.3.13 '*'
            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 ->
0.5.11,5.11-0.175.3.13.0.4.0:20160929T175502Z
...
```

Be sure to use the `-nv` options and check the output before performing the actual update. When you perform the actual update, use the `--be-name` option to give the new BE a meaningful name.

Some installed packages might not be dependent on any package that is constrained by the `entire` constraint package. Those packages will not be updated by updating just the `entire` package. You can add those packages by name to the same `pkg update` command or specify `*` in addition to `entire@`*`version`* .

> **Note:**
>
> Specifying a version constraint at the command line is not the best way to manage a large number of systems. The best scalable solution is to use a constraint package as described in Using an Oracle Solaris Constraint Package and Installing a Custom Constraint Package.

Specifying a Version Constraint Prior to Updating

If you want to allow updates to any Oracle Solaris 11.3 version but not allow update to Oracle Solaris 11.4, you can freeze the `pkg:/entire` constraint package as shown in the following command. Specifying `0.5.11,5.11-0.175.3` means the `entire` package can be updated to `0.5.11,5.11-0.175.3.16`, for example, but not to `11.4`.

```
$ pkg freeze -c "Keep this image at 11.3." entire@0.5.11,5.11-0.175.3
entire was frozen at 0.5.11-0.175.3
$ pkg freeze
NAME    VERSION         DATE                       COMMENT
entire  0.5.11-0.175.3  15 Nov 2016 13:30:44 PST  Keep this image at 11.3.
$ pkg list entire
NAME (PUBLISHER)  VERSION                    IFO
entire            0.5.11-0.175.3.13.0.4.0    if-
```

For more information about package freezing, see Locking Packages to a Specified Version.

To update beyond Oracle Solaris 11.3, you will need to `pkg unfreeze entire` or freeze `entire` at a newer version.

> **Note:**
>
> Freezing a package is not the best way to manage a large number of systems. The best scalable solution is to use a constraint package as described in Using an Oracle Solaris Constraint Package and Installing a Custom Constraint Package.

Using an Oracle Solaris Constraint Package

Similar to using the `pkg freeze` command as shown in the previous section, you can install a package that constrains the version of `pkg:/entire` that can be installed. See Constraint Packages for more information about constraint packages and their use in Oracle Solaris.

Oracle Solaris 11.4 provides a package that constrains `pkg:/entire` to any version of Oracle Solaris 11.4.

```
$ pkg list -s solaris-11.4
NAME (PUBLISHER)                SUMMARY
release/constraint/solaris-11.4  Constraint Package for Oracle Solaris 11.4
```

The content of this package shows that if you install this package, you will be able to update your system to any Oracle Solaris 11.4 release (`entire@11.4-11.4`), such as an Oracle Solaris 11.4 SRU, but not to any newer Oracle Solaris 11.*n* release.

```
$ pkg contents -m solaris-11.4
set name=pkg.fmri value=pkg://solaris/release/constraint/
solaris-11.4@0,5.11:20170724T163053Z
set name=pkg.summary value="Constraint Package for Oracle Solaris 11.4"
set name=variant.arch value=sparc value=i386
set name=pkg.depend.install-hold value=core-os
depend fmri=entire@11.4-11.4 type=incorporate
depend fmri=entire type=require
```

- If you install the `solaris-11.4` package and then attempt to update when you already have the newest available Oracle Solaris 11.4 release installed, you will receive a message that no updates are available.
- If you install the `solaris-11.4` package and then attempt to update to a newer Oracle Solaris 11.*n* release, you will receive a message that your system is overly

constrained. To update to a newer Oracle Solaris 11.*n* release, you must first uninstall the `solaris-11.4` constraint package.

Installing a Custom Constraint Package

Similar to using the Oracle Solaris constraint packages described in the previous section, you can create your own custom constraint package to specify the constraints you want. You might want to specify a constraint that is different from the constraints that are available in Oracle Solaris packages.

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

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

Create a Custom Constraint Package

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

Create the Custom Constraint Package Manifest

The following example shows a manifest named `upgradectrl.p5m` for a custom constraint 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="Package 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` constraint package 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 *Creating Package Repositories in Oracle Solaris 11.4*.

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                    O VERSION
site      upgradectrl               1.0,5.11:20131104T072336Z
$ pkg list -vg myrepo
FMRI                                                IFO
pkg://site/upgradectrl@1.0,5.11:20131104T072336Z    ---
```

A value in the `O` column indicates whether the package is obsolete (`o`) or renamed (`r`).

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

```
$ pkgrecv -s myrepo -d /var/share/pkgrepos/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 /var/share/pkgrepos/solaris upgradectrl
          Name: upgradectrl
       Summary: Package 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 11.4* 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 /var/share/pkgrepos/solaris site
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F https://pkg.oracle.com/solaris/support/
site                      origin   online F file:///var/share/pkgrepos/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 `pkg:/entire` constraint package 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 `entire` package, `0.175.1.10`.

```
set name=pkg.fmri value=upgradectrl@1.10
set name=pkg.summary value="Package 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 /var/share/pkgrepos/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` constraint package that is specified in the new `upgradectrl` constraint package.

```
$ 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
```

Maintain a Separate Package Repository for Oracle Solaris 11.4

One way to prevent Oracle Solaris 11.3 or earlier systems from upgrading to Oracle Solaris 11.4 is do not provide access to Oracle Solaris 11.4. Provide separate Oracle Solaris 11.3 and Oracle Solaris 11.4 IPS package repositories and configure publishers appropriately on each system.

Oracle Solaris 11.3 Systems that are Not Ready to Upgrade to Oracle Solaris 11.4

If these systems have been using an IPS package repository at an Oracle URL, create a local Oracle Solaris 11.3 package repository as described in *Creating Package Repositories in Oracle Solaris 11.4*.

- Create the repository from Oracle Solaris 11.3 repository archive files for the latest public release and all subsequent SRUs. The repository must contain all packages that are installed on any Oracle Solaris 11.3 systems that will use the repository. Do not try to create an Oracle Solaris 11.3 package repository from a subset of an Oracle Solaris 11.4 package repository.
- Maintain the repository by adding new Oracle Solaris 11.3 SRUs from SRU repository archive files. Do not maintain this repository by using the `pkg/mirror` service with an Oracle URL as the source.
- Reset the `solaris` publisher on these Oracle Solaris 11.3 systems so that this new local Oracle Solaris 11.3 repository location is the only origin.

Oracle Solaris 11.3 Systems that are Upgrading to Oracle Solaris 11.4

Systems that are using an IPS package repository at an Oracle URL will be upgraded to Oracle Solaris 11.4 if no constraint is applied such as a constraint package as described in Using an Oracle Solaris Constraint Package.

For systems that are using a local repository that is shared by systems that are not ready to upgrade to Oracle Solaris 11.4, use one of the following options:

- Change the `solaris` publisher origin to an Oracle Solaris repository URL.
- Create a new local Oracle Solaris 11.4 package repository as described in *Creating Package Repositories in Oracle Solaris 11.4*. Reset the `solaris` publisher on these systems to add an origin for this new local Oracle Solaris 11.4 repository to the existing origin for the local Oracle Solaris 11.3 repository.

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 11.3 SRU 10 to Oracle Solaris 11.3 SRU 13 and then realized you need an SRU 12 image, reboot to your SRU 10 BE and update to SRU 12 from there.

Alternatively, you could install a kernel zone or Oracle VM Server at the Oracle Solaris version you need.

Applying Support Updates

The Oracle Solaris support repository is kept updated with important fixes including security updates. See Accessing Support Updates for information about the Oracle Solaris support repository.

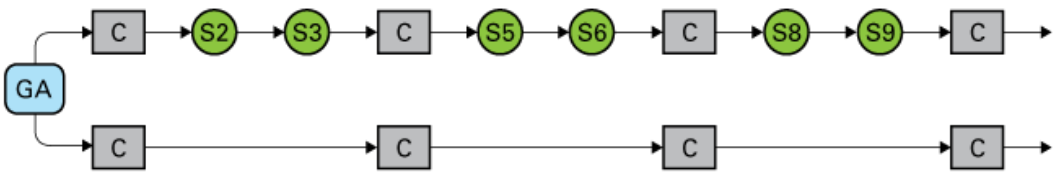
Oracle Solaris provides Support Repository Updates (SRUs) to deliver these fixes. Every third SRU is a Critical Patch Update (CPU SRU). The timing of CPU SRU releases matches the release of critical patch updates for other Oracle products.

The following figure shows two system upgrade strategies. In the figure, GA = a release such as Oracle Solaris 11.3 or Oracle Solaris 11.4, S = SRU, and C = CPU SRU.

- Updating every time a new SRU is available is the best way to keep your system up-to-date with important security fixes.
- If you do not believe you can reboot every month, update at least every quarter to the Oracle critical patch update.

When you update to a CPU SRU or any other SRU, you get all the fixes and enhancements that were delivered in all preceding SRUs. The following figure shows the monthly SRU and quarterly CPU update paths. Every third SRU is a CPU SRU. You can switch paths at any time.

Monthly SRU or Quarterly CPU System Updates



The following table describes differences between SRUs and CPU SRUs.

Table 4-1 Comparison of SRUs and CPUs

Characteristic	SRUs	CPU SRUs
Update release frequency	Monthly	Quarterly See Critical Patch Updates, Security Alerts and Bulletins for the CPU release schedule.
Update content	Any kind of fix or enhancement In addition to new fixes, each SRU provides all fixes and enhancements from preceding Oracle Solaris releases and from all preceding SRUs for the current release. For example, updating from Oracle Solaris 11.2 SRU 14 to Oracle Solaris 11.3 SRU 2 installs all fixes from Oracle Solaris 11.2 SRU 15 and Oracle Solaris 11.3 GA, SRU 1, and SRU 2.	New fixes are only critical fixes, including fixes for Common Vulnerabilities and Exposures (CVEs). Provides all fixes and enhancements from preceding Oracle Solaris releases and from all preceding SRUs for the current release, but new change introduced in this SRU is only critical fixes.
Reboot required	Yes	Yes
New series begins	Each Oracle Solaris release: for example, Oracle Solaris 11.2, Oracle Solaris 11.3, Oracle Solaris 11.4	CPU SRUs are part of the SRU series. Every third SRU is a CPU SRU.
Length of series	Same as the associated Oracle Solaris release as stated in “Solaris Operating System End Of Life Matrix” (Doc ID 1001343.1). SRUs might no longer be produced for a release once a subsequent release is available. You can update directly from an SRU for one Oracle Solaris release to an SRU for a different Oracle Solaris release. The “Oracle Solaris Binary and Source Guarantee Program” (Doc ID 1391762.1) ensures that updating across release boundaries is low risk.	CPU SRUs are part of the SRU series. Every third SRU is a CPU SRU.

Accessing Support Updates

To apply support updates, update your systems from one of the following sources:

- The Oracle Solaris support repository, which is available at `https://pkg.oracle.com/solaris/support/`. To access the support repository, use your Oracle support credentials to create SSL certificates at the `https://pkg-register.oracle.com/` Oracle Solaris package repository certificate request site.
- Your local repository that you update from one of the following sources:
  - The Oracle Solaris support repository.
  - SRU repository files downloaded from My Oracle Support.

To download repository files, search for “Oracle Solaris 11.4 Support Repository Updates (SRU) Index” on `https://support.oracle.com/`. The Readme file for each SRU includes lists of bugs fixed, packages updated, and Interim Diagnostic or Relief (IDR) updates superseded in this SRU. See Installing an IDR Custom Software Update for a description of IDR updates. The Installation Guide for the SRU contains a copy of the SRU Readme file, a separate readme file that explains how to install the SRU package repository files, a checksum file, and the script that installs the SRU repository files into your local package repository. The Repository download contains the SRU repository files.

See *Creating Package Repositories in Oracle Solaris 11.4* for information about how to create and maintain a local IPS package repository and the minimum required content for a repository.

Perform the update as described in Image Update Overview. To update to an SRU that is older than the latest released SRU, use one of the methods described in Updating to a Version Older Than the Newest Version Allowed.

While each SRU includes all fixes and enhancements that were delivered by previously released SRUs as described in Comparison of SRUs and CPUs, an SRU does not contain any other SRUs: An SRU contains only one version of `pkg:/entire`. To update systems to a particular SRU, you must have access to that SRU by using the Oracle Solaris support repository or by adding the content of the repository file for that SRU to your local repository.

For example, if you did not add SRU 28 repository content to your local repository, but you did add SRU 29 repository content, you would have all fixes that were initially delivered in any SRU for this release through SRU 29, but you would not be able to update systems to the SRU 28 level. A query would show that your local repository does not contain `entire@0.5.11-0.175.3.28`, even though it does contain `entire@0.5.11-0.175.3.29`. See Check Available Versions.

Critical Patch Update Packages

The following critical patch update package is available with each monthly SRU. Most of the content of this package is information about CVE fixes delivered through that SRU.

`pkg:/support/critical-patch-update/solaris-11-cpu@`*`YYYY.MM-version`*

Table 4-2 `solaris-11-cpu` Package Version String Components

Component	Description
YYYY	The year in which the SRU associated with this CPU package was released.
MM	The month in which the SRU associated with this CPU package was released. This value is one or two digits; leading zeros are not used.
version	An integer that is incremented when the CPU package is re-released in the same month.

The `solaris-11-cpu` package is not installed by default. If you want this package, you must explicitly install it. This package is not required in order to update to a newer SRU. Advantages to installing this package include:

- Easily list which CVEs are fixed on this system.
- Easily show which SRU is running on this system.

- Easily upgrade to a specific SRU by updating this package to that specific version. All components are moved to the specified SRU level, including any components that are unlocked from their constraint packages.
- Ensure that all packages that are needed to fix these CVEs are installed at the right version.

The following command lists all CVE fixes that are installed on this system if this system has the `solaris-11-cpu` package installed:

```
$ pkg search -Hlo value info.cve:
```

If this system does not have the `solaris-11-cpu` package installed, identify the `solaris-11-cpu` package for the SRU that is installed, and query that package remotely. For example, if this system is running Oracle Solaris 11.3 SRU 28, which was released in January 2018, the corresponding `solaris-11-cpu` package is `solaris-11-cpu@2018.1`.

```
$ pkg contents -ro value -t set -a name=info.cve solaris-11-cpu@2018.1
```

To check whether additional fixes are available, use the following command to show whether a version of the `solaris-11-cpu` package is available that is newer than the version you have installed:

```
$ pkg list -n solaris-11-cpu
```

If a newer package is available, use the following command to list the CVE fixes that are available from the newer package, and compare that list with the list of installed CVE fixes.

```
$ pkg contents -ro value -t set -a name=info.cve solaris-11-cpu@YYYY.MM
```

Use the `pkg update` command to update to the newest available SRU or to a specified SRU and install the new fixes and enhancements for that SRU.

```
$ pkg update --be-name Solaris-11.3-SRU30 solaris-11-cpu@2018.3 '*'
```

The following command shows all the versions of the `solaris-11-cpu` package that deliver the fix for the specified CVE:

```
$ pkg search -Hpo pkg.shortfmri CVE-YYYY-NNNN:
```

This output shows which version of the `solaris-11-cpu` package first delivered the fix for this CVE and which version most recently delivered this fix. Note that these packages are not necessarily listed in date order because, for example, month 10 sorts older than month 9.

For a specific CVE identifier, the following command lists all packages that were modified to fix that CVE:

```
$ pkg search -Ho value CVE-YYYY-NNNN:
```

See *Oracle Solaris 11.4 Compliance Guide* for more information about CVEs.

Platform Firmware Updates for SPARC Systems

Platform firmware updates for selected SPARC systems are available in the Oracle Solaris IPS support repository in addition to `.zip` downloads from My Oracle Support (`http://support.oracle.com`). If you have your `solaris` package publisher

configured to retrieve packages from the Oracle Solaris support repository, you can use the `pkg install` command instead of downloading and unpacking the `.zip` file.

Installing the IPS package does not automatically update the firmware of the server because that update requires manual steps by the system administrator and requires power cycling the server. Installing or updating the firmware update package delivers the files to `/var/firmware/system/` *ServerType*. To install the firmware update, follow the instructions in the included README file.

Use the following command to identify the newest available firmware update packages:

```
$ pkg list -n 'firmware/system/*'
```

Packages are named `pkg:/firmware/system/`*ServerType*. You do not need to install older updates. You can install just the newest update, and then continue to update from there.

Use the `pkg contents` and `pkg info` commands to get more information about the packages for your platform.

Example 4-1 Showing Information About SPARC T7 Firmware Updates

The `firmware/system/T7-4` package installs the `T7-4/sysfw9-7` and `T7-4/hwprogrammables1-0-7` packages:

```
$ pkg install -nv firmware/system/T7-4
           Packages to install:         3
     Estimated space available: 830.81 GB
Estimated space to be consumed: 367.54 MB
       Create boot environment:        No
Create backup boot environment:        No
          Rebuild boot archive:        No

Changed packages:
solaris
  firmware/system/T7-4
    None -> 1.1.0.4.0,5.11:20170726T000710Z
  firmware/system/T7-4/hwprogrammables1-0-7
    None -> 1.0.7,5.11:20160817T161305Z
  firmware/system/T7-4/sysfw9-7
    None -> 9.7.6.2,5.11:20170726T000706Z

Release Notes:
  Sun System Firmware component revisions included with this release:

          Sun System Firmware 9.7.6.b 2017/07/10 15:33
          ----------------------------------------
          ILOM 3.2.10.1.b Mon Jul 10 16:22:42 PDT 2017
          Hostconfig 1.8.6 2017/06/07 09:34
          POST 5.5.6 2017/06/07 09:22
          FWP 1.3.6 2017/06/07 09:15
          Hypervisor 1.17.6.a 2017/06/14 10:14
          OpenBoot 4.40.6 2017/06/07 09:00
          GM 1.8.6 2017/06/07 09:40

          Checksum of Sun_System_Firmware-9_7_6_b-SPARC_T7-4.pkg : 2715457817
          (generated by the /usr/bin/cksum command)
```

The `pkg info` command shows that the `firmware/system/T7-4/sysfw9-7` package provides Patch p26407491_976b:

```
$ pkg info -r firmware/system/T7-4/sysfw9-7
             Name: firmware/system/T7-4/sysfw9-7
          Summary: T7-4 firmware
      Description: T7-4 firmware: MOS Patch p26407491_976b
            State: Not installed
        Publisher: solaris
          Version: 9.7.6.2 (9.7.6.b)
    Build Release: 5.11
           Branch: None
   Packaging Date: July 26, 2017 12:07:06 AM
             Size: 33.62 MB
             FMRI: pkg://solaris/firmware/system/T7-4/
sysfw9-7@9.7.6.2,5.11:20170726T000706Z
```

Follow the instructions in `/var/firmware/system/T7-4/sysfw9-7/p26407491_976b/README.html` to install the firmware update.

After you have installed the `pkg:/firmware/system/`*ServerType* package for your platform, you can use the `pkg update` command to download newer firmware updates. Use the `pkg update` command with no FMRI to download the most current firmware update package and also install the most current SRU.

For more information about firmware updates, see *Oracle ILOM Administrator's Guide for Configuration and Maintenance Firmware Release 3.2.x* or your SPARC server administration guide.

Installing an IDR Custom Software Update

Interim Diagnostic or Relief (IDR) updates deliver custom temporary workarounds or diagnostics for specific issues. An IDR is not a patch and is not generally available to all customers. An IDR might provide an interim workaround for the issue until a final fix can be delivered, or an IDR might gather diagnostic data related to the specific issue. Even if the IDR alleviates the problem, you must proceed to a permanent solution as soon as a permanent solution is available. IDRs are not delivered with SRUs, though a permanent fix for the issue might be delivered in a subsequent SRU. See Applying Support Updates for information about SRUs.

IDRs prevent you from updating the system in a way that would remove the IDR if the issue addressed by the IDR is not fixed in the update. For example, if the IDR modifies the `system/network` package and the SRU updates the `system/network` package but does not provide a fix to the problem addressed by the IDR, then that update will fail.

If an SRU delivers fixes for the issues that are addressed in an IDR, then that SRU will also include a package for that IDR that automatically removes the IDR during the update. This IDR package enables you to update to the SRU without explicitly removing the IDR.

- IDRs are not delivered in SRUs.
- If an SRU does not provide a fix for issues that are addressed by an installed IDR, attempting to update to that SRU will fail.
- If an SRU provides a fix for issues that are addressed by an installed IDR, updating to that SRU will remove the IDR. The updated system will contain a new, final version of the IDR package that has no content except a message that the IDR has been fixed.

An IDR is delivered as a package archive file with a `p5p` extension. A package archive is a file that contains publisher information and one or more packages provided by that publisher.

Installing IDRs

Before you install an IDR, get information about the IDR such as which problems are addressed by the IDR and what content the IDR delivers.

Example 4-2 Get Information About the IDR Package Archive

The following command shows that the archive contains three packages published by the `solaris` publisher:

```
$ pkgrepo info -s idr1929.2.p5p
PUBLISHER PACKAGES STATUS           UPDATED
solaris   3        online           2015-06-24T09:01:20Z
```

The `pkgrepo list` and `pkg list` commands give similar but different information about the packages in the package archive. The name of the package that you want to install or update is `idr1929`. The name of the package archive indicates that the archive is version 2 of IDR 1929. The following output indicates that `idr1834` has been renamed as described in Show Whether an IDR is Renamed.

```
$ pkgrepo list -s idr1929.2.p5p
PUBLISHER NAME                            O VERSION
solaris   idr1834                         r 2,5.11:20150624T090119Z
solaris   idr1929                           2,5.11:20150624T090120Z
solaris   system/install/media/internal
0.5.11,5.11-0.175.2.9.0.3.2.1929.2:20150624T090120Z
$ pkg list -afg idr1929.2.p5p
NAME (PUBLISHER)                          VERSION                       IFO
idr1834                                   2                             --r
idr1929                                   2                             ---
system/install/media/internal             0.5.11-0.175.2.9.0.3.2.1929.2 ---
```

Example 4-3 Show What This IDR Delivers

The following command shows that the `idr1929` package incorporates a custom version of the `system/install/media/internal` package. The version of the `system/install/media/internal` package specifies this IDR. See the `1929.2` fields at the end of the branch portion of the version string. See Fault Management Resource Identifiers for an explanation of the fields in the package version string. The commands in Get Information About the IDR Package Archive showed that the IDR package archive includes this custom version of the package that the IDR package delivers.

```
$ pkg contents -g idr1929.2.p5p -o type,fmri -t depend idr1929
TYPE        FMRI
incorporate pkg:/system/install/media/internal@0.5.11,5.11-0.175.2.9.0.3.2.1929.2
```

The following command shows that this custom version of the `system/install/media/internal` package has a `require` dependency on the `idr1929@2` package: This version of `system/install/media/internal` will only be installed when this IDR is installed.

```
$ pkg contents -g idr1929.2.p5p -o type,fmri -t depend media/internal
TYPE    FMRI
require idr1929@2
```

Example 4-4 Show Which Problems are Addressed by the IDR

The following command displays the number of the problem report that `idr1929` is addressing. Multiple problem reports could be listed.

```
$ pkg contents -Hg idr1929.2.p5p -o value -t set -a name='*bug*' idr1929
16857802
```

Example 4-5 Determine the Oracle Solaris Releases Where the IDR Can Be Installed

The following command displays which Oracle Solaris release this IDR was built for:

```
$ pkg contents -Hg idr1929.2.p5p -o value -t set -a name="*description*" idr1929
i386 IDR built for release : Solaris 11.2 SRU # 10.5.0
```

This IDR might be installable on Oracle Solaris releases other than just the release for which the IDR was built. List the versions of all packages delivered by this IDR as shown in Show What This IDR Delivers, and compare that list with the list of all available versions of those packages. The IDR can be installed on any Oracle Solaris release that allows installation of the same versions of the packages delivered by the IDR. In this example, `idr1929` can be installed on any Oracle Solaris release that allows installation of the `system/install/media/internal` package at version `5.11-0.175.2.9.0.3.2`.

Example 4-6 Show the Location of the Release Notes for an IDR

The following command shows where the release notes for an installed IDR are located:

```
$ pkg contents -Ht file idr1929
usr/share/doc/release-notes/idr1929.txt
```

You can also use the `pkg history` command to display the release notes. Use the `-n`, `-t`, and `-N` options as described in Viewing Operation History.

If the IDR is not installed, you can display the release notes without installing the IDR by using the `pkg install` command with the `-nv` options as shown in How to Install an IDR.

Example 4-7 Show Whether an IDR is Renamed

The `pkgrepo list` and `pkg list` commands in Get Information About the IDR Package Archive show that the `idr1834` package has been renamed.

IDR packages use renaming to enable a system with an IDR installed to update to a new SRU version without requiring an administrator to explicitly remove the installed IDR. For more information about updating with an IDR installed and why an IDR might be renamed, see Installing Superseding IDRs and Support Updates.

The following command shows that `idr1834` has been renamed to `idr1929`. If you explicitly install `idr1834`, `idr1929` will be installed instead, and `idr1834` will be removed if it is already installed.

```
$ pkg info -g idr1929.2.p5p idr1834
          Name: idr1834
       Summary: Superseding pkg for idr1834.1
         State: Not installed (Renamed)
    Renamed to: pkg://solaris/idr1929@2,5.11
     Publisher: solaris
```

```
       Version: 2
 Build Release: 5.11
        Branch: None
Packaging Date: June 24, 2015 09:01:19 AM
          Size: 5.46 kB
          FMRI: pkg://solaris/idr1834@2,5.11:20150624T090119Z
```

The following command shows that the `idr1834` package has no content other than a dependency on the `idr1929` package. The `signature` action is omitted for brevity.

```
$ pkg contents -mg idr1929.2.p5p idr1834
set name=pkg.fmri value=pkg://solaris/idr1834@2,5.11:20150624T090119Z
set name=pkg.summary value="Superseding pkg for idr1834.1"
set name=pkg.renamed value=true
depend fmri=pkg://solaris/idr1929@2,5.11 type=require
```

How to Install an IDR

1. **Make sure you do not have other IDRs installed that address this same problem.**

   The following command lists the identifiers of all problems addressed by the IDR that you want to install:

   ```
   $ pkg contents -g idr1929.2.p5p -Ho value -t set -a name='*bug*' idr1929
   ['problemID1, 'problemID2']
   ```

   The following command lists all installed packages that address these same problems:

   ```
   $ pkg search -lo value,pkg.name problemID1 OR problemID2
   ```

   If any problem addressed by the IDR you want to install is also being addressed by an IDR that is currently installed, check whether the IDR you want to install will remove the installed IDR that addresses this same problem. Use the `pkg list` and `pkg info` commands or use the `-nv` options with the `pkg install` command to determine whether the older IDR will be removed by installing the new IDR. If both IDRs would be installed, contact your support representative.

2. **Add the package archive as a publisher origin.**

   This step must be done so that non-global zones can access this repository. Even if this system does not have non-global zones, adding these packages to a configured publisher origin is best practice.

   ```
   $ pkg set-publisher -g idr1929.2.p5p solaris
   $ pkg publisher
   PUBLISHER   TYPE     STATUS P LOCATION
   solaris     origin   online F file:///var/share/repos/idr1929.2.p5p/
   solaris     origin   online F https://pkg.oracle.com/solaris/support/
   ```

   You might prefer to create a local repository for all your IDR packages. See *Creating Package Repositories in Oracle Solaris 11.4*. Make sure the `pkg5srv` user can access the repository content. Make sure non-global zones can access the repository content.

3. **Make sure this IDR can be installed on this system.**

   The following `pkg contents` command shows that the IDR incorporates the `media/internal` package at version `5.11-0.175.2.9.0.3.2.1929.2`.

   ```
   $ pkg contents -ro type,fmri -t depend idr1929
   TYPE        FMRI
   incorporate pkg:/system/install/media/internal@0.5.11,5.11-0.175.2.9.0.3.2.1929.2
   ```

This IDR can be installed if the system has the `media/internal` package installed at version `5.11-0.175.2.9.0.3.2`.

The following command shows you what would be installed but does not actually perform the installation. Specify the `-r` option if this image has non-global zones. Notice that the installation operation displays release notes for the IDR. The release notes include the bug that is addressed by the IDR, the packages that are installed by the IDR package, and instructions for removing the IDR.

```
$ pkg install -nvr idr1929
           Packages to install:         1
     Estimated space available:  31.42 GB
Estimated space to be consumed: 122.34 MB
       Create boot environment:        No
Create backup boot environment:        No
          Rebuild boot archive:        No

Changed packages:
solaris
  idr1929
    None -> 2,5.11:20150624T090120Z
Release Notes:
  # Release Notes for IDR : idr1929
  # ------------------------------

  Release         : Solaris 11.2 SRU # 10.5.0

  Platform         : i386

  Bug(s) addressed     :
      16857802 : AI zlib download timeout should be longer

  Package(s) included     :
      pkg:/system/install/media/internal

  Removal instruction     :
  # /usr/bin/pkg uninstall -r idr1929
  If this IDR is installed on a system running Solaris 11.2 SRU9 or less, use
  this command to remove:
  # /usr/bin/pkg update --ignore-missing -r --reject
pkg:/system/install/media/
internal@0.5.11,5.11-0.175.2.9.0.3.2:20150321T014312Z

  Generic Instructions     :

  1) If system is configured with 'Zones', ensure that
  IDR is available in a configured repository.

  Special Instructions for : idr1929

  None.
```

4. **Install the IDR.**

   The output from the previous command shows that the `idr1929` package would be installed. Note that if the system already has a version of `idr1834` installed, then installation of `idr1929` will be blocked. In that case, you should update `idr1834` instead of installing `idr1929`. The result of updating `idr1834` is that `idr1929` is installed, as shown in Install a Superseding IDR.

When you install an IDR, make sure a new BE or a backup BE will be created. If you have problems after you install the IDR, you can reboot to the old BE or the backup BE.

The output of the previous command shows that no new BE and no backup BE will be created for this installation. The following command creates a backup BE before installing the IDR into the current BE:

```
$ pkg install --backup-be-name pre-idr1929 idr1929
```

Installing Superseding IDRs and Support Updates

An IDR can be superseded by a new IDR that addresses the same problem or by an SRU that fixes the problem. Get information about the content of the IDR package and use the `-nv` options to understand what the result of the `pkg update` or `pkg install` command will be.

- Superseding IDR. IDRs can supersede other IDRs. Usually, an existing IDR is updated with new fixes, and a superseding IDR is not necessary. In some cases, modifying an existing IDR is not feasible and a new IDR is created that supersedes the previous IDR.

  A superseding IDR is needed when a change would modify a package that is already modified by a different IDR. For example, IDR1 delivers a fix to `usr/sbin/cron`, which is part of the `system/core-os` package. IDR2 delivers unrelated fixes to other packages and includes a fix for `usr/sbin/svcadm`, which also is part of `system/core-os`. IDR1 and IDR2 cannot both modify the `system/core-os` package. A superseding IDR package is necessary and can take one of the following forms:

  – IDR2 includes the fix for IDR1. IDR2 supersedes IDR1. IDR1 is renamed to IDR2 so that when you explicitly install or update the IDR1 package, the IDR2 package is installed instead, and the IDR1 package is removed if it was already installed on your system.

  – IDR3 includes the fix for IDR1 and the fix for IDR2. IDR3 supersedes both IDR1 and IDR2.

- Superseding SRU. If the issue addressed by the IDR is fixed in an SRU, then the IDR package will be renamed to the package where the fix was made. If you update to the SRU that installs the version of the package to which the IDR package was renamed, then the IDR package will be removed.

  Update to an SRU that includes a newer version of the fixed package but does not fix the problem will be blocked. To accomplish the update, uninstall the IDR or reject the IDR package. Note, however, that using these methods to remove the IDR leaves the system exposed to the problems that the IDR addresses.

Example 4-8 Install a Superseding IDR

The following output shows that this image has version 1 of `idr1834` installed. Version 2 of `idr1834` is available from `idr1929.2.p5p`, which was added as a `solaris` publisher origin in How to Install an IDR.

```
$ pkg list -af 'idr*'
NAME (PUBLISHER)      VERSION IFO
idr1834               2       --r
idr1834               1       i--
idr1929               2       ---
```

Show Whether an IDR is Renamed showed that in version 2, the `idr1834` package is renamed to `idr1929`. When you update `idr1834`, the installed version 1 is removed and `idr1929` is installed. No version of the `idr1834` package is installed after the update.

```
$ pkg update -v idr1834
            Packages to remove:         1
           Packages to install:         1
     Estimated space available:  30.42 GB
Estimated space to be consumed: 718.51 MB
       Create boot environment:        No
Create backup boot environment:        No
          Rebuild boot archive:        No

Changed packages:
solaris
  idr1834
    1,5.11:20150708T000258Z -> None
  idr1929
    None -> 2,5.11:20150624T090120Z
...

$ pkg list -af 'idr*'
NAME (PUBLISHER)      VERSION IFO
idr1834               2       --r
idr1834               1       ---
idr1929               2       i--
```

If you tried to install `idr1929` instead of update `idr1834`, that installation operation would have failed because `idr1834` would still constrain the packages on the system to a lower version than the version to which `idr1929` moves.

Example 4-9 Update to an SRU That Fixes the IDR Issue

To determine whether an installed IDR is fixed in an SRU, look for a version of that IDR package in the support repository. Since IDRs are not delivered in SRUs, an IDR package in the support repository indicates that the IDR is fixed. In this example, the support repository is configured for the `solaris` publisher.

The following `pkg search` command shows that the problem that was addressed in `idr4494` is fixed in SRU 24. Use the `-f` option to show packages that would otherwise be excluded by installed incorporations:

```
$ pkg search -fo name,value idr4494
NAME            VALUE
pkg.description idr4494 terminal package for Oracle Solaris 11.4SRU24.75.2
pkg.fmri        pkg://solaris/idr4494@4,5.11:20200821T162221Z
pkg.summary     idr4494 terminal package
```

The following `pkg info` command also shows that `idr4494` is obsolete because the problem that was addressed in `idr4494` is fixed in SRU 24, as shown in the Description. Use the `-r` option to show information about the package in the support repository, not information about the installed package.

```
$ pkg info -r idr4494
          Name: idr4494
       Summary: idr4494 terminal package
   Description: idr4494 terminal package for Oracle Solaris 11.4SRU24.75.2
         State: Not installed (Renamed)
    Renamed to: pkg:/consolidation/osnet/osnet-
```

```
incorporation@11.4,5.11-11.4.24.0.1.75.3
     Publisher: solaris
       Version: 4
        Branch: None
Packaging Date: August 21, 2020 at  4:22:21 PM
          Size: 2.52 kB
          FMRI: pkg://solaris/idr4494@4:20200821T162221Z
```

The version of the IDR that is currently installed in the image, for example `idr4494@3`, is automatically removed when the image is updated to the SRU that is named in the terminal IDR package.

Example 4-10 Update to an SRU That Does Not Fix the IDR Issue

If the SRU does not update any packages that the IDR modifies, then the `pkg update` command will update to the SRU and the IDR will remain installed.

If the SRU updates any packages that the IDR modifies but does not fix the problem addressed by the IDR, then the update operation will fail. You could use the following command to remove the IDR and proceed with the update if you are willing to lose the fix that the IDR provides:

```
$ pkg update --reject idr1929
```

If you are not willing to lose the fix that the IDR provides, then contact your support representative and request a new IDR for the new SRU. A new IDR will be generated that supersedes the existing IDR, enabling the system to be updated to the new SRU and IDR without losing the fixes.

If you update to an SRU that is not the newest available SRU, then you need to specify the superseding IDR package in the update. One way to do this is shown in the following example:

```
$ pkg update entire@intermediate-version superseding-idr '*'
```

Removing IDRs

In most cases, you should not explicitly remove an IDR because this leaves your system vulnerable to the issue that the IDR addresses. The IDR should be removed as part of the process of updating to the SRU that includes the fix for the issue that the IDR addresses.

One case when you could explicitly remove an IDR is when the IDR performs only diagnostic work and you no longer need to gather that diagnostic information.

How to Remove an IDR

1. **Check the IDR package summary for package removal instructions.**

   The IDR package summary includes instructions for how to remove this package.

   The `info`, `contents`, and `search` subcommands can all show the package summary.

   - `pkg info`

     ```
     $ pkg info idr2353
        Name: idr2353
     Summary: To back out This IDR : # /usr/bin/pkg uninstall -r idr2353
     ...
     ```

   - `pkg contents`

```
$ pkg contents -Ho value -a name=pkg.summary idr2353
To back out This IDR : # /usr/bin/pkg uninstall -r idr2353
```

- `pkg search`

```
$ pkg search -Hlo value idr2353::pkg.summary:
To back out This IDR : # /usr/bin/pkg uninstall -r idr2353
```

You can also use the command shown in Show the Location of the Release Notes for an IDR to view this information in the installed release notes file.

2. **Run the prescribed command to remove the IDR package.**

   Use the `pkg uninstall` command as shown in the package summary:

   ```
   $ pkg uninstall -r idr1929
   ```

   The `-r` option performs the removal in all non-global zones associated with this global zone.

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 constraint packages
- 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 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.

If you are using Ops Center, see Configuring Publishers in Oracle Enterprise Manager Ops Center.

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:///var/share/pkgrepos/isvrepo/
devtool        (disabled)   origin  online F http://pkg.example.com/
```

The `TYPE` column indicates whether the `LOCATION` value is an origin or a mirror. See Repository Origins and Mirrors 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:///var/share/pkgrepos/isvrepo/    -
devtool   true   false  false   origin online http://pkg.example.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 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: https://pkg.oracle.com/solaris/support/
        Origin Status: Online
              SSL Key: /var/pkg/ssl/458a3a3690dbdd688dc47aa2e66317b7ecc3ff12
             SSL Cert: /var/pkg/ssl/01bffa3f4f7bc8cf164e8719fbf1a3a43a82f893
 Cert. Effective Date: Thu Jun 22 17:09:43 2017
Cert. Expiration Date: Sun Jun 30 17:09:43 2019
          Client UUID: f892c512-6c3f-11e5-befc-7f447de68cea
      Catalog Updated: Mon Aug 06 19:36:55 2018
              Enabled: Yes
```

The Client UUID is the Universally Unique Identifier for this image.

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 https://pkg.oracle.com/solaris/
support/
```

Use the `-n` option to display only enabled publishers.

```
$ pkg publisher -n
PUBLISHER                   TYPE    STATUS P LOCATION
solaris                     origin  online F https://pkg.oracle.com/solaris/
support/
isvpub         (non-sticky) origin  online F file:///var/share/pkgrepos/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.example.com/release/ devtool
```

Example 5-2 Import Publisher Configuration

Use the `-p` option to retrieve publisher configuration information from the specified repository URI.

```
$ pkg publisher
PUBLISHER                   TYPE     STATUS P LOCATION
solaris                     origin   online F http://pkg.oracle.com/solaris/release/
$ pkgrepo -s /var/share/pkgrepos/myrepo/ info
PUBLISHER     PACKAGES STATUS           UPDATED
site          7        online           2018-04-14T18:32:46.854674Z
solaris       6143     online           2018-04-15T00:45:52.227891Z
solarisstudio 112      online           2018-04-12T19:29:52.173011Z
```

If a publisher is specified, then only the matching publisher is added or updated.

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

If no publisher is specified, all publishers are added or updated as appropriate.

```
$ pkg set-publisher -p /var/share/pkgrepos/myrepo
  Updated publisher(s): solaris
$ pkg publisher
PUBLISHER                   TYPE     STATUS P LOCATION
```

```
solaris                   origin   online F file:///var/share/pkgrepos/myrepo/
solaris                   origin   online F http://pkg.oracle.com/solaris/
release/
site                      origin   online F file:///var/share/pkgrepos/myrepo/
solarisstudio             origin   online F file:///var/share/pkgrepos/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. If you use the same set of origins for a publisher for systems in different geographic locations, for example, IPS clients will attempt to choose the best origin for each different system at that particular time.

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
$ pkg set-publisher -g /var/share/pkgrepos/solaris solaris
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
solaris                   origin   online F file:///var/share/pkgrepos/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:///var/share/pkgrepos/isvrepo/ isvpub
```

Enabling and Disabling Publisher Origins

In the following example, the new origin is added as a disabled origin. The solaris publisher and the original origin are already enabled.

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F http://pkg.oracle.com/solaris/release/
$ pkg set-publisher --disable -g /var/share/pkgrepos/solaris solaris
```

The `/var/share/pkgrepos/solaris/` origin is disabled, as shown by the `disabled` in the `STATUS` column, the `Disabled` on the `Origin Status` line, and the `false` in the `ENABLED` column:

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS   P LOCATION
solaris                   origin   disabled F file:///var/share/pkgrepos/solaris/
solaris                   origin   online   F http://pkg.oracle.com/solaris/release/
$ pkg publisher solaris
            Publisher: solaris
                Alias:
           Origin URI: file:///var/share/pkgrepos/solaris/
        Origin Status: Disabled
              SSL Key: None
             SSL Cert: None
           Origin URI: http://pkg.oracle.com/solaris/release/
        Origin Status: Online
              SSL Key: None
             SSL Cert: None
```

```
         Client UUID: 97f0e8c2-217e-11e4-b444-13b4eaeb0609
      Catalog Updated: Wed Aug 02 05:10:59 2017
              Enabled: Yes
           Properties:
                        signature-policy = verify
$ pkg publisher -F tsv
PUBLISHER  STICKY  SYSPUB  ENABLED TYPE    STATUS  URI                                        PROXY
solaris    false   false   false   origin  online  file:///var/share/pkgrepos/solaris/        -
solaris    false   false   true    origin  online  http://pkg.oracle.com/solaris/release/     -
```

In the following example, an existing disabled origin is enabled:

```
$ pkg set-publisher --enable -g /var/share/pkgrepos/solaris solaris
$ pkg publisher
PUBLISHER                   TYPE     STATUS P LOCATION
solaris                     origin   online F file:///var/share/pkgrepos/solaris/
solaris                     origin   online F http://pkg.oracle.com/solaris/release/
```

The following two commands are equivalent:

```
$ pkg set-publisher -g /var/share/pkgrepos/solaris solaris
$ pkg set-publisher --enable -g /var/share/pkgrepos/solaris solaris
```

The following two commands have equivalent effect:

```
$ pkg set-publisher --disable solaris
$ pkg set-publisher --disable -g '*' solaris
```

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 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://pkg2.example.com/ devtool
$ pkg publisher
PUBLISHER                   TYPE     STATUS P LOCATION
devtool                     origin   online F http://pkg.example.com/
devtool                     mirror   online F http://pkg2.example.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.

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. The hashes of the key and certificate are listed in the output of the `pkg publisher` command for this publisher. See Displaying Publisher Information and SSL Certificate Problem.

```
$ pkg set-publisher -k /tmp/keyfile -c /tmp/certfile publisher-name
```

Each publisher can have only one key and certificate specified. If a publisher has multiple secure origins configured, all secure origins share the one key and certificate.

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. See Specifying a Proxy for more detail and for 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.

In the following example, the value of *proxyURI* is the following, where *protocol* is `http` or `https` and `:port` is optional:

protocol`://`*host-name*`[:`*port*`]`

The `--proxy` option cannot be used to set an authenticated proxy. The following form for the value of *proxyURI* is *not* supported:

protocol`://`*user*`:`*password*`@`*host*

```
$ pkg publisher
PUBLISHER                 TYPE     STATUS P LOCATION
solaris                   origin   online F file:///var/share/pkgrepos/solaris/
$ pkg publisher -F tsv
PUBLISHER STICKY SYSPUB ENABLED TYPE   STATUS URI                                PROXY
solaris   true   false  true    origin online file:///var/share/pkgrepos/solaris/ -
$ pkg set-publisher -g http://pkg.oracle.com/solaris/release/ --proxy proxyURI solaris
$ pkg publisher solaris
      Publisher: solaris
          Alias:
     Origin URI: file:///var/share/pkgrepos/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:///var/share/pkgrepos/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:///var/share/pkgrepos/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 shows an example of what the system repository publisher looks like in the non-global zone.

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 and the list of accepted environment variable names.

In the following example, the value of *proxyURI* is the following:

`[http://|https://](`*`host-name`*`|`*`ipv4_address`*`|[`*`ipv6_address`*`])[:`*`port`*`]`

- The scheme can be `http://` or `https://`. If the scheme is not specified, `http://` is assumed.
- The proxy must contain the host name or dotted numerical IP address.

  A numerical IPv6 address must be written within brackets: `[]`.
- To specify the port number, append `:`*`port`* to the end of the host name. If not specified, the default port is 1080.

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.

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

```
$ svccfg -s system-repository:default setprop config/http_proxy = 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.

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 architecture. Optional components are called *facets* and mutually exclusive components are called *variants*.

How Facet and Variant Values Affect Package Installation

Both facets and variants are comprised of the following two components:

- A property set on the image
- Tags set on actions in a package manifest

Facet and variant tag values set on an action in a package manifest compared with the corresponding facet and variant property values set in the image determine whether that action can be installed.

Each facet and variant property and tag has a name and a value. A single package action can have multiple facet and variant tags.

Values of facet and variant properties and tags do not need to be set explicitly. Values can be default values or inherited values. For example, a non-global zone can inherit values from its parent global zone. Some variant property values are set in the image at initial system installation and cannot be changed.

The following algorithm describes how the facet and variant property values set on the image affect whether a particular action is installed. Values are described in more detail in the following sections.

- An action that has no facet or variant tags in the package manifest is always installed.
- An action that has facet tags in the package manifest is installed if the following conditions exist in the image:
  - All facet tags on the action that have a value of `all` have a facet property value of `true` in the image, either by default or by being set explicitly.
  - If any facet tag on the action has a value of `true`, at least one of those facet property values is also `true` in the image, either by default or by being set explicitly.

  In the following example, the file `test.txt` is installed only if *both* `devel` and `optional.test` are `true` in the image and *either* `doc.info` or `doc.help` is `true` in the image.

  ```
  file path=usr/share/doc/test.txt facet.devel=all facet.optional.test=all
  facet.doc.info=true facet.doc.help=true
  ```
- An action with variant tags in the package manifest is installed only if the values of all the variant tags are the same as the values of the corresponding variant properties set in the image.

In the following example, the file `x86test.txt` will not be installed. The following package manifest excerpt shows two variant tags set on the file `x86test.txt`:

```
file path=usr/share/doc/x86test.txt variant.arch=i386
variant.debug.osnet=true
```

The following command shows the values of these variant properties in the image:

```
$ pkg variant arch debug.osnet
VARIANT                   VALUE
arch                      i386
debug.osnet               false
```

- An action with both facet and variant tags is installed if both the facet and the variant property values allow the action to be installed.

Showing and Changing Variant Values

Most variants have descriptive values. For example, the value of `variant.arch` can be `i386` or `sparc`. The value of a variant tag on an action in a package manifest must match the value of the corresponding variant property in the image in order for that action to be installed. The `arch` and `zone` variant property values are set by the program that creates the image and installs its initial content; these values cannot be changed.

Values of `variant.debug.*` variants and values of unknown variants can only be `true` or `false` and are `false` in the image by default. An unknown variant is a variant that is introduced as a variant tag on an action in a package manifest, not known at the time the image was created.

If both debug and non-debug versions of an action are delivered, both versions must have the applicable `debug` variant tag explicitly set, as shown in the following example. In this example, if `debug.osnet` is `false` in the image, then the non-debug version of this file is installed; if `debug.osnet` is `true` in the image, then the debug version of this file is installed. Usually debug versions of a file also have `variant.arch` tags.

```
file payload chash=hash group=sys mode=0644 overlay=allow owner=root
path=etc/motd pkg.csize=116 pkg.size=106 preserve=true variant.debug.osnet=true
file payload chash=hash group=sys mode=0644 overlay=allow owner=root
path=etc/motd pkg.csize=70 pkg.size=50 preserve=true variant.debug.osnet=false
```

Use the `pkg variant` command to display the values of variant properties that are set in the image.

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

Use the `-a` option to display the values of all variants that are explicitly set in the image and all variants that are set in installed packages.

```
$ pkg variant -a
VARIANT                   VALUE
arch                      i386
debug.container           false
debug.osnet               false
opensolaris.zone          global
```

Use the `-v` option to display all possible variant values that can be set for installed packages.

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

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

> **Note:**
>
> Changing variant property values might update a large number of packages and might require a new BE. Use the `-nv` options to review what changes will be made before you make any changes.

In the following command, a new BE would be created. See Boot Environment Policy Image Properties for information about when BEs are created. When a new BE is created, the current BE is not changed; boot to the new BE to use the debug files in this example. The `-n` option shows what would change if you performed the operation without `-n`, but the command makes no actual changes. No new BE was created in this example.

```
$ pkg change-variant -nv variant.debug.osnet=true
            Packages to change:       232
     Variants/Facets to change:         1
     Estimated space available: 306.74 GB
Estimated space to be consumed:   1.49 GB
       Create boot environment:       Yes
     Activate boot environment:       Yes
Create backup boot environment:        No
          Rebuild boot archive:       Yes

Changed variants/facets:
  variant debug.osnet: true

Changed packages:
solaris
  ...

Editable files to change:
  Update:
    etc/motd
```

Showing and Changing Facet Values

Facet properties have default values in the image and are not required to be explicitly set. Facet properties whose names begin with `facet.debug` or `facet.optional` have the value `false` by default. All other facet properties have the value `true` by default.

A facet tag on an action in a package manifest can only have the value `true` or `all`. See How Facet and Variant Values Affect Package Installation for a description and example. Actions tagged with facets whose names begin with `facet.debug` or `facet.optional` will not be installed by default. To install such actions, you must change the facet property value in the image to `true`. Actions tagged with any other facet will be installed by default. To avoid installing such actions, you must change the facet property value in the image to `false`.

Changing the value of a version-lock facet (`version-lock.`*package-name*) does not cause actions to be installed or uninstalled. Instead, version-lock facets enable or disable an `incorporate` type `depend` action. See Updating a Package Constrained by a Constraint Package.

Facet property values have one of the following three sources. These source names appear in the SRC column of `pkg facet` command output.

- `system` – Values assigned by the system. These values usually are values of facet tags specified in system packages.
- `local` – Values set by using the `pkg change-facet` command or by using IPS APIs.
- `parent` – Values inherited from a parent image. For example, a non-global zone inherits facet settings from the global zone.

Use the `pkg facet` command to display the current value and source of that value of all facet properties that either have been set locally in this image or have been inherited from a parent image.

Use the `pkg facet -a` command to display the values of all facets that are explicitly set in the image and all facets that are set in installed packages.

The following `pkg facet` commands show that no `doc.` facet properties are set locally or inherited:

```
$ pkg facet doc.*
FACET                     VALUE SRC
pkg facet: no matching facets found
$ pkg facet -a doc.*
FACET                     VALUE SRC
doc.help                  True  system
doc.html                  True  system
doc.info                  True  system
doc.man                   True  system
doc.pdf                   True  system
doc.ps                    True  system
```

Use the `pkg change-facet` command to change the value of a facet property. You can set a facet property value to `true`, `false`, or `none`. Setting a facet property to `none` applies the system default value to that facet (either `true` or `false`), and the source is shown as `system`.

> **Note:**
>
> Changing facet property values might update a large number of packages and might require a new BE. Use the `-nv` options to review what changes will be made before you make any changes.

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 that namespace. 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 locale.en_US=true
```

In the following `pkg change-facet` command, a new BE was not created, but a backup BE was created. See Boot Environment Policy Image Properties for information about when BEs are created. When a backup BE is created, the original image is saved in the backup, and the current BE is changed. The original value of `None` under Changed Variants/Facets indicates that no local setting existed prior to this command execution; only the system settings existed, as shown by the `pkg facet` commands above.

```
$ pkg change-facet -v doc.*=false doc.man=true
            Packages to change:        97
     Variants/Facets to change:         2
            Services to change:         1
     Estimated space available: 306.34 GB
Estimated space to be consumed: 936.87 MB
       Create boot environment:        No
Create backup boot environment:       Yes
          Rebuild boot archive:        No

Changed variants/facets:
    facet doc.* (local): None -> False
    facet doc.man (local): None -> True

Changed packages:
solaris
  ...

Services:
  restart_fmri:
    svc:/application/texinfo-update:default

PHASE                                          ITEMS
...
```

The following `pkg facet` commands show the `doc.` facet property changes that resulted from the `pkg change-facet` command:

```
$ pkg facet doc.*
FACET                     VALUE SRC
doc.*                     False local
doc.man                   True  local
$ pkg facet -a doc.*
FACET                     VALUE SRC
doc.*                     False local
doc.help                  False local
doc.html                  False local
doc.info                  False local
doc.man                   True  local
doc.pdf                   False local
doc.ps                    False local
```

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.

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.

```
$ pkg list openssl
NAME (PUBLISHER)                                  VERSION                    IFO
library/security/openssl                          1.0.2.15-11.4.0.0.1.10.0   i--
$ pkg freeze -c "Prevent update until testing complete." openssl
library/security/openssl was frozen at 1.0.2.15-11.4.0.0.1.10.0:20180702T170444Z
```

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.

```
$ pkg freeze
NAME                    VERSION
DATE                    COMMENT
library/security/openssl 1.0.2.15-11.4.0.0.1.10.0:20180702T170444Z 06 Aug 2018
15:56:16 PDT Prevent
update until testing complete.
```

The `f` in the package listing indicates that the package is frozen.

```
$ pkg list openssl
NAME (PUBLISHER)                                  VERSION                    IFO
library/security/openssl                          1.0.2.15-11.4.0.0.1.10.0   if-
```

The State field of the pkg info output also indicates that this package is frozen.

```
$ pkg info openssl
          Name: library/security/openssl
       Summary: OpenSSL - a Toolkit for Secure Sockets Layer (SSL v2/v3) and
                Transport Layer (TLS v1) protocols and general purpose
                cryptographic library
   Description: OpenSSL is a full-featured toolkit implementing the Secure
                Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1)
                protocols as well as a full-strength general purpose
                cryptography library.
      Category: System/Security
         State: Installed (Frozen)
```

```
     Publisher: solaris
       Version: 1.0.2.15 (1.0.2o)
        Branch: 11.4.0.0.1.10.0
Packaging Date: Mon Jul 02 17:04:44 2018
          Size: 16.05 MB
          FMRI: pkg://solaris/library/security/
openssl@1.0.2.15-11.4.0.0.1.10.0:20180702T170444Z
   Project URL: https://www.openssl.org/
    Source URL: https://www.openssl.org/source/openssl-1.0.2o.tar.gz
```

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

```
$ pkg list -n openssl
NAME (PUBLISHER)                                  VERSION                    IFO
library/security/openssl                          1.0.2.15-11.4.0.0.1.12.0   ---
$ pkg update openssl
No updates available for this image.
$ pkg update openssl@1.0.2.15-11.4.0.0.1.12.0
Creating Plan (Solver setup): /
pkg update: No matching version of library/security/openssl can be installed:
  Reject:  pkg://solaris/library/security/openssl@1.0.2.15-11.4.0.0.1.12.0
  Reason:  This version is excluded by a freeze on library/security/openssl at
version 1.0.2.15-11.4.0.0.1.10.0:20180702T170444Z.  The reason for the freeze is:
Prevent update until testing complete.
```

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 Constraint Packages

A constraint package specifies which versions of packages can be installed. These version constraints help keep the system in a supportable state across updates. See Constraint Packages for more information about constraint packages and version constraints.

Some packages that are `incorporate` dependencies might be safe to downgrade or upgrade at a version different from the version specified by the `incorporate` dependency of the constraint package. The version constraint is expressed by a `version-lock.`*`package`* facet tag specified on the `depend` action in the constraint package. The default value of the `version-lock.`*`package`* facet property is `true`. To relax the version constraint on a package, set the value of its `version-lock.` facet to `false`.

> **Note:**
>
> Unlocking packages can result in an unsupported configuration. Best practice is to keep systems updated and keep version locks set to `true`. See Applying Support Updates.

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 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 constraint package, setting the `version-lock` facet for this package to `true` installs the version specified in the constraint 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 constraint packages. In this case, setting the `version-lock` facet to `false` enables you to unlock the constraint package from the rest of the system. Though the constraint package is unlocked, its `incorporate` dependency packages 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, 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.

If you sometimes must use one version and other times must use a different version, consider specifying the full path rather than changing the mediated link.

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. Your list will be different, depending on what you have installed.

```
$ pkg mediator
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
apache       system    2.4     system
apr-1-config system    1.5     system
apu-1-config system    1.5     system
csh          vendor            vendor     suncsh
gcc          system    5       system
java         system    1.8     system
openssl      vendor            vendor     default
perl         vendor    5.22    vendor
python       local     2.7     vendor
ruby         vendor    2.1     vendor
sendmail     vendor            vendor     sendmail
which        vendor            vendor     gnu
```

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    1.8     system
java        system    1.7     system
```

The following output shows all packages that deliver the java mediation:

```
$ pkg search -lo pkg.name,mediator-version link:mediator:java
PKG.NAME           MEDIATOR-VERSION
runtime/java/jre-7 1.7
runtime/java/jre-8 1.8
```

The following output confirms that two different versions of the Java Runtime Environment are installed in this image and version 1.8.0_112 is the currently selected preferred version:

```
$ pkg list -s '*jre*'
NAME (PUBLISHER)        SUMMARY
runtime/java/jre-7      Java Platform Standard Edition Runtime Environment
(1.7.0_131-b12)
runtime/java/jre-8      Java Platform Standard Edition Runtime Environment
(1.8.0_121-b13)
$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

The following output shows the mediated links (PATH) and where they link (TARGET):

```
$ pkg contents -o mediator-version,path,target -t link -a mediator=java jre-8 jre-7
MEDIATOR-VERSION PATH                 TARGET
1.7              usr/java             jdk/jdk1.7.0_131
1.8              usr/java             jdk/jdk1.8.0_121
1.7              usr/jdk/jdk1.7.0_131 instances/jdk1.7.0
1.8              usr/jdk/jdk1.8.0_121 instances/jdk1.8.0
1.7              usr/jdk/latest       jdk1.7.0_131
1.8              usr/jdk/latest       jdk1.8.0_121
$ ls -l /usr/java
lrwxrwxrwx   1 root     root          16 Mar 23 16:11 /usr/java -> jdk/jdk1.8.0_121/
```

Both the `jre-8` and `jre-7` packages define a symbolic link whose path is `/usr/bin/java`. In the `jre-7` package, the target of the link is `jdk1.6.0`. In the `jre-8` package, the target of the link is `jdk1.7.0`. The `pkg mediator` and `java -version` commands above show that version 1.8 is currently the preferred version, the target of the `/usr/bin/java` link.

Identifying Mediation Participants That Are Not Currently Available

The following output shows all currently available participants in the `sendmail` mediation:

```
$ pkg mediator -a sendmail
MEDIATOR      VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
sendmail      vendor            vendor     sendmail
```

The following output shows that the `fips-140` implementation of `sendmail` will be available after you install the `sendmail-fips-140` package:

```
$ pkg search -o pkg.name,mediator-implementation link:mediator:sendmail
PKG.NAME                      MEDIATOR-IMPLEMENTATION
service/network/smtp/postfix  postfix
service/network/smtp/sendmail sendmail
$
```

Setting a Preferred Path that is Not Available: Consequences and Recovery

The following output shows only one implementation available in this image for the `perl` command:

```
$ pkg mediator -a perl
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
perl         vendor    5.22    vendor
perl         system    5.12    system
```

Though only one implementation is available, the path is still a mediated link:

```
$ which perl
/usr/bin/perl
$ ls -l /usr/bin/perl
lrwxrwxrwx   1 root     root          22 Mar 23 16:06 /usr/bin/perl -> ../
perl5/5.12/bin/perl
```

Perhaps you know that a newer version exists, and you enter the following command to update the link to point to the newer version. The following command is incorrect because the specified version is not currently available in this image:

```
$ pkg set-mediator -vV 5.22 perl
            Packages to change:         2
```

```
           Mediators to change:         1
     Estimated space available: 867.06 GB
Estimated space to be consumed: 241.78 MB
       Create boot environment:        No
Create backup boot environment:       Yes
          Rebuild boot archive:        No

Changed mediators:
  mediator perl:
           version: 5.12 (vendor default) -> 5.22 (local default)
...
```

The `pkg mediator` command shows the updated setting:

```
$ pkg mediator perl
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
perl         local     5.22    system
```

However, the `pkg mediator -a` command shows that the current setting is not available:

```
$ pkg mediator -a perl
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
perl         vendor    5.12    vendor
```

The result of the incorrect version specification is that the mediated link was removed:

```
$ which perl
no perl in /usr/sbin /usr/bin
$ ls -l /usr/bin/perl
/usr/bin/perl: No such file or directory
```

You could work around this problem by specifying the full path to the command (`/usr/perl5/5.12/bin/perl`). The target path was not affected. Only the mediated link was removed.

Any one of the following actions should fix this problem:

- Set the preferred version back to the version that is available.

  ```
  $ pkg set-mediator -V 5.12 perl
  ```

- Use the `pkg unset-mediator` command to allow the system to select a new implementation from among the available implementations.

  ```
  $ pkg unset-mediator perl
  ```

- Install the packages that deliver the updated version that you want.

  The output from the following command shows which packages deliver `perl` mediated links:

  ```
  $ pkg search -o pkg.name,mediator-version link:mediator:perl
  ```

  The `runtime/perl-522` and `terminal/cssh-522` packages deliver `perl` mediated links for version 5.22. The `terminal/cssh-522` package installs the `runtime/perl-522` package. To fix the problem, you only need to install `terminal/cssh-522`.

  ```
  $ pkg install terminal/cssh-522
  ```

If you cannot fix or work around the issue, reboot to the backup BE that was created when you used the `pkg set-mediator` command.

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. Do not specify a mediation participant that is not shown by `pkg mediator -a`.

> **Caution:**
>
> If you make a typographical error or otherwise specify a mediator version or implementation that is not currently available, any links that are managed by that mediator are removed as shown in Setting a Preferred Path that is Not Available: Consequences and Recovery.

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 following output shows that version 1.8 is the currently selected preferred version of the `java` mediation, and version 1.7 also is available in this image:

```
$ pkg mediator java
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java     system    1.8     system
$ pkg mediator -a java
MEDIATOR    VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java        system    1.8     system
java        system    1.7     system
```

The following command shows setting version 1.7 as the preferred version, which means that invoking `/usr/bin/java` will invoke JRE version 1.7. JRE version 1.8 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 set-mediator -v -V 1.7 java
            Packages to change:         3
           Mediators to change:         1
     Estimated space available: 867.06 GB
Estimated space to be consumed: 238.26 MB
       Create boot environment:        No
Create backup boot environment:       Yes
          Rebuild boot archive:        No
...
$ pkg mediator java
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
java     local     1.7     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.

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. Explicitly 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 are the target of a group dependency of an installed package cannot be removed from the avoid list by using the `pkg unavoid` command. To remove such a package from the avoid list, install the package.

Example 5-5 Add Packages To and Remove 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                                 5.12-5.12.0.0.0.110.0      ---
$ pkg list -a `pkg contents -o fmri -Hrt depend -a type=group group/feature/amp`
NAME (PUBLISHER)                                  VERSION                    IFO
database/mysql-56                                 5.6.29-5.12.0.0.0.110.0    ---
web/php-56                                        5.6.22-5.12.0.0.0.110.0    ---
web/server/apache-24                              2.4.23-5.12.0.0.0.110.0    i--
web/server/apache-24/module/apache-dbd            2.4.23-5.12.0.0.0.110.0    ---
web/server/apache-24/module/apache-fcgid          2.3.9-5.12.0.0.0.110.0     ---
```

```
web/server/apache-24/module/apache-ldap           2.4.23-5.12.0.0.0.110.0    ---
web/server/apache-24/module/apache-ssl            2.4.23-5.12.0.0.0.110.0    i--
```

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 mysql-56
$ pkg avoid
    database/mysql-56
```

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-56                                 5.6.29-5.12.0.0.0.110.0    ---
web/php-56                                        5.6.22-5.12.0.0.0.110.0    i--
web/server/apache-24                              2.4.23-5.12.0.0.0.110.0    i--
web/server/apache-24/module/apache-dbd            2.4.23-5.12.0.0.0.110.0    i--
web/server/apache-24/module/apache-fcgid          2.3.9-5.12.0.0.0.110.0     i--
web/server/apache-24/module/apache-ldap           2.4.23-5.12.0.0.0.110.0    i--
web/server/apache-24/module/apache-ssl            2.4.23-5.12.0.0.0.110.0    i--
```

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

```
$ pkg avoid
    database/mysql-56 (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 mysql-56
pkg unavoid: The following packages are a target of group dependencies; use install to
unavoid these:
    database/mysql-56
$ pkg install mysql-56
$ 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 mysql-56
pkg avoid: The following packages are already installed in this image; use uninstall
to avoid these:
    database/mysql-56
$ pkg uninstall mysql-56
$ pkg avoid
    database/mysql-56 (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-56
$ pkg unavoid mysql-56
```

```
$ 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.

Daemons or other programs that need to know the time an image was last modified can consult the time stamp on the file `/var/pkg/modified`. The time stamp on `/var/pkg/modified` is updated every time an operation occurs that modifies the image. Use the `pkg history` command to display information about the change at that time. See Viewing Operation History for more information.

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 11.4 Boot Environments*.

See Setting Image Properties 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.

The following property affects how a new BE is named:

be-use-suggested-be-name
When a new BE is created and `--be-name` was not specified, use the value of `com.oracle.info.suggested_bename` in the `pkg:/release/name` package of the new BE as the name of the new BE:

```
$ pkg contents -Ha name=com.oracle.info.suggested_bename -o value release/name
```

The default value of `be-use-suggested-be-name` is `True`. To opt out of this automatic naming of a new BE, set `be-use-suggested-be-name` to `False`. When a new BE is created and `--be-name` was not specified – and the value of `be-use-suggested-be-name` is `False`, the name of the new BE is the name of the current BE with *-n* appended, where *n* is the next in the series 1, 2, 3,...

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 package 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 package client to discover link-local content mirrors using mDNS and DNS-SD. If this property is set to `true`, the package client attempts to download package content from mirrors it dynamically discovers. To run a mirror that advertises its content via mDNS, see the `pkg.depotd`(8) man page.
The default value is `False`.

send-uuid
Send the Universally Unique Identifier (UUID) for this image when performing network operations. Although users can disable this option, some network repositories might refuse to talk to package 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 root of the image.
The default value is `etc/certs/CA`.

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 the `pkg.sysrepo`(8) man page 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.

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 Images and Boot Environments 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 that modifies the image. Commands such as `pkg search`, `pkg refresh`, `pkg publisher`, `pkg facet`, and `pkg property` are not recorded in `pkg` history, but commands such as `pkg set-publisher`, `pkg change-facet`, and `pkg set-property` are recorded in `pkg` history. Sometimes `pkg history` shows multiple operations executed for one command entered. For example, `pkg refresh --full` displays `refresh-publishers` and `rebuild-image-catalogs` in `pkg history` output.

To quickly check the time of the last image modification or configuration change, check the time stamp of the `/var/pkg/modified` file. The `/var/pkg/modified` file is an empty file whose time stamp is updated every time an operation occurs that modifies the image. For more information about what changed, use the `pkg history` command with the `-t` option as shown in Getting Information About Image Change.

By default, the `pkg history` command displays the following information:

- 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 the `pkg`(1) man page.

```
$ 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
```

Example 5-6 Getting Information About Image Change

The following command uses the `/var/pkg/modified` file to show the last time this image was modified:

```
$ ls -l /var/pkg/modified
-rw-r--r--   1 root     root           0 Jul 22 10:54 /var/pkg/modified
```

You might not need any more information. You might only need to know whether the image has changed since the last time you performed a certain operation.

The following example shows how you can use the `pkg history` command to get more information.

The following command shows `pkg` operations that occurred within a specified range of time:

```
$ pkg history -t 2016-07-22T10:54:00-2016-07-22T10:55:00
START                   OPERATION               CLIENT              OUTCOME
2016-07-22T10:54:05     update-publisher        pkg                 Succeeded
2016-07-22T10:54:06     refresh-publishers      pkg                 Succeeded
2016-07-22T10:54:53     rebuild-image-catalogs  pkg                 Succeeded
```

The following command shows more information about a particular operation from the preceding list:

```
$ pkg history -lt 2016-07-22T10:54:05
         Operation: update-publisher
           Outcome: Succeeded
            Reason: None
            Client: pkg
           Version: 77d785eb851b
              User: admin (100)
         Boot Env.: 103_104.34291
    Boot Env. UUID: 1831b116-361f-4ef0-9406-851c446ec4c2
     New Boot Env.: None
New Boot Env. UUID: (None)
          Snapshot: (None)
        Start Time: 2016-07-22T10:54:05
          End Time: 2016-07-22T10:55:25
        Total Time: 0:01:20
```

```
           Command: /usr/bin/pkg set-publisher -g /var/share/pkg/repositories/solaris
solaris
     Release Notes: No
       Start State:
None
         End State:
None
```

Example 5-7 Purging History Information

Use the `pkg purge-history` command to delete all existing 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, including:

- Package cannot be installed
- Cannot satisfy constraints

This appendix first describes information that you should check when you first begin to troubleshoot. Following these suggestions can save you a large amount of time.

This appendix also provides tips for increasing performance and minimizing stored metadata.

Initial Troubleshooting Steps

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

- Check which version of the `pkg:/entire` constraint package is installed. You might need to update the `pkg:/entire` package.

  The following command shows the version of the `pkg:/entire` package that is installed in this image:

  ```
  $ pkg list -v entire
  ```

- Check your package publisher origin. You might need to add or remove publisher origins.

  The following command lists publishers that are configured in this image that are enabled:

  ```
  $ pkg publisher -n
  ```

  Use the `--no-network-cache` global option to ensure that you are seeing the current content of configured publisher locations and not a cached copy. See the `pkg`(1) man page for more information.

  See Enabling and Disabling Publisher Origins for information about how to see whether particular publisher locations are currently enabled.

- Check whether any `version-lock` facet is set to `false`. See Relaxing Version Constraints Specified by Constraint Packages for more information.

  ```
  $ pkg facet
  ```

- Check whether any package is frozen at a particular version. See Locking Packages to a Specified Version for more information.

  ```
  $ pkg freeze
  ```

- Check whether an IDR is installed. You might need to reject the IDR. See Installing an IDR Custom Software Update for information about installing superseding IDRs and removing IDRs.

The following command lists any IDRs that are installed:

```
$ pkg list '*idr*'
```

- Check whether packages installed from a different publisher might be blocking your update or installation.

  The following command lists any packages that are installed from the `internalpublisher` publisher:

  ```
  $ pkg list -v '*internalpublisher*'
  ```

Check whether the package you want to install is available from configured publishers and can be installed in this image. The following command shows whether the package you want to install is available from configured publishers. See also Check Available Versions.

```
$ pkg list -af name-of-package-to-install
```

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. See also Preview the 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.

If you are updating, the system must have access to a package repository that provides the packages that are currently installed on the system. For example, if you are updating from Oracle Solaris 11.3 to Oracle Solaris 11.4, the `solaris` publisher must be configured with access to both the installed Oracle Solaris 11.3 packages and the desired Oracle Solaris 11.4 packages. If packages are installed from another publisher, such as `ha-cluster` or `solarisstudio`, those publishers also must be configured with access to currently installed packages as well as desired newer packages.

The cause of most update errors is an incomplete package repository. See Best Practices for Creating and Using Local IPS Package Repositories in *Creating Package Repositories in Oracle Solaris 11.4*.

Check the Installed Version of `pkg:/entire`

Understanding IPS package versioning and how versions of two different packages compare is often very important for troubleshooting package installation and update issues. See Fault Management Resource Identifiers and Package Version in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* for more information.

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

```
$ pkg list -Hv entire
pkg://solaris/entire@0.5.11,5.11-0.175.2.8.0.5.0:20150325T200338Z      i--
```

The version of this `pkg:/entire` constraint package indicates that this system is running Oracle Solaris 11.2 SRU 8.5.

The `pkg:/entire` constraint package constrains the versions of many other packages to help maintain a working, supportable image. See Constraint Packages for more information about packages constrained by constraint packages.

You cannot directly install or update a package that is constrained by the `pkg:/entire` constraint 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. In some cases, you can remove the constraints as shown in Updating a Package Constrained by a Constraint Package.

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 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`, disable the unreachable origin with `pkg set-publisher -dg` as shown in Enabling and Disabling Publisher Origins, or disable the publisher with `pkg set-publisher -d`. 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                                            0.5.11-0.175.2.8.0.5.0     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@0.5.11-0.175.2.8.0.5.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.8.0.5.0
```

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 /var/share/pkgrepos/solaris entire@0.5.11-0.175.2.8.0.5.0
solaris   entire                  0.5.11,5.11-0.175.2.8.0.5.0:20150325T200338Z
```

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.
- Update your package repository to include the packages you need. See Best Practices for Creating and Using Local IPS Package Repositories in *Creating Package Repositories in Oracle Solaris 11.4*.

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 or update your package repository.

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 a constraint package. You can update the constraining package or, in some cases, you can relax the constraint. For more information, see Cannot Satisfy Constraints.
- 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.

Two publishers should not provide packages with the same name. If this situation occurs, take one of the following actions:

- If you no longer need any of the packages from one of the publishers, disable or unset that publisher. If packages from that publisher are currently installed, uninstall them before you disable or unset the publisher.
- If you need both publishers, do one of the following:
  - Make sure the publisher that is not preferred is not sticky (`--non-sticky`), and set the publisher search order so that the publisher that provides the package you want to install is ahead of any other publisher that provides the same-named package.
  - Specify the full name of the package you want to install, including the publisher.

Use the `pkg freeze` and `pkg facet` commands to check whether a frozen or version-locked package is preventing installation or update.

Use the `pkg verify -v` command to check whether packaged content has been modified. Use the `pkg fix` command to fix any errors that were found.

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.

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.

Configuring Publishers in Oracle Enterprise Manager Ops Center

If you are using Ops Center, the IPS package repository is called the Oracle Solaris Software Update Library. For information about how to update the library, see the Oracle Solaris sections of the Software Library section of the Oracle Enterprise Manager Ops Center configuration reference manual.

To associate new certificates with the `solaris` publisher, use the Configure Parent Repos and Add Content actions in the Library → Oracle Solaris section of the Ops Center BUI.

See the Firewall Rules section of the Oracle Enterprise Manager Ops Center Ports and Protocols guide for firewall rules to allow systems to access the external IPS repository. This list can also be useful if you are not using Ops Center.

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`
- `Framework error: code: E_COULDNT_CONNECT (7) reason: Failed to connect`

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.

If the image has non-global zones, see the instructions for non-global zones in Location Not Found.

SSL Certificate Problem

Error messages:

- `Framework error: code: 35 reason: SSL routines`
- `Framework error: code: 60 reason: SSL certificate problem, verify that the CA cert is OK`
- `Framework error: code: 60 reason: SSL certificate problem: self signed certificate`

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

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

- Make sure the time and date are correct on the system.
- Make sure the key and certificate required by the publisher origin are installed and are not expired.

Information about how to obtain a key and certificate should be included with other information about the secure repository. For example, use the `https://pkg-register.oracle.com/` site to obtain a key and certificate for the `https://pkg.oracle.com/solaris/support/` Oracle Solaris support repository.

Use the `-k` and `-c` options with the `pkg set-publisher` command to install the key and certificate files for this publisher. Each publisher can have only one key and certificate specified. If a publisher has multiple secure origins configured, all secure origins share the one key and certificate.

```
$ pkg set-publisher -k /tmp/keyfile -c /tmp/certfile publisher-name
```

Use the `pkg publisher` command for the publisher to verify that the key and certificate files are installed and are not expired.

```
$ pkg publisher solaris

            Publisher: solaris
                Alias:
           Origin URI: https://pkg.oracle.com/solaris/support/
              SSL Key: /var/pkg/ssl/keyfile
             SSL Cert: /var/pkg/ssl/certfile
 Cert. Effective Date: July  1, 2015 04:47:13 PM
Cert. Expiration Date: July  8, 2017 04:47:13 PM
          Client UUID: client-uuid
      Catalog Updated: May 11, 2016 03:28:43 PM
              Enabled: Yes
           Properties:
                        proxied-urls = []
                        signature-policy = require-signatures
```

If the key or certificate is expired, an error message such as the following is shown:

```
Certificate '/var/pkg/ssl/certfile' has expired.
  Please install a valid certificate.
```

- Make sure the key and certificate work with the specified origin.

  The following command fails because the specified origin requires a key and certificate:

  ```
  $ pkgrepo info -s https://pkg.oracle.com/solaris/support/
  ```

  The following command succeeds because the key and certificate file names copied from the `pkg publisher` *publisher-name* output are valid:

  ```
  $ pkgrepo info -s https://pkg.oracle.com/solaris/support/ \
  > --key /var/pkg/ssl/keyfile --cert /var/pkg/ssl/certfile

  PUBLISHER PACKAGES STATUS           UPDATED
  solaris   6711     online           2016-05-19T19:00:10.152688Z
  ```

- Make sure the CA certificate is not corrupted.

  Verify the `crypto/ca-certificates` package.

  ```
  $ pkg verify crypto/ca-certificates
  ```

  If any problems are reported, perform the following steps:

  – Fix the `crypto/ca-certificates` package.

```
$ pkg fix crypto/ca-certificates
```

  - Refresh the `system/ca-certificates` SMF service.

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

- Check proxies and firewalls.

  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 for instructions.

  See the Firewall Rules section of the Oracle Enterprise Manager Ops Center Ports and Protocols guide for firewall rules to allow systems to access the external IPS repository. This list can also be useful if you are not using Ops Center.

- If you are using a self-signed certificate, add the CA certificate to the system as described in Creating a Self-Signed Server Certificate Authority in *Creating Package Repositories in Oracle Solaris 11.4*.
- If you are using Ops Center, see Configuring Publishers in Oracle Enterprise Manager Ops Center.

Location Not Found

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

Check your publisher URIs as described in Cannot Access Package Repository. 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 Package Depot Server Behind a Web Server in *Creating Package Repositories in Oracle Solaris 11.4* 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`(8) 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 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 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
/var/share/pkgrepos/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` constraint package or by some other constraint package. You must update the constraint package. For more information, see Cannot Satisfy Constraints.

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 a constraint package. See Constraint Packages for information about constraint packages and version constraints.

Constraint packages 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 a constraint package. Instead, you should update the constraint 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.

See also Non-Global Zone Cannot Be Installed.

Updating a Package Constrained by a Constraint Package

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

If you still want to update just one package from a constraint package, 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 constraint package. 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 Constraint Packages.

Example A-1 Unlock and Update 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` constraint 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` constraint 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 constraint package. To update the Java software without updating other software, unlock the `version-lock` facet of the Java constraint package and then update the Java constraint package. For more information about version-lock facets, see Relaxing Version Constraints Specified by Constraint Packages.

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 a Constraint Package When a Suitable Dependency Cannot Be Found

Error message: `A version for 'incorporate' dependency cannot be found`

See Constraint Packages for information about constraint packages and their `incorporate` dependencies.

Examples of reasons a constraint package might fail to update include the following installation states of an `incorporate` dependency of the constraint 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 Update `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` constraint 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` constraint 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
5.0.0.0.2.0:20111019T144756Z
  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` constraint 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` constraint package to which the system attempted to update. The version of the `pkg:/entire` constraint 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 a Constraint Package 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` constraint package. The following command shows that the `desktop-incorporation` package is required by the `pkg:/entire` constraint package:

```
$ 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@2.6.4-0.175.1.0.0.15.0:20120501 4T200156Z found:
    Reject:  pkg://solaris/library/python-2/python-extra-26@
2.6.4-0.175.1.0.0.15.0:20120501 4T200156Z
    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` constraint package. 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` constraint package, which is preventing the update of the `pkg:/entire` constraint package. 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 for more information about the `pkg freeze` and `pkg unfreeze` commands.

The `pkg sync-linked` error shown in Sync Linked Package Cannot Be Installed 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.`

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.

See also Non-Global Zone Cannot Be Installed.

Cannot Use Temporary Origins With Child Images

Error message: `pkg install: The proposed operation on this parent image can not be performed because temporary origins were specified and this image has children.`

The children in this message are non-global zones. The parent image is the global zone. Temporary origins were specified by using the `-g` option with the `pkg install` or `pkg update` command. The `-g` option cannot be used if the image has non-global zones.

Instead of using the `-g` option to specify a temporary origin, use the `pkg set-publisher` command to add that origin for that publisher in the global zone. When he publisher is configured in the global zone, the non-global zones can access the origin through the system repository.

Non-Global Zone Cannot Be Installed

Error messages:

- `The following pattern(s) did not match any allowable packages. Try using a different matching pattern, or refreshing publisher information`
- `Linked progress: pkg: update failed (linked image exception(s)): pkg sync-linked: No solution was found to satisfy constraints`

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/VARSHARE/zones/myzone
Progress being logged to /var/log/zones/zoneadm.20160206T181301Z.myzone.install
       Image: Preparing at /system/zones/myzone/root.

Install Log: /system/volatile/install.4606/install_log
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 `/var/share/pkgrepos/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 -Hs /var/share/pkgrepos/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 /var/share/pkgrepos/solaris/ solaris
```

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.

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

See also Sync Linked Package Cannot Be Installed.

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 package 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 cbusl04061, 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.

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 no package installed in the image either explicitly or implicitly references a 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`(7) man page for information about the `IMAGE_META` directory.

To avoid having these files move, avoid storing unpackaged data in packaged directories. For example, store user data in a separate dataset such as a directory in `/var/` *app-name*.

Minimize Stored Image Metadata

> **Note:**
>
> Do not modify anything in the `/var/pkg` directory except by explicit instructions from Oracle Support or official Oracle documentation. Modification of `/var/pkg` or its contents can result in an unusable and unsupported system.

Do not manually remove anything from the `/var/pkg` directory. The `/var/pkg` directory holds metadata for the image and is used to manage image state and cache catalogs, for example.

The `/var/pkg` directory can get quite large. 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 following command 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`(8) 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 90%.

  ```
  $ 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 *Creating Package Repositories in Oracle Solaris 11.4*.
- If `http_proxy` is set, check the performance of the proxy.
- If you are installing or updating a large number of systems, for example a large number of Oracle VM Servers, configure an Apache web server and set the `pkg/threads` property as described in Configuring a Simple Prefixed Proxy in *Creating Package Repositories in Oracle Solaris 11.4*.

Index

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

U

V

Z