
ORACLE®

ORACLE

A brief introduction to IPS

Tim Foster
Solaris Core OS

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Outline

- Why build a new packaging system
- Design assertions
- Packaging basics
- Actions
- Dependency types
- Installation / Upgrade
- Implementing side effects safely – actuators
- Handling diversity – variants and facets
- Tools

Why Build A New Packaging System?

- Existing SVR4 packaging needed work
  - Required better integration of patching with packaging
    - Artificial distinction between what was in a patch vs. what was in a package
  - No automatic installation of dependencies
  - Required better network support
  - Install-time scripting was error prone and scripts broke when new installation contexts were introduced (eg, zones)
  - Too much installer-specific logic required
  - Required better tools

Design Assertions

- All packaging data belongs in packages
- Meta-data must be extensible
- No install-time scripting, will use software *self-assembly* whenever possible
- Package dependencies must define safe surfaces (combinations of package versions)
- No difference between packages, patches or upgrades
- Will use ZFS snapshots & clones

Packaging basics

- A package is a minimisation boundary
- A package has a version
- Only one version of a package may be installed at a time
- A package is defined by a manifest
- A manifest is a collection of actions
- A package is named by an FMRI*:*

  pkg:///{publisher}/{package name}@{version}

pkg://opensolaris.org/package/pkg@0.5.11,5.11-0.149:20100924T020949Z

Actions

- How IPS delivers content and metadata: causes things to occur during installation
- 12 different actions at present
  - file, directory, link, hardlink, license and user, group, set, depend, driver, signature, legacy
- Each action takes a set of attributes containing meta-data
  - file 14d...ec group=bin mode=0555 \
    owner=root path=usr/bin/ls variant.arch=i386
  - dir group=bin mode=0755 owner=root path=usr/bin
  - user gcos-field="pkg(5) server UID" group=pkg5srv \
    uid=97 username=pkg5srv

Dependency Types

- How a package expresses constraints
- depend fmri=pkg:/foo@0.5.11,5.11-0.149 type={type}
  - require: the package must be present (at a version ≥ the version given, if specified)
  - optional*:* the package, if present must be at a version ≥ the version specified. Omitting the version does nothing.
  - exclude*:* the package installed must be at a version < the specified version. If the version has been specified, the package must not be present.
  - incorporate: constrains the package to be within the version specified at that precision
    - eg. pkg:/foo@2 allows 2, 2.1, 2.99.99, but not 1.99 or 3.0

      (groups collections of packages, allowing them to upgrade in lock-step)

Dependency Types

- depend fmri=pkg:/foo@0.5.11,5.11-0.149 type={type}
  - conditional: the listed package must be installed if the package marked by the 'predicate' attribute is installed (version rules as per 'require')
  - require-any: any one of the listed packages must be installed (version rules as per 'require')
  - origin: the given package must have already been installed prior to the package operation installing this action (version rules as per 'require')
- We're considering other dependency types (eg. to help group packages)

Installation / Upgrade

- We install packages to images, either live or non-live full root images, linked-images, user-images
- Installing a newer version of an already installed package is an *upgrade* operation.
  - Actions that are different in the new version of the package are installed
  - Unchanged actions are left alone
  - Any actions that have disappeared from the new manifest are deleted
- Download sizes are determined by amount of change
- Downgrading is limited: use boot environments

Implementing side effects safely – actuators

- An action tag to trigger side-effects when modifying live-images, deferring those to boot for non-live images
  - file 5c2...33ebd group=sys mode=0444 owner=root \
    path=var/svc/manifest/application/pkg-server.xml \
    restart_fmri=svc:/system/manifest-import:default
- Actuators are tied to actions – if the action doesn't change, the actuator doesn't fire
- Actuators with the same name & value are combined and only fire once

Handling diversity – variants and facets

- Variant tags allow a single package to deliver different content depending on image properties
  - eg. variant.arch with current values i386 and sparc
- Facet tags are similar, but used to deliver optional content
- Facets have a hierarchical namespace
  - eg.

facet.locale.*	false
facet.locale.fr	true
facet.locale.fr_FR	true

Matches facet.locale.fr_FR, facet.locale.fr but not facet.locale.fr_CA

Command line tools

- Client
  - pkg(1) packagemanager(1)
- Developer
  - pkgfmt(1), pkgdepend(1), pkgdiff(1), pkglint(1), pkgmogrify(1)
- Publisher
  - pkgsend(1), pkgrecv(1), pkg.depotd(1M)
- Related tools
  - Installadm(1), distro_const(1)

Hardware and Software
Engineered to Work Together

15


ORACLE®