Solaris 11 Developer Webinar Series
Webinar #7

Publishing IPS Packages – Guide for Developers

Eric Reid
Oracle Systems ISV Engineering

Brock Pytlik
Oracle Solaris Engineering
## Solaris 11 for Developers
### Webinar Series

<table>
<thead>
<tr>
<th>Webinar Series Topic</th>
<th>Date</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modern Software Packaging for Enterprise Developers</td>
<td>03-27-12 @ 9am PT</td>
<td>Eric Reid</td>
</tr>
<tr>
<td>Simplify Your Development Environment with Zones, ZFS &amp; More</td>
<td>04-10-12 @ 9am PT</td>
<td>Eric Reid &amp; Stefan Schneider</td>
</tr>
<tr>
<td>Managing Application Services – Using SMF Manifests in Solaris 11</td>
<td>04-24-12 @ 9am PT</td>
<td>Matthew Hosanee</td>
</tr>
<tr>
<td>Optimize Your Applications on Solaris 11: The DTrace Advantage</td>
<td>05-08-12 @ 9am PT</td>
<td>Angelo Rajadurai</td>
</tr>
<tr>
<td>Maximize Application Performance and Reliability on Solaris 11</td>
<td>05-22-12 @ 9am PT</td>
<td>Vijay Taktar</td>
</tr>
<tr>
<td>Writing Solaris 11 Device Drivers</td>
<td>06-05-12 @ 9am PT</td>
<td>Bill Knoche</td>
</tr>
<tr>
<td>Publishing IPS Packages</td>
<td>06-19-12 @ 9am PT</td>
<td>Eric Reid &amp; Brock Pytlik</td>
</tr>
<tr>
<td>The World of IPS, Part 2: Tools and Commands</td>
<td>07-17-12 @ 9am PT</td>
<td>Eric Reid &amp; Brock Pytlik</td>
</tr>
<tr>
<td>Oracle Solaris Remote Lab</td>
<td>07-31-12 @ 9am PT</td>
<td>Ron Larson &amp; Angelo Rajadurai</td>
</tr>
</tbody>
</table>
Eric Reid
Oracle Systems ISV Engineering

• 24-year Sun/Oracle employee

• Has worked with SunOS/Solaris/OpenSolaris since 1985

• Home-based employee for Sun/Oracle since 1996

• Oracle Systems ISV Engineering works closely with our most important Systems and Solaris partners
Brock Pytlik
Oracle Solaris Engineering

• Started at Oracle in 2008
• Has worked on IPS for the last 4 years.
What You’ll Learn Today

• IPS Concepts and Terms
• IPS Developer Tools and Commands
• Architectural Considerations when developing IPS Packages
• Creating a Simple IPS Package
• Publishing an IPS Package

Assumption for this Webinar: some knowledge of IPS concepts and consumer-level commands (\texttt{pkg(1)}, etc.)

Next webinar (17 July): Advanced IPS topics, including how to implement package ‘scripting’
Best Resource for IPS Package Developers

• IPS technology page


• Whitepaper: Packaging and Delivering Software with the Image Packaging System: A developer's guide

http://hub.opensolaris.org/bin/download/Project+pkg/files/ipsdevguide.pdf

• Oracle University Course (Instructor-led or Virtual): Developing and Deploying Applications on Oracle Solaris 11

Why Should Developers Care About IPS?

- Supported, consistent installation mechanism
  - ...with tools!
- Dependency checking
  - Automatically integrated with OS and other 3rd party packages
- Transparent multi-architecture support
- Fast
- Secure
- Zone support
- Full Solaris Boot Environment support
- Patch Update support
- Standalone or repository (network) install
- One-click install from the web
- Integrated with Solaris Automated Installer
- Integrated with Oracle Enterprise Manager Ops Center
IPS Concepts and Terms - 1

- **Action**
  - The atomic units of software delivery into IPS
  - Types: file, directory, link, set, driver, depend, license, user, legacy, signature

- **Manifest**
  - A file which describes a collection of Actions

- **Package**
  - A set of Actions to be taken upon installation, upgrade or removal; described by a Manifest

- **Repository**
  - A collection of Packages, accessible via HTTP or local filesystem

- **Publisher**
  - An entity that develops and constructs Packages
IPS Concepts and Terms - 2

- **Variant**
  - Mechanism to allow a package to be used in multiple situations (multiple CPU architecture; debug/non-debug bits; Global Zone bits vs non-Global Zone bits)

- **Facet**
  - Mechanism to create optional portions of packages (for locales, man pages, etc.)

- **Actuator**
  - Tag applied to any action that causes a system change when that action is installed, removed or updated

- **Package FMRI**
  - Unique Fault Management Resource Identifier
  - Example:

    pkg://solaris/system/library@0.5.11,5.11-0.175.0.0.0.2.1:20111019T082311Z
IPS Concepts and Terms - 3

• For Developers:
  • Actions + Manifest ➔ Package
  • Packages ➔ Repository
  • Packages ➔ Package Archive file (.p5p)
  • Multiple versions of Packages can coexist in Repositories

• For Consumers (target systems):
  • Only one version of a package may be installed at a time
  • Accessing Packages via Repositories:
    • Consumer System subscribes to one or more Publisher’s Repositories
    • Packages are identified by use of pkg:// FMRI
  • Installing Package Archive files:
    • Outside the bounds of Repositories
IPS Developer Tools and Commands - 1 (pkg:/package/pkg)

- `pkgsend(1)`: Publish/update packages
  - `generate` subcommand
    - Create manifest from files/paths/SVR4 package
  - `publish` subcommand
    - Publish package

- `pkgmogrify(1)`: Programmatically edit manifests

- `pkgdepend(1)`: Manage package dependencies
  - `generate` subcommand
    - Generate dependencies for a package by examining package content
  - `resolve` subcommand
    - Resolves package dependencies by examining other packages
IPS Developer Tools and Commands - 2
(pkg:/package/pkg)

- pkgmerge(1): Create multi-variant packages
- pkglint(1): ‘Sanity check’ one or more package manifests
- pkgsign(1): Cryptographically sign a package
- pkgrepo(1): Create and manage packages repositories
- pkgfmt(1): Make more human-readable manifests
Package Dependencies

- Dependencies are not optional!
- Require: Causes another package to be installed
- Require-any: Installs one of a set of packages
- Conditional: If one package is installed, install a second
- Group: A require that a user can override
- Optional: Establishes a minimum version
- Origin: Establishes a minimum version in the current image
- Incorporate: Constrain the version of another package
- Parent: Incorporate enforced on non-global zones
- Exclude: Prevent another package from being installed
General guidelines

• Fixes or features (any change) means a new version of a package

• Package obsoletion – rarely a good idea
  • Upgrade is prevented if any packages depend on the package
  • Usually a better idea to let the package continue to exist

• Moving files between packages
  • Use an optional dependency to ensure that two packages don’t think they both own a path at the same time.
  • If it’s an editable file, set the original_name attribute

• Renaming packages – sometimes necessary
  • Old names persist and are mapped accordingly
How to distribute packages

- Network-based package repositories
  - Simplest for customers
  - Useful for diverse customer base
  - Put apache in front of pkg.depotd
- File-based package repositories
  - Typically faster than network-based repositories
  - Useful for internal deployment
- Package archives
  - Useful for disconnected customers
  - Require customers to retrieve archives manually
Creating a Simple IPS Package

Planning

• The package creation and publication ‘assembly line’
  1. Lay out your package as you want it – the proto area
  2. Create initial manifest from proto area with pkgsend generate
     • Add package name, facets, actuators, description, summary, …
     • Modify file layout, permissions, owners …
  4. Evaluate packages dependencies: pkgdepend generate on Step 3 manifest
  5. Resolve packages dependencies pkgdepend resolve on all Step 4 manifests
  6. Verify: pkglint Step 5 manifests to catch any errors
  7. (Optional) Merge: pkgmerge the package from different repositories to create multi-variant packages
Creating a Simple IPS Package Example

- We wish to create a package called *mysoftware*, which delivers files under `/opt`.
- This package consists of a library, binary, and optional man page.
- The library and the binary both depend on the system `libc.so.1` library.
- The *mysoftware* package will deliver the following files to the system:
  - `/opt/mysoftware/lib/mylib.so.1`
  - `/opt/mysoftware/bin/mycmd`
  - `/opt/mysoftware/man/man1/mycmd.1`
Creating a Simple IPS Package
‘Assembly Line’

1. Lay out your package as you want it – the proto area

We’ll be working in /proto as a non-privileged user

/path/to/your/package/lib/mylib.so.1
/path/to/your/package/bin/mycmd
/path/to/your/package/man/man1/mycmd.1
Creating a Simple IPS Package
‘Assembly Line’

2. Create initial manifest from proto area with pkgsend generate

We use pkgfmt to make things more readable.

```
$ pkgsend generate proto | pkgfmt > mypkg.p5m.gen
```

This first intermediate file looks like this:

```
dir path=opt group=bin mode=0755 owner=root
dir path=opt/mysoftware group=bin mode=0755 owner=root
dir path=opt/mysoftware/bin group=bin mode=0755 owner=root
dir path=opt/mysoftware/lib group=bin mode=0755 owner=root
dir path=opt/mysoftware/man group=bin mode=0755 owner=root
dir path=opt/mysoftware/man/man group=bin mode=0755 owner=root
file opt/mysoftware/bin/mycmd path=opt/mysoftware/bin/mycmd group=bin \mode=0755 owner=root
file opt/mysoftware/lib/mylib.so.1 path=opt/mysoftware/lib/mylib.so.1 \group=bin mode=0644 owner=root
file opt/mysoftware/man/man1/mymodule path=opt/mysoftware/man/man1/mymodule.1 \group=bin mode=0644 owner=root
```
Creating a Simple IPS Package
‘Assembly Line’

3. Generate required metadata: `pkgmogrify` Step 2

We create `mypkg.mog` with metadata to be integrated into the manifest:

```plaintext
set name=pkg.fmri value=mypkg@1.0,5.11-0
set name=pkg.summary value="This is our example package"
set name=pkg.description value="This is a full description of all the interesting attributes of this example package."
set name=variant.arch value=${ARCH}
set name=info.classification value=org.opensolaris.category.2008:Applications/Accessories
link path=usr/share/man/index.d/mysoftware target=opt/mysoftware/man
<transform dir path=opt$->drop>
<transform dir file link hardlink path=opt/.+/man(/.+)? -> 
default facet.doc.man true>
```

We then use `pkgmogrify` to pull it all together:

```bash
$ pkgmogrify -DARCH=`uname -p` mypkg.p5m.gen mypkg.mog | pkgfmt > mypkg.p5m.mog
```
Creating a Simple IPS Package
‘Assembly Line’

4. Evaluate packages dependencies: `pkgdepend generate` on Step 3 manifest

5. Resolve packages dependencies `pkgdepend resolve` on all Step 4 manifests

```
$ pkgdepend generate -md proto mypkg.p5m.mog > mypkg.p5m.dep
$ pkgdepend resolve -m mypkg.p5m.dep
```
Creating a Simple IPS Package
‘Assembly Line’

6. Verify: `pkglint` manifests to catch any errors

```
$ pkglint mypkg.p5m.4.res
Lint engine setup...
Starting lint run...
```

7. (Optional): Merge: `pkgmerge` the package from different repositories to create multi-variant packages.

Create a package that:

- Installs on both Sparc and x86
- Includes debug and non-debug bits
- Works in both the global zone and non-global zone
Publishing a Simple IPS Package

Deployment

• For development, publishing to a private file repository is best
• For deployment, the choice of the type of repository depends on business and customer needs
Publishing a Simple IPS Package
To local file-based repository

Create the repo under /scratch

```
$ pkgrepo create /scratch/my-repository
$ pkgrepo -s /scratch/my-repository set publisher/prefix=mypublisher
$ find /scratch/my-repository/
/scratch/my-repository/
/scratch/my-repository/pkg5.repository
```

Publish the package to the repo, then examine the repo

```
$ pkgsend -s /scratch/my-repository/ publish -d proto mypkg.p5m.4.res
pkg://mypublisher/mypkg@1.0,5.11-0:20120619T034303Z
PUBLISHED
$ pkgrepo -s /scratch/my-repository info
PUBLISHER   PACKAGES STATUS           UPDATED
mypublisher 1        online           2012-06-13T03:43:04.117536Z
```

If desired, this repo can also be served over the network using
pkg.depotd(1M). Use an apache proxy if this repository will have much
load.
Publishing a Simple IPS Package
Testing the package - 1

As a user with Software Installation privileges (or root) configure the publisher

```
$ sudo pkg set-publisher -p /scratch/my-repository
pkg set-publisher:
  Added publisher(s): mypublisher
```

Install the package:

```
$ sudo pkg install mypkg
  Packages to install: 1
  Create boot environment: No
Create backup boot environment: No
```

<table>
<thead>
<tr>
<th>DOWNLOAD</th>
<th>PKGS</th>
<th>FILES</th>
<th>XFER (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed</td>
<td>1/1</td>
<td>3/3</td>
<td>0.7/0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PHASE</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Phase</td>
<td>15/15</td>
</tr>
<tr>
<td>Package State Update Phase</td>
<td>1/1</td>
</tr>
<tr>
<td>Image State Update Phase</td>
<td>2/2</td>
</tr>
<tr>
<td>Reading Existing Index</td>
<td>8/8</td>
</tr>
<tr>
<td>Indexing Packages</td>
<td>1/1</td>
</tr>
</tbody>
</table>
Publishing a Simple IPS Package
Testing the package - 2

Check that the files expected are present and that the metadata is correct.

```
$ find /opt/mysoftware/
/opt/mysoftware/
/opt/mysoftware/bin
/opt/mysoftware/bin/mycmd
/opt/mysoftware/lib
/opt/mysoftware/lib/mylib.so.1
/opt/mysoftware/man
  :

$ pkg info mypkg
    Name: mypkg
    Summary: This is our example package
    Description: This is a full description of all the interesting attributes of this example package.
    Category: Applications/Accessories
    State: Installed
    Publisher: mypublisher
    Version: 1.0
  :
```
Resources for Solaris 11 Developers

Recorded Sessions

<table>
<thead>
<tr>
<th>Webinar Series Topic</th>
<th>Date</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing Oracle Solaris 11 Device Drivers</td>
<td>06-05-12 @ 9am PT</td>
<td>Bill Knoche (Principal Software Engineer)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximize Application Performance and Reliability with Oracle Solaris Studio</td>
<td>05-22-12 @ 9am PT</td>
<td>Vijay Tatkar (Senior Manager Software Development)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimize Your Applications on Oracle Solaris 11: The DTrace Advantage</td>
<td>05-06-12 @ 9am PT</td>
<td>Angelo Rajadurai (Principal Software Engineer)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managing Application Services - Using SMF Manifests in Oracle Solaris 11</td>
<td>04-24-12 @ 9am PT</td>
<td>Matthew Hosanee (Principal Software Engineer)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simplify Your Development Environment with Zones, ZFS &amp; More</td>
<td>04-10-12 @ 9am PT</td>
<td>Eric Reid (Principal Software Engineer)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modern Software Packaging for Enterprise Developers</td>
<td>03-27-12 @ 9am PT</td>
<td>Eric Reid (Principal Software Engineer)</td>
</tr>
<tr>
<td>Replay - Slides</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Today’s Webinar

Developing IPS Packages

Registration:

Agenda

- Click on event to register
- All webinars on Tuesday’s 9-10am PT (Event will support VOIP)

Next Sessions

<table>
<thead>
<tr>
<th>Webinar Series Topic</th>
<th>Date</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishing IPS Packages</td>
<td>06-19-12 @ 9am PT</td>
<td>Eric Reid (Principal Software Engineer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brock Pyttik (Senior Software Engineer)</td>
</tr>
<tr>
<td>The World of IPS 2: Tools and Commands</td>
<td>07-17-12 @ 9am PT</td>
<td>Eric Reid (Principal Software Engineer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brock Pyttik (Senior Software Engineer)</td>
</tr>
<tr>
<td>Oracle Solaris Remote Lab</td>
<td>07-31-12 @ 9am PT</td>
<td>Ron Larson (Project Manager, ISV Engineering)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Angelo Rajadurai (Principal Software Engineer)</td>
</tr>
</tbody>
</table>
Questions
Hardware and Software
Engineered to Work Together
## Publishing a Simple IPS Package

### Alternative: To Package Archive (.p5p)

```bash
$ pkgrecv -s /scratch/my-repository -a -d myarchive.p5p mypkg
Retrieving packages for publisher mypublisher ...
Retrieving and evaluating 1 package(s)...

<table>
<thead>
<tr>
<th>DOWNLOAD</th>
<th>PKGS</th>
<th>FILES</th>
<th>XFER (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed</td>
<td>1/1</td>
<td>3/3</td>
<td>0.7/0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARCHIVE</th>
<th>FILES</th>
<th>STORE (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>myarchive.p5p</td>
<td>14/14</td>
<td>0.7/0.7</td>
</tr>
</tbody>
</table>
Tips and tricks: pkgsend

- Make your build produce a proto area which mirrors how the software should be installed.
- Use -T with files where timestamps matter
  - .py files generated .pyc files if the .py file is older than .pyc files
- Use --target if your proto area has hardlinks for reproducible packaging operations.
Tips and tricks: pkgdepend resolve

- Use -R to decouple build machine from packaging results
  - Resolve usually uses resolves against installed packages
  - -S is an option, but usually leads to unresolved dependencies
  - Create standard reference images and resolve against them
    - pkgdepend -R <path to image> resolve...