ORACLE

SOLARIS

Solaris 11 Developer Webinar Series
Webinar #8

Scripting and Other Advanced IPS Topics

Eric Reid
Oracle Systems ISV Engineering

Brock Pytlik
Oracle Solaris Engineering

Solaris 11 for Developers Webinar Series

ORACLE SOLARIS

Webinar Series Topic	Date	Speaker
Modern Software Packaging for Enterprise Developers	03-27-12 @ 9am PT	Eric Reid
Simplify Your Development Environment with Zones, ZFS & More	04-10-12 @ 9am PT	Eric Reid & Stefan Schneider
Managing Application Services – Using SMF Manifests in Solaris 11	04-24-12 @ 9am PT	Matthew Hosanee
Optimize Your Applications on Solaris 11: The DTrace Advantage	05-08-12 @ 9am PT	Angelo Rajadurai
Maximize Application Performance and Reliability on Solaris 11	05-22-12 @ 9am PT	Vijay Taktar
Writing Solaris 11 Device Drivers	06-05-12 @ 9am PT	Bill Knoche
Publishing IPS Packages	06-19-12 @ 9am PT	Eric Reid & Brock Pytlik
Scripting and Other Advanced IPS Topics	07-17-12 @ 9am PT	Eric Reid & Brock Pytlik
Oracle Solaris Remote Lab	08-14-12 @ 9am PT	Ron Larson & Angelo Rajadurai

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

Yes, I enjoy digital photography…

Brock Pytlik
Oracle Solaris Engineering

- Started at Oracle in 2008
- Has worked on IPS for the last 4 years.

What You'll Learn Today

- Implementing scripting functionality in concert with IPS packages using SMF
- Moving from SVR4 packaging to IPS packaging
- Setting up a production IPS repository
- IPS Tips and Tricks

Assumption for this Webinar: Knowledge of IPS concepts and packaging creation tools and techniques, and some knowledge of Service Management Framework

Reminder - IPS Resources for Developers

- IPS technology page
  http://www.oracle.com/technetwork/server-storage/solaris11/technologies/ips-323421.html

- Whitepaper: *Packaging and Delivering Software with the Image Packaging System: A developer's guide:*
  http://timsfoster.files.wordpress.com/2011/10/book.pdf

- Oracle University Course (Instructor-led or Virtual): *Developing and Deploying Applications on Oracle Solaris 11*
  http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D73486GC10

Scripting and IPS

Scripting and IPS Overview

- IPS introduces the concept of **software self-assembly** – any package should be able to build itself into a working configuration
- Other package formats rely on included scripts to update configuration files at install/modify/remove time
  - Drawback – such scripts can do almost anything, *opaque to the invoking framework* (Stephen Hahn, 2007)
- Moving to IPS forces the developer to rethink this configuration approach
- Solaris Service Management Framework (SMF) can be leveraged implement this configuration functionality

Scripting and IPS
Implementing Software Self-Assembly

- Making the software responsible for its own configuration
- Key concepts:
  - IPS **action**: atomic unit of software delivery in IPS
  - **Composition**: the concept of IPS packages delivering fragments of configuration files
  - IPS **actuators** are tags applied to any **action** that causes a side-effect
- Package developers are encouraged to use **actuators** to trigger side-effects at install, update or uninstall time

Scripting and IPS Leveraging SMF

- SMF: replacement for init.d/rc.d scripts; a framework to identify, observe and manage software **services**
- IPS package developers can create and deploy SMF **services** to implement scripting functionality
- SMF references:
  - Matt's Webinar: http://www.oracle.com/technetwork/server-storage/solaris11/documentation/smf-webinarseries-1602377.pdf
  - Documentation: http://docs.oracle.com/cd/E23824_01/html/821-1451/dzhid.html

Scripting and IPS Leveraging SMF

- IPS implements SMF-specific **actuators**:
  - `disable_fmri` – Disable the service before the packaging operation is performed
  - `refresh_fmri` – Refresh the service after packaging operation finishes
  - `restart_fmri` - Restart the service after packaging operation finishes
  - `suspend_fmri` – Temporarily suspend the service before packaging operation is performed, and re-enable the service once packaging operation is complete
- IPS also implements the `reboot-needed` actuator to require reboot when operating on a live image

Scripting and IPS
Simple Example Using IPS and SMF

- Package *myapplication* is required to configure itself once, at install time

1. Include the following actions in our IPS package manifest file:

```
file path=opt/myapplication/bin/run-once.sh owner=root group=sys mode=0755
file path=lib/svc/manifest/application/myapplication-run-once.xml owner=root
group=sys \
    mode=0644 restart_fmri=svc:/system/manifest-import:default
```

The package includes the install-time script (`run-once.sh`) which is to be run once, via an SMF service

Scripting and IPS
Simple Example Using IPS and SMF

2. The script run-once.sh simply runs and writes a simple log message:

```
#!/usr/bin/sh
. /lib/svc/share/smf_include.sh
assembled=$(/usr/bin/svcprop -p config/assembled $SMF_FMRI)
if [ "$assembled" == "true" ] ; then
    exit $SMF_EXIT_OK
fi
svccfg -s $SMF_FMRI setprop config/assembled = true
svccfg -s $SMF_FMRI refresh
echo "This is output from our run-once method script"
```

Notes:

- Script checks to see if it's already been run (i.e. package has been assembled)
- When testing the package, always a good idea to run pkg verify before and after installing the package

Scripting and IPS
Simple Example Using IPS and SMF

3. The `run-once` SMF service manifest:

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type='manifest' name='MyApplication:run-once'>
<service
    name='application/myapplication/run-once'
    type='service'
    version='1'>
    <single_instance />
    <dependency
        name='fs-local'
        grouping='require_all'
        restart_on='none'
        type='service'>
            <service_fmri value='svc:/system/filesystem/local:default' />
    </dependency>
    <dependent
            name='myapplication_self-assembly-complete'
            grouping='optional_all'
            restart_on='none'>
            <service_fmri value='svc:/milestone/self-assembly-complete' />
    </dependent>
...
```

Scripting and IPS
Simple Example Using IPS and SMF

```
...
    <instance enabled='true' name='default'>
            <exec_method
                type='method'
                name='start'
                exec='/opt/myapplication/bin/run-once.sh'
                timeout_seconds='0'/>
            <exec_method
                type='method'
                name='stop'
                exec=':true'
                timeout_seconds='0'/>
            <property_group name='startd' type='framework'>
                <propval name='duration' type='astring' value='transient' />
            </property_group>
            <property_group name='config' type='application'>
                <propval name='assembled' type='boolean' value='false' />
            </property_group>
    </instance>
</service>
</service_bundle>
```

Scripting and IPS
Hints: Writing SMF methods for Self-Assembly

- Timestamps
  - SMF service method script can use configuration file timestamps: compare packaged config file timestamp to unconfigured file timestamp to determine if config file needs to be recompiled – it's more efficient
- Timeouts
  - Depending on circumstances, developers might want to impose a finite `timeout_seconds` on their self-assembly processes, enabling SMF to drop the service to `maintenance` if something goes wrong. This can assist the developer when debugging.

```
...
            <exec_method
                type='method'
                name='start'
                exec='/opt/myapplication/bin/run-once.sh'
                timeout_seconds='360'/>
...
```

Moving From SVR4 to IPS Packages

Moving from SVR4 to IPS Packages Considerations

- First decision – convert or start with fresh proto area
  - Is there information that would be lost or would be hard to reproduce?
- Same basic package creation steps as outlined in the Creating IPS Packages webinar
  - http://www.oracle.com/technetwork/server-storage/solaris11/documentation/ips-packages-webinarseries-1666681.pdf
- SVR4 metadata can provide start of IPS manifest
- Pre-/Post- scripts in SVR4 package?
  - Actuators/SMF Services (previous section)
  - Previous example could have been done by reading directory of config files
  - Make software capable of detecting if it's being run the first time

Converting a Simple SVR4 Package to IPS Example From First Webinar

- We wish to convert an SVR4 package called *MYSWmysoftware,* 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 *MYSWmysoftware* 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
```

Converting a Simple SVR4 Package to IPS 'Assembly Line' Differences

1. Don't need to explicitly create a proto area, can get the metadata and files from an existing SVR4 package (datastream or filesystem format)
2. Create initial manifest from the SVR4 package with pkgsend generate

```
$ pkgsend generate ./MYSWmypkg| pkgfmt > mypkg.p5m.gen
```

Note: pkgsend is smart enough to see and flag pre- and post- scripts in an SVR4 package:

```
$ pkgsend generate ./MYSWmyscriptedpkg| pkgfmt > myscriptedpkg.p5m.gen
pkgsend generate: ERROR: script present in MYSWmyscriptedpkg: postinstall
```

Converting a Simple SVR4 Package to IPS Other Considerations

- While it is possible to install SVR4 packages directly on a system running IPS, it's strongly discouraged
  - Apart from the `legacy` **action**, there are no links between the SVR4 and IPS packaging systems, and they do not reference package metadata from each other
  - SVR4 packages can cause errors when running `pkg verify`
  - Any IPS operation may modify or remove files delivered by an SVR4 package

Setting Up A Production IPS Repository

Setting Up A Production IPS Repository Planning

- Oracle Doc Reference:
  - http://docs.oracle.com/cd/E23824_01/html/E21803/toc.html
- File-based repository access isn't recommended for internet-facing repositories
- Requirements:
  - OS: Solaris 11
  - Sufficient free disk space (ZFS filesystem recommended)
- pkg/server provides simple HTTP server, which may not support enough simultaneous connections

Setting Up A Production IPS Repository The Process

1. Prepare the System
   - Give repo its own ZFS file system
2. Populate the Repository
   - From packages in existing repo, package archives or newly-created packages
3. Start the Repository Service
   - Repos implemented as SMF service (`pkg/server`)
   - Can have multiple repo service instances
4. Stand up an Apache HTTP Server in front of Repository service

Setting Up A Production IPS Repository 1: Prepare the System

```
S11# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
rpool                   75.2G   108G  5.00G  /rpool
rpool/ROOT              23.0G   108G    31K  legacy
rpool/ROOT/solaris      44.8G   108G  3.52G  /
rpool/dump              1.97G   108G  1.97G  -
rpool/export            43.0G   108G  30.5G  /export
...

# Create a ZFS file system for the package repository in the root pool:

S11# zfs create rpool/export/prodRepo
S11# zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
rpool                        75.2G   108G  5.00G  /rpool
rpool/export/repoSolaris11     31K   108G    31K  /export/prodRepo
...

# Tip - For better performance when updating the repository, set atime to
off.

S11# zfs set atime=off rpool/export/prodRepo
```

Setting Up A Production IPS Repository
2a: Populate from HTTP-based Repository

```
S11# pkgrepo create /export/localRepo

# We're making a local copy of the Solaris repo
S11# pkgrecv -s http://pkg.oracle.com/solaris/release:80/ -d /export/localRepo'*'
Processing packages for publisher solaris...
Creating Plan
Retrieving and evaluating 4288 package(s)...
PROCESS                     ITEMS       GET (MB)         SEND (MB)
developer/build/cmake    446/4288    332.1/4589.7    1000.2/14511.8
...
Completed               4288/4288   4589.7/4589.7   14511.8/14511.8

# Build your search index
S11# pkgrepo -s /export/localRepo refresh
Initiating repository refresh.
```

Setting Up A Production IPS Repository
2b: Populate from file-based Repository

```
S11# pkgrepo create /export/prodRepo

# Assume your local repo is mounted on /mnt
S11# df -k /mnt
Filesystem                      1024-blocks  Used  Available Capacity  Mounted on
/root/myRepo.iso                6778178  6778178          0   100%    /mnt

# Use 'rsync' or 'tar' to copy everything over
S11# rsync -aP /mnt/repo/ /export/prodRepo

# Check your work
S11# ls /export/prodRepo
pkg5.repository            README
publisher
S11# df -k /export/prodRepo
Filesystem                1024-blocks      Used  Available Capacity  Mounted on
rpool/export/prodRepo     191987712  13733450   75787939    16%    /export/prodRepo

# Build your search index
S11# pkgrepo -s /export/prodRepo refresh
Initiating repository refresh.
```

Setting Up A Production IPS Repository 3: Starting the Repository Service

```
S11# svccfg -s application/pkg/server setprop pkg/inst_root=/export/prodRepo
S11# svccfg -s application/pkg/server setprop pkg/readonly=true

# Check your work
S11# svcprop -p pkg/inst_root application/pkg/server
/export/prodRepo

# If you want to use a port other than 80
S11# svccfg -s application/pkg/server setprop pkg/port=<port_number>

# Start the pkg.depotd repository service.
S11# svcadm refresh application/pkg/server
S11# svcadm enable application/pkg/server

# To check whether the repository server is working, open a browser window on # the
localhost location. By default, pkg.depotd listens for connections on # port 80. If
you have changed the port, open a browser window on the
# localhost:port_number location.
```

Setting Up A Production IPS Repository 4: Standing Up an Apache HTTP Server

- Not mandatory, but provides several benefits:
  - Performance, via content caching and load balancing
  - Ability to host multiple repos on one domain name
  - Access logging, 'vanity' URLs, etc.
- Apache HTTP Server 2.2 is a click away:

```
S11# pkg search apache-22
INDEX      ACTION VALUE                          PACKAGE
pkg.fmri   set    solaris/web/server/apache-22 pkg:/web/server/
apache-22@2.2.20-0.175.0.0.0.2.537
```

Setting Up A Production IPS Repository 4: Standing Up an Apache HTTP Server

- Apache caching
  - Generic caching settings
  - Caching Catalog Attributes file
  - Caching for Search
- Generic HTTP server settings
- Multiple Repos in single domain
  - `pkg/server` service capable of multiple instances

Setting Up A Production IPS Repository 4: Standing Up an Apache HTTP Server

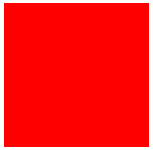
- Simple Example: Configure service

```
# We create a new instance of the service (assuming default already in use)

S11# svccfg -s pkg/server add repo
S11# svccfg -s pkg/server:repo addpg pkg application
S11# svccfg -s pkg/server:repo "setprop pkg/proxy_base = astring: http://
pkg.example.com/myrepo"
S11# svcadm refresh pkg/server:repo
S11# svcadm enable pkg/server:repo

# The pkg(5) client opens 20 parallel connections to the depot server when
# performing network operations. Make sure the number of depot threads matches
# the expected connections to the server at any given time. Use the following
# commands to set the number of threads per depot:

S11# svccfg -s pkg/server:repo "setprop pkg/threads = 200"
S11# svcadm refresh pkg/server:repo
S11# svcadm restart pkg/server:repo
```

Setting Up A Production IPS Repository 4: Standing Up an Apache HTTP Server

- `httpd.conf` modifications for simple single-repo setup, with recommended generic settings
  - Connect http://pkg.example.com/myrepo (external repo URL) to http://internal.example.com:1000 (internal repo URL)

```
...
# Reduces over-the-wire size of package metadata (catalogs, manifests) – up to 90%
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain
# Since packages can include forward slashes, we don't want Adobe interpreting
AllowEncodedSlashes NoDecode
# Allow clients to make more pipelined requests without closing the connection
MaxKeepAliveRequests 10000
# Allow for repo searches with very large number of results – 30 secs should be OK
ProxyTimeout 30
# Make sure proxy forwarding is disabled
ProxyRequests Off

# Redirect external requests to repo server
ProxyRequests OffRedirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ http://internal.example.com:10000 nocanon max=200
...
```



IPS Tips and Tricks

Tips and Tricks: Publishing via Package Archive (.p5p)

- Package as self-contained package archive file
- Useful when transferring packages to other machines which cannot access your package repositories
- Can be set as sources of local publishers
- Creating Package Archive:

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

- Installing Package from .p5p file:

```
$ pkg install –g ./myarchive.p5p mypkg
Retrieving packages for publisher mypublisher …
Retrieving and evaluating 1 package(s)…
...
```

Tips and tricks: pkgsend

- Make your build produce a proto area which mirrors how the software should be installed
  - If you cannot, use `pkgmogrify` to change the paths of actions
- Use -T with files where timestamps matter
  - .py files generate .pyc files if the .py file is newer 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 ...`

Tips and Tricks
Merging Multiple-Architecture Packages

Q: *When creating packages for SPARC and x86, is it best to merge them into one package and how to I do this? If the packages are built on different machines, how are they combined?*

A: Given: SPARC version of foo@1 has been published into a repository at `/SPARC-repo`; the x86 version into a repository at `/x86-repo`. To merge into a single package foo@1 in `/merged`:

```
S11# pkgmerge -d /merged -s arch=sparc,/sparc-repo -s arch=i386,/x86-repo
```

The `pkgmerge` man page has more specific info and several more complicated examples.

Tips and Tricks
Package Versioning

Q: *I note that all package names have 5.11-0 in the version. Is "5.11" always static? Can I modify "-0" ?*

A: Example of the `entire` package:

```
entire@0.5.11,5.11-0.175.1.0.0.19.0:20120625T193029Z
```

0.5.11: Can be any version. We generally recommend having this reflect the external version of software. So if you were packaging foo version 2.7, we'd suggest starting with foo@2.7.

- **Exception**: co-existing multiple versions of the software installed on the system at the same time; then use foo-271@2.7.1, etc.

5.11: Used by Oracle as "the OS version"; don't modify for now

0.175.1.0.0.19.0: Oracle's "packaging version space". You're free to define whatever scheme you want to have here. Make it simpler if you wish

Timestamp (created automatically)

Resources for Solaris 11 Developers

Who should attend?

Application developers and administrators wanting a deep-dive on key features of Oracle Solaris 11 which you can exploit to make your applications superior to your competitors and easier to use.

Please note: (after registering) You will receive details on how to attend the meeting in a separate confirmation e-mail shortly before the webcast.

Agenda - Next Sessions

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

Webinar Series Topic	Date	Speaker
Scripting and Other Advanced IPS Topics ✔	07-17-12 @ 9am PT	Eric Reid (Principal Software Engineer) and Brock Pytlik (Senior Software Engineer
Oracle Solaris Remote Lab	08-14-12 @ 9am PT	Ron Larson (Project Manager, ISV Engineering) and Angelo Rajadurai (Principal Software Engineer)

NEXT Webinar

Recorded Sessions

Webinar Series Topic	Date	Speaker
Publishing IPS Packages Replay - Slides ✔	06-19-12 @ 9am PT	Eric Reid (Principal Software Engineer) and Brock Pytlik (Senior Software Engineer
Writing Oracle Solaris 11 Device Drivers Replay - Slides ✔	06-05-12 @ 9am PT	Bill Knoche (Principal Software Engineer)
Maximize Application Performance and Reliability with Oracle Solaris Studio ✔ Replay - Slides	05-22-12 @ 9am PT	Vijay Tatkar (Senior Manager Software Development)
Optimize Your Applications on Oracle Solaris 11: The DTrace Advantage ✔ Replay - Slides	05-08-12 @ 9am PT	Angelo Rajadurai (Principal Software Engineer)
Managing Application Services - Using SMF Manifests in Oracle Solaris 11 ✔ Replay - Slides	04-24-12 @ 9am PT	Matthew Hosanee (Principal Software Engineer)
Simplify Your Development Environement with Zones, ZFS & More Replay - Slides ✔	04-10-12 @ 9am PT	Eric Reid (Principal Software Engineer) and Stefan Schneider (Chief Technologist ISV-Engineering)
Modern Software Packaging for Enterprise Developers ✔ Replay - Slides	03-27-12 @ 9am PT	Eric Reid (Principal Software Engineer)

Registration:
http://www.oracle.com/technetwork/server-storage/solaris11/overview/webinar-series-1563626.html

Questions

Hardware and Software
Engineered to Work Together