# Service Management Facility (5)

**Bob Netherton**
Principal Engineer, Solaris Adoption
Sun Microsystems
http://blogs.sun.com/bobn

# Agenda

- Motivation for SMF
- Core concepts and terminology
- Administrative Commands
- Migrating a Legacy RC service
- New Boot Process
- Troubleshooting/Recovery
- Resources

# Learning Goals

By the end of this module you should ..

- Have an understanding of the core concepts and commands that comprise SMF
- Use commands to enable, disable, restart or recover a service.
- Boot a system using SMF
- Manage a legacy service
- Migrate a service to SMF
- Restore a corrupted SMF database

# SMF First Impression

- Solaris has a wealth of innovations
  - Most of these features are optional, valuable – but optional
- You cannot avoid SMF
  - You experience it on first boot (loading service definitions)
  - /etc/init.d is significantly smaller than in Solaris 9
  - Where did all of those service scripts go ?
- Legacy methods still work
  - Sequencing is very interesting as you migrate services
- Argh!!! - Why did you do all this to me ?

# SMF First Impression

- SMF seems more complicated than it really is
- Migrate a legacy RC service to be immersed
  - The first one seems hard
  - The second one is a snap
- SMF rocks!

# What is a Service ?

Definition:

A long lived software object with a well-defined state, error boundary, definition of start and stop, and relationship to other services. A service is often critical to operation of system or fulfillment of business objectives.

- A collection of processes (aka daemons)
- A set of configuration files
- Management utilities

# How are services started today ?

init(1M) via rc scripts in /etc/init.d

- Long time running or one time initializations

inetd(1M) as defined by inetd.conf(4)

- Short-lived to provide transient network functions

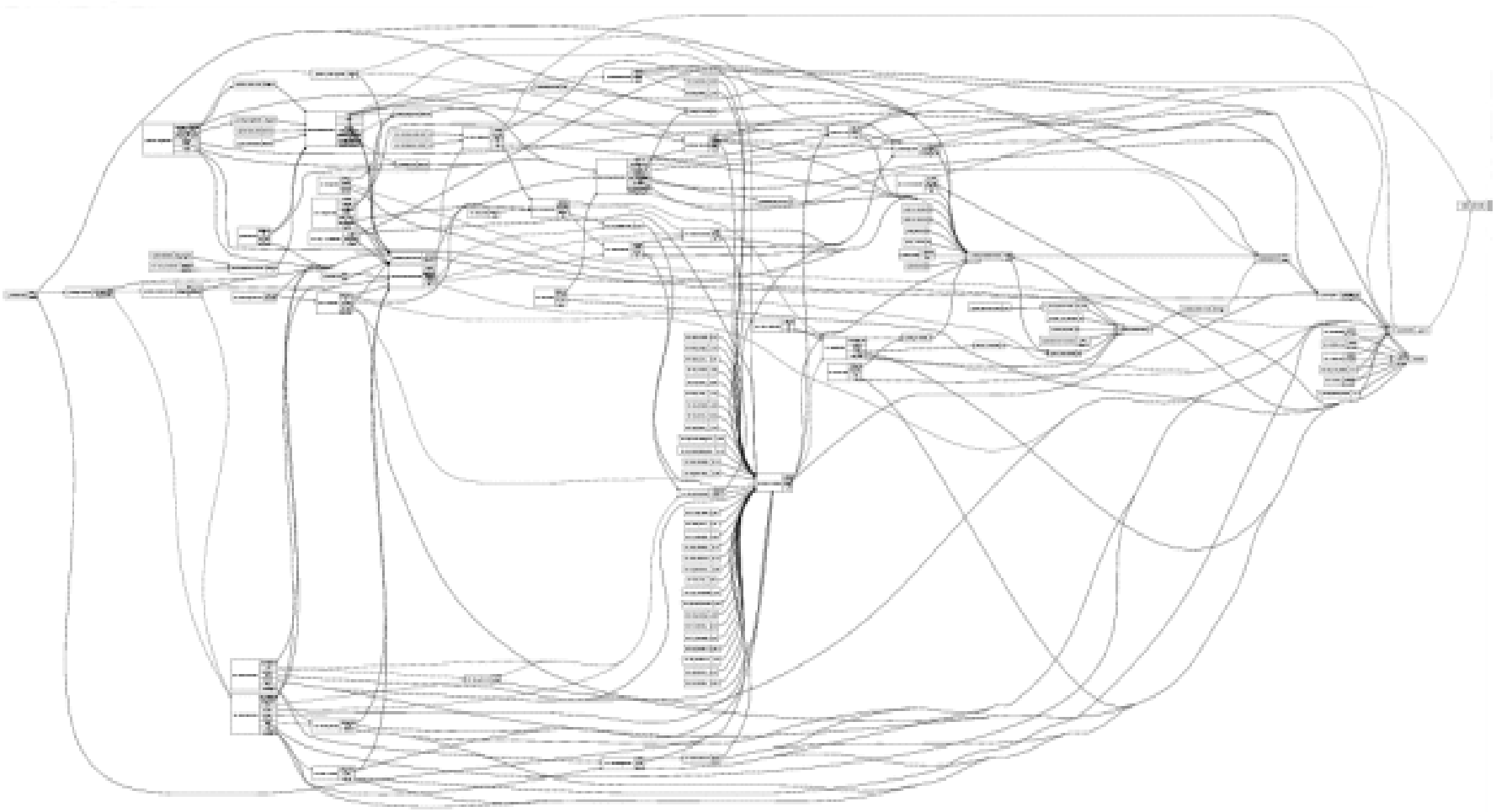
init(1M) as defined by /etc/inittab

- Restartable or one time functions

# What about ...

- Dependencies
- Detailed status
- Services or dependencies that may span multiple hosts (a grid, et al)
- Restarting services (correctly)
- Multiple instances of a service

# Solaris Dependencies today



# Service Management Today

- Multiple administration techniques
- Inconsistent dependencies, often unknown
- Lack of well defined error boundaries
- Service Oriented Architectures (SOA) require a more robust definition of services
- A new system  is required

# Service Management in Solaris 10

- A service is now a first class object that can be managed and observed
- All services have a common framework
- Legacy mechanisms still work
- Automated restart of services in correct order:
  - administrative error
  - software bug
  - uncorrectable hardware error
- Parallel startup improves system boot time

# Service Management in Solaris 10

- Easy access to information about misconfigured/ misbehaving services
  - Easy to script!
- Admins can get a meaningful system view
- Changes persist across upgrades and patches
- Securely delegate tasks to non-root users
- Snapshots and repository backup taken automatically: restore, undo

# Predictive Self-Healing

Solaris Fault Manager (FMA)

- Solaris Fault Manager provides detection and diagnosis of errors, leading to isolation or deactivation of faulty components and precise, accurate administrative messaging.

Solaris Service Manager (SMF)

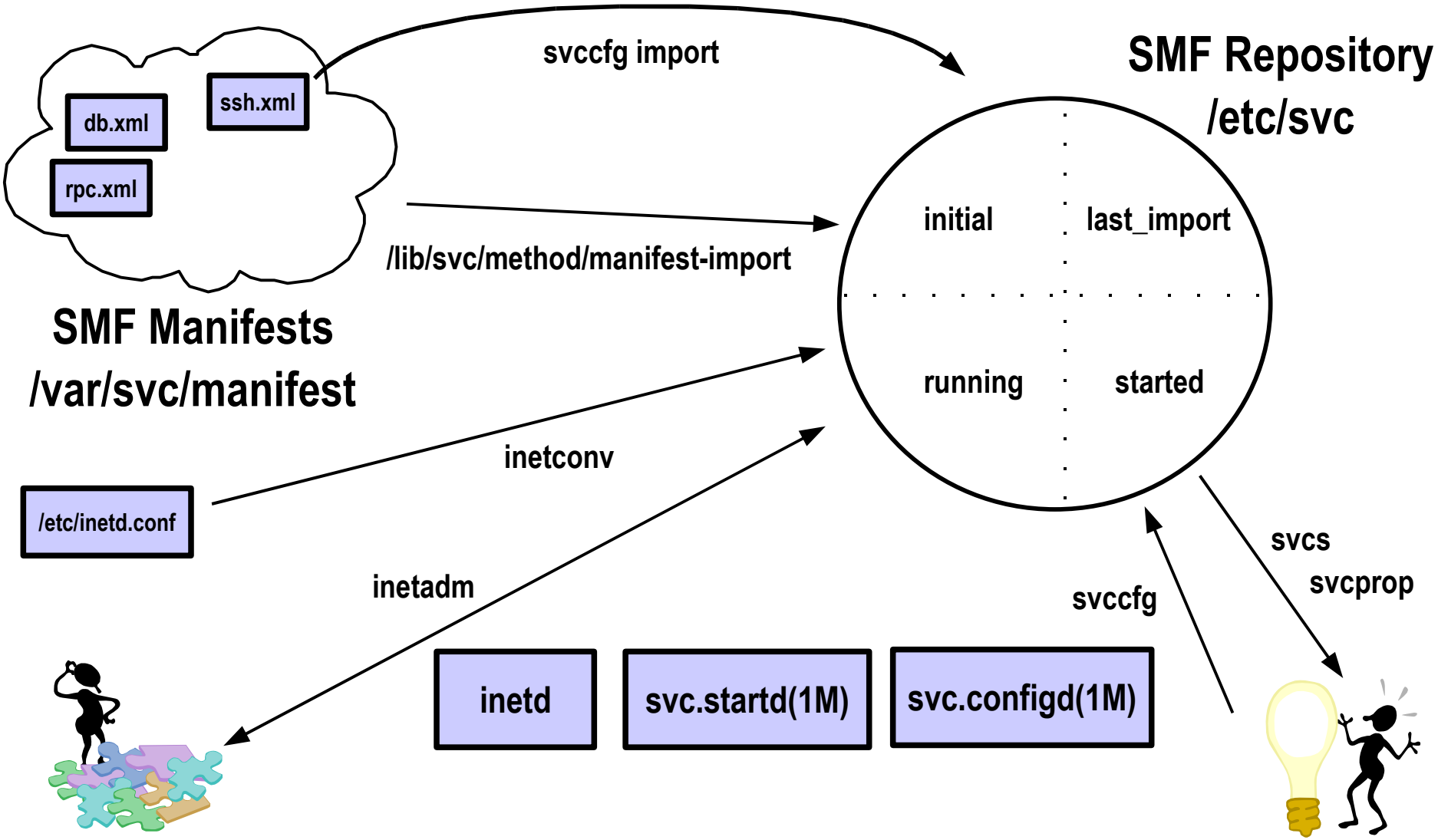
- smf(5) makes Solaris services self-healing.

Hardware faults which previously caused system restart are now isolated to the affected services. Services are also automatically restarted in the face of hardware and software faults.

# SMF Core Concepts

- SMF service definition
- SMF identifiers
- Service States
- SMF Manifests
- Service Configuration Repository
- SMF snapshots
- SMF compatibility

# Solaris SMF Components



# SMF Service

- Definition

  One or more daemons and/or configurations previously configured, started or stopped through System V RC scripts, inetd or /etc/inittab

- A service may have multiple instances
  - A web service
  - A web server (httpd)
  - IPv4 and IPv6

# Service Names in Solaris 10
## FMRI – Fault Management Resource ID

svc://localhost/network/login:rlogin

# SMF Identifiers
## naming and syntax


Scheme:

svc – SMF managed service

lrc – legacy RC script

# SMF Identifiers
## naming and syntax

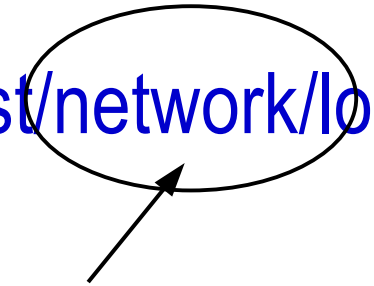

Location:

localhost

<hostname> in future release

# SMF Identifiers
## naming and syntax

svc://localhost/network/login:rlogin



Functional category:

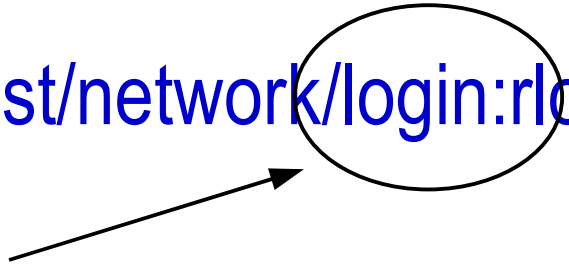
- application
- system
- device
- network
- milestone
- platform
- site

# SMF Functional Categories

- Application – traditional daemon
- Device – useful for dependencies
- Milestone – similar to SVR4 run levels
- Network – inetd converted services
- Platform – platform specific services
- System – platform independent system services
- Site – reserved for a future use

# SMF Identifiers
## naming and syntax

svc://localhost/network/login:rlogin

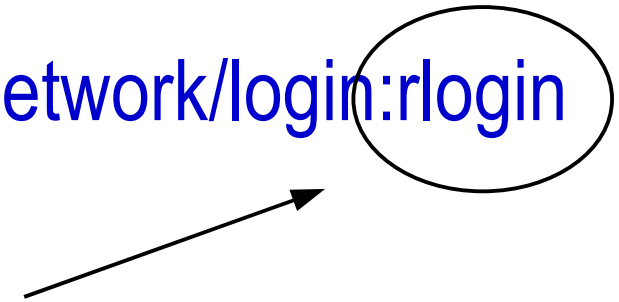


Description:

Relate to method or RC script

# SMF Identifiers
## naming and syntax

svc://localhost/network/login:rlogin



Instance:

default is the ..... default instance

# SMF Identifiers
## aliases and examples

svc://localhost/network/login:rlogin

svc:/network/login:rlogin

network/login:rlogin

rlogin

svc://localhost/system/system-log:default

svc:/system/system-log:default

svc:/platform/i86pc/kdmconfig:default

# Basic Commands

svcs(1) – detailed state information about all service instances

svcadm(1M) – common service management tasks (enable, disable, ...)

svccfg(1M) – manipulate the SMF repository

svcprop(1) – view SMF repository data

inetadm(1M) – view and configure inetd managed services

inetconv(1M) – convert an inetd.conf(4) to an SMF manifest

# Service States

online – The service instance is enabled and has successfully started.

offline – The service instance is enabled, but the service is not yet running or available to run.

disabled – The service instance is not enabled and is not running.

maintenance – The service instance has encountered an error that must be resolved by the administrator.

legacy_run – The legacy service is not managed by SMF, but the service can be observed. This state is only used by legacy services.

# Service States (less common)

degraded – The service instance is enabled, but is running at a limited capacity.

uninitialized – This state is the initial state for all services before their configuration has been read.

offline* - service is transitioning from the offline state by running the start method.

online* - service is transitioning from the online state by running the stop or refresh method

# Service States

```
% svcs -a  (edited for brevity)
 STATE          STIME    FMRI
 legacy_run     Oct_26   lrc:/etc/rcS_d/S50sk98sol
 legacy_run     Oct_26   lrc:/etc/rc2_d/S10lur
 legacy_run     Oct_26   lrc:/etc/rc3_d/S90samba
 disabled       Oct_26   svc:/system/metainit:default
 disabled       Oct_26   svc:/network/rpc/nisplus:default
 online         Oct_26   svc:/system/svc/restarter:default
 online         Oct_26   svc:/milestone/name-services:default
 online         Oct_26   svc:/platform/i86pc/eeprom:default
 online         Oct_26   svc:/system/filesystem/minimal:default
 online         Oct_26   svc:/network/physical:default
 online         Oct_26   svc:/milestone/single-user:default
 online         Oct_26   svc:/network/rpc-100083_1/rpc_tcp:tcp
 online         Oct_26   svc:/network/rpc/rstat:udp
 offline        Oct_26   svc:/application/print/ipp-listener:default
 offline        Oct_26   svc:/application/print/rfc1179:default
 maintenance    Oct_26   svc:/network/rpc/keyserv:default
```

# Service log files

- /var/svc/log/<fmri>.log
  - Due to file name requirements, / translated to -
  - Example: /var/svc/log/network-ssh:default.log
- Generally observable in the restarter property

  svcprop -p restarter/logfile <fmri>
  - May be absent
  - See svclog contributed script (OpenSolaris)

# Service Dependencies

- Dependencies can be to another SMF managed service or a file
- Types of dependencies
  - require_all – all must be online or degraded
  - require_any – at least one online or degraded
  - optional_all – all are online, disabled, degraded, or in maintenance
  - exclude_all – all are disabled, in maintenance, or not present (files)

# Dependency actions

- What do to when a dependent service changes state
  - > Service failure
  - > Administrative stop
  - > Refreshed due to property changes
- Controlled by restart_on attribute of dependency

| | restart_on attribute | | | |
|---|---|---|---|---|
| Reason for Service Stop | None | Error | Restart | Refresh |
| Error | No | Yes | Yes | Yes |
| Non error stop | No | No | Yes | Yes |
| Refresh | No | No | No | Yes |

# Dependency Examples

```
<dependency name='network'    grouping='require_any'    restart_on='error'    type='service'>
        <service_fmri value='svc:/milestone/network' />
</dependency>

<dependency name='nlockmgr'  grouping='require_all'   restart_on='error'   type='service'>
        <service_fmri value='svc:/network/nfs/nlockmgr' />
</dependency>

<dependency name='mapid'    grouping='optional_all'    restart_on='error'    type='service'>
       <service_fmri value='svc:/network/nfs/mapid' />
</dependency>

<dependency name='rpcbind'    grouping='require_all'    restart_on='restart'    type='service'>
        <service_fmri value='svc:/network/rpc/bind' />
</dependency>

<dependency name='keyserv'    grouping='optional_all'    restart_on='none'    type='service'>
        <service_fmri value='svc:/network/rpc/keyserv' />
</dependency>
```

# View detailed status of a service

% svcs -l <fmri>

% **svcs -l ssh**

```
fmri         svc:/network/ssh:default
name         Secure Shell
enabled      true
state        online
next_state   none
restarter    svc:/system/svc/restarter:default
contract_id  24
dependency   require_all/restart
file://localhost/etc/ssh/sshd_config (-)
dependency   require_all/none svc:/system/cryptosvc (online)
dependency   require_all/none svc:/network/loopback (online)
dependency   require_all/none svc:/system/filesystem/usr:default
(online)
```

# SMF common view of an inetd managed service

## % svcs -l network/rpc/rusers

```
svcs -l rpc/rusers
fmri          svc:/network/rpc/rusers:default
name          network user name service
enabled       true
state         online
next_state    none
state_time    Sun Apr 22 21:01:33 2007
restarter     svc:/network/inetd:default
dependency    require_all/restart svc:/network/rpc/bind (online)
```

# Who is dependent on me ?

```
% svcs -D network/physical
STATE          STIME    FMRI
disabled       20:18:22 svc:/network/dns/client:default
disabled       20:18:22 svc:/network/dns/server:bind9
disabled       20:18:22 svc:/network/dns/server:default
disabled       20:18:24 svc:/network/rpc/bootparams:default
disabled       20:18:24 svc:/network/nfs/server:default
disabled       20:21:01 svc:/network/shell:kshell
online         20:18:33 svc:/application/print/cleanup:default
online         20:20:31 svc:/system/identity:domain
online         20:20:31 svc:/system/identity:node
online         20:20:32 svc:/network/initial:default
online         20:20:33 svc:/network/nfs/status:default
online         20:20:33 svc:/network/nfs/mapid:default
online         20:20:33 svc:/milestone/single-user:default
online         20:20:33 svc:/network/nfs/nlockmgr:default
online         20:20:58 svc:/network/inetd:default
online         20:21:02 svc:/network/shell:tcp
online         20:21:02 svc:/network/shell:tcp6only
online         20:21:02 svc:/network/nfs/client:default
```

# What are my dependencies ?

## % svcs -d network/inetd

```
STATE          STIME    FMRI
disabled       20:18:22 svc:/network/inetd-upgrade:default
online         20:18:23 svc:/milestone/name-services:default
online         20:18:24 svc:/network/loopback:default
online         20:20:30 svc:/network/physical:default
online         20:20:32 svc:/network/rpc/bind:default
online         20:20:33 svc:/milestone/single-user:default
online         20:20:57 svc:/system/filesystem/local:default
```

## % svcs -l network/inetd | grep ^depend

```
dependency   require_any/error svc:/network/loopback (online)
dependency   require_all/error svc:/system/filesystem/local (online)
dependency   optional_all/error svc:/milestone/network (online)
dependency   optional_all/error svc:/network/rpc/bind (online)
dependency   optional_all/none svc:/network/inetd-upgrade (disabled)
dependency   require_all/none svc:/milestone/sysconfig (online)
   svc:/milestone/name-services (online)
```

# Lab #1 – examine a service

Let's look at my favorite service: gdm2-login

2) Is it running ?

3) What are it's dependencies

4) What services are dependent on it ?

5) Where is it's log file ?

6) Who is it's designated restarter ?

# Lab #1 – examine a service

- Is it running ?

  **# svcs gdm2-login**

```
STATE          STIME    FMRI
online         22:33:28 svc:/application/gdm2-login:default
```

- What are it's dependencies

  **# svcs -d gdm2-login**

```
STATE          STIME    FMRI
online          8:02:43 svc:/system/filesystem/local:default
online          8:02:48 svc:/system/utmp:default
```

# Lab #1 – examine a service

- What services are dependent on it ?

  # **svcs -D gdm2-login**

  No services are dependant on gdm2-login

- Where is it's log file ?

  # **svcprop -p restarter/logfile gdm2-login:default**

  No logfile property so /var/svc/log/application-gdm2-login:default.log

- Who is it's designated restarter ?

  # **svcs -l gdm2-login | grep ^restarter**

  ```
  restarter       svc:/system/svc/restarter:default
  ```

# Enable a service

- Become superuser or assume a role that includes the Service Management Profile.

  # **svcadm enable network/login:rlogin**

  # **svcs -l network/login:rlogin**

```
fmri            svc:/network/login:rlogin
name            The remote login service.
enabled         true
state           online
next_state      none
restarter       svc:/network/inetd:default
```

- Use **svcadm enable -r** to enable a service plus all of its dependents

# Temporarily Disable a Service

- Become superuser or assume a role that includes the Service Management Profile.
- Check the dependents of the service you want to disable

  # svcs -D [fmri]

  **# svcs -D network/login:rlogin**

  **# svcadm disable -t network/login:rlogin**

  **# svcs network/login:rlogin**

```
STATE    STIME  FMRI
disabled   11:17:24  svc:/network/login:rlogin
```

- Not persistent across reboots

# Permanently Disable a service

- Become superuser or assume a role that includes the Service Management Profile.
- Check the dependents of the service you want to disable

  # svcs -D [fmri]

  **# svcs -D network/login:rlogin**

  **# svcadm disable network/login:rlogin**

  **# svcs network/login:rlogin**

```
STATE    STIME  FMRI
disabled    11:17:24  svc:/network/login:rlogin
```

- Enabled dependent services will go offline

# Restart a service

- Become superuser or assume a role that includes the Service Management Profile.

  **# svcadm restart FMRI**

- The stop method is run
- Dependencies are re-evaluated
- Start method run if all dependencies are satisfied

**Legacy note:** restart is no longer passed to the service script

# Lab #2 – GDM

- Turn off dtlogin

  # **/usr/dt/bin/dtconfig -d**

  # **/usr/dt/bin/dtconfig -kill**
- Log in as root
- Enable gdm

  # **svcadm enable -t gdm2-login**
- Log in a few times and validate
- Reboot and describe what happens

# Lab #2 – Using gdm

1) What happened ?
2) What graphical login greeter is running ?
3) Why or why not ?
4) How do you restore your graphical environment ?

# Lab #2 – Using gdm

- What happened ?
  - gdm replaces dtlogin as the login manager
  - No graphical login after reboot
- Why or why not ?
  - The gdm2-login service was enabled temporarily
- How do you restore a graphical environment ?
  - dtconfig -e  or svcadm enable gdm2-login

# Restoring from maintenance

- Become superuser or assume a role that includes the Service Management Profile.
- Determine if any processes from the service are still running.

  # **svcs -p fmri**

  will output all remaining pid's associated with the service
- Terminate any leftover processes or daemons

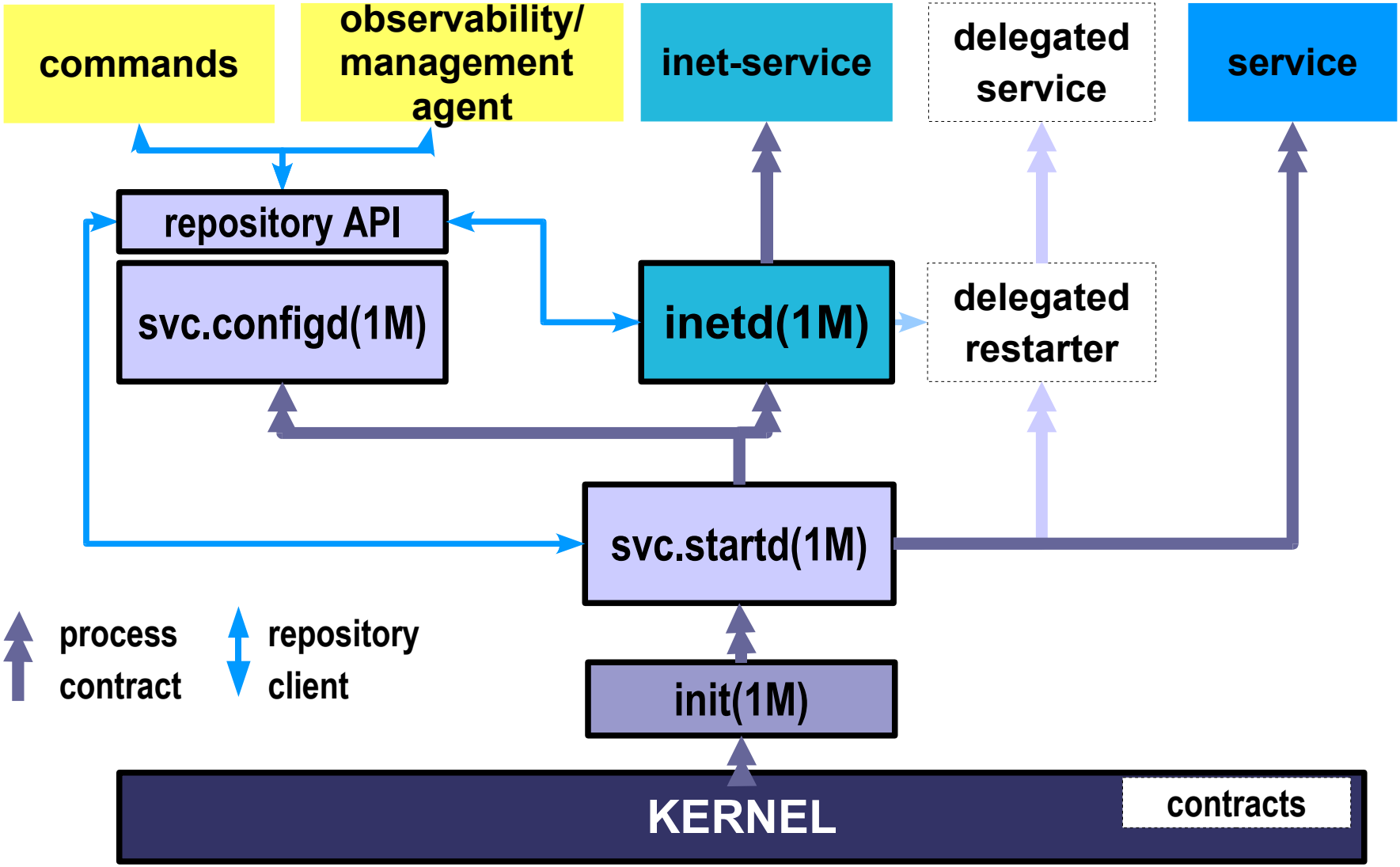
  # **kill** pid(s)
- Look at the fault code reported by **svcs -xv**
- Look in the service log file (/var/svc/log)
- Return the service to the default state

  # **svcadm clear fmri**

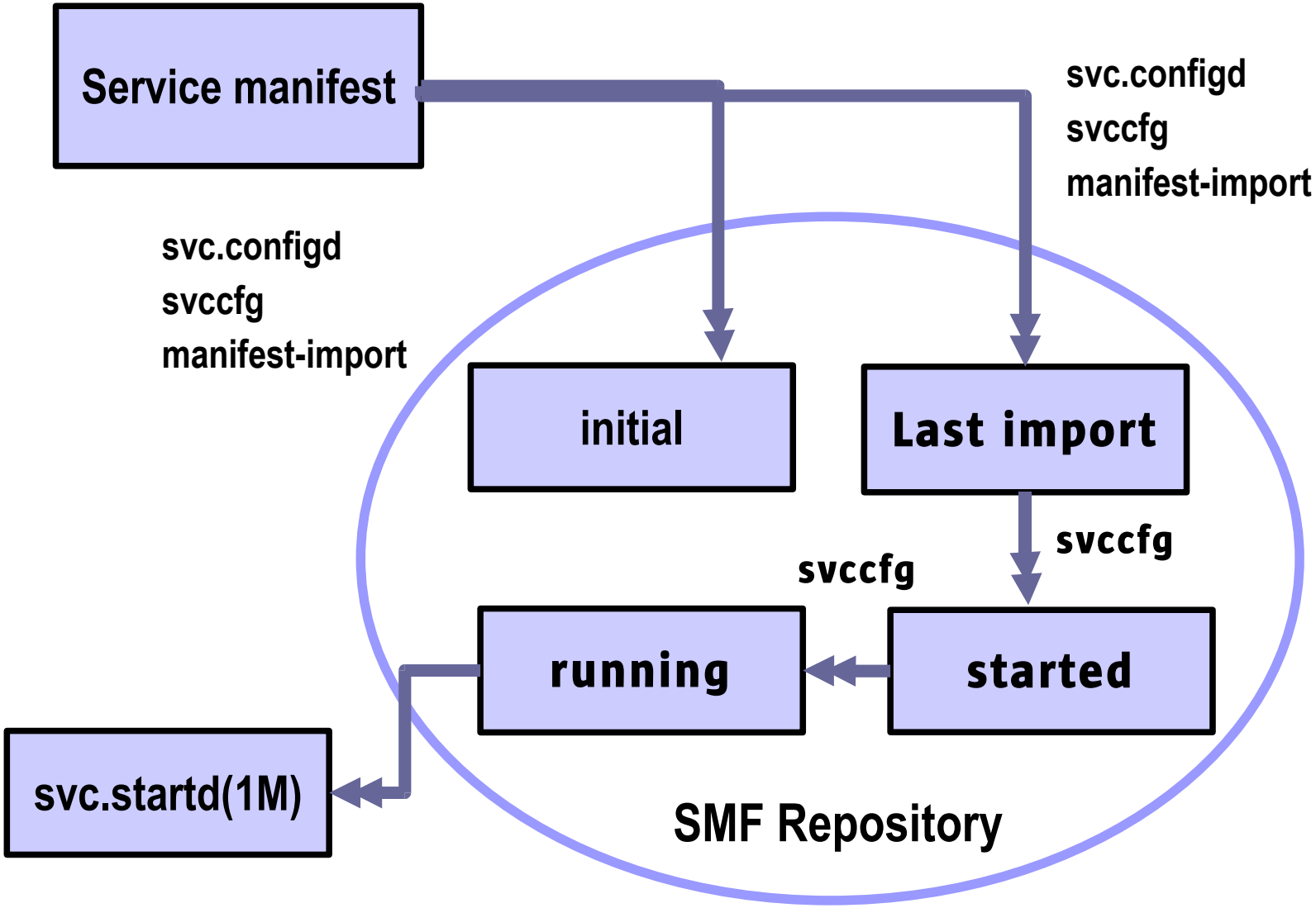
# Lab #3 – Broken Services

- Run the script /workshop/smf/lab3.sh
- Identify all of the broken services
- Why are they broken ?
- How would you fix them ?
- Run /workshop/smf/lab3-cleanup.sh to remove broken services

# Components: Architecture schematic



# Components: Manifest and Repository



# SMF Master Restarter Daemon

- /lib/svc/bin/svc.startd
- Reads the SMF repository and manages dependencies
- The master process starter and restarter
  - > Restarts services that have failed
  - > Shuts down services whose dependencies are no longer satisfied
  - > Runs legacy rc scripts at run level transitions
- Provides system view of service status

# SMF Manifests

- Located in /var/svc/manifest
- Complete XML description of a service
- Loaded into the repository at boot time
- Use svccfg(1M) to manually import a service definition (aka service bundle)
- Manifest DTD at

  /usr/share/lib/xml/dtd/service_bundle.dtd.1
- See service_bundle(4) man page
- ISVs should supply a service bundle

# SMF Manifests

```
<service_bundle type='manifest' name='SUNWzoner:zones'>

<service  name='system/zones'  type='service' version='1'>

<create_default_instance enabled='false' />
<single_instance />

<dependency name='multi-user-server' type='service' grouping='require_all' restart_on='none'>
        <service_fmri value='svc:/milestone/multi-user-server' />
</dependency>

<exec_method type='method' name='start' exec='/lib/svc/method/svc-zones %m' timeout_seconds='60'> </exec_method>
<exec_method type='method' name='stop' exec='/lib/svc/method/svc-zones %m' timeout_seconds='500'> </exec_method>

<property_group name='startd' type='framework'>
          <propval name='duration' type='astring' value='transient' />
</property_group>

<stability value='Unstable' />

<template>
      <common_name>
          <loctext xml:lang='C'>  Solaris zones </loctext>
      </common_name>

      <documentation>
          <manpage title='zones' section='5' manpath='/usr/share/man' />
          <manpage title='zoneadm' section='1M' manpath='/usr/share/man' />
      </documentation>
</template>
</service>
</service_bundle>
```




# SMF Configuration Repository

- Located in /etc/svc
- Nonglobal zones have their own repository
- Persistent configuration information as well as runtime properties for services
- Distributed among local memory (volatile) and local files (repository.db).
- May be placed in a directory (LDAP) in a future Solaris release

# SMF Repository Administration

- svccfg(1M) to modify the repository
- svcprop(1M) to view the repository
- libscf(3LIB) provides repository APIs
- svc.configd(1M) is the repository daemon
  - > Run at boot time to adjust properties
  - > Restarted upon any SMF failure

# SMF Repository Profile

- Set of service instances and enable property
- Generated by svccfg extract
- Activated by svccfg apply
- System profiles located in /var/svc/profile
- Uses
  - Copying service states between systems (DR)
  - Creating a hardened service profile for new installations

# SMF Repository Profile

```
# svccfg extract
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM
  '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='profile' name='extract'>
  <service name='system/console-login'
  type='service' version='0'>
    <instance name='default' enabled='true'/>
  </service>
  <service name='system/device/local'
  type='service' version='0'>
    <instance name='default' enabled='true'/>
  </service>
  <service name='milestone/devices' type='service'
  version='0'>
    <instance name='default' enabled='true'/>
  </service>
```

# SMF Repository Archive

- Complete set of persistent data for all service instances
  - Dump in XML format similar to manifest
  - Does not include transient properties
- Generated by svccfg archive
- Uses
  - Copying service definitions between systems
  - Auditing or revision control
  - Creating a template for a new service definition

# SMF Repository Archive

```
# svccfg archive
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM
   '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='archive' name='none'>
  <service name='system/console-login' type='service'
   version='0'>
    <create_default_instance enabled='true'/>
    <single_instance/>
    <dependency name='fs' grouping='require_all'
   restart_on='none' type='service'>
      <service_fmri
   value='svc:/system/filesystem/minimal'/>
    </dependency>
    <dependency name='identity' grouping='require_all'
   restart_on='none' type='service'>
      <service_fmri value='svc:/system/identity:node'/>
    </dependency>
    <dependency name='utmpx' grouping='require_all'
   restart_on='none' type='service'>
      <service_fmri value='svc:/system/utmp:default'/>
    </dependency>
```

# SMF snapshots

- Complete collection of properties for a service
- Historical view of a service instance
- Simplifies rollback of service configuration changes
- Created automatically or manually as needed

# SMF snapshots

- Standard snapshots
  - *initial* – taken on the first import of the manifest
  - *last_import* – taken during last import
  - *running* – properties of the currently running service instance
  - *start* – taken at the last successful start

# SMF snapshots

- Use svcprop or svccfg(1M) to view snapshot properties
- To activate a snapshot
  - > svccfg(1M) to select snapshot
  - > svcadm refresh to update svc.startd
  - > svcadm restart to start service with new property values

# Revert to a previous snapshot

```
# svccfg
   svc:>
   svc:> select system/console-login:default
   svc:/system/console-login:default> listsnap
   initial
   running
   start
   svc:/system/console-login:default> revert start
   svc:/system/console-login:default> quit
 # svcadm refresh system/console-login
 # svcadm restart system/console-login
```

# Modify a service property

- Service manifests represent factory defaults
  - Avoid modifying service manifests
  - Maintenance scripts may fail due to changes
- Use svccfg to update service properties
  - Record property changes in case of repository rebuild
- Refresh the service after any changes
  - svc.startd will re-read the repository
- May require the service to be restarted

# Example:

```
# svccfg
svc:> select system/console-login
svc:> setprop ttymon/terminal_type = "vt100"
svc:> setprop ttymon/modules = "ldterm,  ttcompat"
svc:> end
# svcadm refresh console-login
```

# Legacy RC scripts

- Identified by the lrc scheme
  Example: lrc:/etc/rcS_d/S35cacheos_sh
- Limited SMF management
- Start status only
- Executed on run level transitions after SMF managed services
- Allows third party applications to run without modification

# Concepts Review

- Each service on the system has a unique FMRI
- XML manifests are kept for each service
- Snapshots of configuration data are kept in the repository
- Legacy /etc/rc*.d scripts are still run

# Lab #3 – Modifying service properties

1. Change console login prompt to
   Do not cry for the Penguins:
2. Restore previous setting
3. Restore original Solaris 10 default (this is important!)
4. How else could you have done this ?
5. What is the most important step
6. Why ?

# Lab #3 – Modifying service properties

1. Change console login prompt to
   Don't cry for the Penguins:

```
# svccfg -s console-login setprop \
    ttymon/prompt = \"Do not cry for the Penguins:\"
# svcadm refresh console-login
```

2. Restore previous setting

```
# svccfg -s console-login:default revert last-import
# svcadm refresh console-login:default
```

3. Restore previous setting

```
# svccfg -s console-login:default revert initial
# svcadm refresh console-login:default
```

# Lab #3 – Modifying service properties

4. How else could you have restored the previous property settings ?
   - Use svccfg archive to dump the repository
   - Save the console-login service definition in a file
   - Import the console-login service definitions
5. What is the most important step
   - # **svcadm refresh console-login**
6. Why ?
   - Informs svc.startd of service property changes
   - svc.startd informs dependent services of property changes

# Perform MySQL setup

- Installation instructions at /etc/sfw/mysql/README.solaris.mysql
  - Watch out for an error on line 15 (chmod -R 770)
  - Don't link mysql.server into /etc/rc3.d

```
# /usr/sfw/bin/mysql_install_db
# groupadd mysql
# useradd -g mysql mysql
# chgrp -R mysql /var/mysql
# chmod -R 770 /var/mysql
# installf SUNWmysqlr /var/mysql d 770 root mysql
# cp /usr/sfw/share/mysql/my-medium.cnf /var/mysql/my.cnf
```

# Migrating a Legacy RC Service

- Configure service for proper operation
- Test manually, if possible
- Create a Service Manifest
- Develop or adapt RC script to SMF methods
- Install manifest and test
- Take advantage of what SMF can do
  - > Decouple large RC scripts into modular components
  - > Leverage Solaris privileges to secure service
  - > Create operator and administrator roles
  - > Apply resource management project limits

# Perform MySQL setup

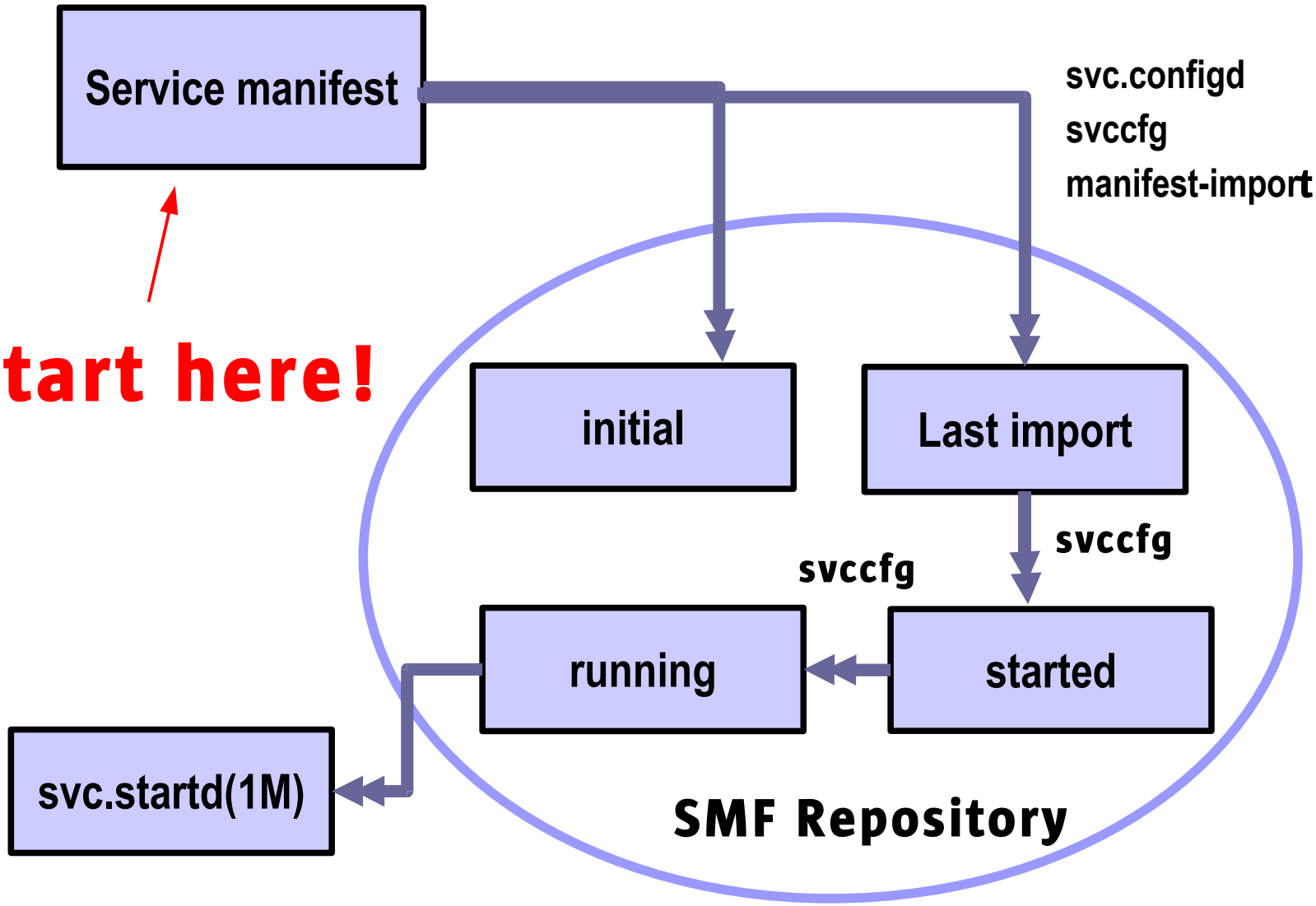
- Start database and test using mysqladmin

```
# /etc/sfw/mysql/mysql.server start
Starting mysqld daemon with databases from /var/mysql

# /usr/sfw/bin/mysqladmin status
Uptime: 32  Threads: 1  Questions: 1  Slow queries: 0  Opens: 6
Flush tables: 1  Open tables: 0  Queries per second avg: 0.031
```

- Stop the database!
  - > We don't want SMF to try to start it if it is already running

```
# /etc/sfw/mysql/mysql.server stop
/etc/sfw/mysql/mysql.server stop Killing mysqld with pid 1212
Wait for mysqld to exit.050907 21:16:19  mysqld ended

 done
```

# Components: Manifest and Repository



# Creating an SMF Manifest

- XML file describing service properties
- Located in /var/svc/manifest/<dir>
  - site is recommended for locally developed services
- DTD at /usr/share/lib/xml/dtd/service_bundle.dtd.1
- service_bundle(4) man page

*Why reinvent the wheel?*

*Copy and modify an existing service manifest!*

# Some suggested sources of manifests

- Explore /var/svc/manifest for similar services
  - system/utmp is a simple standalone daemon
  - system/coreadm is a simple configuration service, and
  - network/telnet is an inetd-managed daemon
- Initial inet service manifests can be created easily by invoking: inetconv -i <file>
- DTD is self-documenting; read it at /usr/share/lib/xml/dtd/service_bundle.dtd.1
- Read the service_bundle(4) man page

# Example: Zones Manifest

```
<service_bundle type='manifest' name='SUNWzoner:zones'>

<service  name='system/zones' type='service' version='1'>

<create_default_instance enabled='false' />
<single_instance />

<dependency name='multi-user-server' type='service' grouping='require_all' restart_on='none'>
        <service_fmri value='svc:/milestone/multi-user-server' />
</dependency>

<exec_method type='method' name='start' exec='/lib/svc/method/svc-zones %m' timeout_seconds='60'> </exec_method>
<exec_method type='method' name='stop' exec='/lib/svc/method/svc-zones %m' timeout_seconds='500'> </exec_method>

<property_group name='startd' type='framework'>
            <propval name='duration' type='astring' value='transient' />
</property_group>

<stability value='Unstable' />

<template>
    <common_name>
        <loctext xml:lang='C'>  Solaris zones </loctext>
    </common_name>

    <documentation>
        <manpage title='zones' section='5' manpath='/usr/share/man' />
        <manpage title='zoneadm' section='1M' manpath='/usr/share/man' />
    </documentation>
</template>
</service>
</service_bundle>
```

Name of the service

Default state

Dependency

Start and Stop methods

Documentation

# Step 1: Start with an existing manifest

- Copy a manifest that most closely resembles your new service
  - Put in /var/svc/manifest/application
- Change the following sections
  - <service_bundle name=”your bundle name”>
  - <service name=”your service name”>
  - <create_default_instance enabled=true | false>
  - <template> to provide text descriptions of service name and man page locations.

# Our new manifest – so far

```
<service_bundle type='manifest' name='MySQL'>

<service  name='application/mysql'  type='service' version='1'>

<create_default_instance enabled='true' />
<single_instance />

<dependency name='multi-user-server' type='service' grouping='require_all' restart_on='none'>
        <service_fmri value='svc:/milestone/multi-user-server' />
</dependency>

<exec_method type='method' name='start' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='60'> </exec_method>
<exec_method type='method' name='stop' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='120'> </exec_method>
<exec_method type='method' name='refresh' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='120'> </exec_method>

<property_group name='startd' type='framework'>
            <propval name='duration' type='astring' value='transient' />
</property_group>

<stability value='Unstable' />

<template>
    <common_name>
        <loctext xml:lang='C'>  MySQL Database </loctext>
    </common_name>

    <documentation>
        <manpage title='mysql' section='1' manpath='/usr/sfw/man' />
        <manpage title='mysqld' section='1' manpath='/usr/sfw/man' />
        <manpage title='mysqladmin' section='1' manpath='/usr/sfw/man' />
    </documentation>
</template>
</service>
</service_bundle>
```

Name of the service

Default state

Documentation references

# Step 2: Start and Stop methods

- You already have these – the legacy RC scripts
- Remove links from the rc<n>.d directories
- Leave them in /etc/init.d or move to some place such as /etc/opt/svc/method
- Modify start, stop, and possibly restart methods to point to the correct RC scripts
  - Use %m in exec method to pass type of method (“start”, “stop”, or “refresh”)
  - exec=”:kill -<signal>” is simple shortcut to replace pkill(1)s
  - Set timeout_seconds if things take a long time to complete

# Service method tokens

%% %

%r Name of the restarter, such as svc.startd

%m Name of the method, such as start or stop

%s Name of the service

%i Name of the instance

%f FMRI of the instance

%{prop[:,]} Value(s) of a property.

See the smf_method(5) manpage for more.

# Looking up service properties

```
. /lib/svc/share/smf_include.sh

if smf_present()
then
        APACHE_HOME=`svcprop -p httpd/home ${SMF_FMRI}`
        if [ $? != 0 ]; then
                APACHE_HOME=/usr/apache
                echo "Apache home property missing: assuming default
    $APACHE_HOME"
     fi
fi
```

See the smf_method(5) manpage for more.

# Service method return codes

- Include /lib/svc/share/smf_include.sh

| | |
|---|---|
| SMF_EXIT_OK | Method exited, successfully |
| SMF_EXIT_ERR_FATAL | Method failed fatally |
| SMF_EXIT_ERR_CONFIG | Unrecoverable configuration error |
| SMF_EXIT_ERR_NOSMF | Running a method outside of SMF |
| SMF_EXIT_MON_DEGRADE | Service deemed in degraded mode |
| SMF_EXIT_MON_OFFLINE | Service is non-responsive |
| SMF_EXIT_ERR_PERM | Permission denied (credentials, etc) |

See the smf_method(5) manpage for more.

# Our new manifest – so far

```
<service_bundle type='manifest' name='MySQL'>

<service  name='application/mysql'  type='service' version='1'>

<create_default_instance enabled='true' />
<single_instance />

<dependency name='multi-user-server' type='service' grouping='require_all' restart_on='none'>
        <service_fmri value='svc:/milestone/multi-user-server' />
</dependency>

<exec_method type='method' name='start' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='60'> </exec_method>
<exec_method type='method' name='stop' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='120'> </exec_method>
<exec_method type='method' name='refresh' exec='/etc/sfw/mysql/mysql.server restart' timeout_seconds='120'> </exec_method>

<property_group name='startd' type='framework'>
        <propval name='duration' type='astring' value='transient' />
</property_group>

<stability value='Unstable' />

<template>
    <common_name>
        <loctext xml:lang='C'>  MySQL Database </loctext>
    </common_name>

    <documentation>
        <manpage title='mysql' section='1' manpath='/usr/sfw/man' />
        <manpage title='mysqld' section='1' manpath='/usr/sfw/man' />
        <manpage title='mysqladmin' section='1' manpath='/usr/sfw/man' />
    </documentation>
</template>
</service>
</service_bundle>
```

Start and Stop methods

# Part 3: Dependencies

- This is where things begin to get complicated
- A dependency graph is invaluable
- Start with the obvious ones
  - Run level
  - Availability of file systems
  - Configuration files
- Then add dependencies on other services
  - Need to know the error boundaries – what happens to this service when other services go offline

# Example of a run level dependency

Example of a Run Level 3 dependency

```
<dependency
        name='multi-user-server'
        type='service'
        grouping='require_all'
        restart_on='none'>
  <service_fmri value='svc:/milestone/multi-user-server' />
  </dependency>
```

Use svc:/milestone/multi-user for Run Level 2 services

# Local filesystems dependency

```
<dependency
        name='filesystem'
        grouping='require_all'
        restart_on='none'
        type='service'>
         <service_fmri value='svc:/system/filesystem/local'/>
</dependency>
```

Be careful with this one: an error from mountall(1M) might prevent service from starting

# Local filesystems dependency (cont)

Other file system services

<service_fmri value='svc:/system/filesystem/root'/>
<service_fmri value='svc:/system/filesystem/usr'/>
<service_fmri value='svc:/system/filesystem/minimal'/>

These are just services, so use SMF to find the mount scripts to see what's really happening!

```
# svcprop -p start/exec minimal
/lib/svc/method/fs-minimal
```

# Dependency on a configuration file

```
<dependency
        name='database_configuration_file'
        type='path'
        grouping='require_all'
        restart_on='refresh'>
 <service_fmri value='file://localhost/var/mysql/my.cnf' />
 </dependency>
```

Extremely handy!!!!  How many times have you wondered why the NFS server didn't start ??????

# MySQL dependencies

- Started at Run Level 3
- After all local file systems are mounted
- Only if there is a configuration file
  - Sample configuration files can be found in /usr/sfw/share/mysql
  - Will keep MySQL from being started in a local zone where setup process hasn't been completed
- Any others ?

# Our new manifest – so far

```
<service_bundle type='manifest' name='MySQL'>

<service  name='application/mysql'  type='service' version='1'>

<create_default_instance enabled='true' />
<single_instance />

<dependency name='multi-user-server' type='service' grouping='require_all' restart_on='restart'>
        <service_fmri value='svc:/milestone/multi-user-server' />
</dependency>

<dependency    name='filesystem'    grouping='require_all'    restart_on='none'    type='service'>
        <service_fmri value='svc:/system/filesystem/local'/>
 </dependency>

  <dependency    name='database_configuration_file'    type='path'    grouping='require_all'    restart_on='refresh'>
        <service_fmri value='file://localhost/var/mysql/my.cnf' />
   </dependency>

<exec_method type='method' name='start' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='60'> </exec_method>
<exec_method type='method' name='stop' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='120'> </exec_method>
<exec_method type='method' name='restart' exec='/etc/sfw/mysql/mysql.server %m' timeout_seconds='120'> </exec_method>
```

# Transient services

- Some services should not be restarted when all processes terminate
  - > Example: one time initializations

```
<property_group name='startd' type='framework'>
                <propval name='duration' type='astring'
   value='transient' />
</property_group>
```

- Include in SMF manifest or modify later using svcadm(1M)
- Should be removed for MySQL

# Let's import our service

- Use svccfg or /lib/svc/method/manifest-import

```
# svcs mysql
svcs: Pattern 'mysql' doesn't match any instances
STATE          STIME    FMRI

# /lib/svc/method/manifest-import
Loaded 1 smf(5) service descriptions

# svcs mysql
STATE          STIME    FMRI
online         22:39:54 svc:/application/mysql:default

# mysqladmin status
Uptime: 1399  Threads: 1  Questions: 1  Slow queries: 0  Opens: 6
Flush tables: 1  Open tables: 0  Queries per second avg: 0.001
```

# Config file dependency in action

- We notice that MySQL isn't starting

```
# svcs mysql
STATE          STIME    FMRI
offline        15:17:09 svc:/application/mysql:default
```

- Using SMF we can ask why

```
# svcs -l mysql
fmri         svc:/application/mysql:default
name         MySQL Database Server
enabled      true
state        offline
next_state   none
state_time   Wed Sep 07 15:17:09 2005
logfile      /var/svc/log/application-mysql:default.log
dependency   require_all/refresh
file://localhost/var/mysql/my.cnf (absent)
```

# SMF in action

- Let's terminate the database daemon and see what happens

```
# svcs mysql
STATE          STIME    FMRI
online         22:39:54 svc:/application/mysql:default

# pkill mysqld

# svcs mysql
STATE          STIME    FMRI
online         22:45:45 svc:/application/mysql:default
```

# Next steps

- Common:
  - Trivial: create simple service manifest, convert init scripts to service methods, minimal testing
  - Add privileges for delegated administration
  - Enable resource management
  - Full restartability: split monolithic services, each separately restartable component becomes its own service
- Advanced:
  - Customized error/restart handling: avoid service restart if fault can be internally handled

# Lab #4 – Migrate an Legacy Service

Perform simple Apache 1 setup

```
# cd /etc/apache
# cp httpd.conf-example httpd.conf
Edit httpd.conf and change “Port 80” to “Port 81”
# /etc/init.d/apache start
```

Lauch a browser and try to connect to

http://localhost:81

Stop Apache

```
/etc/init.d/apache stop
```

# Lab #4 – Migrate an Legacy Service

1) Create a service manifest for Apache 1
   - Create a service property for configuration file

     hint: look at the Apache2 service manifest
   - Modify the Apache1 start method to exit with a configuration error if the configuration file missing
2) Permanently enable Apache 1 service
   - Think of three different ways to do this
3) Kill all Apache 1 processes

   What happens ?

   Where was this event logged ?
4) What happens if we also enable Apache 2 ?

# Lab #4 – Migrate an Legacy Service

1) Create a service manifest for Apache 1

```
# cd /var/svc/manifest/network
# cp http-apache2.xml http-apache1.xml
```

Edit http-apache1.xml

- change the instance name to apache1
- change the start, stop, and restart methods to call /etc/init.d/apache
- add to httpd property group

```
<propval name='config_file' type='astring'
  value='/etc/apache/httpd.conf' />
```

- change the documentation section (optional)
- change the refresh method to a restart method

# Lab #4 – Migrate an Legacy Service

2) Permanently enable Apache 1 service

- svcadm enable apache1
- svccfg -s apache1 setprop general/enabled = “true”
  svcadm refresh apache1
- <instance name='apache1' enabled='true'> in the service manifest

4) Kill all Apache 1 processes

What happens ?

The service instance will be restarted

Where was this event logged ?

/var/svc/log/network-http:apache1.log

# Lab #4 – Migrate an Legacy Service

4) What happens if we also enable Apache 2 ?

If they are configured to use different ports then you have 2 instances of the http service

If they are both using port 80 then one will be online and one will be in maintenance

# Securing SMF Services

- Authorizations
  - Can create service specific authorizations to allow users to run methods (action_authorization)
  - Service property group specific authorizations to allow users to modify (value_authorization) or create/destroy properties (modify_authorization)
- Run the service methods as a specific user with a specific set of rights (or profiles)
- Specify resource management limits automatically

# Approach to Securing SMF Services

- Create operator and administrator authorizations
  - Specify in /etc/security/auth_attr
  - Remember to create a grant role and assign to root if not a solaris.* authorization
  - Assign to a user or role in /etc/user_attr
- Create a non-privileged user to run service methods
  - File permissions become very important
  - Check start scripts, configuration files, and logs
- Determine what privileges the service needs
  - privdebug.pl (OpenSolaris security community)

# Operator and Administrator Roles

- Solaris auth
  - Mechanism used by a privileged application to restrict access or operation
- SMF specific auths
  - action_authorization – run the start or stop method
  - value_authorization – modify service properties
  - modify_authorization – create or delete property groups

```
<property_group name='general' type='framework'>
        <propval name='action_authorization' type='astring'
    value='mysql.operator' />
        <propval name='value_authorization' type='astring'
  value='mysql.administrator' />
</property_group>
```

# Creating and using MySQL auths

```
# echo “mysql.:::MySQL Database Server Auths::” >> /etc/security/auth_attr
# echo “mysql.grant:::Grant MySQL auths::” >> /etc/security/auth_attr
# echo “mysql.operator:::MySQL operator::” >> /etc/security/auth_attr
# echo “mysql.administrator:::MySQL admin::” >> /etc/security/auth_attr

# grep root /etc/user_attr
root::::auths=solaris.*,solaris.grant,mysql.*,mysql.grant;
profiles=Web Console Management,All;lock_after_retries=no

# useradd -d /tmp -A mysql.operator mysqlop
# useradd -d /tmp -A mysql.administrator,mysql.operator mysqldba

# su – mysqlop
% /usr/sbin/svcadm disable -t mysql
% /usr/sbin/svcadm disable mysql
svcadm: svc:/application/mysql:default: Permission denied.
% exit

# su - mysqldba
% /usr/sbin/svcadm disable mysql
```

# Using privdebug.pl

```
# privdebug.pl -f -v  -e "su - mysql /usr/sfw/sbin/mysqld_safe --user=mysql"
STAT TIMESTAMP          PPID   PID    PRIV                 CMD
USED 2005619300419      2211   2212   proc_taskid          su
USED 2005620883559      2211   2212   proc_setid           su
USED 2005621147993      2211   2212   proc_setid           su
USED 2005621161490      2211   2212   proc_setid           su
USED 2005621165094      2211   2212   proc_setid           su
USED 2005630560973      2211   2212   proc_exec            su
Starting mysqld daemon with databases from /var/mysql
                                  contract_event
USED 2005679230394      2211   2212   proc_fork            sh
USED 2005783110622      2212   2223   proc_exec            sh
USED 2005792812264      2211   2212   proc_fork            sh
USED 2005794010658      2212   2225   proc_exec            sh
USED 2005795756145      2212   2225   proc_exec            nohup
USED 2005797704273      2212   2225   proc_exec            nohup
NEED 2005799674735      2211   2212   file_dac_write       sh
USED 2005800708905      2211   2212   proc_fork            sh
```

# Specify credentials for methods

```
<exec_method
  type='method'
  name='start'
  exec='/etc/sfw/mysql/mysql.server %m'
  timeout_seconds='60'>

  <method_context working_directory='/var/mysql'>
    <method_credential
        user='mysql' group='mysql'
        privileges='proc_fork,proc_exec'  />
  </method_context>
</exec_method>
```

# Check File Permissions

- Start and stop methods
  - > Change permissions on /etc/sfw/mysql/mysql.server
- Database files
  - > Configuration file should be writable by mysqladmin
  - > Database files in /var/mysql are fine
- Other objects
  - > pid files in /var/tmp
  - > /var/run

# Putting this all together

```
# svcadm enable mysql
# ps -ef | grep mysql
   mysql  1649  1630   1 10:19:30 ?  0:00  /usr/sfw/sbin/mysqld
   mysql  1630     1   0 10:19:30 ?  0:00  /bin/sh /usr/sfw/sbin/mysqld_safe

# ppriv 1649 1630
1649:   /usr/sfw/sbin/mysqld --basedir=/usr/sfw
flags = <none>
        E: basic,!file_link_any,!proc_info,!proc_session
        I: basic,!file_link_any,!proc_info,!proc_session
        P: basic,!file_link_any,!proc_info,!proc_session
        L: all
1630:   /bin/sh /usr/sfw/sbin/mysqld_safe
flags = <none>
        E: basic,!file_link_any,!proc_info,!proc_session
        I: basic,!file_link_any,!proc_info,!proc_session
        P: basic,!file_link_any,!proc_info,!proc_session
        L: all
```

# Lab 5: Securing Apache

- Create an administrator and operator role for apache2
- Use privdebug.pl to determine what rights are required to start and stop apache2
- Create a set of non-root users
  - To run the apache2 service
  - To be able to start and stop apache2 (operator)
  - To be able to permanently enable or disable apache2 (administrator)
- Configure apache2 to run as a normal user

# Lab 5: Securing Apache

- Create an administrator and operator role for apache2

```
# echo “apache2.:::Apache2 Database Server Auths::” >>
  /etc/security/auth_attr
# echo “apache2.grant:::Grant Apache2 auths::” >>
  /etc/security/auth_attr
# echo “apache2.operator:::Apache2 operator::” >>
  /etc/security/auth_attr
# echo “apache2.administrator:::Apache2 admin::” >>
  /etc/security/auth_attr


# grep root /etc/user_attr
root::::auths=solaris.*,solaris.grant,mysql.*,mysql.grant;
profiles=Web Console Management,All;lock_after_retries=no


# roleadd -d /tmp -A apache2.operator webop
# roleadd -d /tmp -A apache2.administrator,apache2.operator
  webadmin
```

# Lab 5: Securing Apache2

- Add authorizations to the Apache2 manifest

```
<property_group name='general' type='framework'>
        <propval name='action_authorization'  type='astring'
            value='apache2.operator' />
        <propval name='value_authorization'
      type='astring' value='apache2.administrator' />
        <propval name='modify_authorization'
      type='astring' value='apache2.administrator' />

</property_group>
```

# Lab 5: Securing Apache

- Use privdebug.pl to determine what rights are required to start and stop apache2

```
# cp /etc/apache2/httpd.conf-example
    /etc/apache2/httpd.conf
# privdebug.pl -f -v -n httpd
# svcadm enable apache2
      proc_fork, proc_exec, net_privaddr
# svcadm disable apache2
# svcadm refresh apache2
# privdebug.pl -f -v -e “/lib/svc/method/httpd-apache2 start”
# privdebug.pl -f -v -e “/lib/svc/method/httpd-apache2 stop”
# privdebug.pl -f -v -e “/lib/svc/method/httpd-apache2 refresh”
```

# Lab 5: Securing Apache2

- Start method

```
<exec_method    type='method'    name='start'
        exec='/lib/svc/method/http-apache22 start'
                    timeout_seconds='60' >
<method_context working_directory='/var/apache2/2.2'>
<method_credential
                  user='webservd' group='webservd'
                privileges='proc_fork,proc_exec,net_privaddr' />
          </method_context>
</exec_method>
```

# Lab 5: Securing Apache2

- Stop method

```
<exec_method    type='method'    name='stop'
        exec='/lib/svc/method/http-apache22 stopt'
                    timeout_seconds='60' >
<method_context working_directory='/var/apache2/2.2'>
<method_credential
                  user='webservd' group='webservd'
                privileges='proc_fork,proc_exec,proc_session' />
          </method_context>
</exec_method>
```

# Lab 5: Securing Apache2

- Refresh method

```
<exec_method    type='method'    name='refresh'
        exec='/lib/svc/method/http-apache22 refresh'
                    timeout_seconds='60' >
<method_context working_directory='/var/apache2/2.2'>
<method_credential
                  user='webservd' group='webservd'
                privileges='proc_fork,proc_exec,proc_session' />
          </method_context>
</exec_method>
```

# Lab 5: Securing Apache2

- Fix lazy file permissions

  # chown -R webservd:webservd /var/apache2/logs

- Change the location of the pid file (httpd.conf)
- Uncomment the accept lock definition (httpd.conf)

# Boot Process

In this module we will

- define milestones
- learn how milestones relate to run levels
- boot to a specific milestone

# Milestones vs Run Levels

- Legacy run level model
  - The run level is set at the beginning of a run level transition
  - A lower run level is complete when the next run level is started
- An SMF milestone indicates the completion of a part of the boot process
  - Set the corresponding run level
  - Start the services whose dependencies are now satisfied (in parallel)
  - Start the corresponding legacy RC scripts (serially)
  - Set the milestone as online

# Boot Process

- System V RC scripts are only now run for legacy services.
- All Solaris services are started from methods in /lib/svc/method
- There are new boot milestones

  *none* – before any services are started

  *all* – all available services started
- /etc/init.d may eventually be empty

# Milestones and Run Levels

| SVR4 Run Level | SMF Milestone |
|---|---|
| - | none |
| s, S | single-user |
| 2 | multi-user |
| 3 | multi-user-server |
| - | All |

# Boot Process
## run levels and milestones

- Set the default milestone
  - **svcadm milestone -d single-user**
- Transition immediately to a milestone
  - **svcadm milestone single-user**
- What is the current milestone ?
  - **svcprop -p options_ovr/milestone \ system/svc/restarter:default**
  - If property is missing then milestone is all

# Boot Process-w/o services

ok **boot -m milestone=none**

login as root

Remount root filesystem as writeable

Perform required maintenance

Enable all services.

```
# svcadm milestone all
# exit
```

# Boot Process-review

- SMF methods replace SysV rc scripts
- There are 5 milestones – 2 new
- Boot to a milestone instead of a run level
- Use init(1M) to transition between run levels

# Lab #6 – Differences between milestones and run levels

- Boot single user
- Log in as root
- Exit the shell
- What happens ?
- Why ?
- What happens if you boot to milestone none ?
- Explain the output of the mount command at milestone none

# Lab #6 – Differences between milestones and run levels

- What happens ?
  - The current root shell exits
  - The system transitions to the default milestone
- Why ?
  - Consistent with previous versions of Solaris
- What else could you have done ?

  **svcadm milestone all**

# SMF Delegated Restarters

- Provides a mechanism to handle a class of services with common startup or shutdown requirements
- Can support different methods but provides a consistent SMF interface
- The restarter's name is stored with the service.
- Example: inetd(1M)
  - > Starts services on demand
  - > Maintains additional service configuration data
- Can you think of another delegated restarter ?

# inetd Managed Services

- /etc/inetd.conf converted to SMF manifest and repository on initial boot
- FMRI for converted inetd services
    - svc:/network/<servicename>:default
- inetconv(1M) adds new network services
    - Does not modify existing entries (no deletes)\
    - Use inetconv -f to overwrite existing manifest
    - Allows installation of third party applications that create /etc/inetd.conf entries
- Use inetadm(1M) to modify properties of inetd managed services

# Viewing inetd managed services

```
% inetadm
ENABLED   STATE          FMRI
enabled   offline        svc:/application/print/rfc1179:default
disabled  disabled       svc:/network/tname:default
enabled   online         svc:/network/security/ktkt_warn:ticotsord
enabled   online         svc:/network/telnet:default
enabled   online         svc:/network/rpc/smserver:default
disabled  disabled       svc:/network/rpc/mdcomm:tcp
disabled  disabled       svc:/network/rpc/mdcomm:tcp6
enabled   online         svc:/network/rpc/gss:ticotsord
disabled  disabled       svc:/network/time:stream
enabled   online         svc:/network/nfs/rquota:ticlts
enabled   online         svc:/network/nfs/rquota:udp
enabled   online         svc:/network/ftp:default
enabled   online         svc:/network/finger:default
disabled  disabled       svc:/network/login:eklogin
disabled  disabled       svc:/network/login:klogin
disabled  disabled       svc:/network/login:rlogin
disabled  disabled       svc:/network/rexec:tcp
disabled  disabled       svc:/network/rexec:tcp6udp6
enabled   online         svc:/network/rpc-100235_1/rpc_ticotsord:ticotsord
enabled   online         svc:/network/rpc-100083_1/rpc_tcp:tcp
enabled   online         svc:/network/rpc-100068_2-5/rpc_udp:udp
enabled   online         svc:/network/fs/tcp6:default
enabled   online         svc:/network/rpc-100424_1/rpc_ticotsord:ticotsord
#
```

# Managing inetd services

- Become superuser or assume a role that includes the Service Management Profile.
- List the properties for the specific service.

  **# inetadm -l FMRI**

- Change the property for the service.

  **# inetadm -m FMRI property-name=value**

# Example: inetd service (telnet)

```
# inetadm -l network/telnet
    SCOPE    NAME=VALUE
             name="telnet"
             endpoint_type="stream"
             proto="tcp6"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/sbin/in.telnetd"
             user="root"
    default  bind_addr=""
    default  bind_fail_max=-1
    default  bind_fail_interval=-1
    default  max_con_rate=-1
    default  max_copies=-1
    default  con_rate_offline=-1
    default  failrate_cnt=40
    default  failrate_interval=60
    default  inherit_env=TRUE
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE
```

# Example: telnet (cont)

- Change the property

**# inetadm -m network/telnet tcp_trace=TRUE**

**# inetadm -l network/telnet**

```
SCOPE    NAME=VALUE
         name="telnet"
         endpoint_type="stream"
         proto="tcp6"
         isrpc=FALSE
         wait=FALSE
         exec="/usr/sbin/in.telnetd"
         user="root"
default  bind_addr=""
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=TRUE
default  tcp_wrappers=FALSE
```

# Monitoring services

## How to Convert inetd.conf Entries

When adding 3rd part service to inetd.conf, you will need to update the inetd smf

```
# inetconv -i filename
```

eg.

```
# inetconv -i /etc/inetd.conf
```

# Basic Commands review

- Use svcs command to see info about services on system
- Use svcadm to make changes to services
- Use inetconv to convert inetd.conf to SMF manifests
- Use inetadm to view and make changes to inetd services

# Lab #7 – inetd delegated restarter

- Enable trivial file transfer daemon (tftpd)
- Change location of tftpd home directory
  - > Do this at least two different ways
  - > What is the implication of each method ?
- Disable tftpd
- Reimport the tftp manifest
- Should tftp be enabled at this point ? Why or why not ?

# Lab #7 – inetd delegated restarter

- Enable trivial file transfer daemon (tftpd)

  Edit /etc/inetd.conf and uncomment tftp line

  # **inetconv**

- Change location of tftpd home directory
  - > Method 1:

    # **inetadm -m tftp/udp6 exec="/usr/sbin/in.tftpd -s /tftpboot2"**
  - > Method 2:

    Edit /etc/inetd.conf and change the tftp line

    # **inetconv -f**
  - > What is the implication of each method ?
    - > inetadm doesn't change the service manifest, so an import as a part of a repository rebuild wouldn't include the new value

# Lab #7 – inetd delegated restarter

- Disable tftpd

  **# svcadm disable tftp/udp6**

- Reimport the tftp manifest

  **# touch /var/svc/manifest/network/tftp-udp6.xml**

  **# /lib/svc/method/manifest-import**

- Should tftp be enabled at this point ? Why or why not ?
  - > It remains disabled. Manifest importing will never override the enable service property

# Approaches to Hardening Solaris

- Package Minimization
  - Do not install packages that aren't immediately needed
  - Extra effort to install packages later when needed
  - Difficult to patch software not installed
- Harden after installation
  - Solaris Security Toolkit (JASS) is one example
  - System is installed with network services enabled
  - Late first boot processing to shut down unneeded services
  - Window of vulnerability exists during first boot

# Solaris Secure By Default

- Implemented as SMF profiles to secure first boot
  - Select the appropriate SMF profiles (generic.xml)
  - Settings will survive a repository rebuild
  - Services can be turned off or limited before they have a chance to run
  - New service properties that allow local only connections
- New jumpstart keyword (service_profile)
- netservices(1M)
  - Apply open or limited_net profiles
  - Refresh or restart services for new properties to take effect

http://opensolaris.org/os/community/security/projects/sbd

# Lab #8: Solaris Secure by Default

- Apply netservices limited to lock down network (inet) services
- Enable rlogin
- Create a site.xml from the current hardened configuration
- Replace SMF repository with seed
- Reboot
- Are ssh and rlogin both running ?  Telnet ?

# Troubleshooting and recovery

SMF contains a database that can be recovered if there is corruption

we will learn how to recover a corrupted database

# Troubleshooting and recovery

Become root or appropriate role

Stop the svc.startd daemon

```
# pstop ‘pgrep svc.startd‘
```

kill the svc.configd daemon.

```
# pkill svc.configd
```

Make sure / is writeable

```
# mount -o rw,remount /
```

# Troubleshooting and recovery

Save the current repository for debugging.

**# cp /etc/svc/repository.db /etc/svc/repository.bad**

Copy the default repository.

**# cp /lib/svc/seed/global.db /etc/svc/repository.db**

reboot

Now done by

/lib/svc/bin/restore_repository

# Lab #9 - Boot without services and recover

1) Assume that your services database has been corrupted

   dd if=/dev/zero of=/etc/svc/repository.db bs=1024 count=1000

2) A boot to the current default state will leave the system unusable

3) What do you do ?

# Lab #9 - Boot without services and recover

Boot single user

(b)oot or (i)nterpreter: **b -m milestone=none**

Restore database from snapshot or seed

Import additional manifests via svccfg

- why would you do this ?

Reboot

# Troubleshooting and recovery

```
# svcs -x
svc:/application/print/server:default (LP Print Service)
 State: disabled since Fri Oct 29 09:13:24 2004
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: lpsched(1M)
Impact: 2 services are not running.

svc:/network/rpc/keyserv:default (RPC Encryption Key Storage)
 State: maintenance since Fri Oct 29 09:13:39 2004
Reason: Start method failed repeatedly, last exited with status
   1.
   See: http://sun.com/msg/SMF-8000-KS
   See: keyserv(1M)
Impact: 0 services are not running.
```

# Lab #10 – Clean Boot

```
Select (b)oot or (i)nterpreter:
SunOS Release 5.10 Version j10_70 32-bit
Copyright 1983-2004 Sun Microsystems, Inc.  All rights reserved
Use is subject to license terms.
Hostname: pandora
checking ufs filesystems
/dev/rdsk/c0d0s7: is logging

Nov 14 20:18:56 svc.startd[7]: svc:/network/rpc/keyserv:default: Method
   /usr/sbin/keyserv" failed with status 1
Nov 14 20:18:56 svc.startd[7]: svc:/network/rpc/keyserv:default: Method
   /usr/sbin/keyserv" failed with status 1
Nov 14 20:18:56 svc.startd[7]: svc:/network/rpc/keyserv:default: Method
   /usr/sbin/keyserv" failed with status 1
[svc:/network/rpc/keyserv:default failed (see 'svcs -x' for details) ]

Nov 14 20:19:02 pandora sendmail[285]: unable to qualify my own domain name
   (localhost) --using short name

Nov 14 20:19:02 pandora sendmail[286]: unable to qualify my own domain name
   (localhost) --using short name



pandora console login:
```

# Lab #10 – Clean Boot

Your boot should be nearly as clean as this

```
Select (b)oot or (i)nterpreter:
SunOS Release 5.10 Version j10_70 32-bit
Copyright 1983-2004 Sun Microsystems, Inc.  All rights reserved
Use is subject to license terms.
Hostname: pandora
checking ufs filesystems
/dev/rdsk/c0d0s7: is logging


pandora console login:
```

# Lab #10 – Clean Boot

1) Figure out what services are complaining
2) Correct the problem or make it go away

# Lab #10 – Clean Boot

- Sendmail
  - svcs sendmail shows enabled so it was just complaining at boot time
  - Add a host.domain to your localhost in /etc/hosts
  - Or disable sendmail
- keyserv
  - svcs keyserv shows service is in maintenance mode
  - Disable keyserv or add /etc/defaultdomain

# Troubleshooting common errors

- Start with svcs -x
- Make sure MySQL isn't running from a previous step
  - Service will fail to start and go into maintenance state
- Check the service log
  - Can be found using svcprop -p restarter/logfile mysql
- Check dependencies (svcs -l mysql)

# Using SMF to diagnose common zone configuration problem

- Configured and installed local zone
- Boot the local zone
- Can zlogin to the local zone, but no network access
- First check svcs -a and look for uninitialized services
  - Telltale sign that SMF not started due to incomplete initialization
  - Finish sysidconfig
  - zlogin -C to complete identification process

# Using SMF to diagnose common zone configuration problem

```
# zoneadm -z zone1 boot
# zlogin zone1
# svcs -a
...
offline         20:58:01 svc:/system/sysidtool:system
offline         20:58:01 svc:/milestone/sysconfig:default
...
uninitialized   20:58:01 svc:/network/rpc/gss:default
uninitialized   20:58:01 svc:/application/font/stfsloader:default
uninitialized   20:58:02 svc:/application/print/rfc1179:default
uninitialized   20:58:02 svc:/application/x11/xfs:default
...
```

# Additional Resources

- OpenSolaris SMF Community
  http://opensolaris.org/os/community/smf
- Additional quickstart and developer documentation
  http://www.sun.com/bigadmin/content/selfheal/
- Solaris System Administration Guide
  http://docs.sun.com/app/docs/doc/817-1985
- Blogs:
  - http://blogs.sun.com/sch
  - http://blogs.sun.com/lianep
  - http://blogs.sun.com/jwadams
  - http://blogs.sun.com/dep
  - http://blogs.sun.com/dminer
  - http://blogs.sun.com/bobn (this workshop)

# Questions ?

# Thank you!

# Solaris 10 Workshop
# Service Management Facility

**Bob Netherton**
bob.netherton@sun.com
http://blogs.sun.com/bobn