

# Service Management Facility: Administration and Development

**Liane Praza**

Solaris Kernel Development

Sun Microsystems

# smf(5) tour

- Definitions
- Features
- Compatibility and changes
- Components
- Commands
- Restart relationships
- Troubleshooting and repair
- Service development

# Terms and Definitions

## Service

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.

## Service Management

- > service delivery
- > administrative interaction
- > management of service by system

smf(5): Service Management Facility

# Predictive Self-Healing

## Solaris Fault Manager Solaris Service Manager (`smf(5)`)

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

`smf(5)` makes Solaris services self-healing. Hardware faults which previously caused system restart are now isolated to the effected services. Services are also automatically restarted in the face of hardware and software faults.

# Features: Administering Services

- Service is a first class object
  - > meaningful *system* view
  - > manage services, not processes
- Easy access to information about misconfigured/misbehaving services
- Supported enable/disable; changes persist across upgrades and patches
- Securely delegate tasks to non-root users
- Boot messages under control of administrator
- More reliable console access

# Features: Self-Healing Services

- Services automatically restart in dependency order:
  - > Administrative error
  - > Software bug
  - > Uncorrectable hardware error
- Parallel startup
- Automatic service snapshots and repository backup:
  - > restore to known-good
  - > undo changes
  - > recover from backup
- Legacy mechanisms still work

# Features: Developing Services

- Unify service delivery
- Single API for management, configuration, and observation
- Enable service-based resource management
- Simplify debugging of boot process

# Compatibility

- Existing SysV init scripts (`/etc/rc?.d`) will just work unless they
  - > rely on being run before a Solaris-provided script or effect Solaris-provided infrastructure services
  - > require input from console during boot (strongly discouraged)
- Legacy services visible in smf
- Documented `/etc/init.d` scripts work
- Customer or ISV additions to `/etc/inittab` continue to work
- No public configuration files absorbed except...

# Changes: inetd.conf

- `inetd.conf` is no longer primary configuration source
- Most Solaris-delivered `inetd(1M)` services have been converted to `smf(5)`
- Remaining entries auto-converted during upgrade by `inetconv(1M)`
- Admin must run `inetconv(1M)` manually after creating entries or adding packages which do so
- Warning messages during boot if unconverted entries exist

# Changes: boot

- boot is quieter by default

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: demobox
NIS domain name is testlab.example.com
checking ufs filesystems
/dev/rdisk/clt0d0s7: is logging.
demobox console login:
```

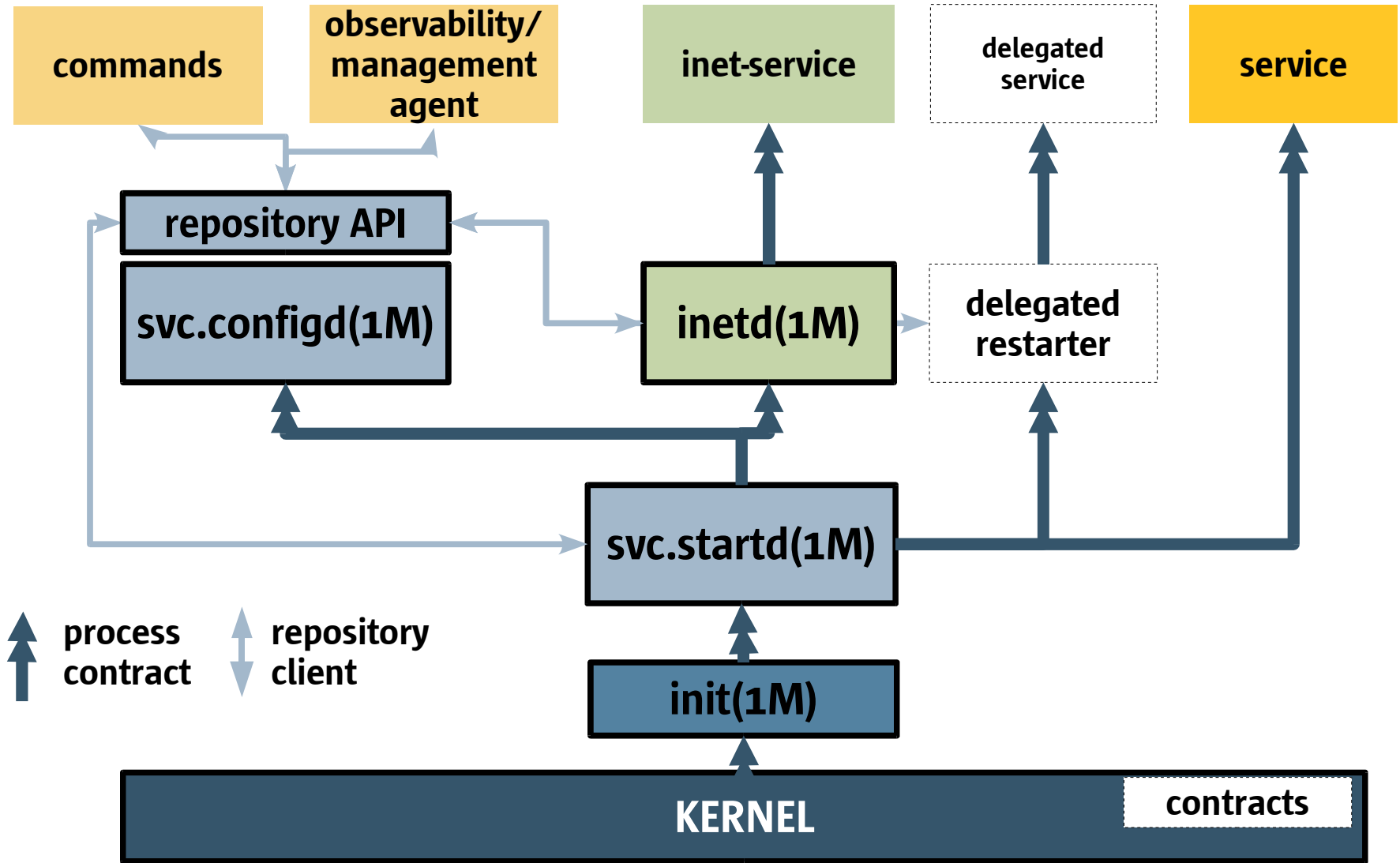
- messages logged to files
- verbose boot available (boot -m verbose)

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
[ network/loopback:default starting (loopback network interface) ]
[ system/filesystem/root:default starting (root file system mount) ]
[ network/pfil:default starting (packet filter) ]
[ network/physical:default starting (physical network interfaces) ]
...
```

# Changes

- Processes refuse to die
  - > Service processes restarted automatically
  - > Use `svcadm(1M)` to keep a service from being restarted
- `/etc/init.d` and `/etc/rc*.d` directories, as well as the `/etc/inittab` file, are quite a bit emptier than in previous releases of Solaris

# Components: Architecture schematic



# Components: Service

- Consistent specification of a software object on a Solaris system
- Stored in the `smf(5)` repository
- Properties divided into property groups
- Includes
  - > name
  - > localized descriptions and doc references
  - > dependencies
  - > method specifications (start, stop, etc.)
  - > restarter
  - > service-specific startup properties

# Components: Service Instance

- Running view of a service
- Inherits configuration from service unless explicitly overridden
- One service may have many instances (e.g. Web server on multiple ports)
- Has a well-defined state [*uninitialized, offline, online, degraded, disabled, or maintenance*] and a well-defined error boundary [process contract]
- Includes non-persistent property groups
- Named by an FMRI

# Components: FMRI

## Fault-Managed Resource Identifier

- Category and service  
`svc:/network/smtp:sendmail`
- Property group and property  
`svc:/network/smtp:sendmail/:properties/general/enabled`
- Commands take abbreviations: 'svc:/' inserted automatically, 'globbed' on instance or service name
- Shell globbing also supported in commands; remember to escape meta-characters if your shell requires it

# Components: Service Manifest

- XML description of a service, or set of services
- Delivery mechanism for service descriptions
- Located in `/var/svc/manifest`
- Automatically imported into repository on install, upgrade, boot (by `svc:/system/manifest-import`), and `pkgadd(1M)`
- Ignored once imported
- Sun-delivered manifests NOT for modification  
Customizations should occur in repository, or as a distinct service/instance.

# utmpd(1M) service description

```

<service name='system/utmp' type='service' version='1'>
  <create_default_instance enabled='true' />
  <single_instance />
    <dependency name='milestone' grouping='require_all'
      restart_on='none' type='service'>
      <service_fmri value='svc:/milestone/sysconfig' />
    </dependency>
    <dependent name='utmpd_multi-user' grouping='optional_all'
      restart_on='none'>
      <service_fmri value='svc:/milestone/multi-user' />
    </dependent>
    <exec_method type='method' name='start'
      exec='/lib/svc/method/svc-utmpd' timeout='60' />
    <exec_method type='method' name='stop'
      exec=':kill' timeout='60' />
  <stability value='Unstable' />
  <template>
    <common_name><loctext xml:lang='C'>
      utmpx monitoring
    </loctext></common_name>

    <documentation>
    <manpage title='utmpd' section='1M'
      manpath='/usr/share/man' />
    </documentation>
  </template>
</service>

```

# Components: Service Profile

- An XML file containing listing of service instances and settings for “enabled”
- Located in `/var/svc/profile`
- `/var/svc/profile/site.xml` available for administrators to make local customizations; it is always applied after Solaris-provided profiles
- Apply manually with `svccfg apply`
- See `smf_bootstrap(5)` for more detail

# Components: Legacy Services

- Started and stopped due to presence in appropriate `/etc/rc?.d` directory
- Faults not handled by `smf(5)`; no automated restart
  - > software error results in process death
  - > administrative error undetected
  - > hardware error results in process death
- Script names visible in `svcs(1)` (prefixed by `rc:`)
- Start time during boot displayed, but not updated if administrator stops or restarts the service

# Components: Security

- Enhance security by providing uniform mechanism to disable services
- Provide alternate profiles: `generic_limited_net.xml`
- Specify administrative authorizations to manage and configure services via RBAC (`smf_security(5)`):
  - > `solaris.smf.modify`: change all service properties
  - > `solaris.smf.manage`: request restart, refresh, etc.
- Define a `method_context` to allow services to easily run as non-root users and with restricted privileges

# Components: Repository

- All data (services, methods, etc.) stored in persistent, transaction-based repository
  - > Transactions/snapshots allow “undo”, rollback to safe configuration
  - > Repository can be local, in directory [later], or mixed [later]
- NOT a giant registry: mainly svc mgmt properties
- Can contain simple configurations (a few properties)
  - > All configurations in repository can be read/written using a common API  $\Rightarrow$  lowers management s/w development times

# Components: *svc.configd*(1M)

- Single access point to repository
- Manages
  - > back-end database
  - > access control
  - > snapshots
  - > backups

# Components: *svc.startd(1M)*

- Graph engine
  - > records service state changes and manages all service restart relationships
  - > signals individual restarters on dependency events and administrative requests
  - > responsible for run-level management
- Master restarter
  - > writes contracts and responds to contract events
  - > manages service environment and bindings
  - > executes service methods based on graph engine requests

# Components: inetd(1M)

- A delegated restarter for inet services
- Manages inet-specific properties
  - > tcp\_wrappers
  - > max connection rate
- Listens for connections, etc.
- Manages instance state for inet services

# Components: `init(1M)`

- Remains the primordial user process
- Continues its process-reaping duties
- Delegates run-level maintenance to `svc.startd(1M)`
- Restarted by the kernel
- Restarts `svc.startd(1M)`

# Commands

- General commands:
  - > `svcs(1)` service status listings
  - > `svcadm(1M)` administrative actions
  - > `svccfg(1M)` general property manipulation
  - > `svccprop(1)` property reporting (scripting)
- `inetd(1M)` management commands:
  - > `inetadm(1M)` administrative actions/property mods
  - > `inetconv(1M)` conversion of legacy `inetd.conf` entries

# Commands: `svcs(1)`

- List enabled or all (`-a`) instances, sorted by state, time
- Explanations for errors/states (`-x`)
- Show dependencies (`-d`) and dependents (`-D`)
- Show member processes (`-p`), additional details (`-v/-l`)

```

$ svcs
STATE          STIME          FMRI
....
online         18:18:30      svc:/network/http:apache2
online         18:18:29      svc:/network/smtp:sendmail
....

$ svcs -p sendmail
STATE          STIME          FMRI
online         18:18:29      svc:/network/smtp:sendmail
                18:18:29      100180 sendmail
                18:18:29      100181 sendmail

$ svcs -d sendmail
STATE          STIME          FMRI
online         18:17:44      svc:/system/identity:domain
online         18:17:52      svc:/network/service:default
....

```

# Commands: svcadm(1M)

- Enable, disable, refresh, restart service instances
- Mark in special states (maintenance)
- Synchronously wait for changes (-s)

```

$ grep lianep /etc/user_attr
lianep::::auths=solaris.smf.modify,solaris.smf.manage
$ svcs apache2
STATE          STIME          FMRI
-              ?              svc:/network/http:apache2
$ # create /etc/apache2/httpd.conf
$ svcadm enable apache2
STATE          STIME          FMRI
online         19:19:01      svc:/network/http:apache2
$ # edit /etc/apache2/httpd.conf
$ svcadm refresh apache2
$ svcs apache2
STATE          STIME          FMRI
online         19:19:33      svc:/network/http:apache2
$ svcadm disable apache2
$ svcs apache2
STATE          STIME          FMRI
disabled      19:20:07      svc:/network/http:apache2

```

# Commands: `svcadm(1M)` actions

- **enable**: allow start once dependencies are satisfied
- **disable**: stop service and do not allow it to start again
  - > **-t**: enable/disable until the system is rebooted
  - > **-s**: enable/disable synchronously (wait for it...)
- **refresh**: reload service configuration and run the refresh method (if any)
- **restart**: stop the service, then allow it to start once its dependencies are satisfied (no configuration change made)
- **clear**: mark service as repaired

# Commands: `svcadm(1M)` milestone

- Milestone: A service which specifies a collection of dependencies which declare a specific state of system-readiness
- `init S`, `boot -s` and friends still work
- Major milestones, which are analogous to system run-levels can be reached directly from boot (`-m milestone=`), the standard `init` invocations, or via `svcadm`:  
`milestone/single-user`, `milestone/multi-user`,  
`milestone/multi-user-server`
- Transitions to milestones are implemented by temporary disable. `svcs(1M)` will show all enabled services that are not part of the milestone as temporarily disabled

# Commands: svccfg(1M)

- Import, export manifests; apply, extract profiles
- Interactive mode for modifying properties

```

$ grep lianep /etc/user_attr
lianep:::auths=solaris.smf.modify,solaris.smf.manage
$ svccfg -v import /var/svc/manifest/network/http-apache2.xml
svccfg: Refreshed network/http:/apache2
svccfg: Successful import.
$ svccfg
svc:> select network/http:apache2
svc:/network/http:apache2> listprop
...
general                                framework
general/enabled                        boolean  false
...
start                                   method
start/exec                              astring  "/lib/svc/method/http-apache2 start"
start/timeout_seconds                   count    60
start/type                               astring  method
svc:/network/http:apache> editprop
[$EDITOR launches, allows direct editing of properties]
svc:/network/http:apache2> exit

$ svcadm refresh apache2      # read changed config
$ svcadm restart apache2     # restart with changed config

```

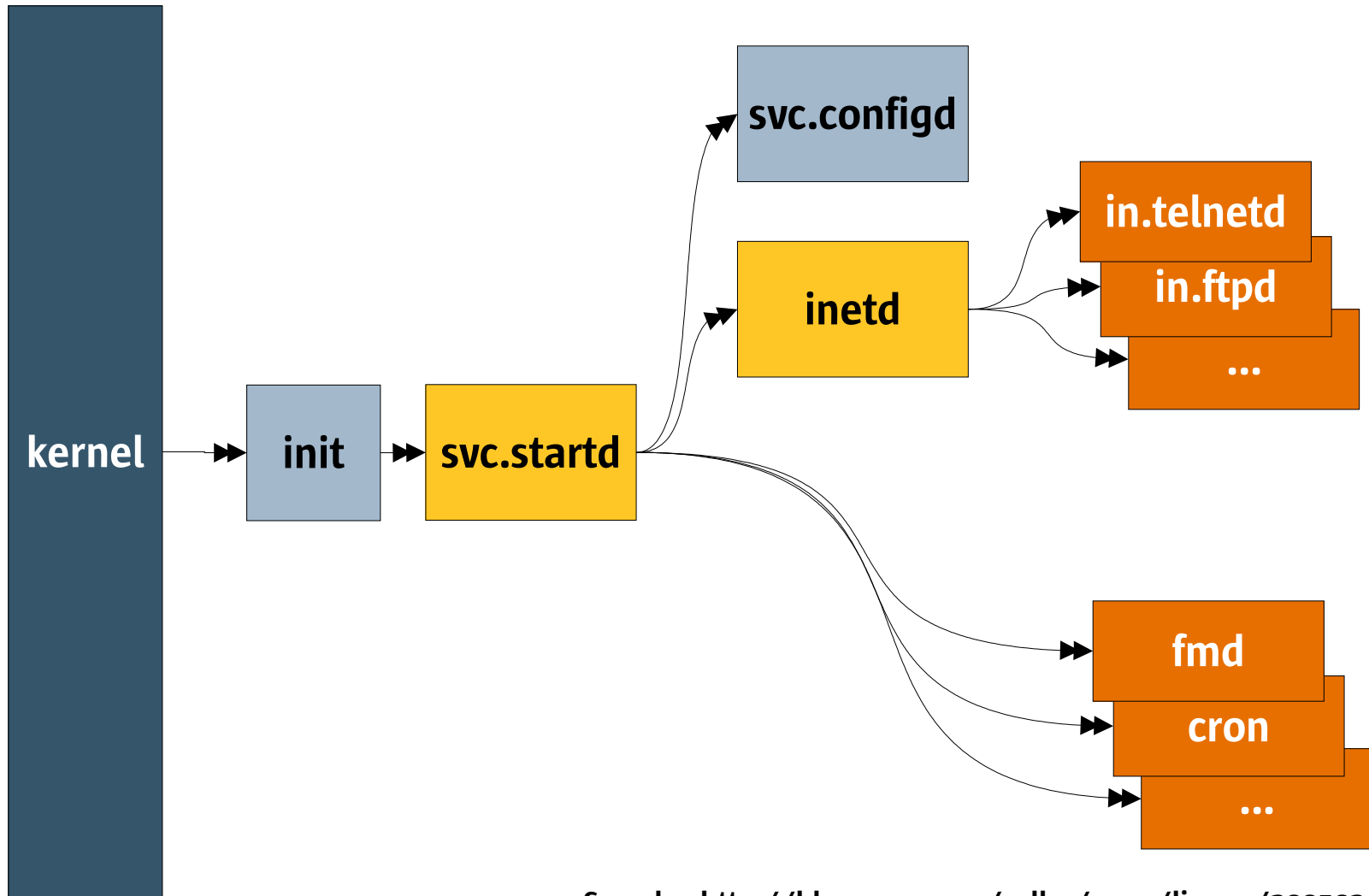
# Commands: svcprop(1)

- List properties of services and instances
- Fetch in convenient forms for scripting
- View running or current props (-c), uncomposed (-C)

```
$ svcprop network/http:apache2
...
physical/entities fmri svc:/network/physical:default
physical/grouping astring optional_all
physical/restart_on astring error
physical/type astring service
start/exec astring /lib/svc/method/http-apache2\ start
start/timeout_seconds count 60
start/type astring method
stop/exec astring /lib/svc/method/http-apache2\ stop
stop/timeout_seconds count 60
stop/type astring method
restarter/auxiliary_state astring none
restarter/next_state astring none
restarter/state astring disabled
restarter/state_timestamp time 1102030556.737590000

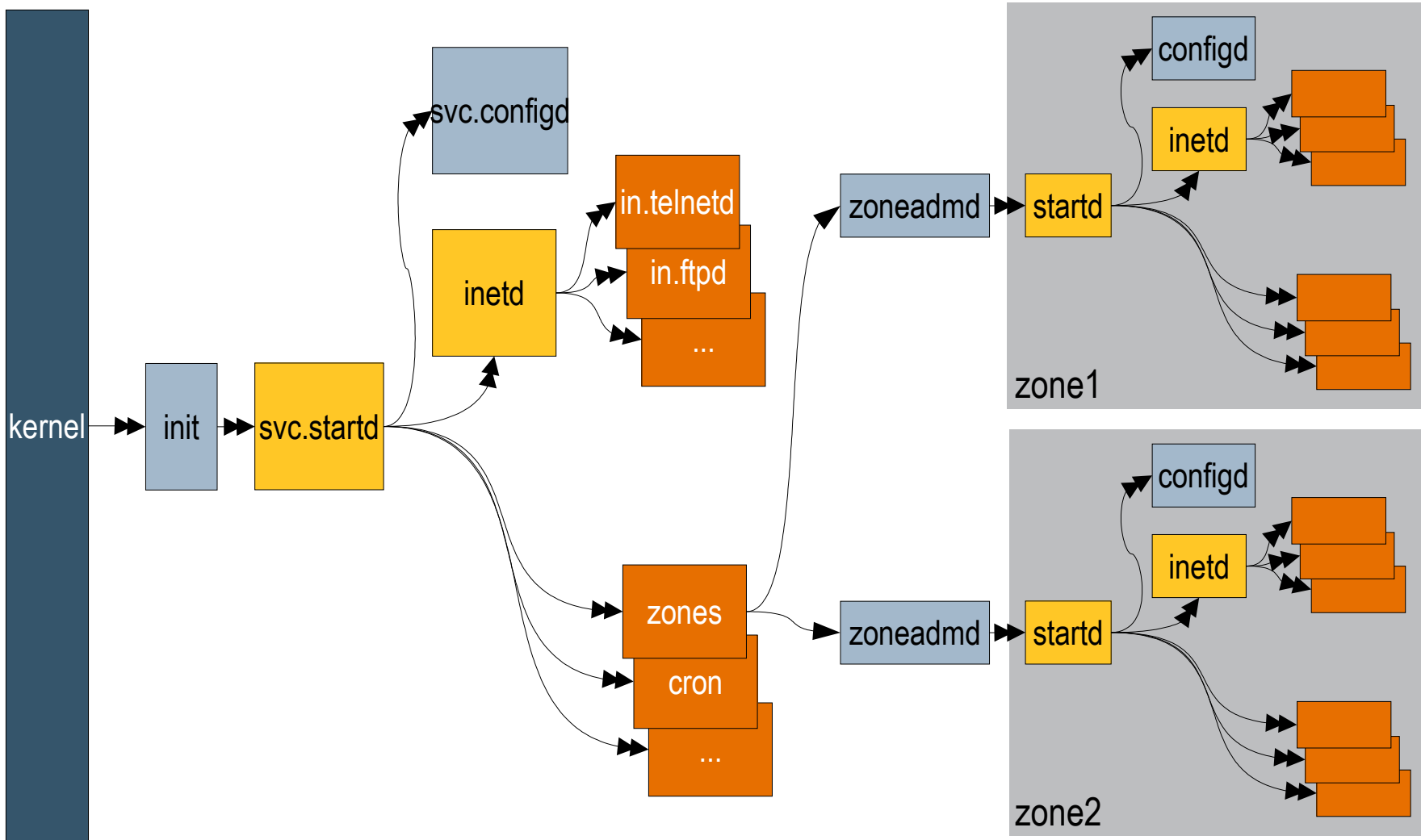
$ svcprop -p enabled network/http:apache2
false
```

# Restart relationships



See also <http://blogs.sun.com/roller/page/lianep/20050316>

# Zones



# Troubleshooting

- Service failures printed to console, syslog
- Always start with `svcs -x`; <http://sun.com/msg>
- `svcs -x` will display the service logfile, if it exists:
  - > `/var/svc/log`
  - > `/etc/svc/volatile`
- See service start messages with `boot -m verbose`
- For a system that hangs during boot:
  - > `boot -m milestone=none`
  - > log in at prompt
  - > `svcadm milestone all`
  - > Watch system progress with `svcs`

# Recovery

- If a single service is broken, make sure you've got the latest service config; `svcadm refresh <fmri>`
- Follow instructions from `svcs -x` pointer
- Revert to a previous snapshot.

```
$ svccfg -s system/cron:default
svc:/system/cron:default> listsnap
initial
last-import
previous
running
start
svc:/system/cron:default> revert start
svc:/system/cron:default> exit
$ svcadm refresh cron
$ svcadm restart cron
```

- Read `/lib/svc/share/README`
- Restore repository from backup:  
<http://sun.com/msg/SMF-8000-MY>

# Service Development

- Conversion may be done piecemeal and is a lightweight act
- Work needed for existing Solaris software is usually:
  - > Create a service manifest
  - > Make minimal modifications to `/etc/rc?.d` scripts and re-invoke them as service methods

# Development: Benefits

- Services appear with `smf(5)` FMRI's
  - > Visible using standard Solaris tools; your service appears in administrative heads-up displays
  - > Manageable using standard Solaris tools; admin can leverage existing knowledge to use your service
  - > New generic tools developed will automatically see your service
- Built-in restart due to administrative error, software, or hardware fault
- Participation in future software diagnosis capabilities

# Development: Levels of integration

- Common:
  - > Trivial: create simple service manifest, convert init scripts to service methods, minimal testing
  - > 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

# Service Lifecycle

- Determine service suitability
- Write service methods
- Write service manifest
- Import manifest
- Test/modify
- Package manifest and methods
- Install service
- Service startup/runtime
- Configure/modify service
- Remove service

# Method creation

- Start with `/etc/init.d` script
- Include `/lib/svc/share/smf_include.sh`
  - > `SMF_EXIT_OK`
  - > `SMF_EXIT_ERR_FATAL`
  - > `SMF_EXIT_ERR_CONFIG`
- Fail with a non-0 exit code, and emit a message to `stdout` or `stderr`
- `start` and `stop` required, `refresh` optional
- `:kill [signal]` and `:true`
- Don't exit until service is completely started
- remove script from `/etc/rc?.d` locations and `/etc/init.d`

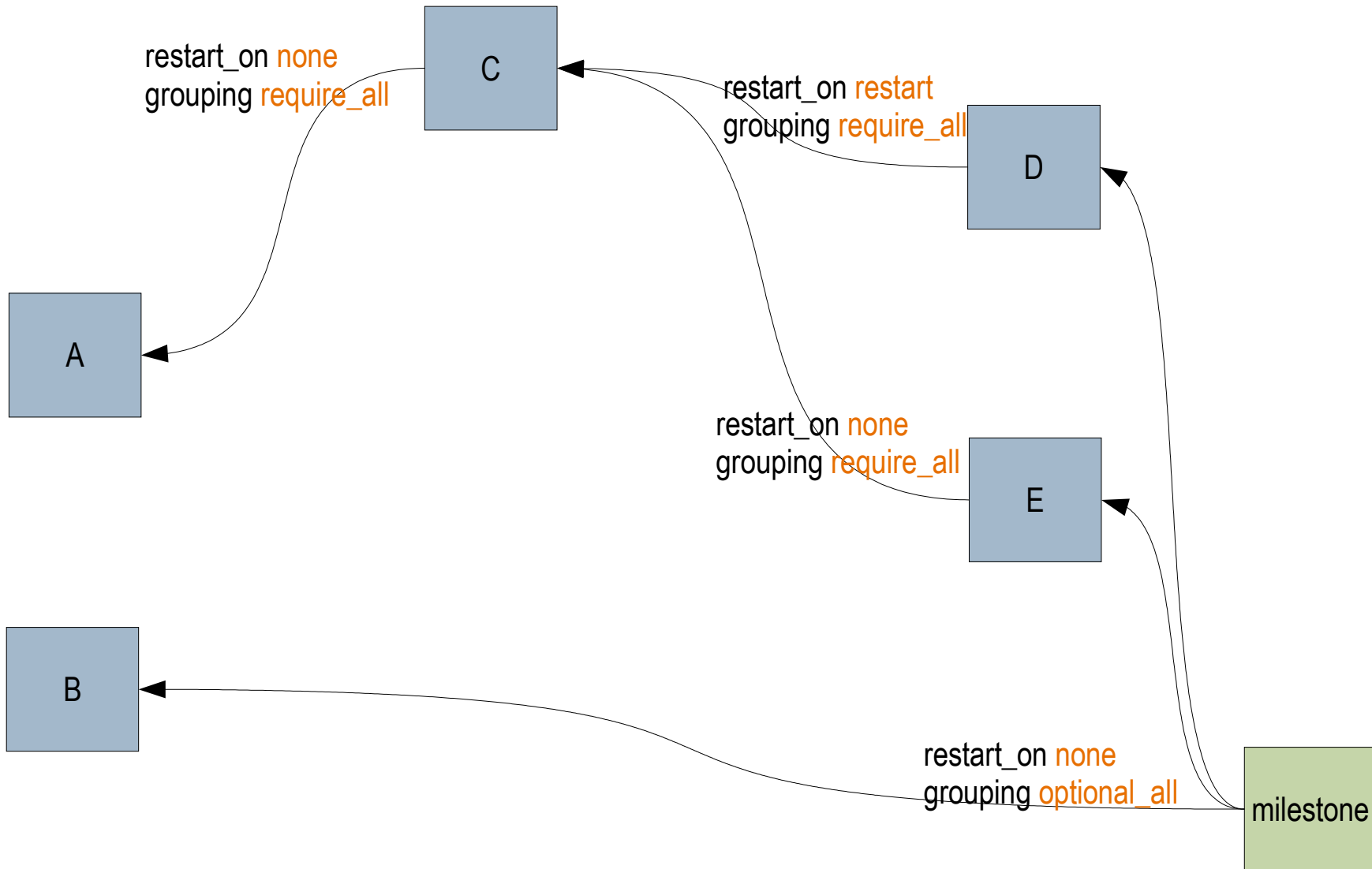
# Manifest Creation

- Name your service
- Identify whether your service may have multiple instances
- Identify your service model: `contract`, `transient`, or `wait`
- Identify how your service is started/stopped
- Determine faults to be ignored
- Identify dependencies
- Identify dependents
- Insert your service into a milestone
- Create, if appropriate, a default instance
- Create template information to describe your service

# Service relationships

- Restart relationships defined by dependency groups
  - > grouping
    - > require\_all: all services are running
    - > require\_any: at least one service running
    - > optional\_all: all services running, disabled, or maint
    - > exclude\_all: all services disabled, maint, or absent
  - > restart\_on
    - > none: required only for startup
    - > error: stop if dependency fails due to h/w or s/w error
    - > restart: stop if dependency restarts for any reason
    - > refresh: stop if dependency restarts or is refreshed

# Service relationships



# Development: utmpd(1M) example

```

<service name='system/utmp' type='service' version='1'>
  <create_default_instance enabled='true' />
  <single_instance />
    <dependency name='milestone' grouping='require_all'
      restart_on='none' type='service'>
      <service_fmri value='svc:/milestone/sysconfig' />
    </dependency>
    <dependent name='utmpd_multi-user' grouping='optional_all'
      restart_on='none'>
      <service_fmri value='svc:/milestone/multi-user' />
    </dependent>

    <exec_method type='method' name='start'
      exec='/lib/svc/method/svc-utmpd' timeout='60' />
    <exec_method type='method' name='stop'
      exec=':kill' timeout='60' />
  <stability value='Unstable' />
  <template>
    <common_name><loctext xml:lang='C'>
      utmpx monitoring
    </loctext></common_name>

    <documentation>
    <manpage title='utmpd' section='1M'
      manpath='/usr/share/man' />
    </documentation>
  </template>
</service>

```

# Service Packaging

- Use `i.manifest` and `r.manifest` from a Sun-delivered package (e.g. `SUNWcsr`)
- Manifests delivered into `/var/svc/manifest` with type “f” and class “manifest”
  - > Use `/var/svc/manifest/site` if the service is specific to your site
  - > Use another directory if you're an ISV, but remember a uniquifier
- Methods delivered with your application binaries (`/opt` strongly recommended)

# Development: Other Examples

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

# Additional Resources

- Additional quickstart and developer documentation available at  
<http://www.sun.com/bigadmin/content/selfheal/>
- Solaris System Administration Guide has smf information:  
<http://docs.sun.com/app/docs/doc/817-1985>
- smf(5) manpage introduces the facility
- Blogs:
  - > <http://blogs.sun.com/sch>
  - > <http://blogs.sun.com/lianep>

**smf(5)**

**Liane Praza**

<http://blogs.sun.com/lianep>

[liane.praza@sun.com](mailto:liane.praza@sun.com)