Zones as SMF services

One aspect of default local zones is that they are managed as one big bunch – all zones marked with autoboot="true" are started along with the service svc://system/zone, and all running zones are stopped along with it. These zones are started and stopped in parallel, with little respect to possible dependencies (i.e. an LDAP server or database should better start before application servers and stop after them). Using some simple tricks from this article, it is possible to set up local zones as SMF services of the global zone, along with all the goodies this brings regarding setup of dependencies, in OpenIndiana as well as similar SMF environments (including legacy ones – verified to include Sun Solaris 10 and OpenSolaris SXCE).

Technical details

- Zones are managed by zoneadm which is spawned separately from the command-line zoneadm processes; SMF can not see it as a managed child process in contract of wait service types, so we are limited to transient (fire and forget) services. If any zones are stopped, the SMF framework would still consider the service running because it is not monitored. This matches what is available with the default zones service.
- It may be possible to add-on active monitoring of the zone status and/or responsiveness of processes inside the zone with techniques like those in vboxsvc (see http://sourceforge.net/projects/vboxsvc), but this is not pursued in this setup.
- The service manifest contains all the code needed for startup and shutdown without relying on method scripts. The solution should also be portable to Solaris 10 and similar OSes.
- The shutdown methods are customizable to an extent. However, they do rely on zlogin -S providing a shell into the local zone OS, so that the shutdown routines can be started. This probably limits this implementation to Solaris-like zones (iPKG, SVR4, solaris10 brand and such).
- The service does not currently impose a timeout on zone shutdown by default [timeout_seconds="0"]). In case of dependencies properly set up, this would mean that a hung zone will block a clean shutdown of its service, and won’t allow the system/zone service to stop (and in particular to call zoneadm halt). It might be beneficial to set the timeout to a large, but finite, value (1800 seconds?) either in the overall zone service definition, or in certain instances for local zones which you expect (or find) to be troublemakers.
- Note that a timed-out shutdown would retry 3 times (by default) and then place the service instance into the maintenance state which, until cleared (by an admin who would revise or ensure that the zone has stopped, or would otherwise fix the problem), would block subsequent startup of this zone and those instances which depend on it.
- If a timeout is enabled, the default stop/exec method tries to detect that the previous attempt to stop the service timed out and failed, and falls back to zoneadm ... halt.
- If any zones remain with autoboot="true", their startup will be managed (fire and forget) by svc:/system/zone/default first, and any dependencies (the zone-group instances and zone instances) will not start until the system/zone service has reported that it is online. If some of these zones are also SMFized (and autobooted, still), their SMF instance startup should wait until check_start succeeds (see below) and further dependencies will be fulfilled.

Set-up

First, we add a service zone-groups which does not do anything and is just a dependency for the local SMF-ized zones. Its instances have the same role – if you have a number of zones which are grouped by some role (say, a networking lab on appserver with database, or a set of zones for software compilation and testing, which you don’t use daily 24x7), you can set these zones’ SMF instances to depend not only on each other, but also on such a grouping service which would allow single-command startups and shutdowns of whole zone farms.

By default this service depends on svc://system/zone/default and creates one :default instance of itself, and by default all the SMF-ized zones will depend on it.

Download zone-group.xml and install with svccfg:

```bash
;; wget http://wiki.openindiana.org/download/attachments/27230238/zone-group.xml
;; svccfg import zone-group.xml
```

Second, we install zone.xml which delivers the svc://system/zone service with no instances. There are pre-configured start, stop and refresh methods, as well as some variables and the dependency on zone-group which can be overridden and/or extended in particular instances.

Installation is as simple:

```bash
;; wget http://wiki.openindiana.org/download/attachments/27230238/zone.xml
;; svccfg import zone.xml
```

As discussed above, the default timeouts in this service are not enforced, which may be undesirable for your production installation. If you want all the zone shutdown attempts to abort after a while (then the system/zone service takes over and may be more “rude” in its methods), do set a service-level timeout, i.e.:

```bash
;; svccfg -s zone setprop stop/timeout_seconds = count: 1800
;; svcadm refresh zone
```

Adding zones to SMFization

Download zone.xml and install with svccfg:

```bash
;; wget http://wiki.openindiana.org/download/attachments/27230238/zone.xml
;; svccfg import zone.xml
```

As discussed above, the default timeouts in this service are not enforced, which may be undesirable for your production installation. If you want all the zone shutdown attempts to abort after a while (then the system/zone service takes over and may be more “rude” in its methods), do set a service-level timeout, i.e.:

```bash
;; svccfg -s zone setprop stop/timeout_seconds = count: 1800
;; svcadm refresh zone
```
Let's add some zones to be managed by this framework:

```bash
; zoneadm list -cv
ID  NAME          STATUS   PATH                        BRAND IP
 0  global        running  /                             ipkg shared
 26 zone1        running  /zones/build/zone1           ipkg excl
 -  build-ss12sun installed /zones/build/build-ss12sun ipkg excl
```

```bash
; svccfg -s zone add zone1
; svccfg -s zone add build-ss12sun
```

```bash
; svcs -a | grep zone
online Nov_09 svc:/system/zones-monitoring:default
online 5:17:43 svc:/system/zones:default
```

So, here we have the default instance of the grouping service, which is automatically enabled, and an instance for each of zone1 and build-ss12sun which have no service history.

If you enable the service, the zone should start... or remain working (as it was running now):

```bash
; svcadm enable build-ss12sun
; tail /var/svc/log/*build-ss12sun*log
[ Nov 18 05:27:23 Enabled. ]
[ Nov 18 05:27:43 Executing start method ("zoneadm -z build-ss12sun list -p | awk -F: '{print $3}' | egrep -vi 'CONFIGURED|INCOMPLETE' | exit; zoneadm -z build-ss12sun boot && sleep 5 && while ; do zlogin -S build-ss12sun 'eval svcadm enable -ts svc:/milestone/multi-user-server:default' && exit 0; sleep 3; done"). ]
installed
[ Nov 18 05:27:52 Method "start" exited with status 0. ]
```

Most of the activity performed during start can be seen above: Verify that the zone is in the installed or higher state, boot it, and loop until the requested command succeeds. During this time the service instance is in offline* SMF state and its dependents, if any, don't start just yet.

One trick is to disable auto-starting of the SMF-ized zone with the general zones service. This can be done by the new instance's refresh method:

```bash
; svcadm -v refresh build-ss12sun
Action refresh set for svc:/system/zone:build-ss12sun.
; tail /var/svc/log/*build-ss12sun*log
[ Nov 18 05:24:34 Rereading configuration. ]
[ Nov 18 05:24:34 Executing refresh method ("zonecfg -z build-ss12sun set autoboot=false"). ]
[ Nov 18 05:24:34 Method "refresh" exited with status 0. ]
```

If for some reason refresh does not cut it, you can just do the task manually (as in the log-file snippet above).

Likewise, shutdown of a zone with SMF is simply:
This logic is not so simple: Verify that the zone is in the installed or higher state, and loop until the zone reaches the installed state; if not there yet – verify the instance's log file to see if the last attempt had timed out and in this case try the rude `zoneadm halt`; for the first attempt try the requested shutdown and verification commands. During this time the service instance is in online state and its dependencies, like the zone-group and then system-wide zones services, should not be stopping just yet.

**Customizations**

The service defines the following overridables:

- Dependency on `zone-group:default` – this can be replaced with another group (of course, you can also add dependency on several groups at once – not detailed here):

```plaintext
::; DEPPG=zone-group
::; DEPINST=zones-dummies
::; ZONE=zone1
::; svccfg -s zone-group add "$DEPINST"
::; svcs -d zone:$ZONE

STATE          STIME    FMRI
online         6:18:10  svc:/system/zone-group:default

::; svccfg -s zone:$ZONE addpg $DEPPG dependency && 
svccfg -s zone:$ZONE setprop $DEPPG/grouping = astring: require_all && 
svccfg -s zone:$ZONE setprop $DEPPG/restart_on = astring: restart && 
svccfg -s zone:$ZONE setprop $DEPPG/type = astring: service && 
svccfg -s zone:$ZONE setprop $DEPPG/entities = fmri: "svc:/system/zone-group:$DEPINST" && 
svcadm refresh zone:$ZONE
::; svcs -d zone:$ZONE

STATE          STIME    FMRI
uninitialized   7:02:21  svc:/system/zone-group:zones-dummies
```

As can be verified by some disable and enable actions, the service `zone1` no longer reacts to enablement and disenablement of `zone-group: default`, but does start and stop in line with the new grouping instance.

Of course, dependencies on any other FMRI, including resources like files, SMF services, service-wrapped zones and VMs (via `vboxsvc`) can be set up similarly, each in its property-group, or as a list of FMRI in one group (usually if you require not all of some resource instances, but any one on the list).

- Variables for startup (`zone/check_start`) and shutdown (`zone/check_stop`) verification routines, as well as the shutdown method (`zone/init_stop`):

```plaintext
### Revise defaults inherited from common service
::; svccfg zone | egrep "^zone/"
zone/check_start astring svcadm\ enable\ -ts\ svc:/milestone/multi-user-server:default
zone/check_stop astring svcadm\ disable\ -ts\ svc:/system/svc/restarter:default
zone/init_stop astring init\ 5
```
For my example, zone1 is an unconfigured zone used as a template for cloning. It never reaches the multi-user-server milestone, because it is stuck on console waiting for user input regarding system settings. Its startup as an SMF service with these settings never completes, so it can not be stopped either. Also, init 5 or graceful shutdown -y -g 0 -i 5 never complete either, but the rude halt works in this zone as expected. So let’s fix it up:

```plaintext
:: ZONE=zone1
:: svccfg -s zone:$ZONE addpg zone application
:: svccfg -s zone:$ZONE setprop zone/check_start="/bin/true"
:: svccfg -s zone:$ZONE setprop zone/init_stop=halt
:: svcadm refresh $ZONE
```

I don't bother about check_stop here, because halt does the job, and the SMF service will loop until the zone is merely initialized. While the default zone/check_start value does allow to track a generic zone to the end of its startup, including legacy init-scripts, you might want to customize some zones with scripts that would be called from this method which do the proper verification of the services you need (i.e. ld apsearch -h localhost ... returns an expected answer, for an LDAP server zone). For heavy services like some Java application servers or some databases, startup of “useful” state may be many minutes after the services’ processes have all started and naive initialization methods have reported that everything is okay now. Quite often, it is not.

Also remember that SMF attributes with whitespaces must be passed as double-quoted strings; on command-line this means encasing that with single quotes (""/myscript test"").

As discussed above, the default timeouts in this service are not enforced, which may be undesirable for your production installation. If you want a particular zone’s shutdown attempts to abort after a while (then the system/zone service takes over and may be more “rude” in its methods), do set a instance-level timeout, i.e.:

```plaintext
:: svccfg -s zone:$ZONE addpg stop method
:: svccfg -s zone:$ZONE setprop stop/timeout_seconds = count: 300
:: svcadm refresh zone:$ZONE
```

And note that a timed-out shutdown (where even two zoneadm halt attempts did not help) would place the service instance into maintenance state which, until cleared (by an admin who would revise or ensure that the zone has stopped, or would otherwise fix the problem), would block subsequent startup of this zone and those instances which depend on it.

### Restarting zones

I found that svcadm restart can allow the dependency zone (such as a DBMS one) to stop first, and then it initiates shutdown and restart of dependents (appservers), which is not good.

However, svcadm disable -ts mysql; svcadm enable -s mysql is processed correctly – the appservers stop (given a restart or refresh requirement in restart_on), then the database stops, then vice-versa.

So this solution should work properly during OS startup and shutdown, but service maintenance during the lifetime requires some care.

### SMFizing all zones

All zones in this setup depend on different instances of zone-group service, which in turn depend on svc:/system/zone:default, for backwards compatibility at least (disabling or enabling the system service is still a single point of command to toggle the zones' boot-states). If any zones are not "SMFized" and remain under control of zones, they will start first and block startup of the dependent zone-group instances and ultimately of the "SMFized" zones.

To ensure predictability in the system startup-shutdown, and that some of your “higher priority” zones do indeed start up ASAP, you might want to move all defined zones under the SMF umbrella. This would mark them with autoboot="false" and thus out of zones management; start-up of the system service would essentially become a quick no-op (shutdown would still process and stop all zones which might remain after the SMF wrappers complete).

One trick is to place your normally disabled zones (“golden” templates, etc.) under a zone-group which stays disabled. This still allows you to boot them manually (zoneadm -z zonename boot), but they won’t autostart with the global zone. The snippet below finds all zones with autoboot="false" set which are not yet SMF instances, and makes them SMF instances which depend on a new zone-group:zones-disabled:
This slab of code should find all of your zones, filter out those already put under SMF control, and for remaining ones it will create SMF instances, further filtering based on the `autoboot` flag value:

- those with `autoboot=true` and a state other than `configured` or `incomplete` will be enabled (WARNING: zones which are not currently running will be booted in the process – unless their dependency such as the main `svc:/system/zones:default` service is disabled and `fline` at this moment) and refreshed to set `autoboot=false` (just in case the `refresh` does not to as asked, the code includes an explicit disablement of the zone's `autoboot`);
- those with `autoboot=true` and in `configured` or `incomplete` state will be disabled and refreshed to set `autoboot=false` (as well as explicit command);
- those with `autoboot=false` will be set to depend on the newly created (and disabled) `zone-group:zones-disabled` grouping instead of `zone-group:default`, and explicitly disabled, just in case.

**Cheatsheet**

To summarize this page, a cheat-sheet for copy-pasting 🎨
Hope this helps,
//Jim Klimov

```bash
; DEPINST=zones-dummies
; DEPFMRI="svc:/system/zone-group:$DEPINST"
; DEPPG=zone-group
### For additional zone-group dependencies, other than overriding the default:
#:; DEPPG=zone-group_$DEPINST

; ZONE=zone1
### Put an existing autobooting zone under SMF control
; svccfg -s zone add $ZONE
; svcadm enable -s zone:$ZONE
; svcadm refresh zone:$ZONE
### Change dependency from default to another zone-group
#:; svccfg -s zone-group add "$DEPINST"
; svccfg -s zone:$ZONE addpg $DEPPG dependency &&
   svccfg -s zone:$ZONE setprop $DEPPG/grouping = astring: require_all &&
   svccfg -s zone:$ZONE setprop $DEPPG/restart_on = astring: restart &&
   svccfg -s zone:$ZONE setprop $DEPPG/type = astring: service &&
   svccfg -s zone:$ZONE setprop $DEPPG/entities = fmri: "$DEPFMRI" &&
   svcadm refresh zone:$ZONE
### Add some other dependency, i.e. on another SMFized local zone
; DEPZONE=mysql
; DEPPG=zone_$DEPZONE
; DEPFMRI="svc:/system/zone:$DEPZONE"
; DEPGROUPING=require_all
; DEPRESTART_ON=restart
### Note that depending on the nature of your systems, you might want other restart_on
### values, including "none" for appserver-pooled reconnections to database services.
; svccfg -s zone:$ZONE addpg $DEPPG dependency &&
   svccfg -s zone:$ZONE setprop $DEPPG/grouping = astring: "$DEPGROUPING" &&
   svccfg -s zone:$ZONE setprop $DEPPG/restart_on = astring: "$DEPRESTART_ON" &&
   svccfg -s zone:$ZONE setprop $DEPPG/type = astring: service &&
   svccfg -s zone:$ZONE setprop $DEPPG/entities = fmri: "$DEPFMRI" &&
   svcadm refresh zone:$ZONE
### Change startup monitoring method
; svccfg -s zone:$ZONE addpg zone application;
   svccfg -s zone:$ZONE setprop zone/check_start='"/mytestscript.sh test"' &&
   svcadm refresh zone:$ZONE
### Review states of all zone-related services, and zone states
; svcs -a | grep zone; zoneadm list -cv
```