Assigning System Resources to Solaris 10 Zones Without Reboot

Gabriel Simona, May 2008

Introduction

Assigning some resources, such file systems, raw devices, tape devices, IP addresses, and so on, from the global zone to a non-global zone in the Solaris 10 Operating System generally requires a reboot of the non-global zone. In a production environment, it's unfortunate but necessary to assign resources and reboot, and the Solaris administrator has to wait for the database administrator and the backup operator's approval before rebooting the zone. Often reboot opportunities occur only for a few minutes at night on the weekend.

The way to avoid this nuisance, at least with file systems and raw devices, is to organize the zone's configuration at first and later deliver the administrative operations in a suggested order, as follows.

The methods described here apply only to environments in which all the administrative operations on file systems will be performed from the global zone. These methods work fine with any release of the Solaris 10 OS for SPARC platforms, even with Sparse Zones or Whole Root Zones, but these methods have not been tested on the Solaris 10 OS for x86 platforms or Branded Zones because of their distinct nature.

The types of file systems to be considered here are UFS, ZFS and LOFS (Loopback File System). In the case of ZFS, the file systems must be set with legacy mount point, which means that they do not have a mount point by default. The raw devices considered here are the ones created with Solaris Volume Manager, although the concepts here could be exploited for other Solaris devices.

File Systems

First, we create the file systems and directories in the global zone:

```
# newfs /dev/md/rdsk/d35
# zpool create my_zpool1 c0t2d0
# zfs create my_zpool/my_zfs
# zfs set mountpoint=legacy my_zpool/my_zfs
# mkdir /my_lofs
```

Then we configure the non-global zone called `my_zone`:

```
# zonecfg -z my_zone
zonecfg:my_zone> add fs
zonecfg:my_zone:fs> set dir=/my_ufs
zonecfg:my_zone:fs> set special=/dev/md/dsk/d35
zonecfg:my_zone:fs> set raw=/dev/md/rdsk/d35
zonecfg:my_zone:fs> set type=ufs
zonecfg:my_zone:fs> end
zonecfg:my_zone> add fs
zonecfg:my_zone:fs> set dir=/my_zfs
zonecfg:my_zone:fs> set special=my_zpool/my_zfs
zonecfg:my_zone:fs> set type=zfs
zonecfg:my_zone:fs> end
zonecfg:my_zone> add fs
zonecfg:my_zone:fs> set dir=/my_lofs
zonecfg:my_zone:fs> set special=/my_lofs
zonecfg:my_zone:fs> set type=lofs
zonecfg:my_zone:fs> end
zonecfg:my_zone> commit
zonecfg:my_zone> exit
#
```

To mount file systems in a non-global zone at boot, we do the following:

```
# zoneadm -z my_zone reboot
```

However, we can mount every file system in the zone while in the online state, for example, as in the following, where `/my_zone` is the zonepath:

```
# mkdir /my_zone/root/my_ufs
# mkdir /my_zone/root/my_zfs
# mkdir /my_zone/root/my_lofs
# mount -F ufs /dev/md/dsk/d35 /my_zone/root/my_ufs
# mount -F zfs my_zpool/my_zfs /my_zone/root/my_zfs
# mount -F lofs /my_lofs /my_zone/root/my_lofs
```

Note that the mount point directories must be created previously in the zone.

Now, executing the `df` command in the non-global zone returns something like this:

```
# zlogin -l root my_zone df -h
Filesystem        size   used  avail capacity  Mounted on
/                 6.9G   5.5G   1.4G    81%    /
/dev              6.9G   5.5G   1.4G    81%    /dev
proc                0K     0K     0K     0%    /proc
ctfs                0K     0K     0K     0%    /system/contract
swap               13G   272K    13G     1%    /etc/svc/volatile
mnttab              0K     0K     0K     0%    /etc/mnttab
fd                  0K     0K     0K     0%    /dev/fd
swap               13G    16K    13G     1%    /tmp
swap               13G    32K    13G     1%    /var/run
/dev/md/dsk/d35   6.9G   5.5G   1.4G    81%    /my_ufs
my_zpool/my_zfs   7.9G   2.0G   5.8G    27%    /my_zfs
/my_lofs           22G    15G   6,1G    72%    /my_lofs
```

However, in the global zone, the same command displays a bit different output:

```
# df -h
Filesystem        size   used  avail capacity  Mounted on
/dev/md/dsk/d0     22G    15G   6,1G    72%    /
/devices            0K     0K     0K     0%    /devices
ctfs                0K     0K     0K     0%    /system/contract
proc                0K     0K     0K     0%    /proc
mnttab              0K     0K     0K     0%    /etc/mnttab
swap               13G   1,1M    13G     1%    /etc/svc/volatile
objfs               0K     0K     0K     0%    /system/object
/platform/sun4u-us3/lib/libc_psr/libc_psr_hwcap1.so.1
                   22G    15G   6,1G    72%
                   /platform/sun4u-us3/lib/libc_psr.so.1
/platform/sun4u-us3/lib/sparcv9/libc_psr/libc_psr_hwcap1.so.1
                   22G    15G   6,1G    72%
                   /platform/sun4u-us3/lib/sparcv9/libc_psr.so.1
fd                  0K     0K     0K     0%    /dev/fd
swap               13G    19M    13G     1%    /tmp
swap               13G    80K    13G     1%    /var/run
/dev/md/dsk/d3     20G   6.8G    13G    35%    /my_zone
/dev/md/dsk/d35   6.9G   5.5G   1.4G    81%    /my_zone/root/my_ufs
my_zpool/my_zfs   7.9G   2.0G   5.8G    27%    /my_zone/root/my_zfs
```

Instead, the `mount` command for `my_zone` delivers the following lines, partially cut off, both from the global and the non-global zone:

```
# zlogin -l root my_zone mount | awk '{print $1, $2, $3, $4}'
/ on / read/write/setuid/devices/intr/largefiles/logging/xattr/
onerror=panic/dev=154001f
/dev on /dev read/write/setuid/devices/zonedevfs/dev=4e0000e
/proc on proc read/write/setuid/nodevices/zone=identity/dev=4bc000f
/system/contract on ctfs read/write/setuid/nodevices/zone=identity/
dev=4b8000f
/etc/svc/volatile on swap read/write/setuid/nodevices/xattr/zone=
identity/dev=4c4002b
/etc/mnttab on mnttab read/write/setuid/nodevices/zone=identity/
dev=4c0000f
```

```
/dev/fd on fd read/write/setuid/nodevices/zone=identity/dev=4e4000f
/tmp on swap read/write/setuid/nodevices/xattr/zone=identity/dev=
4c4002c
/var/run on swap read/write/setuid/nodevices/xattr/zone=identity/
dev=4c4002d
/my_ufs on /my_ufs read/write/setuid/devices/intr/largefiles/logging/
xattr/onerror=panic/dev=1540048
/my_zfs on my_zpool/my_zfs read/write/setuid/devices/exec/xattr/
atime/dev=401000b
/my_lofs on /my_lofs read/write/setuid/devices/dev=1540000
#
# mount | grep my_zone | awk '{print $1, $2, $3, $4}'
/my_zone/root/dev on /my_zone/dev read/write/setuid/devices/
zonedevfs/dev=4e0000e
/my_zone/root/proc on proc read/write/setuid/nodevices/zone=
identity/dev=4bc000f
/my_zone/root/system/contract on ctfs read/write/setuid/nodevices/
zone=identity/dev=4b8000f
/my_zone/root/etc/svc/volatile on swap read/write/setuid/nodevices/
xattr/zone=identity/dev=4c4002b
/my_zone/root/etc/mnttab on mnttab read/write/setuid/nodevices/
zone=identity/dev=4c0000f
/my_zone/root/dev/fd on fd read/write/setuid/nodevices/zone=
identity/dev=4e4000f
/my_zone/root/tmp on swap read/write/setuid/nodevices/xattr/
zone=identity/dev=4c4002c
/my_zone/root/var/run on swap read/write/setuid/nodevices/
xattr/zone=identity/dev=4c4002d
/my_zone/root/my_ufs on /dev/md/dsk/d35 read/write/setuid/
devices/intr/largefiles/logging/xattr/onerror=panic/dev=1540048
/my_zone/root/my_zfs on my_zpool/my_zfs read/write/setuid/devices
/exec/xattr/atime/dev=401000b
/my_zone/root/my_lofs on /my_lofs read/write/setuid/devices/
dev=1540000
```

We can dismount the non-global zone's file systems without reboot as well:

```
# umount /my_zone/root/my_ufs
# umount /my_zone/root/my_zfs
# umount /my_zone/root/my_lofs
```

Remember that an LOFS can refer to almost any directory in the global zone, which can be shared by many zones concurrently. This means that if we have an LOFS mounted in a non-global zone and we mount a new file system into the primitive directory in the global zone, the new file system will be accessible in every zone in the respective LOFS mount point.

A special case is the `/cdrom` directory. For example, we can mount it as an LOFS in read-only mode in a non-global zone:

```
# mkdir /my_zone/root/cdrom
# mount -F lofs -o ro /cdrom /my_zone/root/cdrom
```

Then we insert media into the DVD unit and it's mounted into `/cdrom/cdrom0` by the `vold` daemon in the global zone. The media's content will be available online in `my_zone` as it is in the global zone.

It might be useful, furthermore, to mount file systems into a non-global zone in order to overwrite some native mount points. Let's suppose we need the zone's `/tmp` directory outside the swap space to preserve temporary files after reboot. In such a case, we could dismount the original zone's `tmpfs` type file system, as long as nobody is using it and we realize that its content will be lost. Then, we could mount a long-term file system on `/my_zone/root/tmp` anytime:

```
# umount /my_zone/root/tmp
# cp /my_zone/root/etc/vfstab /my_zone/root/etc/vfstab.ori
# sed '/tmpfs/s/swap/#swap/' /my_zone/root/etc/vfstab.ori > \
/my_zone/root/etc/vfstab
# cat > tmpconfig.txt << EOF
add fs
set dir=/tmp
set special=/dev/md/dsk/d36
set raw=/dev/md/rdsk/d36
set type=ufs
end
EOF
# zonecfg -z my_zone -f tmpconfig.txt
# mount -F ufs /dev/md/dsk/d36 /identity/root/tmp
# chmod 1777 /my_zone/root/tmp
```

To revert the recent action without reboot, we have to do the following:

```
# umount /my_zone/root/tmp
# mv /my_zone/root/etc/vfstab.ori /my_zone/root/etc/vfstab
# zonecfg -z my_zone remove fs dir=/tmp
# mount -F tmpfs swap /identity/root/tmp
```

However, if we want the zone's `/var` directory to be in a separate file system than the / one, then it must be mounted on `/my_zone/root/var` prior to installing the zone (included with the initial zone's configuration). Optionally, if the zone is already installed, we could halt it, then move the zone's `/var` directory content to a new file system, and finally add the new file system to the zone on its `/var` mount point before booting the zone again. It's not possible to do such migration without rebooting the non-global zone.

At this point, it's important to decide on the zone's `autoboot` parameter, because every file system on the entire server must be available before the zone boots. Zones always boot after the `svc:/system/zones` service is initiated. Sometimes, file systems are mounted later in the global zone because they depend on other resources, and some of them might have to be mounted as LOFS in a non-global zone. In such situations, we can set the zone's `autoboot` parameter to the value `false` and boot the zone through a legacy run-control script so that we can start a zone under certain conditions, such as testing a file, for instance, `/mounted_path/flag_file`:

```
cat >> /etc/rc3.d/S99zones << EOF
[ -f /mounted_path/flag_file ] && zoneadm -z my_zone boot
EOF
chmod 744 /etc/rc3.d/S99zones
chown root:sys /etc/rc3.d/S99zones
echo 'zoneadm -z my_zone halt' >> /etc/rc0.d/K01zones
chmod 744 /etc/rc0.d/K01zones
chown root:sys /etc/rc0.d/K01zones
```

Raw Devices

To add the raw device `/dev/md/rdsk/d321` to a non-global zone called `my_zone`, we do the following:

```
# zonecfg -z my_zone
zonecfg:my_zone> add device
zonecfg:my_zone:device> set match=/dev/md/rdsk/d321
zonecfg:my_zone:device> end
zonecfg:my_zone> commit
zonecfg:my_zone> exit
#
```

After that, to make the changes effective, we need to reboot the zone:

```
# zoneadm -z my_zone reboot
```

Now, if we want to add the new raw device `/dev/md/rdsk/d322` to the same zone, we must repeat the procedure and reboot the zone again. To avoid doing that, the solution is to use some regular expressions with matching devices at the beginning. It's like configuring tape devices:

```
# cat > tapeconfig.txt << EOF
add device
set match=/dev/rmt/1*
end
EOF
# zonecfg -z my_zone -f tapeconfig.txt
```

This configuration implies the matching of every device's composition from `/dev/rmt/1`, such as `/dev/rmt/1n`, `/dev/rmt/1bc`, and so on:

```
# zlogin -l root my_zone ls /dev/rmt
1   1bn 1cb  1cn  1hb  1hn  1lb   1ln  1mb   1mn  1u   1ubn
1b  1c  1cbn 1h  1hbn  1l   1lbn  1m   1mbn  1n   1ub  1un
```

Taking advantage of this feature, we can assign tens or even hundreds of raw devices with just one matching line:

```
# cat > rawconfig.txt << EOF
add device
set match=/dev/md/rdsk/d3*
end
EOF
# zonecfg -z my_zone -f rawconfig.txt
```

However, we might need to match raw devices consecutively from `d301` to `d320` only. At this time, it's risky to match any other devices, such as `d3` or `d35`, because they could be used in another zone, especially the global zone, since a privileged user in `my_zone` could write on those devices by mistake.

```
# ls /my_zone/root/dev/md/rdsk
d3   d302  d304  d306  d308  d310  d312  d314  d316  d318  d320
d301 d303  d305  d307  d309  d311  d313  d315  d317  d319
```

Another wrong solution would be to match all the disk slices with the expression `/dev/rdsk/c0t2d0s*`, since `c0t2d0s2` represents the entire disk. In fact, it's highly improbable that someone will write on forbidden devices, but a smart administrator wouldn't present the main devices to a non-global zone.

To prevent such situations, we can assign raw devices to a non-global zone, as follows:

```
# cat > rawconfig.txt << EOF
add device
set match=/dev/md/rdsk/d3??
end
EOF
# zonecfg -z my_zone -f rawconfig.txt
```

This brings every raw device between `d300` and `d399` into the zone, but doesn't match `d3` or `d35`, as intended. A sufficient solution is to assign different groups of soft partitions by the hundreds to each zone, in order to rule out overlapping.

The second and more interesting advantage is that we can create new soft partitions in the matching range from the global zone, and they will be online in the non-global zone without reboot. For instance, assume there is a zone that has 20 raw devices from `d301` to `d320` matching `/dev/md/rdsk/d3??`:

```
# ls /my_zone/root/dev/md/rdsk
d301  d303  d305  d307  d309  d311  d313  d315  d317  d319
d302  d304  d306  d308  d310  d312  d314  d316  d318  d320
```

Now we create a new soft partition in the matching range and it appears immediately in the non-global zone:

```
# metainit d321 -p /dev/rdsk/c0t2d0s0 5g
# ls /my_zone/root/dev/md/rdsk
d301  d303  d305  d307  d309  d311  d313  d315  d317  d319  d321
d302  d304  d306  d308  d310  d312  d314  d316  d318  d320
```

Suppose now that we concatenate two disk slices as `d300` and then we create a new soft partition `d322` in it, although this is not recommended, as inferred above.

```
# metainit d300 2 1 /dev/rdsk/c0t2d0s6 1 /dev/rdsk/c1t2d0s6
# metainit d322 -p d300 5g
# ls /my_zone/root/dev/md/rdsk
d300  d302  d304  d306  d308  d310  d312  d314  d316  d318  d320  d322
d301  d303  d305  d307  d309  d311  d313  d315  d317  d319  d321
```

Device `d300` is now available in the non-global zone. We should hinder the zone's users from accessing it.

The `/dev` directory in a non-global zone is an LOFS referenced to `/my_zone/dev` mounted at boot with a special option called `zonedevfs`. The Solaris 10 OS generates one special file in the `/my_zone/dev` directory for each matching device:

```
# file /my_zone/dev/md/rdsk/d300
/my_zone/dev/md/rdsk/d300:      character special (85/300)
# ls -l /my_zone/dev/md/rdsk | head -6
total 0
crw-r-----   1 root     sys       85, 300 Mar  4 10:39 d300
crw-r-----   1 root     sys       85, 301 Mar  4 10:13 d301
crw-r-----   1 root     sys       85, 302 Mar  4 10:13 d302
crw-r-----   1 root     sys       85, 303 Mar  4 10:13 d303
crw-r-----   1 root     sys       85, 304 Mar  4 10:13 d304
```

These are not symbolic links pointing to files located in the global zone's `/device` directory, but they are proper files that can be managed separately. The privileged users in the non-global zone can change permissions on these files inside the `/dev` directory, but they cannot delete them. However, we can remove the zone's devices from the global zone:

```
# rm /my_zone/dev/md/rdsk/d300
```

Though we can clear away some online zones' devices this way, they will be there again after the next reboot. The least bad, most permanent solution is to create empty files replacing the original ones:

```
# touch /my_zone/dev/md/rdsk/d300
# zloqin -l root my_zone "file /dev/md/rdsk/d300"
/dev/md/rdsk/d300:      empty file
# zlogin -l root my_zone "ls -l /dev/md/rdsk" | head -6
total 0
-rw-r--r--   1 root     root           0 Mar  4 12:05 d300
crw-r-----   1 root     sys       85, 301 Mar  4 10:13 d301
crw-r-----   1 root     sys       85, 302 Mar  4 10:13 d302
crw-r-----   1 root     sys       85, 303 Mar  4 10:13 d303
crw-r-----   1 root     sys       85, 304 Mar  4 10:13 d304
```

The new empty files can't be modified by the non-global zone's users. After reboot, these files will remain there.

On the other hand, when we don't need a raw device anymore in the zones, we should delete it everywhere:

```
# metaclear d322
# rm /my_zone/dev/md/rdsk/d322
# metaclear d300
# rm /my_zone/dev/md/rdsk/d300
```

The zone's configuration doesn't change and it's not necessary for the zone to reboot.

Finally, to accomplish all the practices suggested above, the general zone's configuration should look as follows:

```
# zonecfq -z my_zone info
zonepath: /my_zone
autoboot: true
pool:
fs:
        dir: /my_ufs
        special: /dev/md/dsk/d35
        raw: /dev/md/rdsk/d35
        type: ufs
        options: []
fs:
        dir: /my_zfs
        special: my_zpool/my_zfs
        raw not specified
        type: zfs
        options: []
fs:
```

```
fs:
        dir: /my_lofs
        special: /my_lofs
        raw not specified
        type: lofs
        options: []
fs:
        dir: /cdrom
        special: /cdrom
        raw not specified
        type: lofs
        options: [ro]
net:
        address: 192.168.0.35/24
        physical: bge0
device
        match: /dev/rmt/1*
device
        match: /dev/md/rdsk/d3??
#
```