

IBM Lotus Domino in Solaris 10 Zones

Eric Sosman
Sun Microsystems, Inc.
August 2009

Introduction

The Solaris™ 10 operating system provides a number of virtualization technologies collectively known as *Solaris containers*, including the encapsulation technology called *zones*. Applications and services can be assigned to their own zones, within which they can operate freely without interference from activities in other zones. A suite of processes running in a zone appears to have an entire Solaris environment of its own; it cannot even detect the existence of other zones.

This paper explains how to set up and run IBM® Lotus® Domino® in zones. The reader should already be familiar with the essentials of Solaris management (booting and shutting down, mounting file systems, and so on) and of Domino administration (installing and setting up Domino, using scripts to start and stop Domino, and the like). Consult the References for more information on such topics; this paper addresses only the zone-specific aspects of Domino management.

The procedures described here have been tested on Solaris 10 with Domino 6.5, 7, 8.0, and 8.5. It is possible that future Domino versions will require changes to these procedures or new procedures entirely. Consult the release notes for each Domino version as it becomes available.

Why run Domino in zones?

“After all, I've managed without them up to now.”

There are probably as many reasons to use zones for Domino as there are Domino installations. Here are a few scenarios where zones may be helpful; you can probably think of others:

- *Functional separation:* At many installations, the Domino administrators and the Solaris administrators are separate groups, yet the Domino administrators need to control and monitor some aspects of the Solaris environment. Such needs are typically met by having the Domino administrators ask the Solaris administrators to do some simple task that's outside their authorization; this can introduce delays, opportunities for mis-communication, and other procedural snarls. By granting the Domino administrators root access to the Domino zones but not to the entire machine, the Solaris administrators can delegate some of this work to the Domino staff without placing the health of the system as a whole at risk. (Some procedures, however, still require privileged access to the entire machine and therefore still need the attention of the Solaris staff.)
- *Multiple services:* Using zones, a Solaris system can be partitioned to allow disparate applications to run on the same system without interfering with each other. Level-of-resource guarantees can be established, so that, for example, the LDAP server in Zone 1 is assured of at least 20% of the machine's CPU cycles (if it needs them) even if Domino in Zone 2 and the database in Zone 3 are experiencing heavy load and trying to monopolize the CPU. Also, if one of these services should malfunction, whether through programming error, misconfiguration, or malicious action, the damage is contained within a single zone and does not propagate to the rest of the system.
- *Economy:* As a special case of the above, you may choose to set up different zones for different Domino partitions. For example, you might run your production Domino servers in one zone, granting it virtually all the system resources it chooses to consume and restricting access to the authorized cadre

of Domino administrators. Another zone could be set up for testing and staging new applications prior to deployment on the production servers; even if a new application should misbehave badly it will not disturb the production servers running in a different zone. Still more zones could be created for the application developers, whose work-in-progress code is not ready for prime time. Since these activities can safely coexist on a single machine, you need not pay for separate machines merely to achieve isolation and then see those machines sit underutilized most of the time.

Zone Basics

A Solaris 10 system always contains at least one zone, called the *global zone*. The global zone is unique in that it is the only zone that has unrestricted access to the entire machine. The root account for the global zone is the master of the whole system, just as in non-zoned Solaris versions. The global zone starts operating when Solaris boots, and continues operating until Solaris shuts down.

Once the global zone is running, subordinate *non-global zones* can be started. (“Non-global” is such an unattractive term that almost no one uses it except under duress, so this paper uses the less official but less clumsy term *local zone*.) Each local zone appears to be a complete Solaris system, with its own host name, its own network addresses, its own root account, and so on. A local zone can be booted, shut down, and rebooted without affecting the global zone or any other local zones.

However, a local zone can access only those devices and file systems that were assigned to it when it was configured, and cannot even detect (much less interfere with) other devices, file systems, and zones. In order to maintain this isolation, processes in a local zone—even privileged processes—are subject to some restrictions that do not apply to the global zone or to earlier Solaris versions. These security restrictions affect Domino in three areas:

- *Installation:* The scripts that install Domino add some components to the Solaris kernel, and place files in a directory that is usually shared by all zones running Domino. Both these actions are forbidden in local zones, so the installation of Domino must be performed in the global zone. Installing Domino upgrades, maintenance releases, patches, and hot fixes must be done in the global zone for the same reasons.
- *Setup:* The process of setting up a newly-installed Domino server requires no special privileges, but does require that all the data directories for all the installed server partitions be available during setup. If you wish to run different Domino partitions in different zones, you will need to mount all their data directories in a single zone (perhaps the global zone) while performing the setup. Once setup is complete, you can rearrange the mounts across multiple zones as needed.
- *Diagnostics:* Lotus' `nsd` diagnostic program uses the `prtconf` utility, whose functions are restricted when run in a local zone. `prtconf` lists the system's installed CPUs and physical memory, and ordinarily also reports the entire physical device tree. This last operation is forbidden in a local zone, so that part of the `prtconf` output is absent from the `nsd` report.

Most of this paper discusses ways of dealing with the first two difficulties, which must be solved before you can run Domino in local zones. Fortunately, the difficulties can be solved with straightforward

procedures. The final issue cannot be solved, and `nsd` logs will simply not contain some of the information IBM's Support staff are accustomed to seeing. However, none of the missing information seems to have any material effect on Support's ability to provide customer service: Domino always uses virtual constructs like "sockets" and "files," so a listing of the physical devices behind such virtualizations is largely irrelevant.

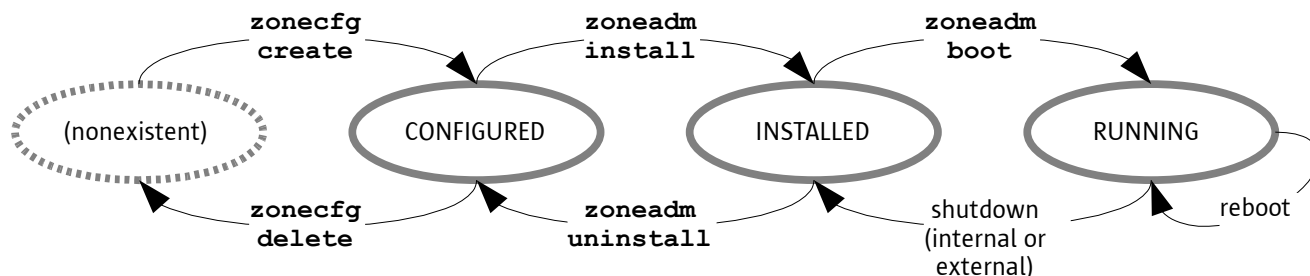
Root file systems

Local zones are categorized by the nature of their root file systems. A *whole root zone* has a complete root file system of its own, containing all (or nearly all) the files one would find in the root file system of a non-zoned Solaris. A *sparse root zone* "inherits" most of the root file system from the global zone, with private versions of only a few directories like `/etc` and `/var`. Since a sparse root zone shares the inherited material with the global zone and with other sparse root zones, the inheritance is read-only: The sparse root zone can write to its `/etc` and `/var` file systems, but cannot modify `/usr` or other inherited portions of the root file system.

Since a sparse root zone shares most of its root file system and needs only a hundred megabytes or so of disk space for itself, a large number of such zones can exist on a single system quite inexpensively. Also, a patch or update applied to the root file system in the global zone is automatically inherited by all sparse root zones. Because of their disk space economy and low maintenance burden, sparse root zones are preferable to whole root zones except in fairly special circumstances. Therefore, this paper only describes how to use sparse root zones; consult the References for information on whole root zones.

A zone's life cycle

At any moment, a local zone is in one of several possible states. The most important of these are illustrated below, with the events or commands that move the zone from one state to another:



- *Configured:* A description of the zone's configuration has been created and saved, using the `create` subcommand of the `zonecfg` utility. You can use `zonecfg` as many times as necessary to modify the configuration, or use the `delete` subcommand to remove it altogether. `zonecfg` can make some kinds of changes to an Installed or Running zone to take effect the next time the zone boots, but others can only be made to a Configured zone.

- *Installed:* The `install` subcommand of the `zoneadm` utility builds the zone's root file system and other components, in accordance with the description prepared by `zonecfg`. The `uninstall` subcommand erases the zone's root file system and returns the zone to the Configured state.
- *Running:* The `boot` subcommand of `zoneadm` causes an Installed zone to initialize and start executing. If desired, you can arrange to have a zone boot automatically when Solaris itself boots. A zone can shut down and return to the Installed state on its own initiative, or the `halt` subcommand of `zoneadm` can force an abrupt shutdown if necessary.

A few other states exist: for example, the transition from Running to Installed goes through Shutting_Down and Down states not shown in the diagram. These states are usually transitory unless something goes wrong, so this paper does not discuss them. See the References for information on the full suite of zone states and how to troubleshoot them.

At any time, the root user in the global zone can list all the system's zones and their current states by running the `list` subcommand of the `zoneadm` utility:

```
# zoneadm list -cv
ID NAME           STATUS           PATH
 0 global          running          /
28 twilight       running          /zoneroots/twilight
- comfort         installed        /zoneroots/comfort
```

An Introductory Example

This section describes how to set up perhaps the simplest of all possible zoned configurations for Domino: One local zone running one Domino partition on one network address, storing its databases on one file system. Later sections will build on this one and discuss more complex configurations.

Although Domino executes quite happily in a local zone, some of the procedures for installing, updating, and maintaining Domino are not “zone-aware” and must be carried out in the global zone. In order to let both the global and the local zone access Domino's files, we will use the *loopback file system (lofs)* to enable them to be mounted in both zones simultaneously. This is not ideal from the standpoint of zone isolation, but is quite simple to set up and manage. Procedures that maintain better isolation will be described later.

In broad outline, the steps for setting up this configuration are:

- (Global zone): Install Domino from the IBM Lotus distribution kit.
- (Global zone): Prepare the `/dev/listen` device for export to local zones.
- (Global zone): Configure a local zone for Domino.
- (Global zone): Install and boot the local zone.
- (Local zone): Perform first-time boot configuration.
- (Local zone): Set up and run the Domino server.

These steps are described in the following subsections.

Installing Domino

Install Domino from its distribution kit just as you would on a non-zoned Solaris system. As the root user in the global zone, mount the CD or unpack the compressed download archive, then change to the kit's topmost directory (the one containing the `install` shell script), execute `./install`, and answer the prompts as usual.

The only special concern for a zoned system is that the Unix user account and group that own Domino's data directory must be valid in the global zone. If user and group information is obtained from a network service like NIS or LDAP this will probably require no extra effort. However, if it is to be obtained from the system's `/etc/passwd` and `/etc/group` databases you will need to define them manually before starting the installation. The user account's only purpose in the global zone is to own the installed files, so it does not actually need a home directory—in fact, you may wish to prevent the user account from running in the global zone at all, for example, by specifying `/usr/bin/false` as its “shell.” Whichever way the user account is made known to the global zone, it must have the same user and group ID numbers as it will in the local zone where Domino will eventually run.

When the installation completes, do not attempt to set up the new Domino server until you have created and initialized the local zone as described below.

If you wish to tune Solaris' parameters in `/etc/system`, you must make these adjustments in the global zone and reboot the entire system. However, few adjustments are actually necessary nowadays; Solaris 10 is far more “dynamic” than earlier versions in the sense that it adjusts itself to adapt to changing circumstances. We recommend that you should not simply inherit `/etc/system` adjustments from earlier releases, but start with a clean or nearly clean slate and only tune as the need becomes apparent. Consult the References for more tuning information.

Preparing `/dev/listen` for export

Domino adds a device driver to the Solaris kernel, and you must export the device to the local zone so Domino can use it when it runs there. Domino's installation script makes the device accessible to the global zone, but some versions leave it in an “unexportable” state. To make it “exportable” you need to edit one file and run one command.

As the root user in the global zone, edit the file `/etc/devlink.tab` and search for this line:

```
type=ddi_pseudo;name=listen;addr=0;minor=listen \M0
```

Some Domino versions install this line themselves, but others do not. If the line is not present, add it at the end of the file. The line must begin at the left margin and must contain no space characters at all; the blank just before `\M0` must be exactly one TAB character. Save the edited file.

Still as the root user, run the `devfsadm -v` command. You should see output like

```
# devfsadm -v
devfsadm[387]: verbose: removing link /dev/listen -> /devices/pseudo/listen@
0:listen invalid contents
devfsadm[387]: verbose: symlink /dev/listen -> ../devices/pseudo/listen@0:li
sten
```

The `listen` device is now registered with the *device file system (devfs)*, which enables you to export it to local zones for Domino's use.

Configuring a local zone

In this step you will define the configuration of the local zone, describing its name, its network addresses, its file systems, and so on. Begin by choosing a name for the zone; one common convention is to give the zone the same name as its primary Internet host name. (Note that the local zone's Internet name will necessarily be different from the global zone's name: different Internet addresses correspond to different names.) The local zone in this example will be named `twilight`.

As the root user in the global zone, run the `zonecfg` command and specify the local zone's name with the `-z` option. Then use the `create` subcommand to start building a new zone configuration:

```
# zonecfg -z twilight
twilight: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:twilight> create
```

Choose a place in the global zone's file system where the local zone's root file system will reside. You can use the same boot disk as the global zone, or any file system that the global zone will mount before the local zone boots. (See "Isolating zone roots" on page 16 for some suggestions.) Specify the chosen location with the `set zonepath` subcommand:

```
zonecfg:twilight> set zonepath=/zoneroots/twilight
```

Decide whether you want the local zone to boot automatically whenever Solaris boots. If so (most Domino installations will probably want to do this), use the `set autoboot` subcommand:

```
zonecfg:twilight> set autoboot=true
```

Export the `listen` device to the new zone with these three subcommands:

```
zonecfg:twilight> add device
zonecfg:twilight:device> set match=/dev/listen
zonecfg:twilight:device> end
```

NOTE: In some early versions of Solaris 10 the proper command was `set match=listen` instead of `set match=/dev/listen` as shown here. If you cannot find a `/dev/listen` device in the zone after you install and initialize it (and you have double-checked the zone configuration for typographical errors), try changing to `set match=listen`, and rebooting the zone (you do not need to reinstall it). Better, upgrade to a contemporary Solaris.

Specify the physical device the local zone will use for network communications, and the network address it will use. Along with the address, specify the subnet mask: the number of high-order address bits that identify the subnet (for example, the mask that in some other contexts is written as `255.255.255.0` or `ffffff00` describes a subnet that is identified by the top twenty-four address bits; you would use the suffix `/24` to specify this subnet to `zonecfg`):

```
zonecfg:twilight> add net
zonecfg:twilight:net> set physical=ce0
zonecfg:twilight:net> set address=129.148.63.27/24
zonecfg:twilight:net> end
```

Note that when specifying the physical network device you should name only the “real” device (like `ce0` or `hme1`) and not a pseudo-device like `ce0:1` or `hme1:3`. Solaris manages the assignment of pseudo-device numbers in zones based on which zones are active at the moment.

If the local zone will use more than one network address and perhaps more than one physical device (for example, to separate client traffic from Domino cluster traffic), use the above sequence of subcommands as many times as necessary to specify the additional network connections.

Make Domino's executables available to the local zone in read-only mode by exporting the directory that contains them, typically `/opt/lotus` or `/opt/ibm/lotus`, or whatever alternate directory you specified during Domino installation:

```
zonecfg:twilight> add fs
zonecfg:twilight:fs> set dir=/opt/lotus
zonecfg:twilight:fs> set special=/opt/lotus
zonecfg:twilight:fs> set type=lofs
zonecfg:twilight:fs> add options [nodevices,ro]
zonecfg:twilight:fs> end
```

Use the same method to export file systems for the Domino data directory, the transaction logs, and any other Domino databases (omitting the `ro` option, which would make them read-only). This example assumes that the data directory is in the default location `/local/notesdata`, that the transaction logs will reside on a file system mounted at `/noteslogs`, and that no other file systems are needed; make adjustments as needed for your system:

```
zonecfg:twilight> add fs
zonecfg:twilight:fs> set dir=/local/notesdata
zonecfg:twilight:fs> set special=/local/notesdata
zonecfg:twilight:fs> set type=lofs
zonecfg:twilight:fs> add options [nodevices]
zonecfg:twilight:fs> end
zonecfg:twilight> add fs
zonecfg:twilight:fs> set dir=/noteslogs
zonecfg:twilight:fs> set special=/noteslogs
zonecfg:twilight:fs> set type=lofs
zonecfg:twilight:fs> add options [nodevices]
zonecfg:twilight:fs> end
```

You can use the `info` subcommand to review the information you have entered, the `select` command to navigate to and change erroneous or incomplete items, the `remove` subcommand to delete some information altogether, and of course as many more `set` and `add` subcommands as you need. When you are through, use the `exit` subcommand; this will automatically commit the configuration, saving it for later use. Use `man zonecfg` or see the References for more information on these and other subcommands.

You can run `zonecfg` again if you need to change the configuration: simply specify the same local zone name with the `-z` option and the existing configuration will be loaded for you to modify and re-commit.

Installing and booting the local zone

Once its configuration has been specified, install the local zone with the `zoneadm` utility:

```
# zoneadm -z twilight install
Preparing to install zone <twilight>.
(... more output ...)
The file </zoneroots/twilight/root/var/sadm/system/logs/install_log> contains
a log of the zone installation.
```

Installation creates the new zone's root file system, and takes about five minutes. When installation finishes, review the installation log to be sure all went well. You will see quite a few packages that refuse to be installed in the local zone; these packages are only intended for use in the global zone and their omission from the local zone is entirely normal. You may also see a few errors concerning file permissions issues with the `man` pages; these can safely be ignored. Other errors should be investigated; consult the References.

If installation fails because of a problem with the zone configuration, return to the previous step and use `zonecfg` to correct the problem, then use `zoneadm` to start the installation again. The first attempt may have left behind a partially-built zone root, and if so `zoneadm install` will refuse to overwrite it. If this happens, uninstall the partially-built root before re-installing:

```
# zoneadm -z twilight uninstall
Are you sure you want to uninstall zone twilight (y/[n])? y
# zoneadm -z twilight install
Preparing to install zone <twilight>.
(... more output ...)
```

Once the zone is installed, it can be booted. The first time the zone boots it builds various internal databases and then engages in a console dialog much like the one Solaris goes through when it is first installed. To boot the zone for the first time and connect to its console, use the `zoneadm` and `zlogin` utilities:

```
# zoneadm -z twilight boot ; zlogin -C twilight
[Connected to zone 'twilight' console]
Hostname: twilight
Hostname: twilight
Loading smf(5) service descriptions: 1/103
```

First-boot configuration of the local zone

The progress indicator that appears when the zone boots for the first time will count to completion in about seven or eight minutes. Then the new zone will ask for some of the same kinds of configuration information that Solaris itself needs when it is first installed. Some of the “full Solaris” questions will not be asked because the answers are already known; for example, the Internet addresses are defined by the zone configuration and need not be re-entered. However, you must still specify a time zone, choose host

names, describe network naming services, and so on. The answers you provide can be different from those the global zone uses; for example, a local zone might use the organization's LDAP naming service even if the global zone does not.

The first question asks for the kind of terminal that is connected to the zone's console device, so the rest of the configuration procedure can format its prompts and menus appropriately. The “X Terminal Emulator” selection is usually the best choice for virtual terminal windows (including the Gnome and Putty terminals), but you may need to make a different selection if you are using some other device.

The exact sequence of questions that follows depends in part on the way you defined the zone's configuration. If you defined more than one network connection, for example, you will be asked to designate one of them as the primary connection—but if there is only one network connection it is obviously “primary” and this question does not appear.

One of the last questions asks you to specify the password for the zone's root account. This password—in fact, the root account itself—is specific to the local zone and has no connection to the root account in the global zone or in any other local zone. Once the zone is set up, you can give its root password to whatever people you choose, and they will be able to perform super-user functions within the local zone but will not even be able to log in to the global zone. Thus, you can partition the system into non-interfering “spheres of influence.”

CAUTION: The configuration dialog in some early Solaris 10 versions has a bug: If you specify the naming service as “None,” the dialog does not ask for a root password and creates a root account that is not password-protected. If this occurs, you should log in to the new zone's root account *immediately* after the zone reboots, and use the `passwd` command to set a password. To eliminate even that brief window of vulnerability you can specify a different naming service initially and create a root password in the usual way, and (after the zone reboots) log in and copy `/etc/nsswitch.files` to `/etc/nsswitch.conf` to set up the “None” naming service.

Once the configuration dialog is complete, the new local zone reboots automatically; this will take only a short time, since the configuration databases have already been built. You can then log in using the zone's console, but it is usually more convenient to disconnect from the console and reconnect using `zlogin` in a different mode. Type the tilde and period characters (`~.`) in quick succession to disconnect from the console (you may need to try more than once), then use `zlogin` in interactive mode to log in to the zone:

```
twilight console login: ~.
[Connection to zone 'twilight' console closed]
# zlogin twilight
[Connected to zone 'twilight' pts/2]
Last login: Fri Jun  9 13:36:32 on console
Sun Microsystems Inc.   SunOS 5.10       Generic January 2005
#
```

Observe that `zlogin` (which is only available to a super-user in the global zone) does not require a password to log in as the root user in the local zone. Even if the local zone's administrator changes or forgets the zone's root password, you as a global zone administrator can still exercise super-user privileges in any local zone. Consult `man zlogin` and the References for more information on the `zlogin` utility.

Check to see whether the `/dev/listen` device is accessible to the local zone by using the `ls` command. If all is well, you should see an entry for a “character special” file, like this:

```
# ls -l
crw-rw-rw-  1 root      sys          268,  0 Aug  6 17:19 /dev/listen
```

If the `/dev/listen` file exists in the zone, as shown by `ls`, all is well and you can proceed to the next step. If instead you receive a “no such file” error message, see the note on page 7; the recipe for exporting the device to a zone was different in the very earliest Solaris 10 versions.

Setting up and running Domino

As the root user in the local zone, make any adjustments that are needed to enable the user account that will run Domino: define the user and group if they're not imported through a network naming service (be sure to use the same user and group ID numbers as when you installed Domino in the global zone), add NFS mounts to the zone's `/etc/vfstab`, set up a home directory for the user account, and so on. You may also wish to make changes to the network services that run in the zone; for instance, you might want disable `telnet` in favor of the more secure `ssh`. Finally, if you plan to use Domino transaction logging you must make the directory where the logs will reside writeable by the Domino user account (the data directory is already writeable, having been made so during the installation of Domino.)

Now switch to the Domino user account. You can use `su - notesuser` from the session that's already running, or you can log out (returning to the global zone) and use `zlogin -l notesuser twilight`, or if an appropriate network service like `ssh` or `telnet` is enabled you can log in to the zone over the network.

As the Domino user in the local zone, set up Domino exactly as you would on any other system: change to the Domino data directory, set the `DISPLAY` environment variable (if necessary) to point to your screen, and start the server setup process with `/opt/lotus/bin/server` or `/opt/ibm/lotus/bin/server` (or substitute a different directory path if you specified one during installation). From this point on, Domino runs in the local zone exactly as it would on a non-zoned system: you set it up, start it, stop it, administer it, edit its `notes.ini` file and so on just as you would on any other system.

Once Domino has been set up, you may wish to install startup and shutdown scripts to start and stop Domino when the local zone boots and shuts down. You do this exactly as you would in a non-zoned system: log in to the local zone as the root user and proceed as if you had an entire machine to yourself. Consult the References for sample startup and shutdown scripts that you can tailor to your needs, and for other tips about running Domino on Solaris. Some of the References do not discuss zones specifically, but nearly everything they say is as applicable to running Domino in a local zone as to running on a non-zoned system.

Beyond the Introduction

The example above described how to run a single instance of Domino in a single local zone. Few systems, though, are quite that simple! This section describes some variations on the preceding simple system, and explains some of the special concerns that can arise. Fortunately, most of the procedures needed for these more elaborate installations are the same as those described above or are simple extensions of them.

Multiple partitions in one zone

This configuration is nearly as easy to set up as the first. Follow the same procedures as for the single-partition case, with these straightforward alterations:

- During Domino installation, specify additional data directories and additional user accounts to own them.
- While defining the local zone's configuration, export the additional data directories (and transaction log volumes, if used) to the zone.
- Define additional network addresses for the zone. They can share the same physical network device, or can be spread across multiple physical devices. (Since the network addresses are provided by the zone configuration, you do not need to do anything extra to set them up; the instructions in Domino's release notes and in the Administrator Help about creating additional network addresses can be ignored.)

Once the local zone is booted and configured, proceed just as you would when running multiple partitions on a non-zoned system: Edit each partition's `notes.ini` to define which network addresses the partition binds to, set each partition's `PercentAvailSysResources` value, and so on. The local zone behaves almost exactly like a non-zoned Solaris system, and you manage Domino partitions within it in the same way.

Multiple partitions in multiple zones (the hard way)

It is simple enough to configure, install, and boot multiple local zones, just by repeating the procedures described earlier for a single zone. And it is simple to install multiple data directories and export each to its chosen zone. However, a problem arises when you try to perform the server setup step in one of the local zones: Domino's server setup refuses to run unless *all* the installed data directories are “visible,” even if you are trying to set up Domino in just one of them. If you have installed Domino data directories in `/local/notesdata` and `/local/notesdata2` and exported one of these to the `twilight` zone and the other to the `comfort` zone, you will be unable to complete Domino setup in either zone because each will be missing one of the data directories.

One way to circumvent this difficulty is to run server setup in the global zone, and then move to the local zones for actual operation. This will require that the Unix user accounts that own the server data files must not only be known to the global zone (as they were for installation), but also be able to run in it—of course, you can disable them if you like after completing setup.

Another work-around is to mount all the data directories in one local zone temporarily, set up all the servers, and then redistribute the data directories to the intended local zones. This is more involved than using the global zone for setup, but may be attractive for installations that would like to keep application-level activities out of the global zone as much as possible. The remainder of this subsection describes this second approach.

Begin by adding all the “foreign” data directories to the local zone where you will set up the servers. As above, assume that the `twilight` and `comfort` zones mount `/local/notesdata` and `/local/notesdata2`, respectively, and that server setup will run in `twilight`. Assume also that `/zoneroots/twilight` is the point in the global zone's file system where `twilight`'s root file system is located (the value of `zonepath` from when the zone was configured). To mount `/local/notesdata2` in `twilight` temporarily, first shut down the `comfort` zone and as the root user in the global zone do

```
# mkdir -p /zoneroots/twilight/root/local/notesdata2
# mount -F lofs /local/notesdata2 /zoneroots/twilight/root/local/notesdata2
```

(If there are additional “foreign” data directories, add them to the chosen local zone in the same way.) Now log in to the local zone and set up the server partitions as described in “Multiple partitions in one zone” on page 12. When the setups are completed, return to the global zone and unmount the foreign file systems from `twilight`:

```
# umount /zoneroots/twilight/root/local/notesdata2
# rmdir /zoneroots/twilight/root/local/notesdata2
```

(The second line removes the directory that `twilight` used for the mount point. This is not strictly necessary, but is recommended to help avoid confusion.)

Multiple partitions in multiple zones (an easier way)

Much of the clumsiness in the procedures of the previous section can be avoided if you are willing to maintain a private copy of the Domino executables for each zone running Domino. This will cost an extra 700MB or so of disk space for each copy, plus extra effort in applying patches and upgrades, but may smooth out other operations enough to be worth considering.

The basic idea is simple: Each Domino installation has a file like `/opt/ibm/lotus/.install.dat` that stores information about the installation, including the locations of the installed data directories. This allows the setup and management tools to locate those data directories when performing upgrades and so on. By giving each zone its own `.install.dat` file we divide the data directories into separate “universes” that each zone can manage independently, without needing access to the directories of the other zones. This section will show how to set up two Domino instances in the `twilight` zone and three more in `comfort`, installing Domino once for each zone.

In the global zone, install the Domino kit for the `twilight` zone. Choose a suitable directory name for the Domino executables, something like `/opt/twilight/lotus`, and specify this name when the installation asks for it, instead of `/opt/lotus` or `/opt/ibm/lotus`. Install data directories only for the two Domino partitions `twilight` will handle; we'll suppose they are `/notesdat1` and `/notesdata2`.

Configure the twilight zone in the usual way, exporting the `/opt/twilight/lotus`, `/notesdata1`, and `/notesdata2` file systems as illustrated above. You can then start `twilight` at any time, and proceed as in “Multiple Partitions in One Zone” on page 12.

Meanwhile, install the Domino kit a second time, this time on behalf of the `comfort` zone. Specify a different location like `/opt/comfort/lotus` to hold the Domino executables, and install data directories for `comfort`'s partitions: `/notesdata3`, `/notesdata4`, and `/notesdata5`, say. Include these executable and data directories in the file systems listed in `comfort`'s zone configuration, start `comfort`, and set up its partitions just as you did `twilight`'s.

Since the two zones have their own executable directories, with installation information recorded independently in `/opt/twilight/lotus/.install.dat` and `/opt/comfort/lotus/.install.dat`, Domino maintenance procedures running in either zone have no need to “see” the data directories belonging to the other. All the unmounting and remounting and shuffling about of the previous section can be dispensed with.

Remember, though, that there are now two distinct sets of Domino executables on the system. If you apply a Lotus patch or upgrade to one of them, it will not automatically propagate to the other; you need to install it twice. This is a bit of extra work, but perhaps not too much more. Also, you now have the ability to test a Domino upgrade in one zone before putting it into production on the other.

Better file system isolation

The examples thus far have used the *loopback file system (lofs)* to export mounted file systems from the global zone to local zones. This is a convenient choice, but it reduces the degree of isolation because each file system is mounted at least twice: once by the global zone and once by each local zone it is exported to. Mounting the same file system in two (or more) zones creates a channel through which the zones can communicate with and possibly interfere with each other. Some files (like the Domino executables in `/opt/lotus` or `/opt/ibm/lotus`) *should* be shared by all the Domino zones, and are mounted read-only to prevent inter-zone interference. But the file systems that hold Domino databases and transaction logs cannot be mounted read-only, so another approach is desirable.

One alternative is to leave the “dedicated” file systems unmounted in the global zone and to mount them only in their local zones. This is done by exporting to each local zone not the file systems themselves, but the disk devices that hold them. Given access to the physical disk devices (access which was denied to the zones in the earlier examples), the local zone can mount file systems on those devices just as ordinary Solaris does. This section shows how to configure a local zone for direct access to the disk device.

It's actually quite simple: all you need do is describe the file systems differently to `zonecfg`. Assume as before that `twilight`'s Domino directory and transaction logs are on `/local/notesdata` and `/noteslogs`, and assume further that these file systems are on the physical devices `/dev/dsk/c1t0d0s6` and `/dev/dsk/c2t0d0s6`. When adding the file systems to the zone's configuration with `zonecfg`, instead of the subcommands shown earlier use the following:

```

zonecfg:twilight> add fs
zonecfg:twilight:fs> set dir=/local/notesdata
zonecfg:twilight:fs> set special=/dev/dsk/c1t0d0s6
zonecfg:twilight:fs> set raw=/dev/rdisk/c1t0d0s6
zonecfg:twilight:fs> set type=ufs
zonecfg:twilight:fs> add options [nodevices]
zonecfg:twilight:fs> end
zonecfg:twilight> add fs
zonecfg:twilight:fs> set dir=/noteslogs
zonecfg:twilight:fs> set special=/dev/dsk/c2t0d0s6
zonecfg:twilight:fs> set raw=/dev/rdisk/c2t0d0s6
zonecfg:twilight:fs> set type=ufs
zonecfg:twilight:fs> add options [nodevices]
zonecfg:twilight:fs> end

```

The differences are in the `special` attribute (which now specifies the physical disk device), in the addition of the `raw` attribute (specifying the raw device corresponding to the disk device), and the change of `type` to UFS (or whatever other file system you might be using).

You can make this change at any time: before or after installing the zone, even while the zone is running (although the change will not take effect until the next time the zone boots). To make these changes to an existing configuration, you will use the `select` subcommand to navigate to each file system in turn, then `set` subcommands to change its information; the `dir` and `options` settings will remain as they were, so you need not change them:

```

zonecfg:twilight> select fs dir=/local/notesdata
zonecfg:twilight:fs> set special=/dev/dsk/c1t0d0s6
zonecfg:twilight:fs> set raw=/dev/rdisk/c1t0d0s6
zonecfg:twilight:fs> set type=ufs
zonecfg:twilight:fs> end
zonecfg:twilight> select fs dir=/noteslogs
zonecfg:twilight:fs> set special=/dev/dsk/c2t0d0s6
zonecfg:twilight:fs> set raw=/dev/rdisk/c2t0d0s6
zonecfg:twilight:fs> set type=ufs
zonecfg:twilight:fs> end

```

(Tip: You can use the `info` subcommand to view the current configuration settings, either for the entire zone if you use it before `add` or `select` or after `end`, or just for a single file system if you issue it after starting work on that file system and before issuing `end`.)

Exporting the disk devices themselves gives the local zone exclusive control over a file system, which means that the file system *cannot be mounted in the global zone while the local zone is running*. If you configure a local zone this way and try to boot it while the file system is mounted in the global zone, the local zone will not boot and you will see an error message like

```

# zoneadm -z twilight boot
zoneadm: zone 'twilight': fsck of '/dev/rdisk/c1t0d0s6' failed with exit
status 33; run fsck manually
zoneadm: zone 'twilight': call to zoneadmd failed

```

On the other hand, if you try to mount the file system in the global zone after the local zone has already booted and has mounted it, the mount attempt will fail with a message like

```
# mount /local/notesdata  
mount: /dev/dsk/c1t0d0s6 is already mounted or /local/notesdata is busy
```

Since the purpose of exporting the disk devices to the local zone was to prevent double-mounting, this is as desired. However, it means you will need to take extra care when performing maintenance activities on those file systems in the global zone. For example, installing a new Domino data directory requires (as server setup does) that the all data directories be visible simultaneously in one zone. Installing a Domino update may also require access to the data directories. To carry out such tasks, you will need to shut down the local zone or zones (so they no longer mount the file systems), mount the file systems in the global zone, perform the maintenance, unmount the file systems from the global zone, and boot the local zone(s) again. This sounds like a lot of work—but running Domino is something you do every day, while updating it happens only now and then.

Isolating zone roots

Another way zones can (potentially) interact is through their zone roots, the directories that hold the local zones' simulated root file systems. Each zone root is mounted only by its own local zone, so nothing written in a zone's `/var`, for example, is visible to any other local zone. However, if all these directories reside in the same file system in the global zone, a runaway process in a local zone could write so much data to its own root that it fills up the shared file system, making other zones unable to write to their own zone roots. If the zone roots are on the same file system as the global zone's “real” root, a runaway local zone could even bring down the global zone by filling up the file system.

To contain the damage a runaway local zone can do, consider putting each zone root on a dedicated small file system. Then if a misbehaving zone fills up its own root, it still won't affect the roots of any other zones, including the global zone. The damage is contained to the misbehaving zone and does not spill over to the rest of the system.

The file systems for sparse root zones do not need to be very large: a few hundred megabytes is enough for most purposes, and half a gigabyte is generous. Create the small file system, mount it in the global zone, and specify its mount point as the `zonepath` value when you configure the zone.

You cannot relocate a zone root after the zone has been installed, so if you have already installed a zone you will need to uninstall it before `zonecfg` will allow you to change its `zonepath`. Then re-install it, boot it, and re-configure it as described above under “Installing and booting the local zone.”

Resource Management

Because Solaris supports so many kinds of applications, and because Solaris' users have so many different needs and priorities, Solaris provides many techniques for managing the allocation of computing resources. This section is a brief overview of the technologies that come under the general heading of Solaris *containers*. A full treatment of container technology is beyond the scope of this short paper, which can offer only the briefest of introductions. Consult the References for fuller descriptions and “how-to” guides for managing system resources with containers.

Workload management

In prior Solaris versions, all processes belonging to all users and running all programs competed for system resources on a more or less equal footing. The system's own tasks took precedence over users' tasks, and real-time processes took precedence over both, but all “ordinary” processes were treated pretty much alike. This egalitarian allocation of resources is appropriate for a time-shared system where all the tasks are of similar importance, but it does not lend itself to situations where tasks differ in their importance to the overall mission or where level-of-service guarantees are desirable.

The first step in addressing this issue is to be able to identify the processes that are part of some activity or service, so they can be distinguished from those belonging to less (or more) important activities. In Solaris 10 such a group of related processes is called a *project*. The system administrator defines the projects that operate on the system, and the users and groups that are entitled to run processes in those projects. For example, a system's production Domino servers could execute in the `domino_prod` project, while the test partitions run in `domino_test`.

Once the projects are identified, the system administrator can attach various resource controls to them. Continuing the preceding example, the administrator might grant the `domino_prod` project four “shares” of CPU entitlement while giving `domino_test` just one; this would guarantee the production servers access to at least 80% of the system's CPU power. The administrator can even give `domino_test` *zero* shares, forcing the test servers to get by on whatever remains after the more important projects have had their fill.

The controls on a project can govern resources other than CPU access, and can use more sophisticated actions than a simple allow-or-deny. Consult the References for more information on how to use projects for workload-centric resource management.

Resource pools

A resource pool contains a subset of the physical CPUs in a system. Projects and zones can be associated with a resource pool, causing their processes to execute only on the pool's CPUs and no others. Furthermore, the pool's CPUs will run only those processes; the pool is dedicated to the projects and zones associated with it.

To this extent, resource pools are somewhat like the *processor sets* supported by earlier Solaris versions. However, resource pools are considerably more flexible. For example, you can define minimum and

maximum CPU counts for a pool, and allow Solaris to transfer individual CPUs between pools in response to changing circumstances. You can define *objectives* and allow Solaris to seek out combinations of pool configurations that meet them. And, of course, you can redistribute CPUs among pools manually to adjust to changes in workloads. If daytime and nighttime loads are dissimilar, for example, you might use a `cron` job to redistribute pool resources each evening and morning.

Consult the References for more information on setting up and managing resource pools.

Zones

This paper has concentrated on zones as an isolation and containment mechanism, but zones also participate in resource management. You can assign a zone to a resource pool, for example, so that all its processes run on the pool's CPUs and leave other CPUs for other activities. You can specify the number of “shares” of CPU a zone's processes are entitled to. You can even specify which scheduling class is used to dispatch a zone's processes.

These “external” decisions about the resource allotments for a zone do not prevent the zone's own administrator from making additional “internal” decisions. A local zone can have its own set of projects with its own set of resource controls, for example. However, the local zone administrator cannot override the externally-imposed global system policies: the resources granted to a zone can be subdivided however the local zone administrator chooses, but cannot be increased above the levels specified by the global administrator.

See the References for more information on managing resources within and among local zones.

References

- *System Administration Guide: Solaris Containers—Resource Management and Solaris Zones*, <http://docs.sun.com/app/docs/doc/817-1592>. The definitive document for zones.
- *Solaris Containers—What They Are and How to Use Them*, <http://www.sun.com/blueprints/0505/819-2679.pdf>. A guide to workload-centric resource management. The examples center around a database deployment, but are easily adapted for Domino installations.
- *Domino 7 for Sun Solaris 10*, <http://www.redbooks.ibm.com/abstracts/sg247162.html>. A “best practices” guidebook for installing, tuning, and managing Domino servers on Solaris. Does not describe the use of zones, but contains much information of general applicability.
- *Domino on Solaris: Common Tuning Tips*, <http://www.sun.com/lotus> in the “Technical Documentation” section. A brief guide to tuning Domino and Solaris for best performance.
- Solaris man pages: Brief on-line help for command syntax, option flags, and so on. Of particular interest are
 - man -s5 zones: general introduction to zone concepts and management
 - man zonecfg: defining and modifying the configurations of local zones
 - man zoneadm: installing, booting, halting, uninstalling, and listing zones
 - man zlogin: logging in to a local zone, connecting to a local zone's console

Acknowledgments

The author thanks Craig Swain and Chien-Hua Yen, both formerly of Sun Microsystems, for their assistance in the preparation of this paper.

