17 Administering Solaris Zones

CERTIFICATION OBJECTIVES

17.01 Understanding Solaris Zones

17.02 Configuring and Reporting on Solaris Zones

17.03 Administering Solaris Zones

✓ Two-Minute Drill

Q&A Self Test

Congratulations! You've made it to the final chapter of this book. You are about to learn how to create and configure a Solaris Zone, observe it in its various states of being, and manage it as it runs on your system. For those of you who need to get on with learning about other virtualization techniques available on Solaris and other operating systems, learning how zones work will give you a solid conceptual foundation for those subjects.

You've been learning about zones already—or at least the key parts that make up one. For example, you've learned to configure projects, an abstraction for treating a group of processes as a workload. You've learned how to set limits with process, task, and project resource controls, and you've learned about global resource controls such as memory caps, process scheduling, and CPU provisioning. You can create a new ZFS file system with its own property controls for any occasion you deem fit. And you know that you can now create virtual NICs whenever you need a new logical network resource.

Using properties or attributes as a meme, each of these subsystems provides a common quality, *isolation*, that contributes to zones. Projects and resource controls let you separate a workload from other processes in your system. ZFS file systems give you configuration management as well as a separate namespace for files. VNICs let you create a network presence that looks like another station on the network.

Congratulations if you're now asking, What else could I possibly need to create a zone? You've come to exactly the right place.

CERTIFICATION OBJECTIVE 17.01

Understanding Solaris Zones

Let's cut right to it: What is a Solaris Zone? If I'm talking loosely, I'd say it's a special kind of project that runs in its own boot environment (BE). That is, it runs its own operating system and its own file systems. It maintains its own process scheduler, its own service repository, and it can support its own network services. There are a few subtler elements as well that can make it virtually separate from the machine-level operating system that supports it. From now on, I'll refer to this machine-level environment as the *global zone*.

A zone takes the isolation concept a step further by partitioning its operations from all other processes running on the same machine, whether those processes run in the global zone or in another non-global zone. This additional level of separation means you can run multiple versions of some service, such as an HTTPd server, by using zones as separate process- and namespaces. It means you can run a slew of processes in one zone without fear that misconfiguration, process failures, or other anomalies can cause failures in other zones, including the global zone.

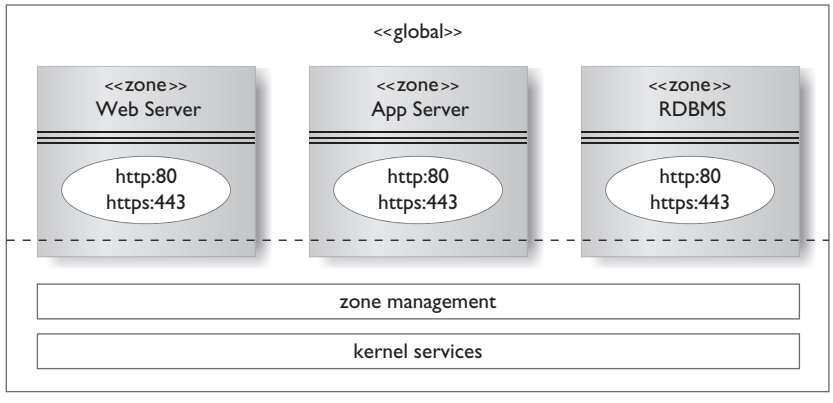
But let's back up for a minute. A lot of computing technology in the last decade, and all of this one so far, has focused heavily on virtualization. When Solaris Zones debuted in 2005, a lot people asked how they related to virtual machine technology such as VMWare. It became common to pitch zones using illustrations such as Figure 17-1.

This figure suggests the roles in a traditional multitiered environment could be fulfilled by zones running on one machine instead of running on separate physical systems. If each software package here supports its own HTTP and HTTPS services, it's no problem. Each zone can have its own network stack (and therefore its own range of ports), its own resource configuration, and so on. This picture is appealing for managers who want to consolidate their software processes, but still lots of people ask, What other operating systems can you run in a zone?

The short answer: not too many. In Solaris 11, zones running either late update versions of Solaris 10 or Solaris 11 are supported. A zone does not run on a *hypervisor* as many virtual machine implementations do. The analogous term sometimes applied to the global zone is *supervisor,* meant to suggest a facility that is much closer in form to a custom BE manager than a resource provider to different operating

FIGURE 17-1

Multiple non-global zones



system instances. In Solaris 10, there was some support for Solaris Zones running an instance of Linux, but it didn't seem to attract an audience. Bottom line: Thinking of zones as a kind of virtualization misses the point, at least in my view.

What you can create instead are as many Solaris instances as your current hardware will support; that's not a small benefit. Each one can run a workload that is, from a resource perspective, unrelated to other workloads. More to the point, it can be configured specifically for the workload(s) that it hosts: a database, an HTTP server, a batch-processing application, or something else. Many resource settings that once had to be made in the global /etc/system file are now applied to a zone, thus removing the implied dependency of all processes on global settings. Consolidation, after all, is not much of a benefit if your database has to work with the same resource configuration as your HTTP server.

Zone isolation doesn't just protect the workload, either. All non-global zones rely on the global zone for kernel resources. Those resources are shared in the sense that every zone relies on them, but they are also moderated in the sense that only the global zone permits every other zone access to them. In addition, Solaris 11 extends the idea of a zone as a separate user space with what is called *delegated administration*. Each zone can be configured for access to a specific non-root user in the global zone. Each zone's resources derive from the global zone, but it's otherwise reserved for the user(s) as a separate process space they can use without fear of conflicting with other non-local zones.

Understanding the Global Zone

The global zone is the operating system instance that manages your physical operating environment (or virtualized one). It therefore has certain privileges and capabilities that other zones you create will not have. For one, only the global zone can host and manage another zone. You cannot create a zone inside a non-global zone. Also, although non-global zones share many aspects of a BE, you cannot boot off the hardware using one. Non-global zones can only run when the global zone runs.

A non-global zone configuration, in one sense, is just a list of resource requests passed to the global zone at startup time. The global zone validates each request before initiating the zone's boot. Some resource requests may even fail without causing the zone boot to fail too. The general actions you can perform include:

- Adding and deleting zone configurations
- Installing and removing zone files
- Booting, halting, restarting, and shutting down zones

You use the zonecfg utility to create, modify, and delete zones (we'll examine this tool in the next section). When you create a new zone, its configuration is stored in the /etc/zones directory in an XML file format called a *manifest*. The full contents of this directory on a freshly installed system are shown here:

```
$ ls /etc/zones
index                   SYSdefault-shared-ip.xml  SYSsolaris10.xml
SUNWdefault.xml         SYSdefault.xml
SYSblank.xml            SYSsolaris.xml
```

This directory includes an index file that records the name and current installation state of every zone. The global zone is listed in the index but is not represented with an XML manifest by default. If, however, you use the zonecfg utility to add resource controls to the global zone, it will write a global.xml file that reflects the changes. Other XML files that start with SUNW or SYS are templates for stock configurations, including legacy Solaris 10 types.

Zone manifests can be modified at any time. However, changes made to a manifest cannot be pushed into the running zone; doing so requires a zone reboot. Resource control changes are different. Any control you can modify using the rctladm and prtcl utilities will take effect on a running zone.

Understanding Zones and Resource Management

The benefit of applying resource controls and caps to suit a zone's workload is hard to overstate. The term Solaris Container, often understood as a synonym for Solaris Zone, is actually meant to describe a zone that has been configured with resource controls and caps to accommodate its workload. Given some of the architectural differences between projects and zones, zones are even better suited to using the Fair-Share scheduler and operating on memory caps, CPU shares, and CPU pools.

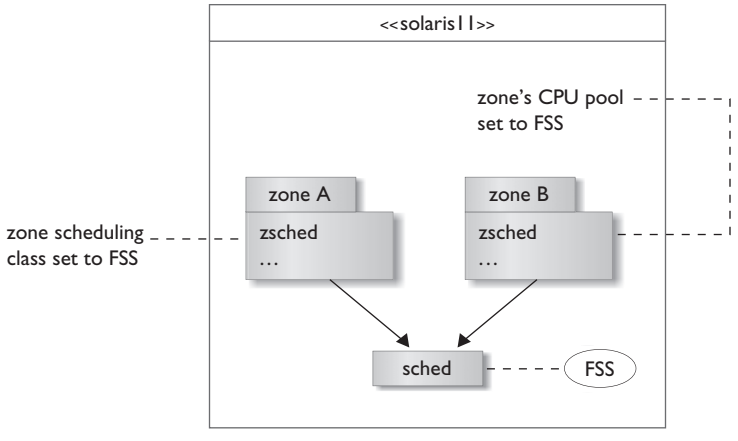
As mentioned earlier, each non-global zone is responsible for its own process scheduling. By default, a zone will just inherit the default scheduling class of the global zone, which you can change with the dispadmin -d command, as shown in Chapter 10. You can also assign a scheduling class specifically to a zone to override the global zone default. Or, you can run the dispadmin -d command inside a local zone and reboot. If you assign a CPU pool to a zone and the pool itself has a scheduling class configured, that will also take precedence over the global zone's default scheduling class. It's a mess to sort out all these options and the precedence among all of them. Fortunately, you're not responsible for preparing that on the OCA exam.

This complexity of control is due in part to the non-global zone scheduling design. Non-global zones manage their process demands through a process called zsched. This process only runs inside a non-global zone. All processes in all zones, however, remain visible in the global zone. Non-global zones can't hide what they're doing from the global zone, only from each other. Processes in non-global zones must work through their local zsched process to get CPU time, which is only supplied by the global zone. Figure 17-2 illustrates this relationship, along with the various ways a scheduling class can be assigned to a zone.

The Fair-Share scheduler, when applied to one or more zones, makes it easy for the global zone to guarantee some minimum chunk of time to each zone, which in turn can distribute the time among its processes, according to the local scheduler of choice. It's not required that global and non-global zones run the same scheduler, but it does make sense that the global zone distribute resources to zones using a system that does not take individual process priority into account but rather the relative importance of each zone as expressed in shares. The same logic holds for distributing time to various projects.

FIGURE 17-2

Non-global zones running zsched processes



Understanding CPU Caps, Assigned CPUs, and Pools

Allocating CPUs, as opposed to CPU time, can take several forms in zone configuration. You can cap CPU resources with the resource control zone.cpu-cap. This control virtualizes the power of a CPU, treating 1 percent of the CPU (available to user processes only) as its unit of measure. There are 100 units of CPU power for every CPU on the system. Using this control, it's possible to assign a fraction of a CPU to a zone, irrespective of its scheduling class or other allocations.

A Solaris 11 zone has a configurable property called capped-cpu. It is an alias for the zone.cpu-cap resource control, but uses fractional values instead of integers to express a portion of a CPU. If you had planned to set the zone.cpu-cap control to 125, you could set the zone's capped-cpu property to 1.25 and achieve the same effect. The configuration property is recorded in the zone's manifest file, so it's a little easier to track than the resource control.

Solaris Zones also have a dedicated-cpu property that is different. It takes two values: an integer representing some number of available CPUs and an importance property, which is optional. With this property set, the global zone will allocate the number of CPUs declared by the non-global zone at startup, place them in a pool that it constructs on the fly, and then assign the pool to that zone. This approach to making a pool for a zone is not only fully automated, it is also constructed and torn down according to the zone's current state. The administrator only has to make sure the CPUs requested are available when the zone is started; if they aren't, the zone will not run. The importance value is applied by the poold process only if the number of CPUs required by a zone is defined by a range. Importance tells the poold process the degree to which a pool should receive all CPUs it will take or just some of them.

Chapter 10 briefly discussed that creating and using pools is mutually exclusive to creating psets by hand. It simply puts undue complexity on the system to moderate between managing psets by hand (using the psradm utility) and using pools to contain them. A similar tension exists between configuring pools by hand and letting the system configure them for zones that use the dedicated-cpu property. You can also define pools by configuration at the same time the system defines them on startup, just not for the same zone. That said, administration may be kept simple by preferring one approach to the exclusion of the other.

For any one zone, then, you may leverage the cpu-shares property, or assign a preconfigured pool to it, or use the dedicated-cpus property. You do not have to choose one way for all zones, but it may be difficult to keep all your CPU configurations straight if you mix and match configuration approaches as you go.

Understanding Zone Networking

The VNIC brings a completely new dimension to configuring network resources for a zone in Solaris 11. Early releases of Solaris 10 supported a shared-ip network configuration only. With a shared-ip attribute, the zone receives a logical interface that is created and added to a NIC controlled by the global zone. These interface names have a format such as bge0:1. The zone has no administrative control over the NIC and only some control over the IP interface supplied to it.

Later versions of Solaris 10 introduced the exclusive-ip interface. Using this property, the zone received complete control of the network interface, including the ability to plumb it, create logical interfaces, and assign IP addresses to it. As a control, the zone could also be configured with a restricted range of IP addresses and a default router address. This arrangement gave the zone administrator more control, but at the cost of removing the NIC, more or less, from the global zone view.

Solaris 11's fully virtualized network components have changed all that, and along with it the Solaris view of what the shared-ip and exclusive-ip properties represent. Now that you can create a VNIC on demand, there is no real need to use a shared-ip arrangement, just as there is no longer an either/or arrangement necessary for this NIC. You can, if you wish, simply assign any zone its own VNIC in exclusive-ip form. The global zone retains full visibility of the physical NIC, and the local zone gets full administrative flexibility. It's a win-win outcome.

In fact, Solaris 11 takes this new flexibility one step further with an automatic network resource, or anet, added to the default zone template. In short, when you create a zone, install it, and boot it, the system will create and configure a VNIC on the fly, get an IP address for it, bind it to the zone, and tear the whole thing down once the zone halts or shuts down. Similar to the effects of the dedicated-cpu property, Solaris Zones have a built-in convenience to make much of the process look like "Things Just Work."

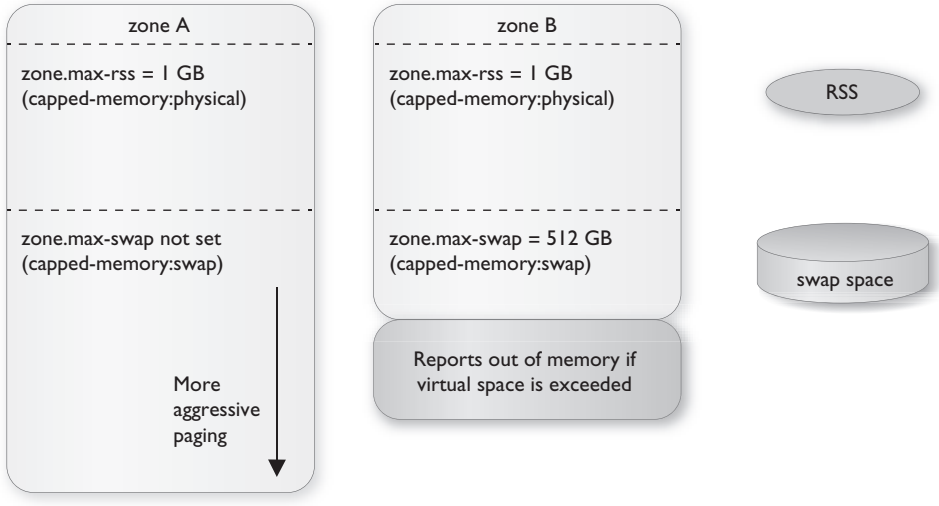
You can continue to use the shared-ip setting or add net resources into a zone configuration, but in most cases there is probably no need. Where you have a custom configuration in mind, you can just remove anet resources from a zone or use a template that doesn't include them.

Understanding Zone Memory

In Chapter 10, you learned a bit about the difference between a process's virtual and resident memory sizes. Process virtual size (or SIZE in ps and prstat command output) is the total memory space the process requires the system to account for. Resident size is the amount charged against system RAM, and in many cases is a smaller number. By applying the rcap.max-rss resource control to a project (or zone), you can regulate the resident memory demand a workload makes against the system. You can also tune the vigor of the system's enforcement policy, which pages resident memory out to swap space (disk) until the workload demand falls below its cap again.

One unfortunate side effect of rcapd's enforcement behavior is that paging memory to disk can cause delays to other processes that get hung up on disk activity due to excessive paging. There are remedies, such as relaxing enforcement or raising the cap limits, but the threat of an unexpected system slowdown from abusive paging doesn't build confidence in the system. The problem is that swap space is a global resource governed only by the global zone. When local zones process pages heavily, the system reaction might impose delays on all other processes as a result. What's needed is a control to limit swap space as well as resident space. With both controls in play, a memory-hungry zone could run out of memory without getting a chance to exhaust the global supply. Figure 17-3 shows the difference between a zone that can access all of system swap and one that can't.

FIGURE 17-3 TZone with system swap vs. zone with limited swap



A zone with 1GB allotted for RSS and 512MB for swap has a total of 1.5GB available for all process maps. If the workload exceeds 1GB of RAM, the rcapd process kicks in and attempts to lower it. If the process demand still increases up to 1.5GB and tries to go further, all subsequent memory requests are denied; the zone cannot start more processes until some memory is released. The zone that places no limit on swap exhibits the same behavior, but to the limit of swap that has been made available for the entire system.

To that end, zones have a capped-memory property with three sub-properties: physical, swap, and locked memory. These sub-properties are aliases for the resource controls zone.max-rss, zone,max-swap, and zone.max-locked-memory, respectively. The zone.max-locked-memory control defines the total physical memory a zone's processes can pin down (that is, prevent from being paged out). If you enable capped-memory, you must define at least one of these three sub-properties to go with it.

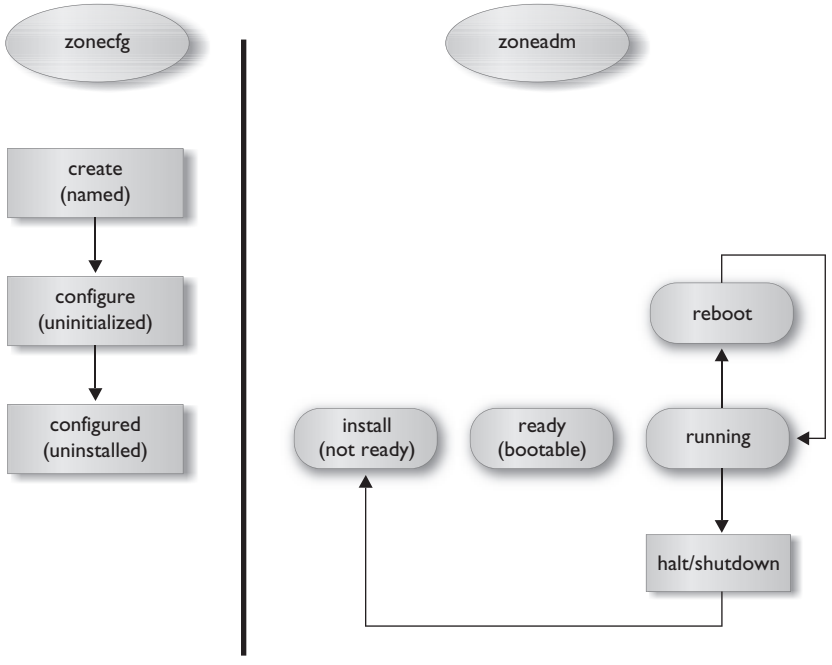
Understanding Zone States

You use the zoneadm utility to install and remove zone files or to change the run state of an installed zone. I glibly said earlier that a zone is a lot like a project inside a customized BE, and in concept there are a lot of similarities. In technical terms, however, a zone that has been configured but not yet installed is in actuality an IPS *target*. That is, it's a location suitable for receiving and installing packages from an IPS repository. When a zone installs, it applies an image to the file system created for the zone. When the installation completes, you can then review it through the log messages that are recorded during the process, just as you would with a machine install. The process doesn't take as long as a machine install, or shouldn't, but it covers the same steps.

Figure 17-4 illustrates the stages from creating a zone to changing run states from the global zone's perspective. Note that this flow of changes is observed and enforced by the utilities in both directions. That is, the tools will not delete a running zone without halting it first, or let you delete a zone before you uninstall it, and so on. The current state is protected simply by recording it in the /etc/zones/index file.

Changing a zone's state is a nontrivial assignment, and it's not handled by the sched process in the global zone. The zoneadm process instead will spawn a zoneadmd process to handle the state transition of any one zone. The zoneadmd program, stored in the /usr/lib/zones directory, is not intended for administrative use. It shuts down once the task assigned to it is done, and it's managed through the SMF service svc:/system/zones:default. You are most likely to observe this process while a zone is booting or shutting down.

FIGURE 17-4 Transition from zone creation to run states



Understanding Branded Zones

Although Solaris Zones were not designed with virtual machine hosting in mind, some variations from the global operating system environment called *branded zones* are supported. In Solaris 11, *branded zones* support Solaris 10 instances. This support makes it possible to migrate existing Solaris 10 zones (or physical installations) to hardware that is running on Solaris 11. If you have applications with heavily customized, nonportable configurations, or a setup that is hard-coded to Solaris 10 features and services, a branded zone gives you a way to take advantage of more recent hardware—or at least to not be stuck running on a legacy system that Solaris 11 does not support.

The template for a Solaris 10 branded zone is SYSSolaris10.xml. All templates include the brand attribute, but the SYSSolaris10.xml file is the only one whose brand value isn't "solaris" but "solaris10." The Solaris 10 zone implementation used brand as an optional attribute. You don't have to do anything special to create a branded zone other than use the appropriate template. The mechanisms that support a Solaris 10 branded zone are baked into the kernel and are only otherwise evident by the features and behavior of the branded zone when it runs.

Understanding Access to Non-Global Zones

A local (or non-global) zone resides in the global zone. So how do you access it? It depends on who you are. Users in the global zone with sufficient privileges can treat the zlogin utility as console access, which requires superuser login and authentication by default. To get this type of access, use the -C switch. With the switch, the zlogin utility will access the zone using a baked-in sudo operation; there is no authentication challenge in that case. The utility takes the zone name as a parameter.

All other users can access a zone like any other remote system. A Solaris Zone will generate its own keys for SSH validation. If you set up services inside a zone, clients' systems can of course access them like another network station.

CERTIFICATION OBJECTIVE 17.02

Configuring and Reporting on Solaris Zones

Quite a few configuration options are available with a zone. A good zone configuration is one that complements well the workload you plan to run in it. It's then a matter of learning what each control does and how each can help you create a more complementary environment for your workload.

exam Watch

Items in the OCA exam are specifically checked against what are called* memory stunts, *or questions that emphasize your ability to recall a diverse amount of technical detail. That means you won't (or shouldn't) receive questions that ask you to pick a bogus resource control out of a list or something similar.

Solaris 11 does *not* support the sparse root model that was popular in Solaris 10. A so-called sparse-root zone consumed very little disk space—less than 100MB. Although default zones in Solaris 11 are nothing like the Solaris 10 whole root zone model in size, they also aren't small: about 350MB. Then again, most people don't have the same concerns over storage space these days as ten years ago. Most people won't miss a 200MB here or there, not in a time when new laptops are shipping with 1TB drives in them.

The configuration process for a zone is straightforward: decide where you will support it (on disk) and what special resource requirements it may have. These requirements may include:

- Additional storage dedicated to the zone
- Multiple network interfaces
- Resource controls to support middleware or databases
- Direct access to physical devices in the global zone

These requirements fall into the realm of advanced configuration and fall outside the scope of introduction. For any requirements below this level of complexity, the following section will cover the proper technique for configuring a zone to accommodate them.

Remember that the zone configuration is just that: a manifest file that does not interact with an active zone. If you want to make changes to a zone, it's a good idea to incorporate those changes into the zone manifest, so they can be quickly read in when the zone is restarted. You cannot change the configuration of a running zone, other than its resource controls.

Reporting Zone Utilization

Once you have zones up and running. You can observe them in more or less detail for their resource consumption. The prstat -Z command will give you output similar to what the prstat -J command does for projects:

```
$ prstat -Z -c 1 1
Please wait...
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  8342 mfernest   98M   41M run      1    0   0:00:35 2.6% plugin-containe/3
  1543 mfernest   77M   53M run     59    0   0:22:23 1.3% java/20
…
ZONEID    NPROC  SWAP   RSS MEMORY      TIME  CPU ZONE
     0      119 2841M  738M    47%   0:42:46 8.0% global
     1       31  155M   88M   5.6%   0:01:37 0.6% oca17
Total: 150 processes, 918 lwps, load averages: 0.18, 0.14, 0.27
```

Notice that a local zone without additional configuration runs a modest complement of processes (31) and consumes an equally modest amount of resident memory (88MB). For anyone who thinks comparing a zone to a virtual machine is reasonable, feel free to start here! It's very cheap on system resources to idle a zone.

For more detailed analysis, Solaris 11 offers the zonestat utility. It's a rich tool; you can report on CPU, memory, networking, and resource control utilization against one, many, or all of your zones. You can also define which columns you want in your output and manipulate the report intervals in some rather subtle ways. The reports are also quite busy on the screen. Here, for example, is a summary of zone demand over one five-second interval:

```
$ zonestat 5 1
Collecting data for first interval...
Interval: 1, Duration: 0:00:05
SUMMARY                   Cpus/Online: 1/1   PhysMem: 1571M  VirtMem: 2595M
                    ---CPU----  --PhysMem-- --VirtMem-- --PhysNet--
                 ZONE  USED %PART  USED %USED  USED %USED PBYTE %PUSE
              [total]  0.22 22.6% 1300M 82.7% 1590M 61.2%     0 0.00%
             [system]  0.21 21.6% 1232M 78.4% 1532M 59.0%     -     -
                oca17  0.00 0.94% 67.9M 4.32% 57.9M 2.23%     0 0.00%
```

The summary line shows aggregate values for memory against which the consumption by zones is easier to interpret. The global zone, signified by [system], is eating the lion's share of system resources at the moment. The zonestat utility comes with many different options for producing more specialized reports and greater detail. For the sake of the exam, familiarize yourself with the kinds of information you can retrieve with the zonestat utility.

CERTIFICATION OBJECTIVE 17.03

Administering Solaris Zones

Understanding how zones bring workgroup isolation and resource management together with a BE-like operating state is the hard part. Getting a zone up and running, by contrast, is easy work. The first time through might seem like a lot of details you have to handle, but that fog lifts for most people very quickly. In this section, we'll walk through the steps illustrated in Figure 17-4:

- Configuration
- Installation
- Booting
- Logging in

- Exiting a zone
- Halting
- Removing and deleting

We'll stick to straightforward actions in each step. Bear in mind there are variations of all sorts. Now that you know generally what resources a zone may include, the goal is to get a simple one up and running that you can mangle and re-create as much as you like.

Configuring a Zone

To configure a zone with the fewest possible options, you will need:

- A name for your zone
- A location for the zone's file system

That's it for starting a zone in Solaris 11. Most values require no initial setting or have a default that suffices for simple configurations. You can use the zonecfg utility in interactive mode to declare the zone, configure it, validate it, and write the configuration to a file. The whole session looks like this:

```
$ sudo zonecfg -z oca17
oca17: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:oca17> create
create: Using system default template 'SYSdefault'
zonecfg:oca17> set zonepath=/export/oca17
zonecfg:oca17> verify; commit; exit
$
```

The *-z* switch introduces the zone you want to add (or modify). In interactive mode, the zonecfg utility builds and maintains a zone configuration in memory during the session. You can write the contents of this session to file using the commit subcommand, so long as the in-memory configuration is complete and valid. You can use the verify subcommand to do this on demand, but if you intend to exit the session, the calls to verify and commit are implied. The tool will warn you if you attempt to exit with an incomplete configuration.

The create subcommand invokes a template to give a new zone a starting point. You can also choose another template available in the /etc/zones directory. After that, all you require is a zonepath, which the global zone will guard against use by other processes or newly declared zones.

TABLE 17-1 Select Zone Properties Reported by the zonecfg info Command

Zone Property	Default	Description
zonename	*None*	*Assigned name*
zonepath	*None*	*Install location*
brand	solaris	Version supported (10 or 11)
autoboot	False	Boots with the system
pool	None	Assigned CPU pool
scheduling-class	global default	Process scheduler
ip-type	exclusive	Control type for network resource(s)
anet	linkname: net0	Multiple link properties

You can review your current configuration at any time with the info subcommand. The output is about 30 lines long and includes all the default or blank settings a zone can have. About half of this output is the properties from the anet resource, few of which concern us for the OCA exam. Some of the properties included in the output are listed in Table 17-1.

You can also call the zonecfg info command without an interactive session to see what's what. The command requires that you name the zone before passing the subcommand, like so:

```
$ zonecfg -z oca17 info
zonename: oca17
zonepath: /export/oca17
brand: solaris
autoboot: false
…
```

You can also observe the state of the zone with the zoneadm list command. You can specify whether you'd like to see (c)onfigured or (i)nstalled zones, along with ones that are running, but the switches themselves become redundant for zones in an advanced state. In other words, all zones that are installed are, by definition, configured. All running zones are of course already installed. You only need the -i switch to list zones that are not yet configured, and the -c switch to list zones not currently running. Also, be sure to use the -v option; otherwise, you'll just get a list of zone names, nothing more:

```
$ zoneadm list -cv
  ID NAME             STATUS     PATH                    BRAND    IP
   0 global           running    /                       solaris  shared
   - oca17            configured /export/oca17           solaris  excl
```

If you run this output for yourself, you'll notice that several of the properties discussed earlier, such as capped-memory, aren't listed by default. What you see in the default output are simple top-level properties. Property keys that may have more than one value (or sub-property) are added using the add subcommand. If you wanted to configure a capped-memory control in the oca17 zone, it would look like this:

```
$ sudo zonecfg -z oca17
zonecfg:oca17> add capped-memory
zonecfg:oca17:capped-memory> set physical=1g
zonecfg:oca17:capped-memory> set swap=512m
zonecfg:oca17:capped-memory> end
zonecfg:oca17> verify; commit; exit
$ zonecfg -z oca17 info capped-memory
capped-memory:
      physical: 1G
     [swap: 512M]
```

The zonecfg command has an add subcommand that requires a valid property. It opens a scope for setting the properties it contains and is closed with an end subcommand. The explicit verify and commit subcommands are unnecessary here, but I've gotten in the habit of spelling out what's happening. Also note you can limit the info subcommand output to the property you name.

Delegating Zone Administration

In Solaris 11, you can delegate some zone privileges to a non-root user in the global zone. The property name is admin. It hosts keys for a user and the auths you want to assign to the user. The auth values are login, manage, and copyfrom; this last authorization lets the authorized user clone the zone. The manage authorization gives the user root privileges inside the zone:

```
zonecfg:oca17> add admin
zonecfg:oca17:admin> set user=mfernest
zonecfg:oca17:admin> set auths=login,manage
zonecfg:oca17:admin> end
zonecfg:oca17> exit
Found user in files repository.
UX: /usr/sbin/usermod: mfernest is currently logged in, some changes may not
take effect until next login.
Found user in files repository.
```

You can add other users with an additional admin property, if you like. You can also use the delete subcommand to remove a property. If there's more than one property type, specify a key-value that matches your target.

```
zonecfg:oca17> add admin
zonecfg:oca17:admin> set user=joe
zonecfg:oca17:admin> set auths=login,manage
zonecfg:oca17:admin> end
zonecfg:oca17> remove admin user=mfernest
zonecfg:oca17> exit
Found user in files repository.
Found user in files repository.
```

Once the zone has been installed and started, the user joe may use zlogin from the global zone and enjoy root privileges inside the zone.

Installing a Zone

To install a zone, use the following command:

```
$ sudo zoneadm -z oca17 install
```

The process starts by creating a ZFS file system at the zonepath and opening a log to record the install progress. The initial output looks like this:

```
A ZFS file system has been created for this zone.
Progress being logged to /var/log/zones/zoneadm.20130129T051134Z.oca17.install
       Image: Preparing at /export/oca17/root.

 Install Log: /system/volatile/install.4249/install_log
 AI Manifest: /tmp/manifest.xml.m8a4ri
  SC Profile: /usr/share/auto_install/sc_profiles/enable_sci.xml
    Zonename: oca17
Installation: Starting ...
```

If you're short on storage space, the installation could fail. Otherwise, the process resembles an image install. The process will create a plan, "download" the files necessary, and install them to the zonepath. Once it is finished, your new zone will appear in an installed state and will be ready to boot.

The installation may also proceed slowly if the origin for your solaris publisher is still http://pkg.oracle.com/solaris/release. You're installing at Internet speed, if that's the case. If you're also running the process in a virtual machine with a modest memory complement, it might be worth your while to go for a short walk while the installation does its thing.

Booting and Accessing a Zone

Once you're past the installation, access to a zone is largely unsurprising and straightforward. To boot a zone, for example, simply name the zone and invoke the action:

```
$ sudo zoneadm -z oca17 boot
```

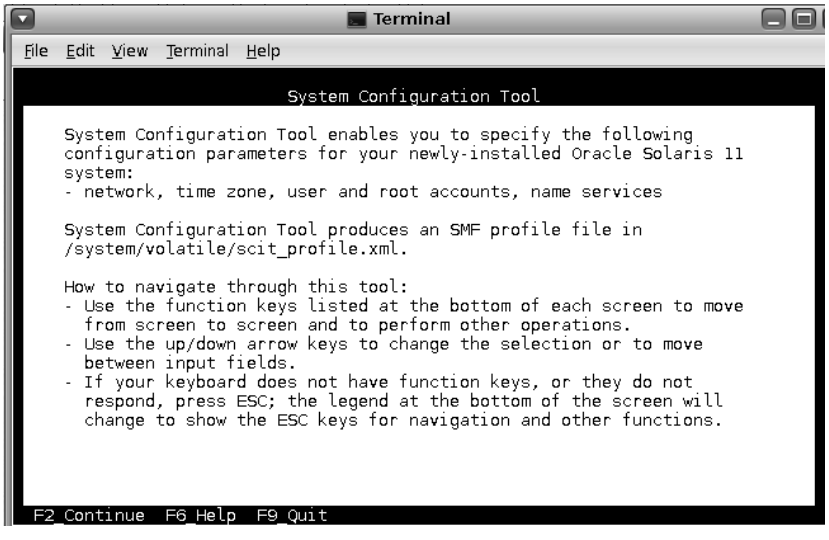
The first time you boot a new zone, it must initialize the services repository and perform some other housekeeping, just like a regular installation. Here, it's important to remember the distinction between the zlogin *zonename* and zlogin -C *zonename* commands. The first command behaves like an su command and lets you assume zone credentials, assuming you have the right credentials in the global zone. The second command gives you a console view of the system, which includes a view of the sysconfig process. *Note that you cannot see this setup process without the console view*. If you use the zlogin command to enter a zone and there doesn't seem to be anything going on, back out and try the zlogin -C command instead:

```
$ sudo zlogin -C oca17
[Connected to zone 'oca17' console]
112/112
```

The numbers show the count of installed SMF services to expected services. After a short pause, you should then see the first screen of the System Configuration Tool Wizard, as it appears in Figure 17-5.

FIGURE 17-5

System Configuration Tool Wizard screen



Refer to this screen progression in Chapter 2 if you need a refresher. The differences between system configuration and a global zone are minor and do not require additional scrutiny here.

In general, a fresh Solaris Zone doesn't hold too many surprises. It is just a replica of the global zone. A zone is supposed to behave transparently in most respects so that applications don't have to be "zone sensitive" by default. Local zones, however, are under no obligation to make their configuration invisible. You can use several commands to determine if you're operating in a non-global environment. Once you've configured your zone and let it reboot, log in and try the following commands:

```
mfernest@michaelz:~$ hostname
michaelz
mfernest@michaelz:~$ zonename
oca17
mfernest@ michaelz:~$ ifconfig net0
net0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 4
        inet 10.0.2.16 netmask ffffff00 broadcast 10.0.2.255
mfernest@michaelz:~$ zpool list
NAME    SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  5.94G  1.09G  4.85G  18%  1.00x  ONLINE  -
mfernest@michaelz:~$ zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
rpool                         370M   398M  30.9K  /rpool
rpool/ROOT                    369M   398M  30.9K  legacy
rpool/ROOT/solaris            369M   398M   339M  /
rpool/ROOT/solaris/var       28.0M   398M  27.0M  /var
rpool/export                 96.1K   398M  31.9K  /export
rpool/export/home            64.2K   398M  31.9K  /export/home
rpool/export/home/mfernest   32.4K   398M  32.4K  /export/home/mfernest
```

That's more than enough indication we're in a different operating space. Incidentally, I configured my net0 VNIC to get an IP from the DHCP server. The pool size derives from a separate 6GB logical volume that I attached to my virtual machine and am using to host the zones I create. From the local zone's perspective, it's the rpool. From the global zone, it looks like this:

```
$ zpool list bloom
NAME    SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
bloom  5.94G  1.09G  4.85G  18%  1.00x  ONLINE  -
```

Notice, however, that the file system's perceived space is much smaller. I applied a quota at the zonepath of the local zone from the global zone, limiting the entire space to 768MB. From the zfs list output, you can see that 370MB or so is consumed by files installed in the rpool, leaving (in this case) just under 400MB to use. I included this part to show there's no zone configuration control that limits storage space. The local zone administrator could impose one, but it would be effectively voluntary.

Exiting a Zone

If you have used zlogin to access a zone, you can just type **exit**. You'll be dropped back into the global zone, just as if you had exited a shell or terminal session:

```
$ sudo zlogin oca17
[Connected to zone 'oca17' pts/3]
Oracle Corporation      SunOS 5.11    11.0   November 2011
root@michaelz:~# ls
root@michaelz:~# exit
logout

[Connection to zone 'oca17' pts/3 closed]
```

If you connected using zlogin -C, escaping the zone is a little trickier. A console session in a local zone behaves like a hard-wired, serial connection. If you type **exit**, the session will simply issue a new login prompt:

```
$ sudo zlogin -C oca17
[Connected to zone 'oca17' console]

michaelz console login: mfernest
Password:
Last login: Tue Jan 29 07:08:08 on console
Oracle Corporation      SunOS 5.11      11.0    November 2011
mfernest@michaelz:~$ ls
mfernest@michaelz:~$ exit
logout

michaelz console login:
```

To break out altogether, you have to signal the terminal controller (managed within the zone) and interrupt it. Pressing the tilde and dot keys, one after the other,

will do the trick, but it can take more than one attempt. Sometimes the terminal controller is interrupted, does not catch your two-key sequence, and appears to ignore you. It's best to keep trying so you signal to the controller that the session is done:

```
michaelz console login: ~.
[Connection to zone 'oca17' console closed]
```

Halting and Deleting a Zone

The zoneadm and zonecfg utilities provide a built-in protection for the state transitions of a zone, as mentioned earlier. You might not think you'll catch yourself trying to delete a running zone. The protection isn't there for intentional acts, however, but for accidents committed in haste and the like. Try this out on your own installation: Use zonecfg to delete a running zone and see what happens. Nothing bad will happen, promise.

On the other hand, the steps for removing a running zone may seem tedious when you know you want to remove it and prefer not to waste time. It's a double-edged sword, of course: One that errs on the side of caution. Even if you follow the steps in the correct order, you'll be challenged each time you want to destroy zone data. In the following example, I uninstall the zone I created and then delete it. I use the zoneadm list command before and after to confirm the state of the zone I am dismantling:

```
$ zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              solaris  shared
   - oca17            installed  /bloom/oca17                   solaris  excl
$ sudo zoneadm -z oca17 uninstall
Are you sure you want to uninstall zone oca17 (y/[n])? Y
Progress being logged to /var/log/zones/zoneadm.20130129T085457Z.oca17.uninstall
$ sudo zonecfg -z oca17 delete
Are you sure you want to delete zone oca17 (y/[n])? Y
Found user in files repository.
$ zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              solaris  shared
```

You can also examine the log file, which records the progress in both directions, building up and tearing down.

CERTIFICATION SUMMARY

Solaris Zones use something from nearly every new technology we've covered in this book. I've repeated a few times that a zone is a project, more or less, running in its own boot environment with its own file system and network resource. Coupled with resource controls, caps, and pools, a zone uses almost everything Solaris has to offer. It's an excellent mode for reviewing all the concepts and skills you need for the OCA exam.

Speaking of the exam, keep in mind that you'll be tested on the basics of zones: how they operate; what a branded zone is; how to configure, install, boot, and shut down a zone; where configuration files are kept; and which commands provide zone-specific reports (such as zonestat). If the idea that zones are mostly a composite of other Solaris technologies seems right to you, I think you're in good shape to review a few questions and get yourself ready to pass the exam. Good luck!

TWO-MINUTE DRILL

Understanding Solaris Zones

- ❑ Solaris Zones add partitioning and run-state management to the project/resource controls paradigm, adding stronger isolation to the workload concept.
- ❑ The global zone acts like a supervisor, managing the requests non-global zones make for resources and monitoring their behavior and state.
- ❑ The gateway to the global zone from a local zone is the zsched process.
- ❑ The zoneadmd process manages local zone transitions from the global zone's process space.

Configuring and Reporting on Solaris Zones

- ❑ All zone models in Solaris 11 are a form of whole root.
- ❑ A local zone is an IPS image, or installation target.
- ❑ The prstat -Z and zonestat utilities provide high-level and detailed views of zone resource consumption, respectively.

Administering Solaris Zones

- ❑ The only required elements to configure a new zone are a name and zonepath; everything else can be configured later.
- ❑ Some zone properties are top level, are always visible with the zonecfg info subcommand, and can be modified with the set command. Other properties must be added because they involve setting multiple sub-properties or more than one value.
- ❑ The zlogin and zlogin -C commands are different ways of connecting to a local zone. Without the -C argument, the user invokes operation, similar to what the su utility does, that overrides the authentication process. The -C option forces a console-like access routine that must be broken with the tilde-dot (~.) key combination.

SELF TEST

Use the following questions to test your recall and understanding of the chapter's material.

Understanding Solaris Zones

1. Which three items are kept in the /etc/zones directory?
   - A A global zone configuration file
   - B A blank template file
   - C A Solaris 10 branded zone
   - D An index file
2. Which two options cannot be made into a zone and hosted on a Solaris 11 machine?
   - A A Solaris 10 zone
   - B A Linux virtual machine
   - C A Solaris 10 physical instance
   - D A Solaris 11 boot environment
3. Under what circumstances can a local zone host another zone?
   - A None
   - B If the global zone detaches it
   - C If it is configured but not yet installed
   - D If it is defined but not configured

Administering Solaris Zones

4. Which of the following zone properties is an alias for one or more resource control?
   - A admin
   - B capped-cpu
   - C scheduling-class
   - D importance
5. Which of the following methods will set a zone's scheduling class to FSS?
   - A Deleting the /etc/dispadmin.conf file
   - B Putting the zsched process in FSS
   - C Putting zoneadmd in FSS
   - D Changing the global zone's default scheduler to FSS

6. Which two are properties of the capped-memory control?
   A crypto
   B swapped
   C physical
   D locked

Configuring and Reporting on Solaris Zones

7. Which option is not a valid authorization of the admin property?
   A copyto
   B login
   C copyfrom
   D manage
8. Which two zone properties must be unique to one zone?
   A zonename
   B vnic
   C pool
   D ncpus
9. Which command-subcommand pair is not valid?
   A zoneadm halt
   B zoneadm info
   C zonecfg boot
   D zonecfg add
10. How many zones can participate in a shared-ip arrangement?
    A OneTwo
    B All the zones on the system
    C None in Solaris 11

SELF TEST ANSWERS

Understanding Solaris Zones

1. ☑ **A, B,** and **D.** The /etc/zones directory maintains an index that tracks all zones on the system, templates for stock zone configurations, and XML files for any local zones. It will also host an XML file for the global zone if resource controls are applied to it.
   ☒ **C** is incorrect. No zones are stored in the /etc/zones directory.

2. ☑ **B** and **D.** Solaris 11 does not support branded zones for Linux as Solaris 10 did. There is no method for converting a BE into a zone.
   ☒ **A** and **C** are incorrect. Either one is the target candidate for a branded zone.

3. ☑ **A.** Only the global zone can host and manage other zones.
   ☒ **B, C,** and **D** are incorrect. There are no conditions under which any zone other than the global zone may manage another zone.

Administering Solaris Zones

4. ☑ **B.** The capped-cpu property is an alias of the zone.cpu-cap resource control that uses a different unit of measure.
   ☒ **A, C** and **D** are incorrect. **A** and **C** are incorrect because they have no counterparts in the resource controls. **D** is incorrect because it is part of the ncpus property.

5. ☑ **D.** If the global zone boots up with the FSS scheduler as its default, local zones will follow suit and use the same scheduler.
   ☒ **A, B,** and **C** are incorrect. **A** is incorrect because deleting /etc/dispadmin.conf will revert the global zone to the TS scheduler after a reboot. **B** and **C** are incorrect because changing the scheduling class of the zsched and zoneadmd processes will not force other processes to follow suit.

6. ☑ **C** and **D.** The three memory controls for the capped-memory property are physical, swap, and locked.
   ☒ **A** and **B** are incorrect. **A** is incorrect because there is no zone property that addresses crypto memory. **B** is incorrect because the control defines swap space, not swapped memory.

Configuring and Reporting on Solaris Zones

7. ☑ **A.** The three available auths of the admin property are login, manage, and copyfrom.
☒ **B, C,** and **D** are incorrect. These are all correct auths for the admin property.

8. ☑ **A** and **B.** The zone namespace is flat, so every zone's name must be unique. Each VNIC can be assigned to one zone.
☒ **C** and **D** are incorrect. **C** is incorrect because any number of zones may be assigned to the same pool. **D** is incorrect because zones are not constrained from declaring the same number of CPUs for themselves as another zone.

9. ☑ **C.** The zonecfg utility cannot boot a zone. The zoneadm utility is solely responsible for managing the lifecycle of local zones.
☒ **A, B,** and **D** are incorrect. These are all valid command/subcommand combinations.

10. ☑ **C.** All local zones can "share" the NIC controlled by the global zone. Each one will receive a different logical interface on which to support an IP address.
☒ **A, B,** and **D** are incorrect. **A** and **B** are incorrect because there is no exclusive or limited relationship between the local zones and a global zone and a NIC. **D** is incorrect because Solaris 11 does support the shared-ip configuration. There's just not many compelling use cases for it.