System Administration Guide: Solaris Containers, Resource Management, and Zones

Beta

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–1592
March 2004

040221@7940

Contents

Preface

System Administration Guide: Solaris Containers, Resource Management, and Zones is part of a multivolume set that covers a significant part of the Solaris™ Operating System administration information. This book assumes that you have already installed the operating system and set up any networking software that you plan to use.

Note – The Solaris Operating System runs on two types of hardware, or platforms—SPARC® and x86. The Solaris Operating System runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

About Solaris Containers

A Solaris container is a complete runtime environment for applications. Resource management and Solaris™ Zones partitioning technology are both parts of the container. These components address different qualities the container can deliver. The zones portion of the container provides a virtual mapping from the application to the platform resources. Zones allow application components to be isolated from one another even though the zones share a single instance of the Solaris Operating System. Resource management features permit you to allocate the quantity of resources that a workload receives.

The container establishes boundaries for resource consumption, such as CPU. These boundaries can be expanded to adapt to changing processing requirements of the application running in the container.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems that run the Solaris 10 release. To use this book, you should have at least one to two years of UNIX® system administration experience.

Related Books

The following books are recommended.

- *Solaris Resource Manager Developer's Guide*
- Other recommended books to be available at a later date

Accessing Sun Documentation Online

The docs.sun.com℠ Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

TABLE P–1 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Example
AaBbCc123	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename.*
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide.* These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

PART I

Resource Management

This part introduces resource management, which enables you to control how applications use available system resources.

CHAPTER 1

Introduction to Solaris 10 Resource Manager

Resource management functionality is a component of the Solaris container environment. Resource management enables you to control how applications use available system resources. You can do the following:

- Allocate computing resources, such as processor time
- Monitor how the allocations are being used, then adjust the allocations as necessary
- Generate extended accounting information for analysis, billing, and capacity planning

This chapter covers the following topics.

What's New in Resource Management?

This section describes new resource managment features in the Solaris 10 release.

New Resource Controls

A new set of resource controls replaces the System V interprocess communication (IPC) `/etc/system` tunables. For more information, see "Available Resource Controls" on page 64.

Pools Changes

Resource pools now provide a mechanism for dynamically adjusting each pool's resource allocation in response to system events and application load changes. Dynamic resource pools simplify and reduce the number of decisions required from an administrator. Adjustments are automatically made to preserve the system performance goals specified by an administrator. For more information, see Chapter 12.

New Extended Accounting Features

The following new features have been added:

- An Internet Protocol Quality of Service (IPQoS) flow accounting module, described in "Using Flow Accounting and Statistics Gathering (Tasks)" in *IPQoS Administration Guide*, allows you to capture network flow information on a system.
- A Practical Extraction and Report Language (Perl) interface to `libexacct` is now available. The interface enables you to develop customized reporting and extraction scripts. See "Perl Interface to `libexacct`" on page 49.

Ability to Regulate Physical Memory Consumption by Processes

The resource capping daemon `rcapd` enables you to regulate physical memory consumption by processes running in projects that have resource caps defined. For more information, see Chapter 10.

Interaction With Zones

Resource management features can now be used with zones to further refine the application environment. Interactions between these features and zones are described in applicable sections in this guide.

Resource Management Overview

Modern computing environments have to provide a flexible response to the varying workloads that are generated by different applications on a system. A workload is an aggregation of all processes of an application or group of applications. If resource

management features are not used, the Solaris Operating System responds to workload demands by adapting to new application requests dynamically. This default response generally means that all activity on the system is given equal access to resources. Solaris resource management features enable you to treat workloads individually. You can do the following:

- Restrict access to a specific resource
- Offer resources to workloads on a preferential basis
- Isolate workloads from each another

The ability to minimize cross-workload performance compromises, along with the facilities that monitor resource usage and utilization, is referred to as *resource management*. Resource management is implemented through a collection of algorithms. The algorithms handle the series of capability requests that an application presents in the course of its execution.

Resource management facilities permit you to modify the default behavior of the operating system with respect to different workloads. *Behavior* primarily refers to the set of decisions that are made by operating system algorithms when an application presents one or more resource requests to the system. You can use resource management facilities to do the following:

- Deny resources or prefer one application over another for a larger set of allocations than otherwise permitted
- Treat certain allocations collectively instead of through isolated mechanisms

The implementation of a system configuration that uses the resource management facilities can serve several purposes. You can do the following:

- Prevent an application from consuming resources indiscriminately
- Change an application's priority based on external events
- Balance resource guarantees to a set of applications against the goal of maximizing system utilization

When planning a resource-managed configuration, key requirements include the following:

- Identifying the competing workloads on the system
- Distinguishing those workloads that are not in conflict from those workloads with performance requirements that compromise the primary workloads

After you identify cooperating and conflicting workloads, you can create a resource configuration that presents the least compromise to the service goals of the business, within the limitations of the system's capabilities.

Effective resource management is enabled in the Solaris system by offering control mechanisms, notification mechanisms, and monitoring mechanisms. Many of these capabilities are provided through enhancements to existing mechanisms such as the `proc`(4) file system, processor sets, and scheduling classes. Other capabilities are specific to resource management. These capabilities are described in subsequent chapters.

Resource Classifications

A resource is any aspect of the computing system that can be manipulated with the intent to change application behavior. Thus, a resource is a capability that an application implicitly or explicitly requests. If the capability is denied or constrained, the execution of a robustly written application proceeds more slowly.

Classification of resources, as opposed to identification of resources, can be made along a number of axes. The axes could be implicitly requested as opposed to explicitly requested, time-based, such as CPU time, compared to time-independent, such as assigned CPU shares, and so forth.

Generally, scheduler-based resource management is applied to resources that the application can implicitly request. For example, to continue execution, an application implicitly requests additional CPU time. To write data to a network socket, an application implicitly requests bandwidth. Constraints can be placed on the aggregate total use of an implicitly requested resource.

Additional interfaces can be presented so that bandwidth or CPU service levels can be explicitly negotiated. Resources that are explicitly requested, such as a request for an additional thread, can be managed by constraint.

Resource Management Control Mechanisms

The three types of control mechanisms that are available in the Solaris Operating System are constraints, scheduling, and partitioning.

Constraint Mechanisms

Constraints allow the administrator or application developer to set bounds on the consumption of specific resources for a workload. With known bounds, modeling resource consumption scenarios becomes a simpler process. Bounds can also be used to control ill-behaved applications that would otherwise compromise system performance or availability through unregulated resource requests.

Constraints do present complications for the application. The relationship between the application and the system can be modified to the point that the application is no longer able to function. One approach that can mitigate this risk is to gradually

narrow the constraints on applications with unknown resource behavior. The resource controls feature discussed in Chapter 6 provides a constraint mechanism. Newer applications can be written to be aware of their resource constraints, but not all application writers will choose to do this.

Scheduling Mechanisms

Scheduling refers to making a sequence of allocation decisions at specific intervals. The decision that is made is based on a predictable algorithm. An application that does not need its current allocation leaves the resource available for another application's use. Scheduling-based resource management enables full utilization of an undercommitted configuration, while providing controlled allocations in a critically committed or overcommitted scenario. The underlying algorithm defines how the term "controlled" is interpreted. In some instances, the scheduling algorithm might guarantee that all applications have some access to the resource. The fair share scheduler (FSS) described in Chapter 8 manages application access to CPU resources in a controlled way.

Partitioning Mechanisms

Partitioning is used to bind a workload to a subset of the system's available resources. This binding guarantees that a known amount of resources is always available to the workload. The resource pools functionality that is described in Chapter 12 enables you to limit workloads to specific subsets of the machine.

Configurations that use partitioning can avoid system-wide overcommitment. However, in avoiding this overcommitment, the ability to achieve high utilizations can be reduced. A reserved group of resources, such as processors, is not available for use by another workload when the workload bound to them is idle.

Resource Management Configuration

Portions of the resource management configuration can be placed in a network name service. This feature allows the administrator to apply resource management constraints across a collection of machines, rather than on an exclusively per-machine basis. Related work can share a common identifier, and the aggregate usage of that work can be tabulated from accounting data.

Resource management configuration and workload-oriented identifiers are described more fully in Chapter 2. The extended accounting facility that links these identifiers with application resource usage is described in Chapter 4.

When to Use Resource Management

Use resource management to ensure that your applications have the required response times.

Resource management can also increase resource utilization. By categorizing and prioritizing usage, you can effectively use reserve capacity during off-peak periods, often eliminating the need for additional processing power. You can also ensure that resources are not wasted because of load variability.

Server Consolidation

Resource management is ideal for environments that consolidate a number of applications on a single server.

The cost and complexity of managing numerous machines encourages the consolidation of several applications on larger, more scalable servers. Instead of running each workload on a separate system, with full access to that system's resources, you can use resource management software to segregate workloads within the system. Resource management enables you to lower overall total cost of ownership by running and controlling several dissimilar applications on a single Solaris system.

If you are providing Internet and application services, you can use resource management to do the following:

- Host multiple web servers on a single machine. You can control the resource consumption for each web site and you can protect each site from the potential excesses of other sites.
- Prevent a faulty common gateway interface (CGI) script from exhausting CPU resources.
- Stop an incorrectly behaving application from leaking all available virtual memory.
- Ensure that one customer's applications are not affected by another customer's applications that run at the same site.
- Provide differentiated levels or classes of service on the same machine.
- Obtain accounting information for billing purposes.

Supporting a Large or Varied User Population

Use resource management features in any system that has a large, diverse user base, such as an educational institution. If you have a mix of workloads, the software can be configured to give priority to specific projects.

For example, in large brokerage firms, traders intermittently require fast access to execute a query or to perform a calculation. Other system users, however, have more consistent workloads. If you allocate a proportionately larger amount of processing power to the traders' projects, the traders have the responsiveness that they need.

Resource management is also ideal for supporting thin-client systems. These platforms provide stateless consoles with frame buffers and input devices, such as smart cards. The actual computation is done on a shared server, resulting in a timesharing type of environment. Use resource management features to isolate the users on the server. Then, a user who generates excess load does not monopolize hardware resources and significantly impact others who use the system.

Setting Up Resource Management (Task Map)

The following task map gives a basic overview of the steps that are involved in setting up resource management on your system.

Task	Description	For Instructions
Identify the workloads on your system.	Review project entries in either the `/etc/project` database file or in the NIS map or LDAP directory service.	"`project` Database" on page 33
Prioritize the workloads on your system.	Determine which applications are critical. These workloads might require preferential access to resources.	Refer to your business service goals.
Monitor real-time activity on your system.	Use performance tools to view the current resource consumption of workloads that are running on your system. You can then evaluate whether you must restrict access to a given resource or isolate particular workloads from other workloads.	"Monitoring by System" on page 160, `cpustat`(1M), `iostat`(1M), `mpstat`(1M), `prstat`(1M), `sar`(1), and `vmstat`(1M)

Task	Description	For Instructions
Make temporary modifications to the workloads that are running on your system.	To determine which values can be altered, refer to the resource controls that are available in the Solaris system. You can update the values from the command line while the task or process is running.	"Available Resource Controls" on page 64, "Actions on Resource Control Values" on page 67, "Temporarily Updating Resource Control Values on a Running System" on page 70, `rctladm`(1M), and `prctl`(1).
Set resource controls and project attributes for every project entry in the `project` database or name service project table.	Each project entry in the `/etc/project` database or the name service project table can contain one or more resource controls or attributes. Resource controls constrain tasks and processes attached to that project. For each threshold value that is placed on a resource control, you can associate one or more actions to be taken when that value is reached. You can set resource controls by using the command-line interface. Certain configuration parameters can also be set by using the Solaris Managment Console.	"`project` Database" on page 33, "Local `project` File Format" on page 34, "Available Resource Controls" on page 64, "Actions on Resource Control Values" on page 67, and Chapter 8
Place an upper bound on the resource consumption of physical memory by collections of processes attached to a project.	The resource cap enforcement daemon will enforce the physical memory resource cap defined in the `/etc/project` database with the `rcap.max-rss` attribute.	"`project` Database" on page 33, Chapter 10
Create resource pools configurations.	Resource pools provide a way to partition system resources, such as processors, and maintain those partitions across reboots. You can add a `project.pool` attribute to each entry in the `/etc/project` database.	"`project` Database" on page 33, Chapter 12
Make the fair share scheduler (FSS) your default system scheduler.	Ensure that all user processes in either a single CPU system or a processor set belong to the same scheduling class.	"FSS Configuration Examples" on page 90 and `dispadmin`(1M)

Task	Description	For Instructions
Activate the extended accounting facility to monitor and record resource consumption on a task or process basis.	Use extended accounting data to assess current resource controls and plan capacity requirements for future workloads. Aggregate usage on a system-wide basis can be tracked. To obtain complete usage statistics for related workloads that span more than one system, the project name can be shared across several machines.	"How to Activate Extended Accounting for Processes, Tasks, and Flows" on page 53 and `acctadm`(1M)
(Optional) If you determine that additional adjustments to your configuration are required, you can continue to alter the values from the command line. You can alter the values while the task or process is running.	Modifications to existing tasks can be applied on a temporary basis without restarting the project. Tune the values until you are satisfied with the performance. Then update the current values in the `/etc/project` database or the name service project table.	"Temporarily Updating Resource Control Values on a Running System" on page 70, `rctladm`(1M), and `prctl`(1)
(Optional) Capture extended accounting data.	Write extended accounting records for active processes and active tasks. The files that are produced can be used for planning, chargeback, and billing purposes. There is also a Practical Extraction and Report Language (Perl) interface to `libexacct` that enables you to develop customized reporting and extraction scripts.	`wracct`(1M), "Perl Interface to `libexacct`" on page 49

CHAPTER 2

Projects and Tasks (Overview)

This chapter discusses the *project* and *task* facilities of Solaris resource management. Projects and tasks are used to label workloads and separate them from one another.

The following topics are covered in this chapter:

Introduction to Projects and Tasks

To optimize workload response, you must first be able to identify the workloads that are running on the system you are analyzing. This information can be difficult to obtain by using either a purely process-oriented or a user-oriented method alone. In the Solaris system, you have two additional facilities that can be used to separate and identify workloads: the project and the task. The project provides a network-wide administrative identifier for related work. The task collects a group of processes into a manageable entity that represents a workload component.

Based on their project or task membership, running processes can be manipulated with standard Solaris commands. The extended accounting facility can report on both process usage and task usage, and tag each record with the governing project identifier. This process enables offline workload analysis to be correlated with online monitoring. The project identifier can be shared across multiple machines through the `project` name service database. Thus, the resource consumption of related workloads that run on (or span) multiple machines can ultimately be analyzed across all of the machines.

Project Identifier

The project identifier is an administrative identifier that is used to identify related work. The project identifier can be thought of as a workload tag equivalent to the user and group identifiers. A user or group can belong to one or more projects. These projects can be used to represent the workloads in which the user (or group of users) is allowed to participate. This membership can then be the basis of chargeback that is based on, for example, usage or initial resource allocations. Although a user must have a default project assigned, the processes that the user launches can be associated with any of the projects of which that user is a member.

Determining a User's Default Project

To log in to the system, a user must be assigned a default project.

Because each process on the system possesses project membership, an algorithm to assign a default project to the login or other initial process is necessary. The algorithm is documented in the man page `getprojent`(3C). The algorithm to determine a default project consists of four steps. If no default project is found, the user's login, or request to start a process, is denied.

The system sequentially follows these steps to determine a user's default project:

1. If the user has an entry with a `project` attribute defined in the `/etc/user_attr` extended user attributes database, then the value of the `project` attribute is the default project. See `user_attr`(4).
2. If a project with the name `user.`*user-id* is present in the `project` database, then that project is the default project. See `project`(4) for more information.
3. If a project with the name `group.`*group-name* is present in the `project` database, where *group-name* is the name of the default group for the user, as specified in the `passwd` file, then that project is the default project. For information on the `passwd` file, see `passwd`(4).
4. If the special project `default` is present in the `project` database, then that project is the default project.

This logic is provided by the `getdefaultproj()` library function. See `getprojent`(3PROJECT) for more information.

`project` Database

You can store project data in a local file, in a Network Information Service (NIS) project map, or in a Lightweight Directory Access Protocol (LDAP) directory service. The `/etc/project` database or name service is used at login and by all requests for account management by the pluggable authentication module (PAM) to bind a user to a default project.

Note – Updates to entries in the project database, whether to the `/etc/project` file or to a representation of the database in a network name service, are not applied to currently active projects. The updates are applied to new tasks that join the project when `login` or `newtask` is used. For more information, see `login`(1) and `newtask`(1).

PAM Subsystem

Operations that change or set identity include logging in to the system, invoking an `rcp` or `rsh` command, using `ftp`, or using `su`. When an operation involves changing or setting identity, a set of configurable modules is used to provide authentication, account management, credentials management, and session management.

The account management PAM module for projects is documented in the `pam_projects`(5) man page. For an overview of PAM, see "PAM (Overview)" in *System Administration Guide: Security Services*.

Name Service Configuration

Resource management supports the name service `project` database. The location where the `project` database is stored is defined in `/etc/nsswitch.conf`. By default, `files` is listed first, but the sources can be listed in any order.

```
project: files [nis] [ldap]
```

If more than one source for project information is listed, the `nsswitch.conf` file directs the routine to start searching for the information in the first source listed, then search subsequent databases.

For more information on `/etc/nsswitch.conf`, see "The Name Service Switch" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and `nsswitch.conf`(4).

Local `project` File Format

If you select `files` as your `project` database in `nsswitch.conf`, the login process searches the `/etc/project` file for project information. See `projects`(1) and `project`(4) for more information.

The `project` file contains a one-line entry for each project recognized by the system, of the following form:

```
projname:projid:comment:user-list:group-list:attributes
```

The fields are defined as follows.

projname The name of the project. The name must be a string that consists of alphanumeric characters, the underline (_) character, and the hyphen (-). The name must begin with an alphabetic character. *projname* cannot contain periods (.), colons (:), or newline characters.

projid The project's unique numerical ID (PROJID) within the system. The maximum value of the *projid* field is UID_MAX (`2147483647`).

comment The project's description.

user-list A comma-separated list of users who are allowed in the project.

Wildcards can be used in this field. The asterisk (*) allows all users to join the project. The exclamation point followed by the asterisk (!*) excludes all users from the project. The exclamation mark (!) followed by a user name excludes the specified user from the project.

group-list A comma-separated list of groups of users who are allowed in the project.

Wildcards can be used in this field. The asterisk (*) allows all groups to join the project. The exclamation point followed by the asterisk (!*) excludes all groups from the project. The exclamation mark (!) followed by a group name excludes the specified group from the project.

attributes A semicolon-separated list of name-value pairs, such as resource controls (see Chapter 6). *name* is an arbitrary string that specifies the object-related attribute, and *value* is the optional value for that attribute.

```
name[=value]
```

In the name-value pair, names are restricted to letters, digits, underscores, and the period. The period is conventionally used as a separator between the categories and subcategories of the `rctl`. The first character of an attribute name must be a letter. The name is case sensitive.

Values can be structured by using commas and parentheses to establish precedence. The semicolon is used to separate name-value pairs. The semicolon cannot be used in a value definition. The colon is used to

separate project fields. The colon cannot be used in a value definition.

Note – Routines that read this file halt when they encounter a malformed entry. Any project assignments that are specified after the incorrect entry are not made.

This example shows the default `/etc/project` file:

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3::::
group.staff:10::::
```

This example shows the default `/etc/project` file with project entries added at the end:

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

To add resource controls to the `/etc/project` file, see "Using Resource Controls" on page 73 To define a physical memory resource cap for a project using the resource capping daemon described in `rcapd`(1M), see "Attribute to Limit Physical Memory Usage" on page 94. To add a `project.pool` attribute to a project's entry in the project file, see "Creating the Configuration" on page 150.

Name Service Configuration for NIS

If you are using NIS, you can specify in the `/etc/nsswitch.conf` file to search the NIS maps for projects:

```
project: nis files
```

The NIS maps, either `project.byname` or `project.bynumber`, have the same form as the `/etc/project` file:

```
projname:projid:comment:user-list:group-list:attributes
```

For more information, see "Network Information Service (NIS) (Overview)" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Directory Service Configuration for LDAP

If you are using LDAP, you can specify in the `/etc/nsswitch.conf` file to search the LDAP `project` database for projects.

```
project: ldap files
```

For more information on the schema for project entries in an LDAP database, see "Solaris Schemas" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Task Identifier

With each successful login into a project, a new *task* that contains the login process is created. The task is a process collective that represents a set of work over time. A task can also be viewed as a *workload component*.

Each process is a member of one task, and each task is associated with one project.

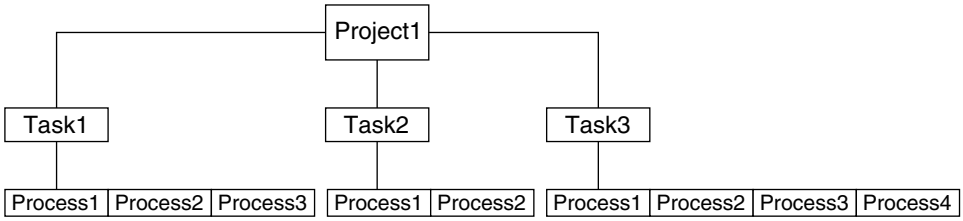


FIGURE 2–1 Project and Task Tree

All operations on sessions, such as signal delivery, are also supported on tasks. You can also bind tasks to processor sets and set their scheduling priorities and classes, which modifies all current and subsequent processes in the task.

Tasks are created at login, by `cron`, by `newtask`, and by `setproject`. For more information, see `login`(1), `cron`(1M), `newtask`(1), and `setproject`(3PROJECT).

The extended accounting facility can provide accounting data for processes. The data is aggregated at the task level.

Commands Used With Projects and Tasks

The commands that are shown in the following table provide the primary administrative interface to the project and task facilities.

Command	Description
`projects`(1)	Prints the project membership of a user.
`newtask`(1)	Executes the user's default shell or specified command, placing the execution command in a new task that is owned by the specified project. `newtask` can also be used to change the task and the project binding for a running process.
`projadd`(1M)	Adds a new project entry to the `/etc/project` file. `projadd` creates a project entry only on the local system. `projadd` cannot change information that is supplied by the network name service.
`projmod`(1M)	Modifies a project's information on the local system. `projmod` cannot change information that is supplied by the network name service. However, the command does verify the uniqueness of the project name and project ID against the external name service.
`projdel`(1M)	Deletes a project from the local system. `projdel` cannot change information that is supplied by the network name service.

CHAPTER 3

Administering Projects and Tasks

This chapter describes how to use the project and task facilities of Solaris resource management.

The following topics are covered.

If you are using these facilities in a zones environment, note that only processes in the same zone will be visible through system call interfaces that take process IDs when these command are run in a non-global zone.

Sample Commands and Command Options

This section provides examples of commands and options used with projects and tasks.

Command Options Used With Projects and Tasks

`ps` Command

Use the `ps` command with the `-o` option to display task and project IDs. For example, to view the project ID, type the following:

```
# ps -o user,pid,uid,projid
USER PID   UID  PROJID
jtd  89430 124  4113
```

`id` Command

Use the `id` command with the `-p` option to print the current project ID in addition to the user and group IDs. If the *user* operand is provided, the project associated with that user's normal login is printed:

```
#  id -p
uid=124(jtd) gid=10(staff) projid=4113(booksite)
```

`pgrep` and `pkill` Commands

To match only processes with a project ID in a specific list, use the `pgrep` and `pkill` commands with the `-J` option:

```
# pgrep -J projidlist
# pkill -J projidlist
```

To match only processes with a task ID in a specific list, use the `pgrep` and `pkill` commands with the `-T` option:

```
# pgrep -T taskidlist
# pkill -T taskidlist
```

`prstat` Command

To display various statistics for processes and projects that are currently running on your system, use the `prstat` command with the `-j` option:

```
% prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 21634 jtd      5512K 4848K cpu0    44    0   0:00.00 0.3% prstat/1
   324 root       29M   75M sleep   59    0   0:08.27 0.2% Xsun/1
 15497 jtd        48M   41M sleep   49    0   0:08.26 0.1% adeptedit/1
   328 root     2856K 2600K sleep   58    0   0:00.00 0.0% mibiisa/11
  1979 jtd      1568K 1352K sleep   49    0   0:00.00 0.0% csh/1
  1977 jtd      7256K 5512K sleep   49    0   0:00.00 0.0% dtterm/1
   192 root     3680K 2856K sleep   58    0   0:00.36 0.0% automountd/5
  1845 jtd        24M   22M sleep   49    0   0:00.29 0.0% dtmail/11
  1009 jtd      9864K 8384K sleep   49    0   0:00.59 0.0% dtwm/8
   114 root     1640K  704K sleep   58    0   0:01.16 0.0% in.routed/1
   180 daemon   2704K 1944K sleep   58    0   0:00.00 0.0% statd/4
   145 root     2120K 1520K sleep   58    0   0:00.00 0.0% ypbind/1
   181 root     1864K 1336K sleep   51    0   0:00.00 0.0% lockd/1
   173 root     2584K 2136K sleep   58    0   0:00.00 0.0% inetd/1
   135 root     2960K 1424K sleep    0    0   0:00.00 0.0% keyserv/4
PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
    10       52  400M  271M    68%   0:11.45 0.4% booksite
     0       35  113M  129M    32%   0:10.46 0.2% system

Total: 87 processes, 205 lwps, load averages: 0.05, 0.02, 0.02
```

To display various statistics for processes and tasks that are currently running on your system, use the `prstat` command with the `-T` option:

```
% prstat -T
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 23023 root       26M   20M sleep   59    0   0:03:18 0.6% Xsun/1
 23476 jtd        51M   45M sleep   49    0   0:04:31 0.5% adeptedit/1
 23432 jtd      6928K 5064K sleep   59    0   0:00:00 0.1% dtterm/1
 28959 jtd        26M   18M sleep   49    0   0:00:18 0.0% .netscape.bin/1
 23116 jtd      9232K 8104K sleep   59    0   0:00:27 0.0% dtwm/5
 29010 jtd      5144K 4664K cpu0    59    0   0:00:00 0.0% prstat/1
   200 root     3096K 1024K sleep   59    0   0:00:00 0.0% lpsched/1
   161 root     2120K 1600K sleep   59    0   0:00:00 0.0% lockd/2
   170 root     5888K 4248K sleep   59    0   0:03:10 0.0% automountd/3
   132 root     2120K 1408K sleep   59    0   0:00:00 0.0% ypbind/1
   162 daemon   2504K 1936K sleep   59    0   0:00:00 0.0% statd/2
   146 root     2560K 2008K sleep   59    0   0:00:00 0.0% inetd/1
   122 root     2336K 1264K sleep   59    0   0:00:00 0.0% keyserv/2
   119 root     2336K 1496K sleep   59    0   0:00:02 0.0% rpcbind/1
   104 root     1664K  672K sleep   59    0   0:00:03 0.0% in.rdisc/1
TASKID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
   222       30  229M  161M    44%   0:05:54 0.6% group.staff
   223        1   26M   20M   5.3%   0:03:18 0.6% group.staff
    12        1   61M   33M   8.9%   0:00:31 0.0% group.staff
     1       33   85M   53M    14%   0:03:33 0.0% system

Total: 65 processes, 154 lwps, load averages: 0.04, 0.05, 0.06
```

Note – The `-J` and `-T` options cannot be used together.

Using `cron` and `su` With Projects and Tasks

`cron` Command

The `cron` command issues a `settaskid` to ensure that each `cron`, `at`, and `batch` job executes in a separate task, with the appropriate default project for the submitting user. The `at` and `batch` commands also capture the current project ID, which ensures that the project ID is restored when running an `at` job.

`su` Command

To switch the user's default project, and thus create a new task, as part of simulating a login, type the following:

```
# su - user
```

To retain the project ID of the invoker, issue `su` without the `-` flag.

```
# su user
```

Project Administration Examples

▼ How to Define a Project and View the Current Project

This example shows how to use the `projadd` command to add a project entry and the `projmod` command to alter that entry.

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **View the default `/etc/project` file on your system.**

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

3. **Add a project with the name *booksite*. Assign the project to a user who is named *mark* with project ID number *4113*.**

```
# projadd -U mark -p 4113 booksite
```

4. **View the `/etc/project` file again to see the project addition.**

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
booksite:4113::mark::
```

5. **Add a comment that describes the project in the comment field.**

```
# projmod -c 'Book Auction Project' booksite
```

6. **View the changes in the `/etc/project` file.**

```
# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
booksite:4113:Book Auction Project:mark::
```

To bind projects, tasks, and processes to a pool, see "Binding to a Pool" on page 145.

▼ How to Delete a Project From the `/etc/project` File

This example shows how to use the `projdel` command to delete a project.

1. **Become superuser.**
2. **Remove the project *booksite* by using the `projdel` command.**

   ```
   # projdel booksite
   ```

3. **Display the `/etc/project` file.**

   ```
   # cat /etc/project
   system:0::::
   user.root:1::::
   noproject:2::::
   default:3::::
   group.staff:10::::
   ```

4. **Log in as user *mark* and type `projects` to view the projects that are assigned to this user.**

   ```
   # su - mark
   # projects
   default
   ```

How to Obtain User and Project Membership Information

Use the `id` command with the `-p` flag to view the current project membership of the invoking process.

```
$ id -p
uid=100(mark) gid=1(other) projid=3(default)
```

▼ How to Create a New Task

1. **Become superuser.**

2. **Create a new task in the *booksite* project by using the `newtask` command with the `-v` (verbose) option to obtain the system task ID.**

```
# newtask -v -p booksite
16
```

The execution of `newtask` creates a new task in the specified project, and places the user's default shell in this task.

3. **View the current project membership of the invoking process.**

```
# id -p
uid=100(mark) gid=1(other) projid=4113(booksite)
```

The process is now a member of the new project.

▼ How to Move a Running Process Into a New Task

This example shows how to associate a running process with a different task and new project. To perform this action, you must either be superuser, or be the owner of the process and be a member of the new project.

1. **Become superuser.**

2. **Obtain the process ID of the *book_catalog* process.**

```
# pgrep book_catalog
    8100
```

3. **Associate process *8100* with a new task ID in the *booksite* project.**

```
# newtask -v -p booksite -c 8100
    17
```

The `-c` option specifies that `newtask` operate on the existing named process.

4. **Confirm the task to process ID mapping.**

```
# pgrep -T 17
    8100
```

CHAPTER 4

Extended Accounting (Overview)

By using the project and task facilities that are described in Chapter 2 to label and separate workloads, you can monitor resource consumption by each workload. You can use the *extended accounting* subsystem to capture a detailed set of resource consumption statistics on both processes and tasks.

The following topics are covered in this chapter.

To begin using extended accounting, see "How to Activate Extended Accounting for Processes, Tasks, and Flows" on page 53.

Introduction to Extended Accounting

The extended accounting subsystem labels usage records with the project for which the work was done. You can also use extended accounting, in conjunction with the Internet Protocol Quality of Service (IPQoS) flow accounting module described in "Using Flow Accounting and Statistics Gathering (Tasks)" in *IPQoS Administration Guide*, to capture network flow information on a system.

Before you can apply resource management mechanisms, you must first be able to characterize the resource consumption demands that various workloads place on a system. The extended accounting facility in the Solaris Operating System provides a flexible way to record system and network resource consumption on a task or process basis, or on the basis of selectors provided by the IPQoS `flowacct` module. For more information, see `ipqos`(7IPP).

Unlike online monitoring tools, which enable you to measure system usage in real time, extended accounting enables you to examine historical usage. You can then make assessments of capacity requirements for future workloads.

With extended accounting data available, you can develop or purchase software for resource chargeback, workload monitoring, or capacity planning.

How Extended Accounting Works

The extended accounting facility in the Solaris Operating System uses a versioned, extensible file format to contain accounting data. Files that use this data format can be accessed or be created by using the API provided in the included library, `libexacct` (see`libexacct`(3LIB)). These files can then be analyzed on any platform with extended accounting enabled, and their data can be used for capacity planning and chargeback.

If extended accounting is active, statistics are gathered that can be examined by the `libexacct` API. `libexacct` allows examination of the `exacct` files either forward or backward. The API supports third-party files that are generated by `libexacct` as well as those files that are created by the kernel. There is a Practical Extraction and Report Language (Perl) interface to `libexacct` that enables you to develop customized reporting and extraction scripts. See “Perl Interface to `libexacct`” on page 49.

With extended accounting enabled, the task tracks the aggregate resource usage of its member processes. A task accounting record is written at task completion. Interim records on running processes and tasks can also be written. For more information on tasks, see Chapter 2.

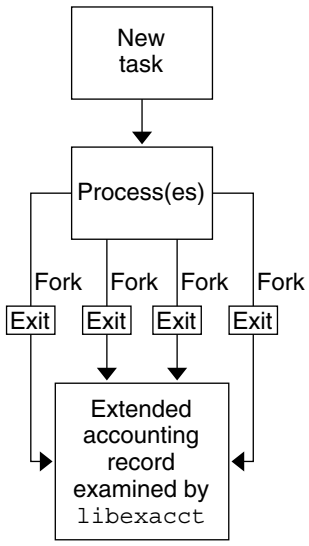


FIGURE 4–1 Task Tracking With Extended Accounting Activated

Extensible Format

The extended accounting format is substantially more extensible than the SunOS™ legacy system accounting software format (see "What is System Accounting?" in *System Administration Guide: Advanced Administration*). Extended accounting permits accounting metrics to be added and removed from the system between releases, and even during system operation.

Note – Both extended accounting and legacy system accounting software can be active on your system at the same time.

`exacct` Records and Format

Routines that allow `exacct` records to be created serve two purposes.

- To enable third-party `exacct` files to be created.
- To enable the creation of tagging records to be embedded in the kernel accounting file by using the `putacct` system call (see `getacct`(2)).

Note – The `putacct` system call is also available from the Perl interface.

The format permits different forms of accounting records to be captured without requiring that every change be an explicit version change. Well-written applications that consume accounting data must ignore records they do not understand.

The `libexacct` library converts and produces files in the `exacct` format. This library is the *only* supported interface to `exacct` format files.

Note – The `getacct`, `putacct`, and `wracct` system calls do not apply to flows. The kernel creates flow records and writes them to the file when IPQoS flow accounting is configured.

Using Extended Accounting in a Zones Environment

The extended accounting subsystem collects and reports information for the entire system (including non-global zones) when run in the global zone. The global administrator can also determine resource consumption on a per-zone basis. See "Extended Accounting in a Zones Environment" on page 241 for more information.

Extended Accounting Configuration

The `/etc/acctadm.conf` file contains the current extended accounting configuration. The file is edited through the `acctadm` interface, not by the user.

The directory `/var/adm/exacct` is the standard location for placing extended accounting data. You can use the `acctadm` command to specify a different location for the process and task accounting-data files. See `acctadm`(1M) for more information.

Commands Used With Extended Accounting

Command	Description
`acctadm`(1M)	Modifies various attributes of the extended accounting facility, stops and starts extended accounting, and is used to select accounting attributes to track for processes, tasks, and flows.
`wracct`(1M)	Writes extended accounting records for active processes and active tasks.
`lastcomm`(1)	Displays previously invoked commands. `lastcomm` can consume either standard accounting-process data or extended-accounting process data.

For information on commands that are associated with tasks and projects, see "Sample Commands and Command Options" on page 39. For information on IPQoS flow accounting, see `ipqosconf`(1M).

Perl Interface to `libexacct`

The Perl interface allows you to create Perl scripts that can read the accounting files produced by the `exacct` framework. You can also create Perl scripts that write `exacct` files.

The interface is functionally equivalent to the underlying C API. When possible, the data obtained from the underlying C API is presented as Perl data types. This feature makes accessing the data easier and it removes the need for buffer pack and unpack operations. Moreover, all memory management is performed by the Perl library.

The various project, task, and `exacct`-related functions are separated into groups. Each group of functions is located in a separate Perl module. Each module begins with the Sun standard `Sun::Solaris::` Perl package prefix. All of the classes provided by the Perl `exacct` library are found under the `Sun::Solaris::Exacct` module.

The underlying `libexacct`(3LIB) library provides operations on `exacct` format files, catalog tags, and `exacct` objects. `exacct` objects are subdivided into two types:

- Items, which are single-data values (scalars)

- Groups, which are lists of Items

The following table summarizes each of the modules.

Module (should not contain spaces)	Description	For More Information
`Sun::Solaris::Project`	This module provides functions to access the project manipulation functions `getprojid`(2), `endprojent`(3PROJECT) , `fgetprojent`(3PROJECT), `getdefaultproj`(3PROJECT), `getprojbyid`(3PROJECT), `getprojbyname`(3PROJECT), `getprojent`(3PROJECT), `getprojidbyname`(3PROJECT), `inproj`(3PROJECT), `project_walk`(3PROJECT), `setproject`(3PROJECT) , and `setprojent`(3PROJECT).	`Project`(3PERL)
`Sun::Solaris::Task`	This module provides functions to access the task manipulation functions `gettaskid`(2) and `settaskid`(2).	`Task`(3PERL)
`Sun::Solaris::Exacct`	This module is the top-level `exacct` module. This module provides functions to access the `exacct`-related system calls `getacct`(2), `putacct`(2), and `wracct`(2). This module also provides functions to access the `libexacct`(3LIB) library function `ea_error`(3EXACCT). Constants for all of the `exacct` EO_*, EW_*, EXR_*, P_*, and TASK_* macros are also provided in this module.	`Exacct`(3PERL)
`Sun::Solaris::Exacct::Catalog`	This module provides object-oriented methods to access the bitfields in an `exacct` catalog tag. This module also provides access to the constants for the EXC_*, EXD_*, and EXD_* macros.	`Exacct::Catalog`(3PERL)
`Sun::Solaris::Exacct::File`	This module provides object-oriented methods to access the `libexacct` accounting file functions `ea_open`(3EXACCT), `ea_close`(3EXACCT), `ea_get_creator`(3EXACCT), `ea_get_hostname`(3EXACCT), `ea_next_object`(3EXACCT), `ea_previous_object`(3EXACCT), and `ea_write_object`(3EXACCT).	`Exacct::File`(3PERL)

Module (should not contain spaces)	Description	For More Information
`Sun::Solaris::Exacct::Object`	This module provides object-oriented methods to access an individual `exacct` accounting file object. An `exacct` object is represented as an opaque reference blessed into the appropriate `Sun::Solaris::Exacct::Object` subclass. This module is further subdivided into the object types Item and Group. At this level, there are methods to access the `ea_match_object_catalog`(3EXACCT) and `ea_attach_to_object`(3EXACCT) functions.	`Exacct::Object`(3PERL)
`Sun::Solaris::Exacct::Object::Item`	This module provides object-oriented methods to access an individual `exacct` accounting file Item. Objects of this type inherit from `Sun::Solaris::Exacct::Object`.	`Exacct::Object::Item`(3PERL)
`Sun::Solaris::Exacct::Object::Group`	This module provides object-oriented methods to access an individual `exacct` accounting file Group. Objects of this type inherit from `Sun::Solaris::Exacct::Object`. These objects provide access to the `ea_attach_to_group`(3EXACCT) function. The Items contained within the Group are presented as a Perl array.	`Exacct::Object::Group`(3PERL)
`Sun::Solaris::Kstat`	This module provides a Perl tied hash interface to the `kstat` facility. A usage example for this module can be found in `/bin/kstat`, which is written in Perl.	`Kstat`(3PERL)

For examples that show how to use the modules described in the previous table, see "Using the Perl Interface to `libexacct`" on page 56.

CHAPTER 5

Extended Accounting (Tasks)

This chapter describes how to administer the extended accounting subsystem. The following procedures are covered.

Using Extended Accounting Functionality

▼ How to Activate Extended Accounting for Processes, Tasks, and Flows

To activate the extended accounting facility for tasks, processes, and flows, use the `acctadm` command. The optional final parameter to `acctadm` indicates whether the command should act on the process, system task, or flow accounting components of the extended accounting facility. See`acctadm`(1M) for more information.

1. **Become superuser.**
2. **Activate extended accounting for processes.**

   ```
   # acctadm -e extended -f /var/adm/exacct/proc process
   ```

3. **Activate extended accounting for tasks.**

   ```
   # acctadm -e extended,mstate -f /var/adm/exacct/task task
   ```

4. **Activate extended accounting for flows.**

```
# acctadm -e extended -f /var/adm/exacct/flow flow
```

How to Activate Extended Accounting With a Startup Script

Activate extended accounting on an ongoing basis by linking the `/etc/init.d/acctadm` script into `/etc/rc2.d`.

```
# ln -s /etc/init.d/acctadm /etc/rc2.d/Snacctadm
# ln -s /etc/init.d/acctadm /etc/rc2.d/Knacctadm
```

The *n* variable is replaced by a number.

See "Extended Accounting Configuration" on page 48 for information on accounting configuration.

How to Display Extended Accounting Status

Type `acctadm` without arguments to display the current status of the extended accounting facility.

```
# acctadm
                Task accounting: active
           Task accounting file: /var/adm/exacct/task
         Tracked task resources: extended
       Untracked task resources: none
              Process accounting: active
        Process accounting file: /var/adm/exacct/proc
      Tracked process resources: extended
    Untracked process resources: host,mstate
                Flow accounting: active
           Flow accounting file: /var/adm/exacct/flow
         Tracked flow resources: extended
       Untracked flow resources: none
```

In the previous example, system task accounting is active in extended mode and `mstate` mode. Process and flow accounting are active in extended mode.

Note – In the context of extended accounting, microstate (`mstate`) refers to the extended data, associated with microstate process transitions, that is available in the process usage file (see `proc`(4)). This data provides much more detail about the activities of the process than basic or extended records.

How to View Available Accounting Resources

Available resources can vary from system to system, and from platform to platform. Use the `acctadm` command with the `-r` option to view the available accounting resources on the system.

```
# acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,flag
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid
basic    taskid,projid,cpu,timeprocess:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,flag
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid
basic    taskid,projid,cpu,time
flow:
extended
saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts,action,ctime,lseen,projid,uid
basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
```

▼ How to Deactivate Process, Task, and Flow Accounting

To deactivate process, task, and flow accounting, turn off each of them individually by using the `acctadm` command with the `-x` option.

1. **Become superuser.**
2. **Turn off process accounting.**

   ```
   # acctadm -x process
   ```

3. **Turn off task accounting.**

   ```
   # acctadm -x task
   ```

4. **Turn off flow accounting.**

   ```
   # acctadm -x flow
   ```

5. **Verify that task accounting, process accounting, and flow accounting have been turned off.**

   ```
   # acctadm
           Task accounting: inactive
      Task accounting file: none
    Tracked task resources: extended
  Untracked task resources: none
   ```

```
          Process accounting: inactive
     Process accounting file: none
   Tracked process resources: extended
 Untracked process resources: host,mstate
             Flow accounting: inactive
        Flow accounting file: none
      Tracked flow resources: extended
    Untracked flow resources: none
```

Using the Perl Interface to `libexacct`

How to Recursively Print the Contents of an `exacct` Object

Use the following code to recursively print the contents of an `exacct` object. Note that this capability is provided by the library as the `Sun::Solaris::Exacct::Object::dump()` function. This capability is also available through the `ea_dump_object()` convenience function.

```
sub dump_object
     {
             my ($obj, $indent) = @_;
             my $istr = '  ' x $indent;

             #
             # Retrieve the catalog tag.  Because we are
             # doing this in an array context, the
             # catalog tag will be returned as a (type, catalog, id)
             # triplet, where each member of the triplet will behave as
             # an integer or a string, depending on context.
             # If instead this next line provided a scalar context, e.g.
             #    my $cat  = $obj->catalog()->value();
             # then $cat would be set to the integer value of the
             # catalog tag.
             #
             my @cat = $obj->catalog()->value();

             #
             # If the object is a plain item
             #
             if ($obj->type() == &EO_ITEM) {
                     #
                     # Note: The '%s' formats provide s string context, so
                     # the components of the catalog tag will be displayed
                     # as the symbolic values. If we changed the '%s'
```

```
                # formats to '%d', the numeric value of the components
                # would be displayed.
                #
                printf("%sITEM\n%s  Catalog = %s|%s|%s\n",
                   $istr, $istr, @cat);
                $indent++;

                #
                # Retrieve the value of the item.  If the item contains
                # in turn a nested exacct object (i.e., an item or
                # group),then the value method will return a reference
                # to the appropriate sort of perl object
                # (Exacct::Object::Item or Exacct::Object::Group).
                # We could of course figure out that the item contained
                # a nested item orgroup by examining the catalog tag in
                # @cat and looking for a type of EXT_EXACCT_OBJECT or
                # EXT_GROUP.
                #
                my $val = $obj->value();
                if (ref($val)) {
                        # If it is a nested object, recurse to dump it.
                        dump_object($val, $indent);
                } else {
                        # Otherwise it is just a 'plain' value, so
                        # display it.
                        printf("%s  Value = %s\n", $istr, $val);
                }

        #
        # Otherwise we know we are dealing with a group.  Groups
        # represent contents as a perl list or array (depending on
        # context), so we can process the contents of the group
        # with a 'foreach' loop, which provides a list context.
        # In a list context the value method returns the content
        # of the group as a perl list, which is the quickest
        # mechanism, but doesn't allow the group to be modified.
        # If we wanted to modify the contents of the group we could
        # do so like this:
        #    my $grp = $obj->value();   # Returns an array reference
        #    $grp->[0] = $newitem;
        # but accessing the group elements this way is much slower.
        #
        } else {
                printf("%sGROUP\n%s  Catalog = %s|%s|%s\n",
                    $istr, $istr, @cat);
                $indent++;
                # 'foreach' provides a list context.
                foreach my $val ($obj->value()) {
                        dump_object($val, $indent);
                }
                printf("%sENDGROUP\n", $istr);
        }
}
```

How to Create a New Group Record and Write It to a File

Use this script to create a new group record and write it to a file named `/tmp/exacct`.

```
#!/usr/perl5/5.6.1/bin/perl

use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
# Prototype list of catalog tags and values.
     my @items = (
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_CREATOR      => "me"       ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_PID     => $$         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_UID     => $<         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_GID     => $(         ],
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_PROC_COMMAND => "/bin/rec" ],
     );

     # Create a new group catalog object.
     my $cat = ea_new_catalog(&EXT_GROUP | &EXC_DEFAULT | &EXD_NONE)

     # Create a new Group object and retrieve its data array.
     my $group = ea_new_group($cat);
     my $ary = $group->value();

     # Push the new Items onto the Group array.
     foreach my $v (@items) {
             push(@$ary, ea_new_item(ea_new_catalog($v->[0]), $v->[1]));
     }

     # Open the exacct file, write the record & close.
     my $f = ea_new_file('/tmp/exacct', &O_RDWR | &O_CREAT | &O_TRUNC)
        || die("create /tmp/exacct failed: ", ea_error_str(), "\n");
     $f->write($group);
     $f = undef;
```

How to Print the Contents of an `exacct` File

Use the following Perl script to print the contents of an `exacct` file.

```
#!/usr/perl5/5.6.1/bin/perl

     use strict;
     use warnings;
     use Sun::Solaris::Exacct qw(:EXACCT_ALL);

     die("Usage is dumpexacct <exacct file>\n") unless (@ARGV == 1);

     # Open the exact file and display the header information.
```

```
my $ef = ea_new_file($ARGV[0], &O_RDONLY) || die(error_str());
printf("Creator:  %s\n", $ef->creator());
printf("Hostname: %s\n\n", $ef->hostname());

# Dump the file contents
while (my $obj = $ef->get()) {
        ea_dump_object($obj);
}

# Report any errors
if (ea_error() != EXR_OK && ea_error() != EXR_EOF)  {
        printf("\nERROR: %s\n", ea_error_str());
        exit(1);
}
exit(0);
```

Example Output From `Sun::Solaris::Exacct::Object->dump()`

Here is example output produced by running `Sun::Solaris::Exacct::Object->dump()` on the file created in "How to Create a New Group Record and Write It to a File" on page 58.

```
Creator:  root
Hostname: localhost
GROUP
       Catalog = EXT_GROUP|EXC_DEFAULT|EXD_NONE
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_CREATOR
         Value = me
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_PID
         Value = 845523
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_UID
         Value = 37845
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_GID
         Value = 10
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_PROC_COMMAND
         Value = /bin/rec
ENDGROUP
```

CHAPTER 6

Resource Controls (Overview)

After you determine the resource consumption of workloads on your system as described in Chapter 4, you can place boundaries on resource usage. Boundaries prevent workloads from over-consuming resources. The resource controls facility is the constraint mechanism that is used for this purpose. Resource controls:

- Extend the concept of UNIX resource limits
- Replace the System V interprocess communication (IPC) `/etc/system` tunables

This chapter covers the following topics.

For information on administering resource controls, see Chapter 7.

Introduction to Resource Controls

In the Solaris Operating System, the concept of a per-process resource limit has been extended to the task and project entities described in Chapter 2. These enhancements are provided by the *resource controls* (*rctls*) facility. In addition, allocations that were set through the `/etc/system` tunables are now automatic or configured through the resource controls mechanism as well.

A resource control is identified by the prefix `project`, `task`, or `process`. Resource controls can be observed on a system-wide basis. It is possible to update resource control values on a running system.

In a zones environment, the prefix `zone` identifies a zone-wide resource control. See "Zone Configuration Data" on page 186 for more information.

Limits, Tunables, and Resource Controls

UNIX systems have traditionally provided a resource limit facility (*rlimit*). The rlimit facility allows administrators to set one or more numerical limits on the amount of resources a process can consume. These limits include per-process CPU time used, per-process core file size, and per-process maximum heap size. Heap size is the amount of scratch memory that is allocated for the process data segment.

The resource controls facility provides compatibility interfaces for the resource limits facility. Existing applications that use resource limits continue to run unchanged. These applications can be observed in the same way as applications that are modified to take advantage of the resource controls facility.

Processes can communicate with each other by using one of several types of interprocess communication (IPC). IPC allows information transfer or synchronization to occur between processes. Previously, IPC tunable parameters were set by adding an entry to the `/etc/system` file. The resource controls facility now provides resource controls that define the behavior of the kernel's IPC facilities, replacing the `/etc/system` tunables.

To observe which IPC objects are contributing to a project's usage, use the `ipcs` command with the `-J` option. See "How to Use `ipcs`" on page 75 to view an example display. For more information about the `ipcs` command, see `ipcs`(1)

For information about the `/etc/system` tunables made obsolete by the resource controls facility, see the *Solaris Tunable Parameters Reference Manual*.

For a list of the resource controls that are available in this release, see "Available Resource Controls" on page 64.

Resource Control Constraint Mechanisms

Resource controls provide a mechanism for the constraint of system resources. Processes, tasks, and projects can be prevented from consuming amounts of specified system resources. This mechanism leads to a more manageable system by preventing over-consumption of resources.

Constraint mechanisms can be used to support capacity-planning processes. An encountered constraint can provide information about application resource needs without necessarily denying the resource to the application.

Project Attribute Mechanisms

Resource controls can also serve as a simple attribute mechanism for resource management facilities. For example, the number of CPU shares made available to a project in the fair share scheduler (FSS) scheduling class is defined by the `project.cpu-shares` resource control. Because the project is assigned a fixed number of shares by the control, the various actions associated with exceeding a control are not relevant. In this context, the current value for the `project.cpu-shares` control is considered an attribute on the specified project.

Another type of project attribute is used to regulate the resource consumption of physical memory by collections of processes attached to a project. These attributes have the prefix `rcap`, for example, `rcap.max-rss`. Like a resource control, this type of attribute is configured in the `project` database. However, while resource controls are synchronously enforced by the kernel, resource caps are asynchronously enforced at the user level by the resource cap enforcement daemon, `rcapd`. For information on `rcapd`, see Chapter 10 and `rcapd`(1M).

The `project.pool` attribute is used to specifiy a pool binding for a project. For more information on resource pools, see Chapter 12.

Configuring Resource Controls and Attributes

The resource controls facility is configured through the `project` database. See Chapter 2. Resource controls and other attributes are set in the final field of the `project` database entry. The values associated with each resource control are enclosed in parentheses, and appear as plain text separated by commas. The values in parentheses constitute an "action clause." Each action clause is composed of a privilege level, a threshold value, and an action that is associated with the particular threshold. Each resource control can have multiple action clauses, which are also separated by commas. The following entry defines a per-process address-space limit and a per-task lightweight process limit on a project entity.

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
  process.max-address-space=(privileged,209715200,deny)
```

Note – On systems that have zones enabled, zone-wide resource controls are specified in the zone configuration using a slightly different format. See "Zone Configuration Data" on page 186 for more information.

The `rctladm` command allows you to make runtime interrogations of and modifications to the resource controls facility, with global scope. The `prctl` command allows you to make runtime interrogations of and modifications to the resource controls facility, with local scope.

Note that in a zones environment, you cannot use `rctladm` in a non-global zone to modify settings. You can use `rctladm` in a non-global zone to view the global logging state of each resource control.

For more information, see `rctladm`(1M) and `prctl`(1).

Available Resource Controls

A list of the standard resource controls that are available in this release is shown in the following table.

The table describes the resource that is constrained by each control. The table also identifies the default units that are used by the `project` database for that resource. The default units are of two types:

- Quantities represent a limited amount.
- Indexes represent a maximum valid identifier.

Thus, `project.cpu-shares` specifies the number of shares to which the project is entitled. `process.max-file-descriptor` specifies the highest file number that can be assigned to a process by the `open`(2) system call.

TABLE 6–1 Standard Resource Controls

Control Name	Description	Default Unit
`project.cpu-shares`	Number of CPU shares granted to this project for use with the fair share scheduler (see `FSS`(7))	Quantity (shares)
`project.max-device-locked-memory`	Total amount of locked memory allowed	Size (bytes)
`project.max-port-ids`	Maximum allowable number of event ports	Quantity (number of event ports)
`project.max-shm-ids`	Maximum number of shared memory IDs allowed for this project	Quantity (shared memory IDs)
`project.max-sem-ids`	Maximum number of semaphore IDs allowed for this project	Quantity (semaphore IDs)

TABLE 6–1 Standard Resource Controls *(Continued)*

Control Name	Description	Default Unit
`project.max-msg-ids`	Maximum number of message queue IDs allowed for this project	Quantity (message queue IDs)
`project.max-shm-memory`	Total amount of shared memory allowed for this project	Size (bytes)
`task.max-cpu-time`	Maximum CPU time that is available to this task's processes	Time (seconds)
`task.max-lwps`	Maximum number of LWPs simultaneously available to this task's processes	Quantity (LWPs)
`process.max-cpu-time`	Maximum CPU time that is available to this process	Time (seconds)
`process.max-file-descriptor`	Maximum file descriptor index available to this process	Index (maximum file descriptor)
`process.max-file-size`	Maximum file offset available for writing by this process	Size (bytes)
`process.max-core-size`	Maximum size of a core file created by this process	Size (bytes)
`process.max-data-size`	Maximum heap memory available to this process	Size (bytes)
`process.max-stack-size`	Maximum stack memory segment available to this process	Size (bytes)
`process.max-address-space`	Maximum amount of address space, as summed over segment sizes, that is available to this process	Size (bytes)
`process.max-port-events`	Maximum allowable number of events per event port	Quantity (number of events)
`process.max-sem-nsems`	Maximum number of semaphores allowed per semaphore set	Quantity (semaphores per set)
`process.max-sem-ops`	Maximum number of semaphore operations allowed per `semop` call (value copied from the resource control at `semget()` time)	Quantity (number of operations)

TABLE 6–1 Standard Resource Controls *(Continued)*

Control Name	Description	Default Unit
`process.max-msg-qbytes`	Maximum number of bytes of messages on a message queue (value copied from the resource control at `msgget()` time)	Size (bytes)
`process.max-msg-messages`	Maximum number of messages on a message queue (value copied from the resource control at `msgget()` time)	Quantity (number of messages)
`process.min-crypto-sessions`	Number of sessions in fixed-sized session table when `/dev/crypto` is opened; value can only be changed by a privileged process	Quantity (number of sessions)
`process.add-crypto-sessions`	Number of additional sessions when session table is full and larger table is allocated	Quantity (number of additional sessions)
`process.max-crypto-sessions`	Maximum number of sessions in session table; value can only be changed by privileged process	Quantity (number of sessions)
`process.crypto-buffer-limit`	Number of bytes allocated for `copyin` of user data; sizes of all buffers allocated for `copyin` are added together and result checked against this resource control; limit applies to each instance of `/dev/crypto`; value can only be changed by privileged process	Size (bytes)

Resource Control Values and Privilege Levels

A threshold value on a resource control constitutes an enforcement point where local actions can be triggered or global actions, such as logging, can occur.

Each threshold value on a resource control must be associated with a privilege level. The privilege level must be one of the following three types.

- Basic, which can be modified by the owner of the calling process
- Privileged, which can be modified only by privileged (superuser) callers

- System, which is fixed for the duration of the operating system instance

A resource control is guaranteed to have one system value, which is defined by the system, or resource provider. The system value represents how much of the resource the current implementation of the operating system is capable of providing.

Any number of privileged values can be defined, and only one basic value is allowed. Operations that are performed without specifying a privilege value are assigned a basic privilege by default.

The privilege level for a resource control value is defined in the privilege field of the resource control block as RCTL_BASIC, RCTL_PRIVILEGED, or RCTL_SYSTEM. See `setrctl`(2) for more information. You can use the `prctl` command to modify values that are associated with basic and privileged levels.

Actions on Resource Control Values

For each threshold value that is placed on a resource control, you can associate one or more actions.

- You can choose to deny the resource requests for an amount that is greater than the threshold.
- You can choose to send a signal to the violating or observing process if the threshold value is reached.

Due to implementation restrictions, the global properties of each control can restrict the set of available actions that can be set on the threshold value. A list of available signal actions is presented in the following table. For additional information on signals, see `signal`(3HEAD).

TABLE 6–2 Signals Available to Resource Control Values

Signal	Description	Notes
SIGABRT	Terminate the process.	
SIGHUP	Send a hangup signal. Occurs when carrier drops on an open line. Signal sent to the process group that controls the terminal.	
SIGTERM	Terminate the process. Termination signal sent by software.	
SIGKILL	Terminate the process and kill the program.	
SIGSTOP	Stop the process. Job control signal.	

TABLE 6–2 Signals Available to Resource Control Values *(Continued)*

Signal	Description	Notes
SIGXRES	Resource control limit exceeded. Generated by resource control facility.	
SIGXFSZ	Terminate the process. File size limit exceeded.	Available only to resource controls with the RCTL_GLOBAL_FILE_SIZE property (`process.max-file-size`). See `rctlblk_set_value`(3C) for more information.
SIGXCPU	Terminate the process. CPU time limit exceeded.	Available only to resource controls with the RCTL_GLOBAL_CPUTIME property (`process.max-cpu-time`). See `rctlblk_set_value`(3C) for more information.

Resource Control Flags and Properties

Each resource control on the system has a certain set of associated properties. This set of properties is defined as a set of flags, which are associated with all controlled instances of that resource. Global flags cannot be modified, but the flags can be retrieved by using either `rctladm` or the `getrctl` system call.

Local flags define the default behavior and configuration for a specific threshold value of that resource control on a specific process or process collective. The local flags for one threshold value do not affect the behavior of other defined threshold values for the same resource control. However, the global flags affect the behavior for every value associated with a particular control. Local flags can be modified, within the constraints supplied by their corresponding global flags, by the `prctl` command or the `setrctl` system call. See `setrctl`(2).

For the complete list of local flags, global flags, and their definitions, see `rctlblk_set_value`(3C).

To determine system behavior when a threshold value for a particular resource control is reached, use `rctladm` to display the global flags for the resource control . For example, to display the values for `process.max-cpu-time`, type the following:

```
$ rctladm process.max-cpu-time
    process.max-cpu-time   syslog=off   [ lowerable no-deny cpu-time inf ]
```

The global flags indicate the following.

`lowerable` Superuser privileges are not required to lower the privileged values for this control.

`no-deny`	Even when threshold values are exceeded, access to the resource is never denied.
`cpu-time`	SIGXCPU is available to be sent when threshold values of this resource are reached.
`inf`	Any value that has RCTL_LOCAL_MAXIMAL defined actually represents an infinite quantity, and the value is never enforced.

Use `prctl` to display local values and actions for the resource control.

```
$ prctl -n process.max-cpu-time $$
    353939: -ksh
    process.max-cpu-time   [ lowerable no-deny cpu-time inf ]
        18446744073709551615 privileged signal=XCPU   [ max ]
        18446744073709551615 system     deny          [ max ]
```

The `max` (RCTL_LOCAL_MAXIMAL) flag is set for both threshold values, and the `inf` (RCTL_GLOBAL_INFINITE) flag is defined for this resource control. Hence, as configured, both threshold quantities represent infinite values that will never be exceeded.

Resource Control Enforcement

More than one resource control can exist on a resource. A resource control can exist at each containment level in the process model. If resource controls are active on the same resource at different container levels, the smallest container's control is enforced first. Thus, action is taken on `process.max-cpu-time` before `task.max-cpu-time` if both controls are encountered simultaneously.

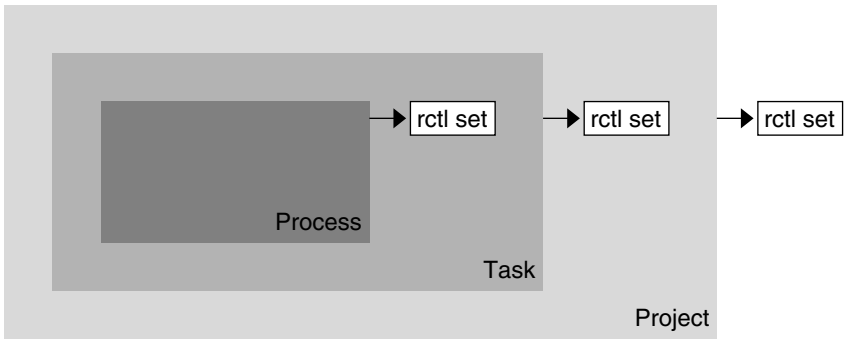


FIGURE 6–1 Process Collectives, Container Relationships, and Their Resource Control Sets

Global Monitoring of Resource Control Events

Often, the resource consumption of processes is unknown. To get more information, try using the global resource control actions that are available with the `rctladm` command. Use `rctladm` to establish a `syslog` action on a resource control. Then, if any entity managed by that resource control encounters a threshold value, a system message is logged at the configured logging level. See Chapter 7 and the `rctladm`(1M) man page for more information.

Applying Resource Controls

Each resource control listed in Table 6–1 can be assigned to a project at login or when `newtask` or the other project-aware launchers `at`, `batch`, or `cron` are invoked. Each command that is initiated is launched in a separate task with the invoking user's default project. See the man pages `login`(1), `newtask`(1), `at`(1), and `cron`(1M) for more information.

Updates to entries in the `project` database, whether to the `/etc/project` file or to a representation of the database in a network name service, are not applied to currently active projects. The updates are applied when a new task joins the project through login or `newtask`.

Temporarily Updating Resource Control Values on a Running System

Values changed in the `project` database only become effective for new tasks that are started in a project. However, you can use the `rctladm` and `prctl` commands to update resource controls on a running system.

Updating Logging Status

The `rctladm` command affects the global logging state of each resource control on a system-wide basis. This command can be used to view the global state and to set up the level of `syslog` logging when controls are exceeded.

Updating Resource Controls

You can view and temporarily alter resource control values and actions on a per-process, per-task, or per-project basis by using the `prctl` command. A project, task, or process ID is given as input, and the command operates on the resource control at the level where the control is defined.

Any modifications to values and actions take effect immediately. However, these modifications apply to the current session only. The changes are not recorded in the `project` database. If the system is restarted, the modifications are lost. Permanent changes to resource controls must be made in the `project` database.

All resource control settings that can be modified in the `project` database can also be modified with the `prctl` command. Both basic and privileged values can be added or be deleted. Their actions can also be modified. By default, the basic type is assumed for all set operations, but processes and users with superuser privileges can also modify privileged resource controls. System resource controls cannot be altered.

Commands Used With Resource Controls

Command	Description
`ipcs`(1)	Allows you to observe which IPC objects are contributing to a project's usage
`prctl`(1)	Allows you to make runtime interrogations of and modifications to the resource controls facility, with local scope
`rctladm`(1M)	Allows you to make runtime interrogations of and modifications to the resource controls facility, with global scope

CHAPTER 7

Administering Resource Controls (Tasks)

This chapter describes how to administer the resource controls facility.

Tasks in the following areas are covered.

Using Resource Controls

How to Set the Maximum Number of LWPs for Each Task in a Project

Type this entry in the `/etc/project` database to set the maximum number of LWPs in each task in project *x-files* to *3*.

```
x-files:100::root::task.max-lwps=(privileged,3,deny)
```

When superuser creates a new task in project *x-files* by joining it with `newtask`, superuser will not be able to create more than three LWPs while running in this task. This is shown in the following annotated sample session.

```
# newtask -p x-files csh

# prctl -n task.max-lwps $$
688: csh
task.max-lwps
                           3 privileged deny
                  2147483647 system     deny
# id -p
```

```
uid=0(root) gid=1(other) projid=100(x-files)

# ps -o project,taskid -p $$
 PROJECT TASKID
 x-files   236

# csh        /* creates second LWP */

# csh        /* creates third LWP */

# csh        /* cannot create more LWPs */
Vfork failed

#
```

How to Set Multiple Controls on a Project

The `/etc/project` file can contain settings for multiple resource controls for each project as well as multiple threshold values for each control. Threshold values are defined in action clauses, which are comma-separated for multiple values.

The following line in the file sets a `basic` control with no action on the maximum LWPs per task for project *x-files*. The line also sets a privileged `deny` control on the maximum LWPs per task. This control causes any LWP creation that exceeds the maximum to fail, as shown in the previous example. Finally, the maximum file descriptors per process are limited at the `basic` level, which forces the failure of any open call that exceeds the maximum.

```
x-files:101::root::task.max-lwps=(basic,10,none),(privileged,500,deny);
    process.max-file-descriptor=(basic,128,deny)
```

How to Use `prctl`

Use the `prctl` command to make runtime interrogations of and modifications to the resource controls associated with an active process, task, or project on the system. See the `prctl`(1) man page for more information.

For example, as superuser, type the following command to display the maximum file descriptor for the current shell that is running:

```
# prctl -n process.max-file-descriptor $$
8437:   sh
process.max-file-descriptor              [ lowerable deny ]
                         256 basic       deny
                       65536 privileged  deny
                  2147483647 system      deny
```

Use the `prctl` command to temporarily add a new privileged value to deny the use of more than three LWPs per task for the *x-files* project. The result is identical to the result in "How to Set the Maximum Number of LWPs for Each Task in a Project" on page 73, as shown in the following annotated sample session:

```
# newtask -p x-files

# id -p
uid=0(root) gid=1(other) projid=101(x-files)

# prctl -n task.max-lwps -t privileged -v 3 -e deny -i project x-files

# prctl -n task.max-lwps -i project x-files
670:    sh
task.max-lwps
                       3 privileged deny
              2147483647 system     deny
```

You can also use `prctl -r` to change the lowest value of a resource control.

```
# prctl -n process.max-file-descriptor -r -v 128 $$
```

How to Use `rctladm`

Use the `rctladm` command to make runtime interrogations of and modifications to the global state of the resource controls facility. See the `rctladm`(1M) man page for more information.

For example, you can use `rctladm` to enable the global `syslog` attribute of a resource control. When the control is exceeded, notification is logged at the specified `syslog` level. Type the following:

```
# rctladm -e syslog process.max-file-descriptor
```

How to Use `ipcs`

Use the `ipcs` utility to display information about active interprocess communication (IPC) facilities. See the `ipcs`(1) man page for more information.

You can use `ipcs` with the `-J` option to see which project's limit an IPC object is allocated against.

```
# ipcs -J
    IPC status from <running system> as of Wed Mar 26 18:53:15 PDT 2003
T         ID      KEY        MODE       OWNER    GROUP    PROJECT
Message Queues:
Shared Memory:
m       3600      0       --rw-rw-rw-   uname    staff    x-files
m        201      0       --rw-rw-rw-   uname    staff    x-files
```

```
m       1802      0       --rw-rw-rw-   uname    staff    x-files
m        503      0       --rw-rw-rw-   uname    staff    x-files
m        304      0       --rw-rw-rw-   uname    staff    x-files
m        605      0       --rw-rw-rw-   uname    staff    x-files
m          6      0       --rw-rw-rw-   uname    staff    x-files
m        107      0       --rw-rw-rw-   uname    staff    x-files
Semaphores:
s          0      0       --rw-rw-rw-   uname    staff    x-files
```

Capacity Warnings

A global action on a resource control enables you to receive notice of any entity that is tripping over a resource control value that is set too low.

For example, assume you want to determine whether a web server possesses sufficient CPUs for its typical workload. You could analyze `sar` data for idle CPU time and load average. You could also examine extended accounting data to determine the number of simultaneous processes that are running for the web server process.

However, an easier approach is to place the web server in a task. You can then set a global action, using `syslog`, to notify you whenever a task exceeds a scheduled number of LWPs appropriate for the machine's capabilities.

See the `sar`(1) man page for more information.

▼ How to Determine Whether a Web Server Is Allocated Enough CPU Capacity

1. **Use the `prctl` command to place a privileged (superuser-owned) resource control on the tasks that contain an `httpd` process. Limit each task's total number of LWPs to 40, and disable all local actions.**

   ```
   # prctl -n task.max-lwps -v 40 -t privileged -d all `pgrep httpd`
   ```

2. **Enable a system log global action on the `task.max-lwps` resource control.**

   ```
   # rctladm -e syslog task.max-lwps
   ```

3. **Observe whether the workload trips the resource control.**

   If it does, you will see `/var/adm/messages` such as:

   ```
   Jan  8 10:15:15 testmachine unix: [ID 859581 kern.notice]
   NOTICE: privileged rctl task.max-lwps exceeded by task 19
   ```

CHAPTER 8

Fair Share Scheduler (Overview)

The analysis of workload data can indicate that a particular workload or group of workloads is monopolizing CPU resources. If these workloads are not violating resource constraints on CPU usage, you can modify the allocation policy for CPU time on the system. The fair share scheduling class described in this chapter enables you to allocate CPU time based on shares instead of the priority scheme of the timesharing (TS) scheduling class.

This chapter covers the following topics.

To begin using the fair share scheduler, see Chapter 9.

Introduction to the Scheduler

A fundamental job of the operating system is to arbitrate which processes get access to the system's resources. The process scheduler, which is also called the dispatcher, is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the concept of scheduling classes. Each class defines a scheduling policy that is used to schedule processes within the class. The default scheduler in the

Solaris Operating System, the TS scheduler, tries to give every process relatively equal access to the available CPUs. However, you might want to specify that certain processes be given more resources than others.

You can use the *fair share scheduler* (FSS) to control the allocation of available CPU resources among workloads, based on their importance. This importance is expressed by the number of *shares* of CPU resources that you assign to each workload.

You give each project CPU shares to control the project's entitlement to CPU resources. The FSS guarantees a fair dispersion of CPU resources among projects that is based on allocated shares, independent of the number of processes that are attached to a project. The FSS achieves fairness by reducing a project's entitlement for heavy CPU usage and increasing its entitlement for light usage, in accordance with other projects.

The FSS consists of a kernel scheduling class module and class-specific versions of the `dispadmin`(1M) and `priocntl`(1) commands. Project shares used by the FSS are specified through the `project.cpu-shares` property in the `project`(4) database.

Note – If you are using the `project.cpu-shares` resource control in a zones environment, see "Zone Configuration Data" on page 186 and "Resource Controls Used in Non-Global Zones" on page 240 for more information.

CPU Share Definition

The term "share" is used to define a portion of the system's CPU resources that is allocated to a project. If you assign a greater number of CPU shares to a project, relative to other projects, the project receives more CPU resources from the fair share scheduler.

CPU shares are not equivalent to percentages of CPU resources. Shares are used to define the relative importance of workloads in relation to other workloads. When you assign CPU shares to a project, your primary concern is not the number of shares the project has. Knowing how many shares the project has in comparison with other projects is more important. You must also take into account how many of those other projects will be competing with it for CPU resources.

Note – Processes in projects with zero shares always run at the lowest system priority (0). These processes only run when projects with nonzero shares are not using CPU resources.

CPU Shares and Process State

In the Solaris system, a project workload usually consists of more than one process. From the fair share scheduler perspective, each project workload can be in either an *idle* state or an *active* state. A project is considered idle if none of its processes are using any CPU resources. This usually means that such processes are either *sleeping* (waiting for I/O completion) or stopped. A project is considered active if at least one of its processes is using CPU resources. The sum of shares of all active projects is used in calculating the portion of CPU resources to be assigned to projects.

The following formula shows how the FSS scheduler calculates per-project allocation of CPU resources.

$$\text{allocation}_{\text{project}^i} = \frac{\text{shares}_{\text{project}^i}}{\sum_{j=1...n} (\text{shares}_{\text{project}^j})}$$

j is the index among all active projects

FIGURE 8–1 FSS Scheduler Share Calculation

When more projects become active, each project's CPU allocation is reduced, but the proportion between the allocations of different projects does not change.

CPU Share Versus Utilization

Share allocation is not the same as utilization. A project that is allocated 50 percent of the CPU resources might average only a 20 percent CPU use. Moreover, shares serve to limit CPU usage only when there is competition from other projects. Regardless of how low a project's allocation is, it always receives 100 percent of the processing power if it is running alone on the system. Available CPU cycles are never wasted. They are distributed between projects.

The allocation of a small share to a busy workload might slow its performance. However, the workload is not prevented from completing its work if the system is not overloaded.

CPU Share Examples

Assume you have a system with two CPUs running two parallel CPU-bound workloads called *A* and *B*, respectively. Each workload is running as a separate project. The projects have been configured so that project *A* is assigned S_A shares, and project *B* is assigned S_B shares.

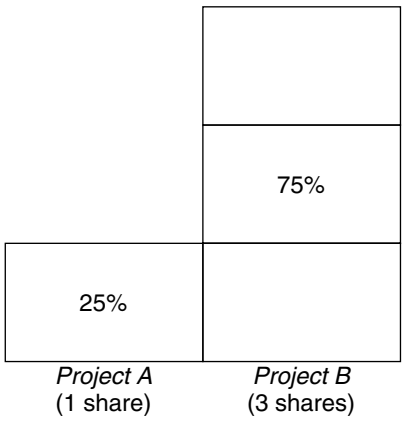
On average, under the traditional TS scheduler, each of the workloads that is running on the system would be given the same amount of CPU resources. Each workload would get 50 percent of the system's capacity.

When run under the control of the FSS scheduler with S_A=S_B, these projects are also given approximately the same amounts of CPU resources. However, if the projects are given different numbers of shares, their CPU resource allocations are different.

The next three examples illustrate how shares work in different configurations. These examples show that shares are only mathematically accurate for representing the usage if demand meets or exceeds available resources.

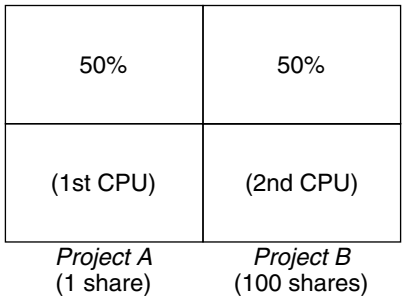
Example 1: Two CPU-Bound Processes in Each Project

If *A* and *B* each have two CPU-bound processes, and $S_A = 1$ and $S_B = 3$, then the total number of shares is 1 + 3 = 4. In this configuration, given sufficient CPU demand, projects *A* and *B* are allocated 25 percent and 75 percent of CPU resources, respectively.



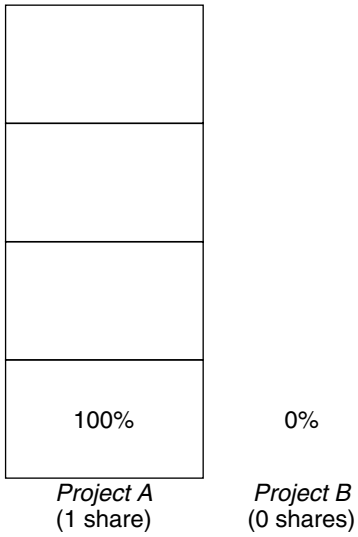
Example 2: No Competition Between Projects

If *A* and *B* have only *one* CPU-bound process each, and $S_A = 1$ and $S_B = 100$, then the total number of shares is 101. Each project cannot use more than one CPU because each project has only one running process. Because no competition exists between projects for CPU resources in this configuration, projects *A* and *B* are each allocated 50 percent of all CPU resources. In this configuration, CPU share values are irrelevant. The projects' allocations would be the same (50/50), even if both projects were assigned zero shares.



Example 3: One Project Unable to Run

If *A* and *B* have two CPU-bound processes each, and project *A* is given 1 share and project *B* is given 0 shares, then project *B* is not allocated any CPU resources and project *A* is allocated all CPU resources. Processes in *B* always run at system priority 0, so they never be able to run because processes in project *A* always have higher priorities.



FSS Configuration

Projects and Users

Projects are the workload containers in the FSS scheduler. Groups of users who are assigned to a project are treated as single controllable blocks. Note that you can create a project with its own number of shares for an individual user.

Users can be members of multiple projects that have different numbers of shares assigned. By moving processes from one project to another project, processes can be assigned CPU resources in varying amounts.

For more information on the `project`(4) database and name services, see "`project` Database" on page 33.

CPU Shares Configuration

The configuration of CPU shares is managed by the name service as a property of the `project` database.

When the first task (or process) that is associated with a project is created through the `setproject`(3PROJECT) library function, the number of CPU shares defined as resource control `project.cpu-shares` in the `project` database is passed to the kernel. A project that does not have the `project.cpu-shares` resource control defined is assigned one share.

In the following example, this entry in the `/etc/project` file sets the number of shares for project *x-files* to *5*:

```
x-files:100::::project.cpu-shares=(privileged,5,none)
```

If you alter the number of CPU shares allocated to a project in the database when processes are already running, the number of shares for that project will not be modified at that point. The project must be restarted for the change to become effective.

If you want to temporarily change the number of shares assigned to a project without altering the project's attributes in the `project` database, use the `prctl` command. For example, to change the value of project *x-files*'s `project.cpu-shares` resource control to *3* while processes associated with that project are running, type the following:

```
# prctl -r -n project.cpu-shares -v 3 -i project x-files
```

See the `prctl`(1) man page for more information.

`-r`	Replaces the current value for the named resource control.
`-n` *name*	Specifies the name of the resource control.
`-v` *val*	Specifies the value for the resource control.
`-i` *idtype*	Specifies the ID type of the next argument.
x-files	Specifies the object of the change. In this instance, project *x-files* is the object.

Project `system` with project ID 0 includes all system daemons that are started by the boot-time initialization scripts. `system` can be viewed as a project with an unlimited number of shares. This means that `system` is always scheduled first, regardless of how many shares have been given to other projects. If you do not want the `system` project to have unlimited shares, you can specify a number of shares for this project in the `project` database.

As stated previously, processes that belong to projects with zero shares are always given zero system priority. Projects with one or more shares are running with priorities one and higher. Thus, projects with zero shares are only scheduled when CPU resources are available that are not requested by a nonzero share project.

The maximum number of shares that can be assigned to one project is 65535.

FSS and Processor Sets

The FSS can be used in conjunction with processor sets to provide more fine-grained controls over allocations of CPU resources among projects that run on each processor set than would be available with processor sets alone. The FSS scheduler treats processor sets as entirely independent partitions, with each processor set controlled independently with respect to CPU allocations.

The CPU allocations of projects running in one processor set are not affected by the CPU shares or activity of projects running in another processor set because the projects are not competing for the same resources. Projects only compete with each other if they are running within the same processor set.

The number of shares allocated to a project is system wide. Regardless of which processor set it is running on, each portion of a project is given the same amount of shares.

When processor sets are used, project CPU allocations are calculated for active projects that run within each processor set, as shown in the following figure.

$$\text{allocation}_{\text{project}_x^i} = \frac{\text{shares}_{\text{project}_x^i}}{\text{processor set } X \quad \sum_{j=1...n} (\text{shares}_{\text{project}^j})}$$

j is the index among all active projects
that run on processor set X

FIGURE 8–2 FSS Scheduler Share Calculation With Processor Sets

Project partitions that run on different processor sets might have different CPU allocations. The CPU allocation for each project partition in a processor set depends only on the allocations of other projects that run on the same processor set.

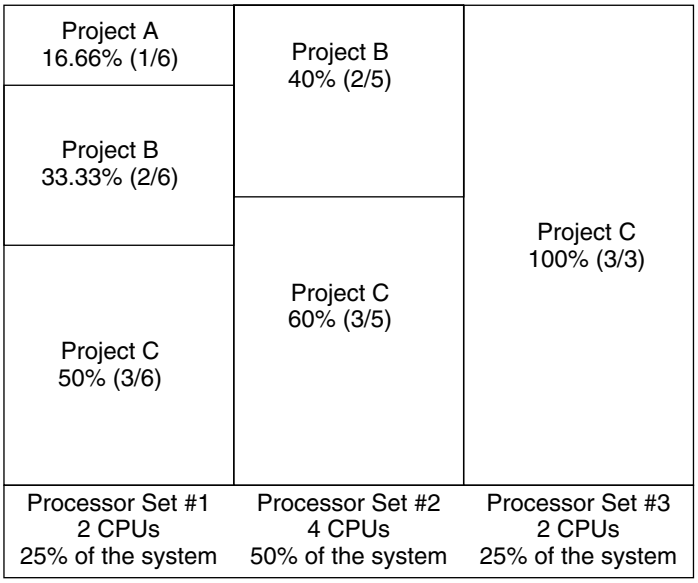
The performance and availability of applications that run within the boundaries of their processor sets are not affected by the introduction of new processor sets. The applications are also not affected by changes that are made to the share allocations of projects that run on other processor sets.

Empty processor sets (sets without processors in them) or processor sets without processes bound to them do not have any impact on the FSS scheduler behavior.

FSS and Processor Sets Examples

Assume that a server with eight CPUs is running several CPU-bound applications in projects *A*, *B*, and *C*. Project *A* is allocated one share, project *B* is allocated two shares, and project *C* is allocated three shares.

Project *A* is running only on processor set 1. Project *B* is running on processor sets 1 and 2. Project *C* is running on processor sets 1, 2, and 3. Assume that each project has enough processes to utilize all available CPU power. Thus, there is always competition for CPU resources on each processor set.



The total system-wide project CPU allocations on such a system are shown in the following table.

Project	Allocation
Project A	$4\% = (1/6 \text{ X } 2/8)_{pset1}$
Project B	$28\% = (2/6 \text{ X } 2/8)_{pset1} + (2/5 * 4/8)_{pset2}$
Project C	$67\% = (3/6 \text{ X } 2/8)_{pset1} + (3/5 \text{ X } 4/8)_{pset2} + (3/3 \text{ X } 2/8)_{pset3}$

These percentages do not match the corresponding amounts of CPU shares that are given to projects. However, within each processor set, the per-project CPU allocation ratios are proportional to their respective shares.

On the same system *without* processor sets, the distribution of CPU resources would be different, as shown in the following table.

Project	Allocation
Project A	16.66% = (1/6)
Project B	33.33% = (2/6)
Project C	50% = (3/6)

Combining FSS With Other Scheduling Classes

By default, the FSS scheduling class uses the same range of priorities (0 to 59) as the timesharing (TS), interactive (IA), and fixed priority (FX) scheduling classes. Therefore, you should avoid having processes from these scheduling classes share *the same* processor set. A mix of processes in the FSS, TS, IA, and FX classes could result in unexpected scheduling behavior.

With the use of processor sets, you can mix TS, IA, and FX with FSS in one system. However, all the processes that run on each processor set must be in *one* scheduling class, so they do not compete for the same CPUs. The FX scheduler in particular should not be used in conjunction with the FSS scheduling class unless processor sets are used. This action prevents applications in the FX class from using priorities high enough to starve applications in the FSS class.

You can mix processes in the TS and IA classes in the same processor set, or on the same system without processor sets.

The Solaris system also offers a real-time (RT) scheduler to users with superuser privileges. By default, the RT scheduling class uses system priorities in a different range (usually from 100 to 159) than FSS. Because RT and FSS are using disjoint ranges of priorities, FSS can coexist with the RT scheduling class within the same processor set. However, the FSS scheduling class does not have any control over processes that run in the RT class.

For example, on a four-processor system, a single-threaded RT process can consume one entire processor if the process is CPU bound. If the system also runs FSS, regular user processes compete for the three remaining CPUs that are not being used by the RT process. Note that the RT process might not use the CPU continuously. When the RT process is idle, FSS utilizes all four processors.

You can type the following command to find out which scheduling classes the processor sets are running in and ensure that each processor set is configured to run either TS, IA, FX, or FSS processes.

```
$ ps -ef -o pset,class | grep -v CLS | sort | uniq
1 FSS
1 SYS
2 TS
2 RT
3 FX
```

To set the default scheduler for the system, see "How to Set the Scheduler Class" on page 90 and `dispadmin`(1M). To move running processes into a different scheduling class, see "FSS Configuration Examples" on page 90 and `priocntl`(1).

Commands Used With FSS

The commands that are shown in the following table provide the primary administrative interface to the fair share scheduler.

Command	Description
`priocntl`(1)	Displays or sets scheduling parameters of specified processes, moves running processes into a different scheduling class.
`ps`(1)	Lists information about running processes, identifies in which scheduling classes processor sets are running.
`dispadmin`(1M)	Sets the default scheduler for the system. Also used to examine and tune the FSS scheduler's time quantum value.
`FSS`(7)	Describes the fair share scheduler (FSS).

CHAPTER 9

Administering the Fair Share Scheduler (Tasks)

This chapter describes how to use the fair share scheduler (FSS). The following topics are covered.

Monitoring the FSS

You can use the `prstat` command (see `prstat`(1M)) to monitor CPU usage by active projects.

You can use the extended accounting data for tasks to obtain per-project statistics on the amount of CPU resources that are consumed over longer periods. See Chapter 4 for more information.

How to Monitor System CPU Usage by Projects

To monitor the CPU usage of projects that run on the system, use the `prstat` command with the `-J` option:

```
% prstat -J
```

How to Monitor CPU Usage by Projects in Processor Sets

To monitor the CPU usage of projects on a list of processor sets, type the following:

```
% prstat -J -C pset-list
```

pset-list List of processor set IDs that are separated by commas

FSS Configuration Examples

The same commands that you use with other scheduling classes in the Solaris system can be used with FSS. You can set the scheduler class, configure the scheduler's tunable parameters, and configure the properties of individual processes.

How to Set the Scheduler Class

Use the `dispadmin` command to set FSS as the default scheduler for the system.

```
# dispadmin -d FSS
```

This change takes effect on the next reboot. After reboot, every process on the system runs in the FSS scheduling class.

▼ How to Manually Move Processes From the TS Class Into the FSS Class

You can manually move processes from one scheduling class to another scheduling class without changing the default scheduling class and rebooting. This example shows how to manually move processes from the TS scheduling class into the FSS scheduling class.

1. **Become superuser.**
2. **Move the `init` process (pid 1) into the FSS scheduling class.**

   ```
   # priocntl -s -c FSS -i pid 1
   ```

3. **Move all processes from the TS scheduling class into the FSS scheduling class.**

   ```
   # priocntl -s -c FSS -i class TS
   ```

All processes again run in the TS scheduling class after reboot.

▼ How to Manually Move Processes From all User Classes Into the FSS Class

You might be using a default class other than TS. For example, your system might be running a window environment that uses the IA class by default. You can manually move all processes into the FSS scheduling class without changing the default scheduling class and rebooting.

1. **Become superuser.**
2. **Move the `init` process (pid 1) into the FSS scheduling class.**

   ```
   # priocntl -s -c FSS -i pid 1
   ```

3. **Move all processes from their current scheduling classes into the FSS scheduling class.**

   ```
   # priocntl -s -c FSS -i all
   ```

All processes again run in the default scheduling class after reboot.

▼ How to Move a Project's Processes Into the FSS Class

You can manually move a project's processes from their current scheduling class to the FSS scheduling class.

1. **Become superuser.**
2. **Move processes that run in project ID *10* to the FSS scheduling class.**

   ```
   # priocntl -s -c FSS -i projid 10
   ```

   The project's processes again run in the default scheduling class after reboot.

How to Tune Scheduler Parameters

You can use the `dispadmin` command to examine and tune the FSS scheduler's time quantum value. *Time quantum* is the amount of time that a thread is allowed to run before it must relinquish the processor. To display the current time quantum for the FSS scheduler, type the following:

```
$ dispadmin -c FSS -g
#
# Fair Share Scheduler Configuration
#
RES=1000
```

```
#
# Time Quantum
#
QUANTUM=110
```

When you use the `-g` option, you can also use the `-r` option to specify the resolution that is used for printing time quantum values. If no resolution is specified, time quantum values are displayed in milliseconds by default. Type the following:

```
$ dispadmin -c FSS -g -r 100
#
# Fair Share Scheduler Configuration
#
RES=100
#
# Time Quantum
#
QUANTUM=11
```

To set scheduling parameters for the FSS scheduling class, use `dispadmin -s`. The values in *file* must be in the format output by the `-g` option. These values overwrite the current values in the kernel. Type the following:

```
$ dispadmin -c FSS -s file
```

CHAPTER 10

Physical Memory Control Using the Resource Capping Daemon

The resource capping daemon `rcapd` regulates physical memory consumption by processes running in projects that have resource caps defined. The following topics are covered.

Resource Capping Daemon Overview

A resource *cap* is an upper bound placed on the consumption of a resource, such as physical memory. Per-project physical memory caps are supported.

The resource capping daemon and its associated utilities provide mechanisms for physical memory resource cap enforcement and administration.

Like the resource control, the resource cap can be defined by using attributes of project entries in the `project` database. However, while resource controls are synchronously enforced by the kernel, resource caps are asynchronously enforced at the user level by the resource capping daemon. With asynchronous enforcement, a small delay occurs as a result of the sampling interval used by the daemon.

For information about `rcapd`, see the `rcapd`(1M) man page. For information about projects and the `project` database, see Chapter 2 and the `project`(4) man page. For information about resource controls, see Chapter 6.

Note – If you are using `rcapd` in a zones environment, you must add a `project` entry and configure the daemon in each zone where you want the daemon to run. `rcapd` will not act on processes in zones other than the one in which it is running.

How Resource Capping Works

The daemon repeatedly samples the resource utilization of projects that have physical memory caps. The sampling interval used by the daemon is specified by the administrator. See "Determining Sample Intervals" on page 98 for additional information. When the system's physical memory utilization exceeds the threshold for cap enforcement, and other conditions are met, the daemon takes action to reduce the resource consumption of projects with memory caps to levels at or below the caps.

The virtual memory system divides physical memory into segments known as pages. Pages are the fundamental unit of physical memory in the Solaris memory management subsystem. To read data from a file into memory, the virtual memory system reads in one page at a time, or *pages in* a file. To reduce resource consumption, the daemon can *page out*, or relocate, infrequently used pages to a swap device, which is an area outside of physical memory.

The daemon manages physical memory by regulating the size of a project workload's resident set relative to the size of its working set. The resident set is the set of pages that are resident in physical memory. The working set is the set of pages that the workload actively uses during its processing cycle. The working set changes over time, depending on the process's mode of operation and the type of data being processed. Ideally, every workload has access to enough physical memory to enable its working set to remain resident. However, the working set can also include the use of secondary disk storage to hold the memory that does not fit in physical memory.

Only one instance of `rcapd` can run at any given time.

Attribute to Limit Physical Memory Usage

To define a physical memory resource cap for a project, establish a resident set size (RSS) cap by adding this attribute to the `project` database entry:

`rcap.max-rss` The total amount of physical memory, in bytes, that is available to processes in the project.

For example, the following line in the `/etc/project` database sets an RSS cap of 10 gigabytes for a project named `db`.

```
db:100::db,root::rcap.max-rss=10737418240
```

Note – The system might round the specified cap value to a page size.

`rcapd` Configuration

You use the `rcapadm` command to configure the resource capping daemon. You can perform the following actions:

- Set the threshold value for cap enforcement
- Set intervals for the operations performed by `rcapd`
- Enable or disable resource capping
- Display the current status of the configured resource capping daemon

To configure the daemon, you must have superuser privileges or have the Process Management profile in your list of profiles. The Process Management role and the System Administrator role both include the Process Management profile.

Configuration changes can be incorporated into `rcapd` according to the configuration interval (see "`rcapd` Operation Intervals" on page 97) or on demand by sending a `SIGHUP` (see the `kill`(1) man page).

If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured.

The following subsections discuss cap enforcement, cap values, and `rcapd` operation intervals.

Memory Cap Enforcement Threshold

The *memory cap enforcement threshold* is the percentage of physical memory utilization on the system that triggers cap enforcement. When the system exceeds this utilization, caps are enforced. The physical memory used by applications and the kernel is included in this percentage. The percentage of utilization determines the way in which memory caps are enforced.

To enforce caps, memory can be paged out from project workloads.

- Memory can be paged out to reduce the size of the portion of memory that is over its cap for a given workload.
- Memory can be paged out to reduce the proportion of physical memory used that is over the memory cap enforcement threshold on the system.

A workload is permitted to use physical memory up to its cap. A workload can use additional memory as long as the system's memory utilization stays below the memory cap enforcement threshold.

To set the value for cap enforcement, see "How to Set the Memory Cap Enforcement Threshold" on page 101.

Determining Cap Values

If a project cap is set too low, there might not be enough memory for the workload to proceed effectively under normal conditions. The paging that occurs because the workload requires more memory has a negative effect on system performance.

Projects that have caps set too high can consume available physical memory before their caps are exceeded. In this case, physical memory is effectively managed by the kernel and not by `rcapd`.

In determining caps on projects, consider these factors.

Impact on I/O system	The daemon can attempt to reduce a project workload's physical memory usage whenever the sampled usage exceeds the project's cap. During cap enforcement, the swap devices and other devices that contain files that the workload has mapped are used. The performance of the swap devices is a critical factor in determining the performance of a workload that routinely exceeds its cap. The execution of the workload is similar to running it on a machine with the same amount of physical memory as the workload's cap.
Impact on CPU usage	The daemon's CPU usage varies with the number of processes in the project workloads it is capping and the sizes of the workloads' address spaces. A small portion of the daemon's CPU time is spent sampling the usage of each workload. Adding processes to workloads increases the time spent sampling usage.

	Another portion of the daemon's CPU time is spent enforcing caps when they are exceeded. The time spent is proportional to the amount of virtual memory involved. CPU time spent increases or decreases in response to corresponding changes in the total size of a workload's address space. This information is reported in the `vm` column of `rcapstat` output. See "Monitoring Resource Utilization With `rcapstat`" on page 99 and the `rcapstat`(1) man page for more information.
Reporting on shared memory	The daemon cannot determine which pages of memory are shared with other processes or which are mapped multiple times within the same process. Since `rcapd` assumes that each page is unique, this results in a discrepancy between the reported (estimated) RSS and the actual RSS. Certain workloads, such as databases, use shared memory extensively. For these workloads, you can sample a project's regular usage to determine a suitable initial cap value. Use output from the `prstat` command with the `-J` option. See the `prstat`(1M) man page.

`rcapd` Operation Intervals

You can tune the intervals for the periodic operations performed by `rcapd`.

All intervals are specified in seconds. The `rcapd` operations and their default interval values are described in the following table.

Operation	Default Interval Value in Seconds	Description
`scan`	15	Number of seconds between scans for processes that have joined or left a project workload. Minimum value is 1 second.
`sample`	5	Number of seconds between samplings of resident set size and subsequent cap enforcements. Minimum value is 1 second.

Operation	Default Interval Value in Seconds	Description
`report`	5	Number of seconds between updates to paging statistics. If set to `0`, statistics are not updated, and output from `rcapstat` is not current.
`config`	60	Number of seconds between reconfigurations. In a reconfiguration event, `rcapadm` reads the configuration file for updates, and scans the `project` database for new or revised project caps. Sending a `SIGHUP` to `rcapd` causes an immediate reconfiguration.

To tune intervals, see "How to Set Operation Intervals" on page 102.

Determining `rcapd` Scan Intervals

The scan interval controls how often `rcapd` looks for new processes. On systems with many processes running, the scan through the list takes more time, so it might be preferable to lengthen the interval in order to reduce the overall CPU time spent. However, the scan interval also represents the minimum amount of time that a process must exist to be attributed to a capped workload. If there are workloads that run many short-lived processes, `rcapd` might not attribute the processes to a workload if the scan interval is lengthened.

Determining Sample Intervals

The sample interval configured with `rcapadm` is the shortest amount of time `rcapd` waits between sampling a workload's usage and enforcing the cap if it is exceeded. If you reduce this interval, `rcapd` will, under most conditions, enforce caps more frequently, possibly resulting in increased I/O due to paging. However, a shorter sample interval can also lessen the impact that a sudden increase in a particular workload's physical memory usage might have on other workloads. The window between samplings, in which the workload can consume memory unhindered and possibly take memory from other capped workloads, is narrowed.

If the sample interval specified to `rcapstat` is shorter than the interval specified to `rcapd` with `rcapadm`, the output for some intervals can be zero. This situation occurs because `rcapd` does not update statistics more frequently than the interval specified with `rcapadm`. The interval specified with `rcapadm` is independent of the sampling interval used by `rcapstat`.

Monitoring Resource Utilization With `rcapstat`

Use `rcapstat` to monitor the resource utilization of capped projects. To view an example `rcapstat` report, see "Producing Reports With `rcapstat`" on page 103.

You can set the sampling interval for the report and specify the number of times that statistics are repeated.

interval Specifies the sampling interval in seconds. The default interval is 5 seconds.

count Specifies the number of times that the statistics are repeated. By default, `rcapstat` reports statistics until a termination signal is received or until the `rcapd` process exits.

The paging statistics in the first report issued by `rcapstat` show the activity since the daemon was started. Subsequent reports reflect the activity since the last report was issued.

The following table defines the column headings in an `rcapstat` report.

`rcapstat` Column Headings	**Description**
`id`	The project ID of the capped project.
`project`	The project name.
`nproc`	The number of processes in the project.
`vm`	The total amount of virtual memory size used by processes in the project, in kilobytes (K), megabytes (M), or gigabytes (G).
`rss`	The estimated amount of the total resident set size (RSS) of the processes in the project, in kilobytes (K), megabytes (M), or gigabytes (G), not accounting for pages that are shared.
`cap`	The RSS cap defined for the project. See "Attribute to Limit Physical Memory Usage" on page 94 or the `rcapd`(1M) man page for information about how to specify memory caps.
`at`	The total amount of memory that `rcapd` attempted to page out since the last `rcapstat` sample.

`rcapstat` Column Headings	Description
`avgat`	The average amount of memory that `rcapd` attempted to page out during each sample cycle that occurred since the last `rcapstat` sample. The rate at which `rcapd` samples collection RSS can be set with `rcapadm`. See "`rcapd` Operation Intervals" on page 97.
`pg`	The total amount of memory that `rcapd` successfully paged out since the last `rcapstat` sample.
`avgpg`	An estimate of the average amount of memory that `rcapd` successfully paged out during each sample cycle that occurred since the last `rcapstat` sample. The rate at which `rcapd` samples process RSS sizes can be set with `rcapadm`. See "`rcapd` Operation Intervals" on page 97.

Commands Used With `rcapd`

Command	Description
`rcapstat`(1)	Monitor the resource utilization of capped projects
`rcapadm`(1M)	Configure the resource capping daemon, display the current status of the resource capping daemon if it has been configured, and enable or disable resource capping
`rcapd`(1M)	Resource capping daemon

CHAPTER 11

Administering the Resource Capping Daemon (Tasks)

This section contains procedures for configuring and using the resource capping daemon `rcapd`. The following topics are covered.

Administering the Resource Capping Daemon With `rcapadm`

This section contains procedures for configuring the resource capping daemon with `rcapadm`. See "`rcapd` Configuration" on page 95 and the `rcapadm`(1M) man page for more information.

If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured.

How to Set the Memory Cap Enforcement Threshold

Caps can be configured so that they will not be enforced until the physical memory available to processes is low. See "Memory Cap Enforcement Threshold" on page 95 for more information.

The minimum (and default) value is `0`, which means that memory caps are always enforced. To set a different minimum, follow this procedure.

1. **Become superuser.**

2. **Use the `-c` option of `rcapadm` to set a different physical memory utilization value for memory cap enforcement.**

   # **`rcapadm -c`** *percent*

   percent is in the range 0 to 100. Higher values are less restrictive. A higher value means capped project workloads can execute without having caps enforced until the system's memory utilization exceeds this threshold.

To display the current physical memory utilization and the cap enforcement threshold, see Reporting Memory Utilization and the Memory Cap Enforcement Threshold.

How to Set Operation Intervals

"`rcapd` Operation Intervals" on page 97 contains information about the intervals for the periodic operations performed by `rcapd`. To set operation intervals using `rcapadm`, follow this procedure.

1. **Become superuser.**
2. **Use the `-i` option to set interval values.**

   # **`rcapadm -i`** *interval=value*`,...,`*interval=value*

All interval values are specified in seconds.

▼ How to Enable Resource Capping

There are two ways to enable resource capping on your system.

1. **Become superuser.**
2. **Enable the resource capping daemon in one of the following ways:**
   - To enable the resource capping daemon so that it will be started now and also be started each time the system is booted, type:

     # **`rcapadm -E`**
   - To enable the resource capping daemon at boot without starting it now, also specify the `-n` option:

     # **`rcapadm -n -E`**

How to Disable Resource Capping

There are two ways to disable resource capping on your system.

1. **Become superuser.**
2. **Disable the resource capping daemon in one of the following ways:**

3.

- To disable the resource capping daemon so that it will be stopped now and not be started when the system is booted, type:

  ```
  # rcapadm -D
  ```

- To disable the resource capping daemon without stopping it, also specify the `-n` option:

  ```
  # rcapadm -n -D
  ```

Note – Use `rcapadm -D` to safely disable `rcapd`. If the daemon is killed (see the `kill`(1) man page), processes might be left in a stopped state and need to be manually restarted. To resume a process running, use the `prun` command. See the `prun`(1) man page for more information.

Producing Reports With `rcapstat`

Use `rcapstat` to report resource capping statistics. "Monitoring Resource Utilization With `rcapstat`" on page 99 explains how to use the `rcapstat` command to generate reports. That section also describes the column headings in the report. The `rcapstat`(1) man page also contains this information.

The following subsections use examples to illustrate how to produce reports for specific purposes.

Reporting Cap and Project Information

In this example, caps are defined for two projects associated with two users. `user1` has a cap of 50 megabytes, and `user2` has a cap of 10 megabytes.

The following command produces five reports at 5-second sampling intervals.

```
user1machine% rcapstat 5 5
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M   50M    0K 3312K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
```

```
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
```

The first three lines of output constitute the first report, which contains the cap and project information for the two projects and paging statistics since `rcapd` was started. The `at` and `pg` columns are a number greater than zero for `user1` and zero for `user2`, which indicates that at some time in the daemon's history, `user1` exceeded its cap but `user2` did not.

The subsequent reports show no significant activity.

Monitoring the RSS of a Project

The following example shows project `user1`, which has an RSS in excess of its RSS cap.

The following command produces five reports at 5-second sampling intervals.

```
user1machine% rcapstat 5 5

    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M  220M  5528K  2764K
376565   user1      3 6249M 6144M 6144M    0M  131M  4912K  1637K
376565   user1      3 6249M 6171M 6144M   27M  147M  6048K  2016K
376565   user1      3 6249M 6146M 6144M 4872M  174M  4368K  1456K
376565   user1      3 6249M 6156M 6144M   12M  161M  3376K  1125K
```

The `user1` project has three processes that are actively using physical memory. The positive values in the `pg` column indicate that `rcapd` is consistently paging out memory as it attempts to meet the cap by lowering the physical memory utilization of the project's processes. However, `rcapd` does not succeed in keeping the RSS below the cap value. This is indicated by the varying `rss` values that do not show a corresponding decrease. As soon as memory is paged out, the workload uses it again and the RSS count goes back up. This means that all of the project's resident memory is being actively used and the working set size (*WSS*) is greater than the cap. Thus, `rcapd` is forced to page out some of the working set to meet the cap. Under this condition, the system will continue to experience high page fault rates, and associated I/O, until one of the following occurs:

- The WSS becomes smaller.
- The cap is raised.
- The application changes its memory access pattern.

In this situation, shortening the sample interval might reduce the discrepancy between the RSS value and the cap value by causing `rcapd` to sample the workload and enforce caps more frequently.

Note – A page fault occurs when either a new page must be created or the system must copy in a page from a swap device.

Determining the Working Set Size of a Project

The following example is a continuation of the previous example, and it uses the same project.

The previous example shows that the `user1` project is using more physical memory than its cap allows. This example shows how much memory the project workload requires.

```
user1machine% rcapstat 5 5
    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
```

Halfway through the cycle, the cap on the `user1` project was increased from 6 gigabytes to 10 gigabytes. This increase stops cap enforcement and allows the resident set size to grow, limited only by other processes and the amount of memory in the machine. The `rss` column might stabilize to reflect the project working set size (WSS), 6247M in this example. This is the minimum cap value that allows the project's processes to operate without continually incurring page faults.

The following two figures graphically show the effect `rcapd` has on `user1` while the cap is 6 gigabytes and 10 gigabytes. Every 5 seconds, corresponding to the sample interval, the RSS decreases and I/O increases as `rcapd` pages out some of the workload's memory. Shortly after the page out completes, the workload, needing those pages, pages them back in as it continues running. This cycle repeats until the cap is raised to 10 gigabytes approximately halfway through the example, and the RSS stabilizes at 6.1 gigabytes. Since the workload's RSS is now below the cap, no more paging occurs. The I/O associated with paging stops as well, as the `vmstat` (see `vmstat`(1M)) or `iostat` (see `iostat`(1M)) commands would show. Thus, you can infer that the project required 6.1 gigabytes to perform the work it was doing at the time it was being observed.

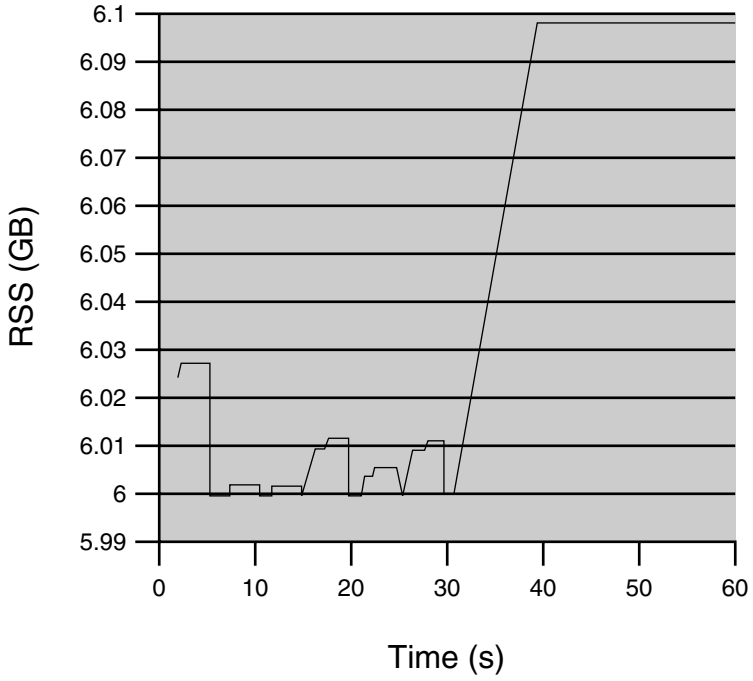


FIGURE 11–1 Stabilizing RSS Values After Raising the Cap of `user1` Higher Than `user1`'s WSS

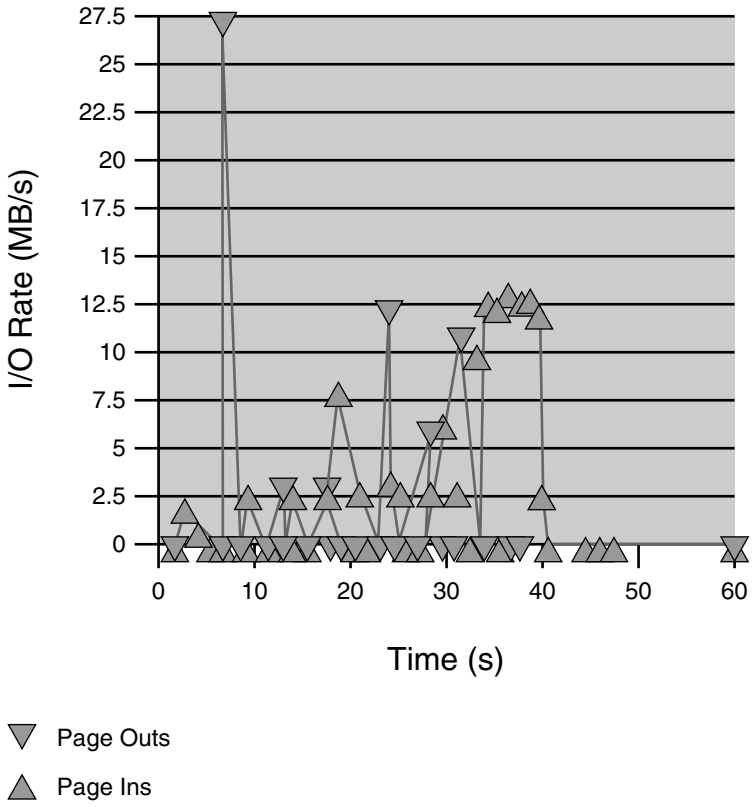


FIGURE 11–2 Relationship Between Page Ins and Page Outs, and the Stabilization of I/O After `user1`'s Cap Is Raised

Reporting Memory Utilization and the Memory Cap Enforcement Threshold

You can use the `-g` option of `rcapstat` to report the following:

- Current physical memory utilization as a percentage of physical memory installed on the system
- System memory cap enforcement threshold set by `rcapadm`

The `-g` option causes a memory utilization and cap enforcement line to be printed at the end of the report for each interval.

```
# rcapstat -g
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
```

```
physical memory utilization: 55%   cap enforcement threshold: 0%
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
```

CHAPTER 12

Dynamic Resource Pools (Overview)

This chapter discusses resource pools, which are used for partitioning machine resources. Dynamic resource pools (DRPs) dynamically adjust each resource pool's resource allocation to meet established system goals.

The following topics are covered.

Introduction to Resource Pools

Resource pools enable you to separate workloads so that workload consumption of certain resources does not overlap. This resource reservation helps to achieve predictable performance on systems with mixed workloads.

Resource pools provide a persistent configuration mechanism for processor set (pset) configuration and, optionally, scheduling class assignment.

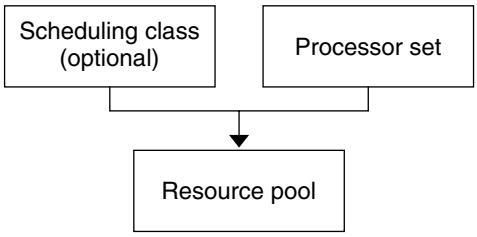


FIGURE 12–1 Resource Pool Framework

A pool can be thought of as a specific binding of the various resource sets that are available on your system. You can create pools that represent different kinds of possible resource combinations:

```
pool1: pset_default
pool2: pset1
pool3: pset1, pool.scheduler="FSS"
```

By grouping multiple partitions, pools provide a handle to associate with labeled workloads. Each project entry in the `/etc/project` database can have a pool associated with that entry, which is specified using the `project.pool` attribute. New work that is started on a project is bound to the appropriate pool. See Chapter 2 for more information.

Resource pools now provide a mechanism for dynamically adjusting each pool's resource allocation in response to system events and application load changes. DRPs simplify and reduce the number of decisions required from an administrator. Adjustments are automatically made to preserve the system performance goals specified by an administrator. The changes made to the configuration are logged. These features are primarily enacted through the resource controller `poold`, a system daemon that should always be active when dynamic resource allocation is required. Periodically, `poold` examines the load on the system and determines whether intervention is required to enable the system to maintain optimal performance with respect to resource consumption. The `poold` configuration is held in the `libpool` configuration. For more information on `poold`, see the `poold`(1M) man page.

When pools are enabled, a *default pool* and a *default processor set* form the base configuration. Additional user-defined pools and processor sets can be created and added to the configuration. User-defined pools and processor sets can be destroyed. The default pool and the default processor set cannot be destroyed.

The default pool has the `pool.default` property set to `true`. The default pset has the `pset.default` property set to `true`. Thus, both the default pool and the default pset can be identified even if their names have been changed. For information about these default elements, see "Resource Pools" in the *Solaris Resource Manager Developer's Guide*.

The user-defined pools mechanism is primarily for use on large machines of more than four CPUs. However, small machines can still benefit from this functionality. On small machines, you can create pools that share noncritical resource partitions. The pools are separated only on the basis of critical resources.

Resource Pools Used in Zones

On a system that has zones enabled, a non-global zone can be associated with one resource pool. Moreover, you cannot bind individual processes in non-global zones to a different pool by using `poolbind` from the global zone. To associate a non-global zone with a pool, see "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197.

The scope of an executing instance of `poold` is limited to the global zone.

The `poolstat` utility run in a non-global zone displays only information about the pool associated with the zone. The `pooladm` command run without arguments in a non-global zone displays only information about the pool associated with the zone.

For information on resource pool commands, see "Commands Used With the Resource Pools Facility" on page 130.

When to Use Pools

Resource pools offer a versatile mechanism that can be applied to many administrative scenarios.

Batch compute server	Use pools functionality to split a server into two pools. One pool is used for login sessions and interactive work by timesharing users. The other pool is used for jobs that are submitted through the batch system.
Application or database server	Partition the resources for interactive applications in accordance with the applications' requirements.
Turning on applications in phases	Set user expectations. You might initially deploy a machine that is running only a fraction of the services that the machine is ultimately expected to deliver. User

	difficulties can occur if reservation-based resource management mechanisms are not established when the machine comes online. For example, the fair share scheduler optimizes CPU utilization. The response times for a machine that is running only one application can be misleadingly fast. Users will not see these response times with multiple applications loaded. By using separate pools for each application, you can place a ceiling on the number of CPUs available to each application before you deploy all applications.
Complex timesharing server	Partition a server that supports large user populations. Server partitioning provides an isolation mechanism that leads to a more predictable per-user response. By dividing users into groups that bind to separate pools, and using the fair share scheduling (FSS) facility, you can tune CPU allocations to favor sets of users that have priority. This assignment can be based on user role, accounting chargeback, and so forth.
Workloads that change seasonally	Use resource pools to adjust to changing demand. Your site might experience predictable shifts in workload demand over long periods of time, such as monthly, quarterly, or annual cycles. If your site experiences these shifts, you can alternate between multiple pools configurations by invoking `pooladm` from a `cron`(1M) job.
Real-time applications	Create a real-time pool by using the RT scheduler and designated processor resources.
Sytem utilization	Enforce system goals that you establish. Use the automated pools daemon feature to identify available resources and then monitor workloads to detect when your specified objectives are no longer being satisfied. The daemon can take corrective action if possible, or the condition can be logged.

Resource Pools Framework

The `/etc/pooladm.conf` configuration file describes the static pools configuration. A static configuration represents the way in which an administrator would like a system to be configured with respect to resource pools functionality. An alternate file name can be specified.

The kernel holds information about the disposition of resources within the resource pools framework. This is known as the dynamic configuration, and it represents the resource pools functionality for a particular system at a point in time. The dynamic configuration can be viewed by using the `pooladm` command. Note that the order in which properties are displayed for pools and resource sets can vary. Modifications to the dynamic configuration are made indirectly, by applying a static configuration file, or directly, by using the `poolcfg` command with the `-d` option.

By default, the resource pools framework is not active. Resource pools must be enabled to create or modify the dynamic configuration. More than one static pools configuration file can exist, for activation at different times. Static configuration files can be manipulated with the `poolcfg` or `libpool` commands even if the resource pools framework is disabled. Static configuration files cannot be created if the pools facility is not active. For more information, see "Creating Pools Configurations" on page 115.

For additional information, see `pooladm`(1M), `poolcfg`(1M), `poold`(1M), and `libpool`(3LIB).

`/etc/pooladm.conf` Contents

The static configuration held at `/etc/pooladm.conf` is a special static configuration. When a system boots, if this file exists, then the resource pools framework is enabled and this static configuration is applied to the system.

All resource pool configurations, including the dynamic configuration, can contain the following elements.

system	Properties affecting the total behavior of the system
pool	A resource pool definition
pset	A processor set definition
cpu	A processor definition

All of these elements have properties that can be manipulated to alter the state and behavior of the resource pools framework. For example, the pool property `pool.importance` indicates the relative importance of a given pool. This property is used for possible resource dispute resolution. For more information, see `libpool`(3LIB).

Pools Properties

The pools facility supports named, typed properties that can be placed on a pool, resource, or component. Administrators can store additional properties on the various pool elements. A property namespace similar to the project attribute is used.

For example, the following comment indicates that a given pset is associated with a particular `Datatree` database.

```
Datatree,pset.dbname=warehouse
```

For additional information about property types, see "`poold` Properties" on page 121 and "Resource Pools" in the *Solaris Resource Manager Developer's Guide*.

Note – A number of special properties are reserved for internal use and cannot be set or removed. See `libpool`(3LIB) for more information.

Implementing Pools on a System

User-defined pools can be implemented on a system by using one of these methods.

1. When the Solaris software boots, an `init` script checks to see if the `/etc/pooladm.conf` file exists. If this file is found, then `pooladm` is invoked to make this configuration the active pools configuration. The system creates a dynamic configuration to reflect the organization that is requested in `/etc/pooladm.conf`, and the machine's resources are partitioned accordingly.
2. When the Solaris system is running, a pools configuration can either be activated if it is not already present, or modified by using the `pooladm` command. By default, `pooladm` operates on `/etc/pooladm.conf`. However, you can optionally specify an alternate location and file name, and use this file to update the pools configuration.

The `poold` resource controller is started with the pools facility.

For information about enabling and disabling resource pools, see “Enabling and Disabling the Pools Facility” on page 133. The pools facility cannot be disabled when there are user-defined pools or resources in use.

To configure resource pools, you must have superuser privileges or have the Process Management profile in your list of profiles. The Process Management role and the System Administrator role both include the Process Management profile. The System Administrator role includes the Process Management profile.

SPARC: Dynamic Reconfiguration Operations and Resource Pools

Dynamic Reconfiguration (DR) enables you to reconfigure hardware while the system is running. A DR operation can increase, reduce, or have no effect on a given type of resource. Because DR can affect available resource amounts, the pools facility must be included in these operations. When a DR operation is initiated, the pools framework acts to validate the configuration.

If the DR operation can proceed without causing the current pools configuration to become invalid, then the private configuration file is updated. An invalid configuration is one that cannot be supported by the available resources.

If the DR operation would cause the pools configuration to be invalid, then the operation fails and you are notified by a message to the message log. If you want to force the configuration to completion, you must use the DR force option. The pools configuration is then modified to comply with the new resource configuration. For information on the DR process and the force option, see the dynamic reconfiguration user guide for your Sun hardware.

It is possible for a partition to move out of `poold` control while the daemon is active. For more information, see “Identifying a Resource Shortage” on page 127.

Creating Pools Configurations

The configuration file contains a description of the pools to be created on the system. The file describes the elements that can be manipulated.

- system

- pool
- pset
- cpu

See `poolcfg`(1M) for more information on elements that be manipulated.

When pools are enabled, you can create a structured `/etc/pooladm.conf` file in two ways.

- You can use the `pooladm` command with the `-s` option to discover the resources on the current system and place the results in a configuration file.

  This method is preferred. All active resources and components on the system that are capable of being manipulated by the pools facility are recorded. The resources include existing processor set configurations. You can then modify the configuration to rename the processor sets or to create additional pools if necessary.
- You can use the `poolcfg` command with the `-c` option and the `discover` or `create system` *name* subcommands to create a new pools configuration.

  These options are maintained for backward compatibility with the previous release.

Use `poolcfg` or `libpool` to modify the `/etc/pooladm.conf` file. Do not directly edit this file.

Directly Manipulating the Dynamic Configuration

It is possible to directly manipulate CPU resource types in the dynamic configuration by using the `poolcfg` command with the `-d` option. There are two methods used to transfer resources.

- You can make a general request to transfer any available identified resources between sets.
- You can transfer resources with specific IDs to a target set. Note that the system IDs associated with resources can change when the resource configuration is altered or after a system reboot.

For an example, see "Transferring Resources" on page 143.

Note that the resource transfer might trigger action from `poold`. See "`poold` Overview" on page 117 for more information.

poold Overview

The pools resource controller, poold, uses system targets and observable statistics to preserve the system performance goals that you specify. This system daemon should always be active when dynamic resource allocation is required.

The poold resource controller identifies available resources and then monitors workloads to determine when the system usage objectives are no longer being met. poold then considers alternative configurations in terms of the objectives, and remedial action is taken. If possible, the resources are reconfigured so that objectives can be met. If this action is not possible, then the daemon logs that user-specified objectives can no longer be achieved. Following a reconfiguration, the daemon resumes monitoring workload objectives.

poold maintains a decision history that it can examine. The decision history is used to eliminate reconfigurations that historically did not show improvements.

Note that a reconfiguration can also be triggered asynchronously if the workload objectives are changed or if the resources available to the system are modified.

Stopping poold

If, for any reason, dynamic resource allocation is not required, then poold can be stopped with the SIGQUIT or the SIGTERM signal. Either of these signals causes poold to terminate gracefully.

Reconfiguring poold

poold will automatically detect changes in the resource or pools configuration. However, you can also force a reconfiguration to occur by using the SIGHUP signal.

Configuration Constraints and Objectives

When making changes to a configuration, `poold` acts on directions that you provide. You specify these directions as a series of constraints and objectives. `poold` uses your specifications to determine the relative value of different configuration possibilites in relation to the existing configuration. `poold` then changes the resource assignments of the current configuration to generate new candidate configurations.

Configuration Constraints

Constraints affect the range of possible configurations by eliminating some of the potential changes that could be made to a configuration. The following constraints, which are specified in the `libpool` configuration, are available.

- The minimum and maximum CPU allocations
- Pinned components that are not available to be moved from a set

See the `libpool`(3LIB) man page and "Pools Properties" on page 114 for more information on pools properties.

The `pset.min` Property and `pset.max` Property Constraints

These two properties place limits on the number of processors that can be allocated to a processor set, both minimum and maximum. See Table 12–1 for more details about these properties.

Within these constraints, a resource partition's resources are available to be allocated to other resource partitions in the same Solaris instance. Access to the resource is obtained by binding to a pool that is associated with the resource set. Binding is performed at login or manually by an administrator who has the PRIV_SYS_RES_CONFIG privilege.

The `cpu.pinned` Property Constraint

The `cpu-pinned` property indicates that a particular CPU should not be moved by DRP from the processor set in which it is located. You can set this `libpool` property to maximize cache utilization for a particular application that is executing within a processor set.

See Table 12–1 for more details about this property.

Configuration Objectives

Objectives are specified similarly to constraints. The full set of objectives is documented in Table 12–1.

There are two categories of objectives.

Workload dependent — A workload-dependent objective is an objective that will vary according to the nature of the workload running on the system. An example is the `utilization` objective. The utilization figure for a resource set will vary according to the nature of the workload that is active in the set.

Workload independent — A workload-independent objective is an objective that does not vary according to the nature of the workload running on the system. An example is the CPU `locality` objective. The evaluated measure of locality for a resource set does not vary with the nature of the workload that is active in the set.

You can define three types of objectives.

Name	Valid Elements	Operators	Values
`wt-load`	`system`	N/A	N/A
`locality`	`pset`	N/A	`loose` | `tight` | `none`
`utilization`	`pset`	`< > ~`	`0–100%`

Objectives are stored in property strings in the `libpool` configuration. The property names are as follows:

- `system.poold.objectives`
- `pset.poold.objectives`

Objectives have the following syntax:

- `objectives = objective [; objective]*`
- `objective = [n:] keyword [op] [value]`

All objectives take an optional importance prefix. The importance acts as a multiplier for the objective and thus increases the significance of its contribution to the objective function evaluation. The range is from 0 to INT64_MAX (`9223372036854775807`). If not specified, the default importance value is `1`.

Some element types support more than one type of objective. An example is `pset`. You can specify multiple objective types for these elements. You can also specify multiple utilization objectives on a single `pset` element.

See "How to Define Configuration Objectives" on page 140 for usage examples.

The `wt-load` Objective

The `wt-load` objective favors configurations that match resource allocations to resource utilizations. A resource set that uses more resources will be given more resources when this objective is active.

Use this objective when you are satisfied with the constraints you have established using the minimum and maximum properties, and you would like the daemon to manipulate resources freely within those constraints.

The `locality` Objective

The `locality` objective influences the impact that locality, as measured by locality group (`lgroup`) data, has upon the selected configuration. An alternate definition for locality is latency. An lgroup describes CPU and memory resources. The lgroup is used by the Solaris system to determine the distance between resources, using time as the measurement. For more information on the locality group abstraction, see "Locality Groups Overview" in *Programming Interfaces Guide*.

This objective can take one of the following three values:

`tight` If set, configurations that maximize resource locality are favored.

`loose` If set, configurations that minimize resource locality are favored.

`none` If set, the favorability of a configuration is not influenced by resource locality. This is the default value for the `locality` objective.

In general, the `locality` objective should be set to `tight`. However, to maximize memory bandwidth or to minimize the impact of DR operations upon a resource set, you could set this objective to `loose` or keep it at the default setting of `none`.

The `utilization` Objective

The `utilization` objective favors configurations that allocate resources to partitions that are not meeting the specified utilization objective.

This objective is specified by using operators and values. The operators are:

< The "less than" operator indicates that the specified value represents a maximum target value.

> The "greater than" operator indicates that the specified value represents a minimum target value.

~ The "about" operator indicates that the specified value is a target value about which some fluctuation is acceptable.

A pset can only have one utilization objective set for each type of operator.

- If the ~ operator is set, then the < and > operators cannot be set.
- If the < and > operators are set, then the ~ operator cannot be set. Note that the settings of the < operator and the > operator cannot contradict each other.

You can set both a < and a > operator together to create a range. The values will be validated to make sure that they do not overlap.

Configuration Objectives Example

In the following example, `poold` is to assess these objectives for the pset:

- The `utilization` should be kept between 30 percent and 80 pecent.
- The `locality` should be maximized for the processor set.
- The objectives should take the default importance of `1`.

EXAMPLE 12–1 `poold` Objectives Example

```
pset.poold.objectives "utilization > 30; utilization < 80;
locality tight"
```

See "How to Define Configuration Objectives" on page 140 for additional usage examples.

`poold` Properties

There are four categories of properties:

- Configuration
- Constraint

- Objective
- Objective Parameter

TABLE 12–1 Defined Property Names

Property Name	Type	Category	Description
`system.poold.log-level`	string	Configuration	Logging level
`system.poold.log-location`	string	Configuration	Logging location
`system.poold.monitor-interval`	unsigned int	Configuration	Monitoring sample interval
`system.poold.history-file`	string	Configuration	Decision history location
`system.poold.pid`	int	Configuration	`poold` PID
`pset.max`	unsigned int	Constraint	Maximum number of CPUs for this processor set
`pset.min`	unsigned int	Constraint	Minimum number of CPUs for this processor set
`cpu.pinned`	boolean	Constraint	CPUs pinned to this processor set
`system.poold.objectives`	string	Objective	Formatted string following `poold`'s objective expression syntax
`pset.poold.objectives`	string	Objective	Formatted string following `poold`'s expression syntax
`pool.importance`	unsigned int	Objective parameter	User-assigned importance

`poold` Features That Can Be Configured

You can configure these aspects of the daemon's behavior.

- Monitoring interval
- Logging level
- Logging location

These options are specified in the pools configuration. You can also control the logging level from the command line by invoking `poold`.

`poold` Monitoring Interval

Use the property name `system.poold.monitor-interval` to specify a value in milliseconds.

`poold` Logging Information

Three categories of information are provided through logging. These categories are identified in the logs:

- Configuration
- Monitoring
- Optimization

Use the property name `system.poold.log-level` to specify the logging parameter. If this property is not specified, the default logging level is `NOTICE`. The parameter levels are hierarchical. Setting a log level of `DEBUG` will cause `poold` to log all defined messages. Note that `INFO` provides a useful balance of information for most administrators.

At the command line, you can use the `poold` command with the `-l` option and a parameter to specify the level of logging information generated.

The following parameters are available:

- `ALERT`
- `CRIT`
- `ERR`
- `WARNING`
- `NOTICE`
- `INFO`
- `DEBUG`

The parameter levels map directly onto their `syslog` equivalents. See "Logging Location" on page 125 for more information on using `syslog`.

For more information about how to configure `poold` logging, see "How to Set the `poold` Logging Level" on page 142.

Configuration Information Logging

The following types of messages can be generated:

`ALERT` Problems accessing the `libpool` configuration, or some other fundamental, unanticipated failure of the `libpool` facility. Causes the daemon to exit and require immediate administrative attention.

`CRIT` Problems due to unanticipated failures. Causes the daemon to exit and require immediate administrative attention.

`ERR` Problems with the user-specified parameters that control operation, such as unresolvable, conflicting utilization objectives for a resource set. Requires administrative intervention to correct the objectives. `poold` attempts to take remedial action by ignoring conflicting objectives, but some errors will cause the daemon to exit.

`WARNING` Warnings related to the setting of configuration parameters that, while technically correct, might not be suitable for the given execution environment. An example is marking all CPU resource as pinned, which means that `poold` cannot move CPU resource between processor sets.

`DEBUG` Messages containing the detailed information that is needed when debugging configuration processing. This information is not generally used by administrators.

Monitoring Information Logging

The following types of messages can be generated:

`CRIT` Problems due to unanticipated monitoring failures. Causes the daemon to exit and require immediate administrative attention.

`ERR` Problems due to unanticipated monitoring error. Could require administrative intervention to correct.

`NOTICE` Messages about resource control region transitions.

`INFO` Messages about resource utiliztion statistics.

`DEBUG` Messages containing the detailed information that is needed when debugging monitoring processing. This information is not generally used by administrators.

Optimization Information Logging

The following types of messages can be generated:

`WARNING` Messages could be displayed regarding problems making optimal decisions. Examples could include resource sets that are too narrowly constrained by their minimum and maximum values or by the number of pinned components.

	Messages could be displayed about problems performing an optimal reallocation due to unforseen limitations. Examples could include removing the last processor from a processor set which contains a bound resource consumer.
`NOTICE`	Messages about usable configurations or configurations that will not be implemented due to overriding decision histories could be displayed.
`INFO`	Messages about alternate configurations considered could be displayed.
`DEBUG`	Messages containing the detailed information that is needed when debugging optimization processing. This information is not generally used by administrators.

Logging Location

The `system.poold.log-location` property is used to specify the location for `poold` logged output. You can specify a location of `SYSLOG` for `poold` output (see `syslog`(3C)).

If this property is not specified, the default location for `poold` logged output is `/var/log/pool/poold`.

When `poold` is invoked from the command line, this property is not used. Log entries are written to `stderr` on the invoking terminal.

Log Management With `logadm`

If `poold` is active, the `logadm.conf` file includes an entry to manage the default file `/var/log/pool/poold`. The entry is:

```
/var/log/pool/poold -N -s 512k
```

See the `logadm`(1M) and the `logadm.conf`(4) man pages.

How Dynamic Resource Allocation Works

This section explains the process and the factors that `poold` uses to dynamically allocate resources.

About Available Resources

Available resources are considered to be all of the resources that are available for use within the scope of the `poold` process. The scope of control is at most a single Solaris instance.

On a system that has zones enabled, the scope of an executing instance of `poold` is limited to the global zone.

Determining Available Resources

Resource pools encompass all of the system resources that are available for consumption by applications.

For a single executing Solaris instance, a resource of a single type, such as a CPU, must be allocated to a single partition. There can be one or more partitions for each type of resource. Each partition contains a unique set of resources.

For example, a machine with four CPUs and two processor sets can have the following setup:

pset 0: 0 1

pset 1: 2 3

where 0, 1, 2 and 3 after the colon represent CPU IDs. Note that the two processor sets account for all four CPUs.

The same machine cannot have the following setup:

set 0: 0 1

set 1: 1 2 3

because CPU 1 can appear in only one set at a time.

Resources cannot be accessed from any partition other than the partition to which they belong.

To discover the available resources, `poold` interrogates the active pools configuration to find partitions. All resources within all partitions are summed to determine the total amount of available resources for each type of resource that is controlled.

This quantity of resources is the basic figure that `poold` uses in its operations. However, there are constraints upon this figure that limit the flexibility that `poold` has to make allocations. For information on available constraints, see "Configuration Constraints" on page 118.

Identifying a Resource Shortage

The control scope for `poold` is defined as the set of available resources for which `poold` has primary responsibility for effective partitioning and management. However, other mechanisms that are allowed to manipulate resources within this control scope can still affect a configuration. If a partition should move out of control while `poold` is active, `poold` tries to restore control through the judicious manipulation of available resources. If `poold` cannot locate additional resources within its scope, then the daemon logs information about the resource shortage.

Determining Resource Utilization

`poold` typically spends the greatest amount of time observing the usage of the resources within its scope of control. This monitoring is performed to verify that workload-dependent objectives are being met.

For example, for processor sets, all measurements are made across all of the processors in a set. The resource utilization is the idle time in the range 0 percent to 100 percent.

Identifying Control Violations

The directives described in "Configuration Constraints and Objectives" on page 118 are used to detect the approaching failure of a system to meet its objectives. These objectives are directly related to workload.

A partition that is not meeting user-configured objectives is a control violation. The two types of control violations are synchronous and asynchronous.

- A synchronous violation is a violation of an objective that is detected by the daemon in the course of its workload monitoring.
- An asynchronous objective violation occurs independently of monitoring action by the daemon.

The following events cause asynchronous objective violations:

- Resources are added to or removed from a control scope.
- The control scope is reconfigured.
- The `poold` resource controller is restarted.

The contributions of objectives that are not related to workload are assumed to remain constant between evaluations of the objective function. Objectives that are not related to workload are only reassessed when a reevaluation is triggered through one of the asynchronous violations.

Determining Appropriate Remedial Action

When the resource controller determines that a resource consumer is short of resources, the initial response is that increasing the resources will improve performance.

Alternative configurations are examined and evaluated in terms of the objectives specified in the configuration for the scope of control.

This process is refined over time as the results of resource movements are monitored and each resource partition is evaluated for responsiveness. The decision history is consulted to eliminate reconfigurations that did not show improvements in attaining the objective function in the past. Other information, such as process names and quantities, are used to further evaluate the relevance of the historical data.

If the daemon cannot take corrective action, then the condition is logged. For more information, see "`poold` Logging Information" on page 123.

Using `poolstat` to Monitor the Pools Facility and Resource Utilization

The `poolstat` utility is used to monitor resource utilization when pools are enabled on your system. This utility iteratively examines all of the active pools on a system and reports statistics based on the selected output mode. The `poolstat` statistics enable you to determine which resource partitions are heavily utilized. You can analyze these statistics to make decisions about resource reallocation when the system is under resource pressure.

The `poolstat` utility includes options that can be used to examine only specified pools and report resource set-specific statistics.

If zones are implemented on your system and you use `poolstat` in a non-global zone, information about the resources associated with the zone's pool display.

For more information about the `poolstat` utility, see the `poolstat`(1M) man page. For `poolstat` task and usage information, see "Using `poolstat` to Report Statistics for Pool-Related Resources" on page 147.

`poolstat` Output

In default output format, `poolstat` outputs a heading line and then displays a line for each pool. A pool line begins with the pool ID and the name of the pool, followed by a column of statistical data for the processor set attached to the pool. Resource sets attached to more than one pool are listed multiple times, once for each pool.

The column headings are:

`id` Pool ID.

`pool` Pool name.

`rid` Resource set ID.

`rset` Resource set name.

`type` Resource set type.

`min` The minimum resource set size.

`max` The maximum resource set size.

`size` The current resource set size.

`used` The measure of how much of the resource set is currently used. This usage is calculated as the percentage of utilization of the resource set multiplied by

the size of the resource set. If a resource set has been reconfigured during the last sampling interval, this value might be not reported. An unreported value appears as a hyphen (`-`).

`load` The absolute representation of the load that is put on the resource set. For more information on this property, see `libpool`(3LIB).

You can determine the column order and specify which headings appear in the `poolstat` output.

Tuning `poolstat` Operation Intervals

You can customize the operations performed by `poolstat`. You can set the sampling interval for the report and specify the number of times that statistics are repeated:

interval Tune the intervals for the periodic operations performed by `poolstat`. All intervals are specified in seconds.

count Specify the number of times that the statistics are repeated. By default, `poolstat` reports statistics only once.

If neither interval nor count are specified, then statistics are reported once. If interval is specified and count is not specified, then statistics are reported indefinitely.

Commands Used With the Resource Pools Facility

The commands described in the following table provide the primary administrative interface to the pools facility. For information on using these commands on a system that has zones enabled, see "Resource Pools Used in Zones" on page 111.

Command	Description
`pooladm`(1M)	Enables or disables the pools facility on your system. Activates a particular configuration or removes the current configuration and returns associated resources to their default status. If run without options, `pooladm` prints out the current dynamic pools configuration.
`poolbind`(1M)	Enables the manual binding of projects, tasks, and processes to a resource pool.

Command	Description
`poolcfg`(1M)	Provides configuration operations on pools and sets. Configurations created using this tool are instantiated on a target host by using `pooladm`. If run with the `info` subcommand argument to the `-c` option, `poolcfg` displays information about the static configuration at `/etc/pooladm.conf`. If a file name argument is added, this command displays information about the static configuration held in the named file. For example, `poolcfg -c info /tmp/newconfig` displays information about the static configuration contained in the file `/tmp/newconfig`.
`poold`(1M)	The pools system daemon. The daemon uses system targets and observable statistics to preserve the system performance goals specified by the administrator. If unable to take corrective action when goals are not being met, `poold` logs the condition.
`poolstat`(1M)	Displays statistics for pool-related resources. Simplifies performance analysis and provides information that supports system administrators in resource partitioning and re-partitioning tasks. Options are provided for examining specified pools and reporting resource set-specific statistics.

A library API is provided by `libpool`(3LIB). The library can be used by programs to manipulate pool configurations.

CHAPTER 13

Administering Dynamic Resource Pools (Tasks)

This chapter describes how to set up and administer resource pools on your system.

The following procedures are covered.

For background information about resource pools, see Chapter 12.

Enabling and Disabling the Pools Facility

Use `pooladm`(1M) to enable the pools facility so that pools can be manipulated or to disable the pools facility so that pools cannot be manipulated.

▼ How to Enable Pools

To enable the pools facility on your system, invoke `pooladm` with the `-e` option, "enable pools."

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing

RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Type the following:**

```
# pooladm -e
```

▼ How to Disable Pools

To disable the pools facility on your system, invoke `pooladm` with the `-d` option, "disable pools."

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Type the following:**

```
# pooladm -d
```

Configuring Pools

▼ How to Create a Static Configuration

Enable pools on your system, then use the `-s` option to `/usr/sbin/pooladm` to create a static configuration file that matches the current dynamic configuration. If a file name is not specified, the default location `/etc/pooladm.conf` is used.

Commit your configuration using the `pooladm` command with the `-c` option. Then use the `pooladm` command with the `-s` option to update the static configuration to match the state of the dynamic configuration.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Update the static configuration file to match the current dynamic configuration.**

```
# pooladm -s
```

3. **View the contents of the configuration file in readable form. Note that the configuration contains default elements created by the system.**

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

4. **To commit the configuration at `/etc/pooladm.conf`, type the following:**

```
# pooladm -c
```

5. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

Note – The new functionality `pooladm -s` is preferred over the previous functionality `poolcfg -c discover` for creating a new configuration that matches the dynamic configuration.

▼ How to Modify a Configuration

To enhance your simple configuration, create a processor set named *pset_batch* and a pool named *pool_batch*. Then join the pool and the processor set with an association.

Note that you must quote subcommand arguments that contain white space.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC" in *System Administration Guide: Security Services*.

2. **Create processor set `pset_batch`.**

   ```
   # poolcfg -c 'create pset pset_batch (uint pset.min = 2; uint pset.max = 10)'
   ```

3. **Create pool `pool_batch`.**

   ```
   # poolcfg -c 'create pool pool_batch'
   ```

4. **Join with an association.**

   ```
   # poolcfg -c 'associate pool pool_batch (pset pset_batch)'
   ```

5. **Display the edited configuration.**

   ```
   # poolcfg -c info
   system tester
           string  system.comment kernel state
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
                   int     pset.sys_id -1
   ```

```
        boolean pset.default true
        uint    pset.min 1
        uint    pset.max 65536
        string  pset.units population
        uint    pset.load 10
        uint    pset.size 4
        string  pset.comment
        boolean testnullchanged true

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line

pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        pset pset_batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
```

```
                        string  cpu.status on-line
```

6. **To commit the configuration at `/etc/pooladm.conf`, type the following:**

   ```
   # pooladm -c
   ```

7. **(Optional) To copy the dynamic configuration to a static configuration file named `/tmp/backup`, type the following:**

   ```
   # pooladm -s /tmp/backup
   ```

▼ How to Associate a Pool With a Scheduling Class

You can associate a pool with a scheduling class so that all processes bound to the pool use this scheduler. To do this, set the `pool.scheduler` property to the name of the scheduler class. This example associates the pool `pool_batch` with the fair share scheduler (FSS).

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Modify pool `pool_batch` to be associated with the FSS.**

   ```
   # poolcfg -c 'modify pool pool_batch (string pool.scheduler="FSS")'
   ```

3. **Display the edited configuration.**

   ```
   # poolcfg -c info
   system tester
           string  system.comment
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
                   int     pset.sys_id -1
                   boolean pset.default true
                   uint    pset.min 1
                   uint    pset.max 65536
                   string  pset.units population
   ```

```
        uint    pset.load 10
        uint    pset.size 4
        string  pset.comment
        boolean testnullchanged true

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line

pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        string pool.scheduler FSS
        pset batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line
```

4. **To commit the configuration at `/etc/pooladm.conf`, type the following:**

```
# pooladm -c
```

5. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

▼ How to Define Configuration Objectives

You can specify objectives for `poold` to consider when taking corrective action.

In the following procedure, the `wt-load` objective is being set so that `poold` tries to match resource allocation to resource utilization. The `locality` objective is disabled to assist in achieving this configuration goal.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC" in *System Administration Guide: Security Services*.

2. **Modify system `tester` to favor the `wt-load` objective.**

```
# poolcfg -c 'modify system host (string system.poold.objectives="wt-load")'
```

3. **Disable the `locality` objective for the default pset.**

```
# poolcfg -c 'modify pset pset_default (string pset.poold.objectives="locality none")'
```

4. **Disable the `locality` objective for the `pset_batch` pset.**

```
# poolcfg -c 'modify pset pset_batch (string pset.poold.objectives="locality none")'
```

5. **Display the edited configuration.**

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
```

```
        int     pset.sys_id -1
        boolean pset.default true
        uint    pset.min 1
        uint    pset.max 65536
        string  pset.units population
        uint    pset.load 10
        uint    pset.size 4
        string  pset.comment
        boolean testnullchanged true
        string  pset.poold.objectives locality none

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line

pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        string pool.scheduler FSS
        pset batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0
        string  pset.poold.objectives locality none

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line
```

```
cpu
        int     cpu.sys_id 4
        string  cpu.comment
        string  cpu.status on-line
```

6. **To commit the configuration at `/etc/pooladm.conf`, type the following:**

   ```
   # pooladm -c
   ```

7. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

   ```
   # pooladm -s /tmp/backup
   ```

▼ How to Set the `poold` Logging Level

To specify the level of logging information that `poold` generates, set the `system.poold.log-level` property in the `poold` configuration. The `poold` configuration is held in the `libpool` configuration. For information, see "`poold` Logging Information" on page 123, `poolcfg`(1M), and `libpool`(3LIB).

You can also use the `poold` command at the command line to specify the level of logging information that `poold` generates.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Set the level by using the `poold` command with the `-l` option and a parameter, for example, `INFO`.**

   ```
   # /usr/lib/pool/poold -l INFO
   ```

For information on available parameters, see "`poold` Logging Information" on page 123. The default logging level is `NOTICE`.

▼ How to Use Command Files With `poolcfg`

The `poolcfg` command with the `-f` option can take input from a text file that contains `poolcfg` subcommand arguments to the `-c` option. This technique is appropriate when you want a set of operations to be performed atomically. When processing multiple commands, the configuration is only updated if all of the commands succeed. For large or complex configurations, this technique can be more useful than per-subcommand invocations.

In command files, the # character acts as a comment for the rest of line.

1. **Create the input file `poolcmds.txt`.**

```
$ cat > poolcmds.txt
create system tester
create pset pset_batch (uint pset.min = 2; uint pset.max = 10)
create pool pool_batch
associate pool pool_batch (pset pset_batch)
```

2. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC" in *System Administration Guide: Security Services*.

3. **Type the following:**

```
# /usr/sbin/poolcfg -f poolcmds.txt
```

Transferring Resources

Use the `transfer` subcommand argument to the `-c` option of `poolcfg` with the `-d` option to transfer resources in the kernel. The `-d` option specifies that the command operate directly on the kernel and not take input from a file. The following example moves two CPUs from processor set *pset1* to processor set *pset2* in the kernel.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC" in *System Administration Guide: Security Services*.

2. **Type the following:**

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

The `from` and `to` sub-clauses can be used in any order. Only one `to` and `from` sub-clause is supported per command.

If specific known IDs of a resource type are to be transferred, then an alternative syntax is provided. To assign two CPUs with IDs *0* and *2* to the `pset_large` pset, type the following:

```
# poolcfg -dc "transfer to pset pset_large (cpu 0; cpu 2)"
```

If a transfer fails because there are not enough resources to match the request or because the specified IDs cannot be located, then the system displays an error message.

Activating and Removing Pools Configurations

Use the `pooladm` command to make a particular pool configuration active or to remove the currently active pool configuration. See `pooladm`(1M) for more information about this command.

▼ How to Activate a Pools Configuration

To activate the configuration in the default configuration file, `/etc/pooladm.conf`, invoke `pooladm` with the `-c` option, "commit configuration." Also use the `-s` to update the static configuration to match the state of the dynamic configuration.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC" in *System Administration Guide: Security Services*.

2. **To commit the configuration at `/etc/pooladm.conf`, type the following:**

   ```
   # pooladm -c
   ```

3. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

   ```
   # pooladm -s /tmp/backup
   ```

▼ How to Validate a Configuration Before Committing the Configuration

You can use the `-n` option with the `-c` option to test what will happen when the validation occurs. The configuration will not actually be committed.

The following command attempts to validate the configuration contained at `/home/admin/newconfig`. Any error conditions encountered are displayed, but the configuration itself is not modified.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Type the following:**

```
# pooladm -n -c /home/admin/newconfig
```

▼ How to Remove a Pools Configuration

To remove the current active configuration and return all associated resources, such as processor sets, to their default status, use the `-x` option for "remove configuration."

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Type the following:**

```
# pooladm -x
```

The `-x` option to `pooladm` removes all user-defined elements from the dynamic configuration. All resources revert to their default states and all pool bindings are replaced with a binding to the default pool.

Note – Mixing scheduling classes within one processor set can lead to unpredictable results. If the use of `pooladm -x` results in mixed scheduling classes within one processor set, you should then use the `priocntl` command to move running processes into a different scheduling class. An example is provided in "How to Manually Move Processes From the TS Class Into the FSS Class" on page 90. You can also see `priocntl`(1) for more information.

Binding to a Pool

You can bind a running process to a pool in two ways.

- You can use the `poolbind`(1M) command to bind a specific process to a named resource pool.

- You can use the `project.pool` attribute in the `project`(4) database to identify the pool binding for a new login session or a task that is launched through `newtask`(1).

▼ How to Bind Processes to a Pool

The following procedure manually binds a process (in this case, the current shell) to a pool named *ohare*.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Use the `poolbind` command with the `-p` option to manually bind a process to a pool:**

   ```
   # poolbind -p ohare $$
   ```

3. **Use `poolbind -q` to verify the pool binding for the process. The system displays the process ID and the pool binding.**

   ```
   $ poolbind -q $$
   155509 ohare
   ```

▼ How to Bind Tasks or Projects to a Pool

To bind tasks or projects to a pool, use `poolbind` with the `-i` option. The following example binds all processes in the *airmiles* project to the *laguardia* pool.

1. **Become superuser, or assume a role that includes the Process Management profile.**

   The System Administrator role includes the Process Management profile. For information on how to create the role and assign the role to a user, see "Managing RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Type the following:**

   ```
   # poolbind -i project -p laguardia airmiles
   ```

How to Use `project` Attributes to Bind New Processes to a Pool

To automatically bind new processes in a project to a pool, add the `project.pool` attribute to each entry in the `project` database.

For example, assume you have a configuration with two pools that are named `studio` and `backstage`. The `/etc/project` file has the following contents.

```
user.paul:1024::::project.pool=studio
user.george:1024::::project.pool=studio
user.ringo:1024::::project.pool=backstage
passes:1027::paul::project.pool=backstage
```

With this configuration, processes that are started by user `paul` are bound by default to the `studio` pool.

▼ How to Use `project` Attributes to Bind a Process to a Different Pool

Using the previous configuration, user `paul` can modify the pool binding for processes he starts. He can use `newtask` to bind work to the `backstage` pool as well, by launching in the `passes` project.

1. **Launch a process in the `passes` project.**

   ```
   $ newtask -l -p passes
   ```

2. **Use the `poolbind` command with the `-q` option to verify the pool binding for the process. The system displays the process ID and the pool binding.**

   ```
   $ poolbind -q $$
   6384  pool backstage
   ```

Using `poolstat` to Report Statistics for Pool-Related Resources

The `poolstat` command is used to display statistics for pool-related resources. See "Using `poolstat` to Monitor the Pools Facility and Resource Utilization" on page 129 and the `poolstat`(1M) man page for more information.

The following subsections use examples to illustrate how to produce reports for specific purposes.

Default `poolstat` Output

Typing `poolstat` without arguments outputs a header line and a line of information for each pool. The information line shows the pool ID, the name of the pool, and resource statistics for the processor set attached to the pool.

```
machine% poolstat
                           pset
       id pool           size used load
        0 pool_default      4  3.6  6.2
        1 pool_sales        4  3.3  8.4
```

Producing Multiple Reports at Specific Intervals

The following command produces three reports at 5-second sampling intervals.

```
machine% poolstat 5 3
                            pset
 id pool                 size used load
 46 pool_sales              2  1.2  8.3
  0 pool_default            2  0.4  5.2
                            pset
 id pool                 size used load
 46 pool_sales              2  1.4  8.4
  0 pool_default            2  1.9  2.0
                            pset
 id pool                 size used load
 46 pool_sales              2  1.1  8.0
  0 pool_default            2  0.3  5.0
```

Reporting Resource Set Statistics

The following example reports statistics for the pset resource set. Note that the resource set `pset_default` is attached to more than one pool, so this pset is listed once for each pool membership.

```
machine% poolstat -r pset
      id pool          type rid rset          min  max size used load
       0 pool_default  pset  -1 pset_default    1  65K    2  1.2  8.3
       6 pool_sales    pset   1 pset_sales      1  65K    2  1.2  8.3
       2 pool_other    pset  -1 pset_default    1  10K    2  0.4  5.2
```

CHAPTER 14

Resource Management Configuration Example

This chapter reviews the resource management framework and describes a hypothetical server consolidation project.

The following topics are covered.

Configuration to Be Consolidated

In this example, five applications are being consolidated onto a single system. The target applications have resource requirements that vary, different user populations, and different architectures.Currently, each application exists on a dedicated server that is designed to meet the requirements of the application. The applications and their characteristics are identified in the following table.

Application Description	Characteristics
Application server	Exhibits negative scalability beyond 2 CPUs
Database instance for application server	Heavy transaction processing
Application server in test and development environment	GUI-based, with untested code execution
Transaction processing server	Primary concern is response time

Application Description	Characteristics
Standalone database instance	Processes a large number of transactions and serves multiple time zones

Consolidation Configuration

The following configuration is used to consolidate the applications onto a single system.

- The application server has a two–CPU processor set.
- The database instance for the application server and the standalone database instance are consolidated onto a single processor set of at least four CPUs. The standalone database instance is guaranteed 75 percent of that resource.
- The test and development application server requires the IA scheduling class to ensure UI responsiveness. Memory limitations are imposed to lessen the effects of bad code builds.
- The transaction processing server is assigned a dedicated processor set of at least two CPUs, to minimize response latency.

This configuration covers known applications that are executing and consuming processor cycles in each resource set. Thus, constraints can be established that allow the processor resource to be transferred to sets where the resource is required.

- The `wt-load` objective is set to allow resource sets that are highly utilized to receive greater resource allocations than sets that have low utilization.
- The `locality` objective is set to `tight`, which is used to maximize processor locality.

An additional constraint to prevent utilization from exceeding 80 percent of any resource set is also applied. This constraint ensures that applications get access to the resources they require. Moreover, for the transaction processing set, the objective of maintaining utilization below 80 percent is twice as important as any other objectives that are specified. This importance will be defined in the configuration.

Creating the Configuration

Edit the `project` database file `/etc/project`. Add entries to implement the required resource controls and to map users to resource pools, and then view the file.

```
# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server:::project.pool=appserver_pool
user.app_db:2002:App Server DB:::project.pool=db_pool;project.cpu-shares(privileged,1,deny)
development:2003:Test and development::staff:project.pool=dev_pool;process.max-address-space=(privileged,
536870912,deny)     keep with previous line
user.tp_engine:2004:Transaction Engine:::project.pool=tp_pool
user.geo_db:2005:EDI DB:::project.pool=db_pool;project.cpu-shares=(privileged,3,deny)
.
.
.
```

Note – The development team has to execute tasks in the development project because access for this project is based on a user's group ID (GID).

Create an input file named `pool.host`, which will be used to configure the required resource pools. View the file.

```
# cat pool.host
create system host
create pset dev_pset (uint pset.min = 0; uint pset.max = 2)
create pset tp_pset (uint pset.min = 2; uint pset.max=8)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.scheduler="TS")
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
modify system tester (string system.poold.objectives="wtload")
modify pset dev_pset (string pset.poold.objectives="locality tight; utilization < 80")
modify pset tp_pset (string pset.poold.objectives="locality tight; 2: utilization < 80")
modify pset db_pset (string pset.poold.objectives="locality tight;utilization < 80")
modify pset app_pset (string pset.poold.objectives="locality tight; utilization < 80")
```

Update the configuration using the `pool.host` input file:

```
# poolcfg -f pool.host
```

Make the configuration active.

```
# pooladm -c
```

The framework is now functional on the system.

Viewing the Configuration

To view the framework configuration, which also contains default elements created by the system, type:

```
# pooladm
system host
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wtload

        pool dev_pool
                int     pool.sys_id 125
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler IA
                pset    dev_pset

        pool appserver_pool
                int     pool.sys_id 124
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    app_pset

        pool db_pool
                int     pool.sys_id 123
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler FSS
                pset    db_pset

        pool tp_pool
                int     pool.sys_id 122
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    tp_pset

        pool pool_default
                int     pool.sys_id 0
                boolean pool.default true
```

```
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    pset_default

pset dev_pset
        int     pset.sys_id 4
        string  pset.units population
        boolean pset.default false
        uint    pset.min 0
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

pset tp_pset
        int     pset.sys_id 3
        string  pset.units population
        boolean pset.default false
        uint    pset.min 2
        uint    pset.max 8
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; 2: utilization < 80

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

pset db_pset
        int     pset.sys_id 2
        string  pset.units population
        boolean pset.default false
        uint    pset.min 4
        uint    pset.max 6
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
```

```
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 6
                string  cpu.comment
                string  cpu.status on-line
pset app_pset
        int     pset.sys_id 1
        string  pset.units population
        boolean pset.default false
        uint    pset.min 1
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80
        cpu
                int     cpu.sys_id 7
                string  cpu.comment
                string  cpu.status on-line

pset pset_default
        int     pset.sys_id -1
        string  pset.units population
        boolean pset.default true
        uint    pset.min 1
        uint    pset.max 4294967295
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line
```

A graphic representation of the framework follows.

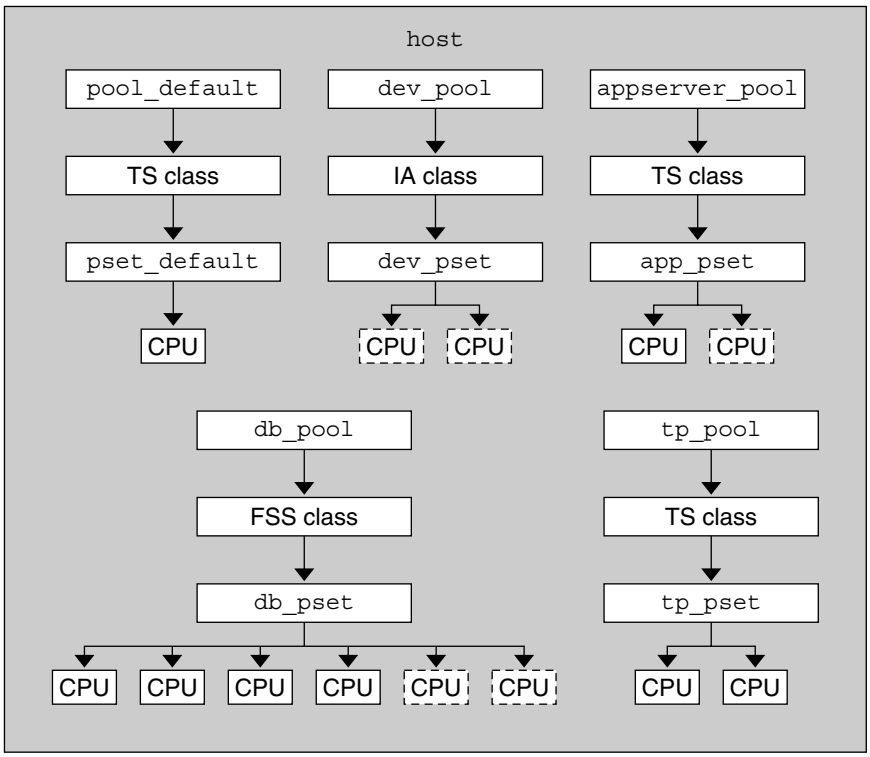


FIGURE 14–1 Server Consolidation Configuration

Note – In the `db_pool`, the standalone database instance is guaranteed 75 percent of the CPU resource.

CHAPTER 15

Resource Control Functionality in the Solaris Management Console

This chapter describes the resource control and performance monitoring features in the Solaris Management Console. Only a subset of the resource management features can be controlled using the console.

You can use the console to monitor system performance and to enter the resource control values shown in Table 15–1 for projects, tasks, and processes. The console provides a convenient, secure alternative to the command-line interface (CLI) for managing hundreds of configuration parameters that are spread across many systems. Each system is managed individually. The console's graphical interface supports all experience levels.

The following topics are covered.

Using the Console (Task Map)

Task	Description	For Instructions
Use the console	Start the Solaris Management Console in a local environment or in a name service or directory service environment. Note that the performance tool is not available in a name service environment.	"Starting the Solaris Management Console" in *System Administration Guide: Basic Administration* and "Using the Solaris Management Tools in a Name Service Environment (Task Map)" in *System Administration Guide: Basic Administration*
Monitor system performance	Access the Performance tool under System Status.	"How to Access the Performance Tool" on page 159
Add resource controls to projects	Access the Resource Controls tab under System Configuration.	"How to Access the Resource Controls Tab" on page 163

Console Overview

Resource management functionality is a component of the Solaris Management Console. The console is a container for GUI-based administrative tools that are stored in collections called toolboxes. For information on the console and how to use it, see "Working With the Management Console (Tasks)" in *System Administration Guide: Basic Administration*.

When you use the console and its tools, the main source of documentation is the online help system in the console itself. For a description of the documentation available in the online help, see "Solaris Management Console (Overview)" in *System Administration Guide: Basic Administration*.

Management Scope

The term *management scope* refers to the name service environment that you choose to use with the selected management tool. The management scope choices for the resource control and performance tools are the `/etc/project` local file, or NIS.

The management scope that you select during a console session should correspond to the primary name service that is identified in the `/etc/nsswitch.conf` file.

Performance Tool

The Performance tool is used to monitor resource utilization. Resource utilization can be summarized for the system, viewed by project, or viewed for an individual user.

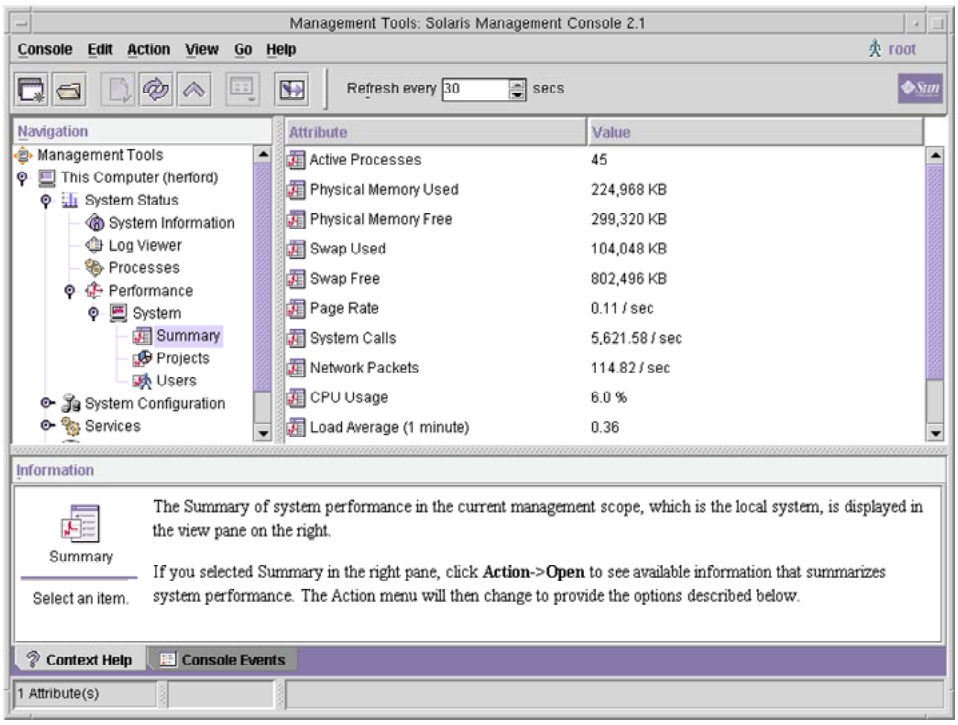


FIGURE 15–1 Performance Tool in the Solaris Management Console

▼ How to Access the Performance Tool

The Performance tool is located under System Status in the Navigation pane. To access the Performance tool, do the following:

1. **Click the System Status control entity in the Navigation pane.**

The control entity is used to expand menu items in the Navigation pane.

2. **Click the Performance control entity.**
3. **Click the System control entity.**
4. **Double-click Summary, Projects, or Users.**
   Your choice depends on the usage you want to monitor.

Monitoring by System

Values are shown for the following attributes.

Attribute	Description
Active Processes	Number of processes that are active on the system
Physical Memory Used	Amount of system memory that is in use
Physical Memory Free	Amount of system memory that is available
Swap Used	Amount of system swap space that is in use
Swap Free	Amount of free system swap space
Page Rate	Rate of system paging activity
System Calls	Number of system calls per second
Network Packets	Number of network packets that are transmitted per second
CPU Usage	Percentage of CPU that is currently in use
Load Average	Number of processes in the system run queue which are averaged over the last 1, 5, and 15 minutes

Monitoring by Project or User Name

Values are shown for the following attributes.

Attribute	Short Name	Description
Input Blocks	`inblk`	Number of blocks read

Attribute	Short Name	Description
Blocks Written	`oublk`	Number of blocks written
Chars Read/Written	`ioch`	Number of characters read and written
Data Page Fault Sleep Time	`dftime`	Amount of time spent processing data page faults
Involuntary Context Switches	`ictx`	Number of involuntary context switches
System Mode Time	`stime`	Amount of time spent in the kernel mode
Major Page Faults	`majfl`	Number of major page faults
Messages Received	`mrcv`	Number of messages received
Messages Sent	`msend`	Number of messages sent
Minor Page Faults	`minf`	Number of minor page faults
Num Processes	`nprocs`	Number of processes owned by the user or the project
Num LWPs	`count`	Number of lightweight processes
Other Sleep Time	`slptime`	Sleep time other than `tftime`, `dftime`, `kftime`, and `ltime`
CPU Time	`pctcpu`	Percentage of recent CPU time used by the process, the user, or the project
Memory Used	`pctmem`	Percentage of system memory used by the process, the user, or the project
Heap Size	`brksize`	Amount of memory allocated for the process data segment
Resident Set Size	`rsssize`	Current amount of memory claimed by the process
Process Image Size	`size`	Size of the process image in Kbytes
Signals Received	`sigs`	Number of signals received
Stopped Time	`stoptime`	Amount of time spent in the stopped state
Swap Operations	`swaps`	Number of swap operations in progress

Attribute	Short Name	Description
System Calls Made	`sysc`	Number of system calls made over the last time interval
System Page Fault Sleep Time	`kftime`	Amount of time spent processing page faults
System Trap Time	`ttime`	Amount of time spent processing system traps
Text Page Fault Sleep Time	`tftime`	Amount of time spent processing text page faults
User Lock Wait Sleep Time	`ltime`	Amount of time spent waiting for user locks
User Mode Time	`utime`	Amount of time spent in the user mode
User and System Mode Time	`time`	The cumulative CPU execution time
Voluntary Context Switches	`vctx`	Number of voluntary context switches
Wait CPU Time	`wtime`	Amount of time spent waiting for CPU (latency)

Resource Controls Tab

Resource controls allow you to associate a project with a set of resource constraints. These constraints determine the allowable resource usage of tasks and processes that run in the context of the project.

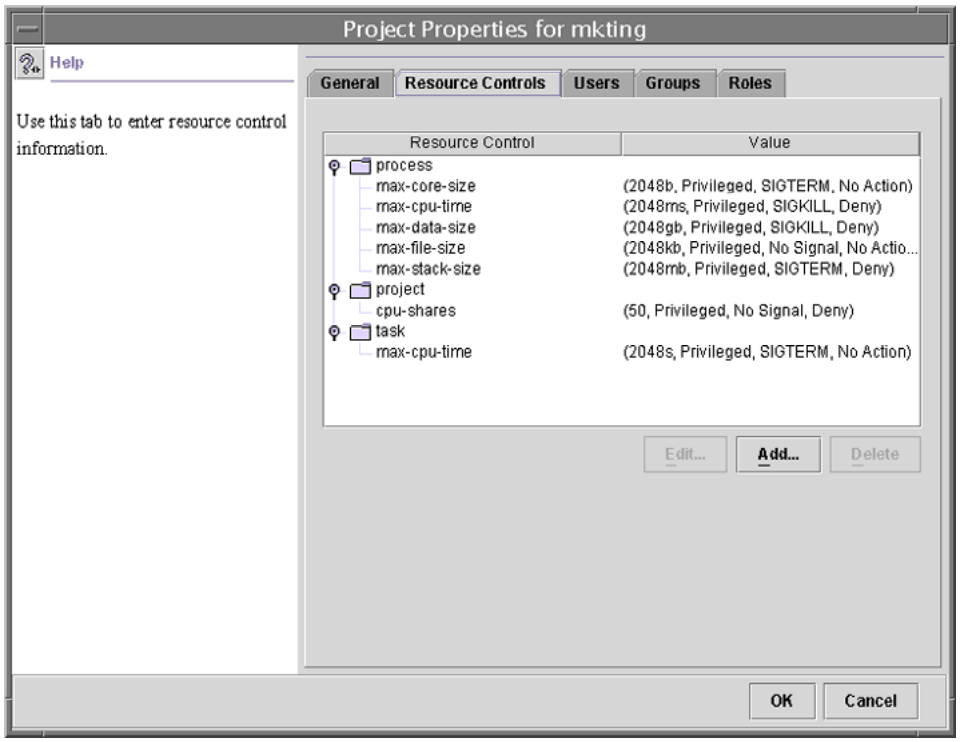


FIGURE 15–2 Resource Controls Tab in the Solaris Management Console

▼ How to Access the Resource Controls Tab

The Resource Controls tab is located under System Configuration in the Navigation pane. To access Resource Controls, do the following:

1. **Click the System Configuration control entity in the Navigation pane.**
2. **Double-click Projects.**
3. **Click on a project in the console main window to select it.**
4. **Select Properties from the Action menu.**
5. **Click the Resource Controls tab.**
   View, add, edit, or delete resource control values for processes, projects, and tasks.

Resource Controls You Can Set

The following table shows the resource controls that can be set in the console. The table describes the resource that is constrained by each control. The table also identifies the default units that are used by the `project` database for that resource. The default units are of two types:

- Quantities represent a limited amount.
- Indexes represent a maximum valid identifier.

Thus, `project.cpu-shares` specifies the number of shares to which the project is entitled. `process.max-file-descriptor` specifies the highest file number that can be assigned to a process by the `open`(2) system call.

TABLE 15–1 Standard Resource Controls Available in the Solaris Management Console

Control Name	Description	Default Unit
`project.cpu-shares`	The number of CPU shares that are granted to this project for use with the fair share scheduler (FSS) (see the `FSS`(7) man page)	Quantity (shares)
`task.max-cpu-time`	Maximum CPU time that is available to this task's processes	Time (seconds)
`task.max-lwps`	Maximum number of LWPs simultaneously available to this task's processes	Quantity (LWPs)
`process.max-cpu-time`	Maximum CPU time that is available to this process	Time (seconds)
`process.max-file-descriptor`	Maximum file descriptor index that is available to this process	Index (maximum file descriptor)
`process.max-file-size`	Maximum file offset that is available for writing by this process	Size (bytes)
`process.max-core-size`	Maximum size of a core file that is created by this process	Size (bytes)
`process.max-data-size`	Maximum heap memory that is available to this process	Size (bytes)
`process.max-stack-size`	Maximum stack memory segment that is available to this process	Size (bytes)

TABLE 15–1 Standard Resource Controls Available in the Solaris Management Console *(Continued)*

Control Name	Description	Default Unit
`process.max-address-space`	Maximum amount of address space, as summed over segment sizes, available to this process	Size (bytes)

Setting Values

You can view, add, edit, or delete resource control values for processes, projects, and tasks. These operations are performed through dialog boxes in the console.

Resource controls and values are viewed in tables in the console. The Resource Control column lists the resource controls that can be set. The Value column displays the properties that are associated with each resource control. In the table, these values are enclosed in parentheses, and they appear as plain text separated by commas. The values in parentheses comprise an "action clause." Each action clause is composed of a threshold, a privilege level, one signal, and one local action that is associated with the particular threshold. Each resource control can have multiple action clauses, which are also separated by commas.

Note – On a running system, values that are altered in the `project` database through the console only take effect for new tasks that are started in a project.

Console References

For information on projects and tasks, see Chapter 2. For information on resource controls, see Chapter 6. For information on the fair share scheduler (FSS), see Chapter 8.

Note – Not all resource controls can be set in the console. See Table 15–1 for the list of controls that can be set in the console.

PART II

Zones

This part introduces Solaris™ Zones partitioning technology, which provides a means of virtualizing operating system services to create an isolated environment for running applications. This isolation prevents processes that are running in one zone from monitoring or affecting processes running in other zones.

CHAPTER 16

Introduction to Solaris Zones

The zones facility in Solaris provides an isolated environment in which to run applications on your system. Zones are a component of the Solaris container environment.

This chapter covers the following topics:

Zones Overview

Zones are a partitioning technology used to virtualize operating system services and provide an isolated and secure environment for running applications. A zone is a virtualized operating system environment created within a single instance of the Solaris Operating System. When you create a zone, you produce an application execution environment in which processes are isolated from the rest of the system. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with superuser credentials cannot view or affect activity in other zones.

A zone also provides an abstract layer that separates applications from the physical attributes of the machine on which they are deployed. Examples of these attributes include physical device paths and network interface names.

Zones can be used on any machine that is supported on the Solaris 10 release. The upper limit for the number of zones on a system is 8192. The number of zones that can be effectively hosted on a single system is dependent upon the total resource requirements of the application software running in all of the zones.

When to Use Zones

Zones are ideal for environments that consolidate a number of applications on a single server. The cost and complexity of managing numerous machines makes it more feasible to consolidate several applications on larger, more scalable servers.

The following figure shows a system with four zones. Each of the zones `apps`, `users`, and `work` is running a workload unrelated to the workloads of the other zones, in a sample consolidated environment. This example illustrates that different versions of the same application can be run without negative consequences in different zones, to match the consolidation requirements. Each zone can provide a customized set of services.

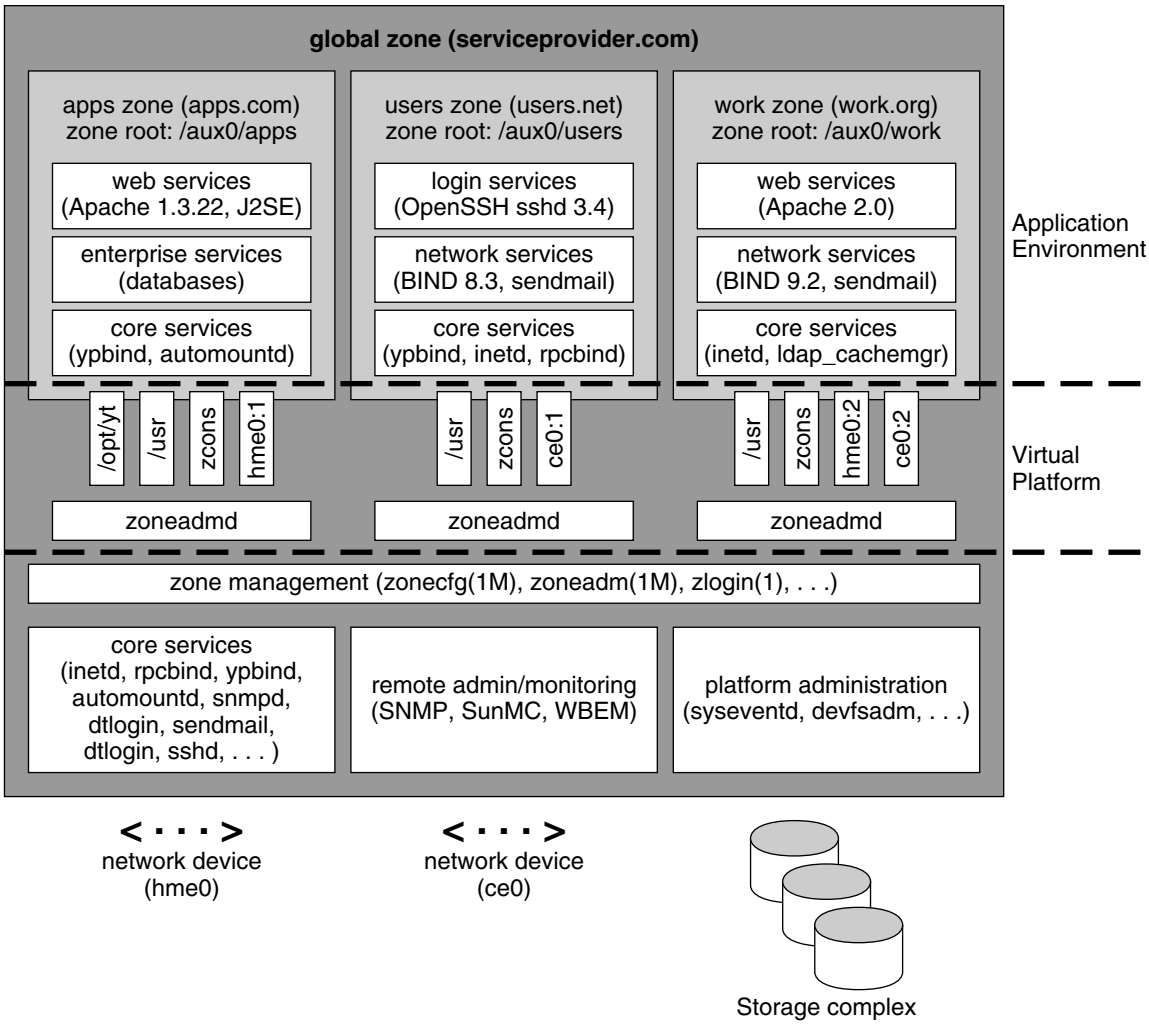


FIGURE 16–1 Zones Server Consolidation Example

Zones enable more efficient resource utilization on your system. Dynamic resource reallocation permits unused resources to be shifted to other containers as needed. Fault and security isolation mean that poorly behaved applications do not require a dedicated and underutilized system. With the use of zones, these applications can be consolidated with other applications.

Zones allow you to delegate some administrative functions while maintaining overall system security.

How Zones Work

A zone can be thought of as a box. One or more applications can run in this box without interacting with the rest of the system. Solaris zones isolate software applications or services by using flexible, software-defined boundaries. Applications that are running in the same instance of the Solaris operating system can then be managed independently of one other. Thus, different versions of the same application can be run in different zones, to match the requirements of your configuration.

Each zone that requires network connectivity has one or more dedicated IP addresses. A process assigned to a zone can manipulate, monitor, and directly communicate with other processes that are assigned to the same zone. The process cannot perform these functions with processes that are assigned to other zones in the system or with processes that are not assigned to a zone. Processes that are assigned to different zones are only able to communicate through network APIs.

Every Solaris system contains a *global zone*. The global zone has a dual function. The global zone is both the default zone for the system and the zone used for system-wide administrative control. All processes run in the global zone if no *non-global* zones, referred to simply as zones, are created by the *global administrator*.

The global zone is the only zone from which a non-global zone can be configured, installed, managed, or uninstalled. Only the global zone is bootable from the system hardware. Administration of the system infrastructure, such as physical devices, routing, or dynamic reconfiguration (DR), is only possible in the global zone. Appropriately privileged processes running in the global zone can access objects associated with other zones.

Unprivileged processes in the global zone might be able to perform operations not allowed to privileged processes in a non-global zone. For example, users in the global zone can view information about every process in the system. If this capability presents a problem for your site, you can restrict access to the global zone.

Each zone, including the global zone, is assigned a zone name. The global zone always has the name `global`. Each zone is also given a unique numeric identifier, which is assigned by the system when the zone is booted. The global zone is always mapped to ID `0`. Zone names and numeric IDs are discussed in "Using the `zonecfg` Command" on page 183.

Each zone also has a node name that is completely independent of the zone name. The node name is assigned by the administrator of the zone. For more information, see "Non-Global Zone Node Name" on page 232

Each zone has a path to its root directory that is relative to the global zone's root directory. For more information, see "Using the `zonecfg` Command" on page 183.

Summary of Zone Features

The following table summarizes the characteristics of global and non-global zones.

Type of Zone	Characteristic
Global	■ Is assigned ID `0` by the system ■ Provides the single instance of the Solaris kernel that is bootable and running on the system ■ Contains a complete installation of the Solaris system software packages ■ Can contain additional software packages or additional software, directories, files, and other data not installed through packages ■ Provides a complete and consistent product database that contains information on all software components installed in the global zone ■ Holds configuration information specific to the global zone only, such as the global zone host name and file system table ■ Is the only zone that is aware of all devices and all file systems ■ Is the only zone with knowledge of non-global zone existence and configuration ■ Is the only zone from which a non-global zone can be configured, installed, managed, or uninstalled
Non-Global	■ Is assigned a zone ID by the system when the zone is booted ■ Shares operation under the Solaris kernel booted from the global zone ■ Contains an installed subset of the complete Solaris Operating System software packages ■ Contains Solaris software packages shared from the global zone ■ Can contain additional installed software packages not shared from the global zone ■ Can contain additional software, directories, files, and other data created on the non-global zone that are not installed through packages or shared from the global zone ■ Has a complete and consistent product database that contains information on all software components installed on the zone, whether present on the non-global zone or shared read-only from the global zone ■ Is not aware of the existence of any other zone(s) ■ Cannot install, manage, or uninstall other zones, including itself ■ Has configuration information specific to the non-global zone only, such as the non-global zone host name and file system table

How Non-Global Zones Are Administered

A global administrator has superuser privileges or the Primary Administrator role. When logged into the global zone, the global administrator can monitor and control the system as a whole.

A non-global zone can be administered by a *zone administrator*. The global zone administrator assigns the Zone Management profile to the zone administrator. The privileges of a zone administrator are confined to a non-global zone.

How Non-Global Zones Are Created

The global administrator configures a zone by specifying various parameters for the zone's virtual platform and application environment. The `zonecfg` command is used to create this configuration. The zone is then installed by the global administrator, who uses the zone administration command `zoneadm` to install software at the package level into the file system hierarchy established for the zone. The `zoneadm` command is then used to boot the zone. After booting the zone, the global administrator can log into the zone by using the `zlogin` command. At first login, the internal configuration for the zone is completed.

For information on zone configuration, installation, and login, see Chapter 17, Chapter 19, and Chapter 21.

Non-Global Zone State Model

A non-global zone can be in one of the following six states.

Configured	The zone's configuration is completely specified and committed to stable storage. However, those elements of the zone's application environment that must be specified after initial boot are not yet present.
Incomplete	During an install or uninstall operation, `zoneadm` sets the state of the target zone to incomplete. Upon successful completion of the operation, the state is set to the correct state.
Installed	The zone's configuration is instantiated on the system. The `zoneadm` command is used to verify that the configuration can be successfully used on the designated Solaris system. Packages are installed under the zone's root path. In this state, the zone has no associated virtual platform.
Ready	The virtual platform for the zone is established. The kernel creates the `zsched` process, network interfaces

	are plumbed, file systems are mounted, and devices are configured. A unique zone ID is assigned by the system. At this stage, no processes associated with the zone have been started.
Running	User processes associated with the zone application environment are running. The zone enters the running state as soon as the first user process associated with the application environment (`init`) is created.
Shutting down and down	These states are transitional states that are visible while the zone is being halted. However, a zone that is unable to shut down for any reason will stop in one of these states.

Chapter 20 and the `zoneadm`(1M) man page describe how to use the `zoneadm` command to initiate transitions between these states.

TABLE 16–1 Commands That Affect Zone State

Current Zone State	Applicable Commands

Configured	`zonecfg -z` *zonename* `verify`
	`zonecfg -z` *zonename* `commit`
	`zonecfg -z` *zonename* `delete`
	`zoneadm -z` *zonename* `verify`
	`zoneadm -z` *zonename*`install`

Incomplete	`zoneadm -z` *zonename* `uninstall`
Installed	`zoneadm -z` *zonename* `ready` (optional) `zoneadm -z` *zonename* `boot` `zoneadm -z` *zonename* `uninstall` uninstalls the configuration of the specified zone from the system.
Ready	`zoneadm -z` *zonename* `boot` `zoneadm halt` and system reboot return a zone in the ready state to the installed state.

Running	`zlogin` *options* `zonename`
	`zoneadm -z` *zonename* `reboot`
	`zoneadm -z` *zonename* `halt` returns ready zone to the installed state.
	`zoneadm halt` and system reboot return a zone in the running state to the installed state.

Non-Global Zone Characteristics

A zone provides isolation at almost any level of granularity you require. A zone does not need a dedicated CPU, a physical device, or a portion of physical memory. These resources can either be multiplexed across a number of zones running within a single domain or system, or allocated on a per-zone basis using the resource management features available in the operating system.

Each zone can provide a customized set of services. To enforce basic process isolation, a process can see or signal only those processes that exist in the same zone. Basic communication between zones is accomplished by giving each zone at least one logical network interface. An application running in one zone cannot observe the network traffic of another zone. This isolation is maintained even though the respective streams of packets travel through the same physical interface.

Each zone is given a portion of the file system hierarchy. Because each zone is confined to its subtree of the file system hierarchy, a workload running in a particular zone cannot access the on-disk data of another workload running in a different zone.

Files used by naming services reside within a zone's own root file system view. Thus, naming services in different zones are isolated from one other and the services can be configured differently.

Using Resource Management Features With Non-Global Zones

If resource management features are used, it is best to align the boundaries of resource management controls with those of the zones. This alignment creates a more complete model of a virtual machine, where namespace access, security isolation, and resource usage are all controlled.

Any special requirements for using the various resource management features with zones are addressed in the individual chapters that document those features.

Features Provided by Zones

Non-global zones provide the following features:

Security — Once a process has been placed in a zone other than the global zone, neither the process nor any of its subsequent children can change zones.

	Network services can be run in a zone. By running network services in a zone, you limit the damage possible in the event of a security violation. An intruder who successfully exploits a security flaw in software running within a zone is confined to the restricted set of actions possible within that zone. The privileges available within a zone are a subset of those available in the system as a whole.
Isolation	Zones allow the deployment of multiple applications on the same machine, even if those applications operate in different trust domains, require exclusive access to a global resource, or present difficulties with global configurations. For example, multiple applications running in different zones on the same system can bind to the same network port by using the distinct IP addresses associated with each zone or by using the wildcard address. The applications are also prevented from monitoring or intercepting each other's network traffic, file system data, or process activity.
Virtualization	Zones provide a virtualized environment that can hide details such as physical devices and the system's primary IP address and host name from applications. The same application environment can be maintained on different physical machines. The virtualized environment allows separate administration of each zone. Actions taken by a non-global zone administrator do not affect the rest of the system.
Granularity	A zone can provide isolation at almost any level of granularity. See "Non-Global Zone Characteristics" on page 176 for more information.
Environment	Zones do not change the environment in which applications execute except when necessary to achieve the goals of security and isolation. Zones do not present a new API or ABI to which applications must be ported. Instead, zones provide the standard Solaris interfaces and application environment, with some restrictions. The restrictions primarily affect applications that attempt to perform privileged operations. Applications in the global zone run without modification, whether or not additional zones are configured.

Setting Up Zones on Your System (Task Map)

The following task map provides a basic overview of the steps that are involved in setting up zones on your system for the first time.

Task	Description	For Instructions
Identify the applications that you would like to run in zones.	Review the applications running on your system. ■ Determine which applications are critical to your business goals. ■ Assess the system needs of the applications you are running.	Refer to your business goals and to your system documentation if necessary.
Determine how many zones to configure.	Assess: ■ The performance requirements of the applications you intend to run in zones ■ The availability of the recommended 100 MB of free disk space per zone to be installed If you are also using resource management features on your system, align the zones with the resource management boundaries.	See "Evaluate Current System Setup" on page 193, Chapter 1.
Perform the preconfiguration tasks.	Determine zone name and zone path, obtain IP addresses, determine required files and devices for each zone.	See Chapter 17, "Evaluate Current System Setup" on page 193.
Develop configurations.	Configure non-global zones.	See "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197 and `zonecfg`(1M).

Task	Description	For Instructions
As global administrator, verify and install configured zones.	Zones must be verified and installed prior to login.	See Chapter 19 and Chapter 20.
As global administrator, log in to the non-global zone.	Log in to perform the initial internal configuration of the zone, including assigning the zone root password.	See Chapter 21 and Chapter 22.
As global administrator, boot the non-global zone.	Boot the zone to place the zone in the running state.	See Chapter 19 and Chapter 20.

CHAPTER 17

About Zone Configuration (Overview)

This chapter provides an introduction to non-global zone configuration.

The following topics are covered:

The Pre-Installation Configuration Process

Before you can install a non-global zone and use it on your system, the zone must be configured.

The `zonecfg` command is used to create the configuration and to determine whether the resources and properties specified are valid on a hypothetical system. The verification check performed by `zonecfg` for a given configuration includes the following:

- Ensures that a zone path is specified.
- Ensures that all of the required properties for each resource are specified.

For more information on the `zonecfg` command, see the `zonecfg`(1M) man page.

Zone Components

This section covers the zone resources and properties that can be configured.

Zone Name and Path

You must choose a name and a path for your zone.

Zone Interfaces

Each zone that requires network connectivity must have one or more dedicated IP addresses. These addresses are associated with logical network interfaces. Zone interfaces configured by the `zonecfg` command will automatically be plumbed and placed in the zone when it is booted. The `zonecfg` command is described in the `zonecfg`(1M) man page.

The `ifconfig` command can be used from the global zone to add or remove logical interfaces in a running zone. For more information, see "Network Interfaces" on page 238.

File Systems Mounted in Zones

Generally, the file systems mounted in a zone include the following:

- The set of file systems mounted when the virtual platform is initialized
- The set of file systems mounted from within the application environment itself

This can include, for example, the following file systems:

- File systems specified in a zone's `/etc/vfstab` file
- `AutoFS` and `AutoFS`-triggered mounts
- Mounts explicitly performed by a zone administrator

Certain restrictions are placed on mounts performed from within the application environment. These restrictions prevent the zone administrator from denying service to the rest of the system, or otherwise negatively impacting other zones.

There are security restrictions associated with mounting certain file systems from within a zone. Other file systems exhibit special behavior when mounted in a zone. See "File Systems and Non-Global Zones" on page 233 for more information.

Configured Devices in Zones

`zonecfg` uses a rule-matching system to specify which devices should appear in a particular zone. Devices matching one of the rules are included in the zone's `/dev` file system. For more information, see "How to Configure the Zone" on page 197.

Resource Controls in Zones

Privileged zone-wide resource controls can be set for a zone by the global zone administrator. Zone-wide resource controls limit the total resource usage of all process entities within a zone, regardless of project. These limits are specified in the `zonecfg` configuration. For more information, see "How to Configure the Zone" on page 197.

Including a Comment for a Zone

You can add a comment for a zone by using the `attr` resource type. For more information, see "How to Configure the Zone" on page 197.

Using the `zonecfg` Command

The `zonecfg` command, which is described in the `zonecfg`(1M) man page, is used to configure a zone. The `zonecfg` command can be used in interactive mode, in command-line mode, or in command-file mode. The following operations can be performed using the command:

- Create or delete (destroy) a zone configuration
- Add resources to a particular configuration
- Set properties for resources added to a configuration
- Remove resources from a particular configuration
- Query or verify a configuration
- Commit to a configuration
- Revert to a previous configuration
- Exit from a `zonecfg` session

The `zonecfg` prompt is of the following form:

`zonecfg:`*zonename*`>`

When you are configuring a specific resource type, such as a file system, that resource type is also included in the prompt:

`zonecfg:`*zonename*`:fs>`

For more information, including procedures that show how to use the various `zonecfg` components described in this chapter, see Chapter 18.

`zonecfg` Modes

The concept of a *scope* is used for the user interface. The default scope is global.

In the global scope, the `add` subcommand and the `select` subcommand are used to select a specific resource. The scope then changes to that resource type. For the `add` subcommand, the `end` or `cancel` subcommands are used to complete the resource specification. For the `select` subcommand, the `end` or `cancel` subcommands are used to complete the resource modification. The scope then reverts back to global.

Certain subcommands, such as `add`, `remove`, and `set`, have different semantics in each scope.

`zonecfg` Interactive Mode

In interactive mode, the following subcommands are supported. For detailed information on semantics and options used with the subcommands, see the `zonecfg`(1M) man page. Note that any subcommand that can result in destructive actions or loss of work confirms with the user before proceeding. You can use the `-F` (force) option to bypass this confirmation.

`help` Print general help, or display help about a given resource.

```
zonecfg:my-zone:inherit-pkg-dir> help
```

See the `zonecfg`(1M) man page for options.

`create` Use the `create` command to begin configuring an in-memory configuration for the specified new zone.

Use `create` to apply the Sun default settings to a new configuration. This method is the default.

Use `create` with the `-t` *template* option to create a configuration that is identical to the specified template. The zone name is changed from the template name to the new zone name.

Use `create` with the `-F` option to overwrite an existing configuration.

Use `create` with the `-b` option to create a blank configuration in which nothing is set.

`export` Print the configuration to `stdout`, or to the output file specified, in a form that can be used in a command file.

`add` In the global scope, add the specified resource type to the configuration. In the resource scope, add a property of the given name with the given value.

See "How to Configure the Zone" on page 197 and the `zonecfg`(1M) man page for more information.

`set` Set a given property name to the given property value. Note that some properties, such as `zonepath`, are global, while others are resource-specific. Thus, this command is applicable in both the global and resource scopes.

`select` Applicable only in the global scope. Select the resource of the given type that matches the given property name-property value pair criteria for modification. The scope is changed to that resource type. You must specify a sufficient number of property name-value pairs for the resource to be uniquely identified.

`remove` In the global scope, the specified resource type is removed. You must specify a sufficient number of property name-value pairs for the resource type to be uniquely identified. In the resource scope, the specified property name-property value is removed from the current resource.

`end` Applicable only in the resource scope. End the resource specification.

`zonecfg` then verifies that the current resource is fully specified. If the resource is fully specified, it is added to the in-memory configuration and the scope will revert back to global. If the specification is incomplete, an error message is issued that describes what needs to be done.

`cancel` Applicable only in the resource scope. End the resource specification and reset the scope to global. Any partially specified resources are not retained.

`delete` Destroy the specified configuration. Delete the configuration both from memory and from stable storage. You must use the `-F` (force) option with `delete`.

Caution – This action is instantaneous. No commit is required, and a deleted zone cannot be reverted.

`info` Display information about the current configuration or about the global resource properties `zonepath`, `autoboot`, and `pool`. If a resource type is specified, display information only about resources of that type. In the resource scope, this subcommand applies only to the resource being added or modified.

`verify` Verify current configuration for correctness. Ensure that all resources have all of their required properties specified.

`commit` Commit current configuration from memory to stable storage. Until the in-memory configuration is committed, changes can be removed with the `revert` subcommand. A configuration must be committed to be used by `zoneadm`. This operation is attempted automatically when you complete a `zonecfg` session. Because only a correct configuration can be committed, the commit operation automatically does a verify.

`revert` Revert configuration back to the last committed state.

`exit -F` Exits the `zonecfg` session. You can use the `-F` (force) option with `exit`.

A `commit` is automatically attempted if needed. Note that an EOF character can also be used to exit the session.

`zonecfg` Command-File Mode

In command-file mode, input is taken from a file. The `export` subcommand described in "`zonecfg` Interactive Mode" on page 184 is used to produce this file. The configuration can be printed to standard output, or the `-f` option can be used to specify an output file.

Zone Configuration Data

Zone configuration data consists of two kinds of entities: resources and properties. Each resource has a type, and each resource can also have a set of one or more properties. The properties have names and values. The set of properties is dependent on the resource type.

The resource types are described as follows.

Zone name The zone name identifies the zone to the configuration utility. The following rules apply to zone names:

- Each zone must have a unique name.
- Zone names are case-sensitive. A zone name must begin with an alpha-numeric character.
- The name can contain alpha-numeric characters, the underbar (_), the hyphen (-), and the period (.).
- The name cannot be longer than 64 characters.

- The name `global` and all names beginning with `SUNW` are reserved and cannot be used.

`zonepath`

The zone path resource is the path to the zone root. Each zone has a path to its root directory that is relative to the global zone's root directory. At installation time, the global zone directory is required to have restricted visibility. It must be owned by `root` with the mode `700`.

The non-global zone's root path will be one level lower. The zone's root directory will have the same ownership and permissions as the root directory in the global zone. The zone directory must be owned by `root` with the mode `755`. This hierarchy ensures that unprivileged users in the global zone are prevented from traversing a non-global zone's file system. See "Traversing File Systems" on page 237 for a further discussion of this issue.

`fs`

The file system resource is the path to the file system mount point. Each zone can have various file systems that are mounted when the zone transitions from the installed state to the ready state. For more information on the use of file systems in zones, see "File Systems and Non-Global Zones" on page 233.

`inherit-pkg-dir`

The `inherit-pkg-dir` resource is used to represent directories that contain packaged software that the non-global zone shares with the global zone.

The contents of software packages transferred into the `inherit-pkg-dir` directory are inherited in a read-only mode by the non-global zone. The zone's packaging database is updated to reflect the packages. These resources cannot be modified or removed after the zone has been installed using `zoneadm`.

Note – Four default `inherit-pkg-dir` resources are included in the configuration. These directory resources indicate which directories should have their associated packages inherited from the global zone. The resources are implemented through a read-only loopback file system mount.

- `/lib`
- `/platform`
- `/sbin`
- `/usr`

`net` The network interface resource is the virtual interface name. Each zone can have network interfaces that should be plumbed when the zone transitions from the installed state to the ready state.

`device` The device resource is the device matching specifier. Each zone can have devices that should be configured when the zone transitions from the installed state to the ready state.

`rctl` The `rctl` resource is used for zone-wide resource controls. The controls should be enabled when the zone transitions from the installed state to the ready state. Note that the only zone-wide resource control implemented in this release is `zone.cpu-shares`.

`attr` This generic attribute can be used for user comments or by other subsystems. The `name` property of an `attr` must begin with an alpha-numeric character. The `name` property can contain alpha-numeric characters, hyphens (`-`), and periods (`.`) . Attribute names beginning with `zone.` are reserved for use by the system.

Some resource types also have properties to configure. The following property types are associated with the resource types shown.

`fs` `dir`, `special`, `type`, `options`

The lines in the following example specify that `/opt/local` in the global zone is to be mounted as `/usr/local` in a zone being configured. The file system type to use is LOFS. The option `nodevices` is added.

```
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/usr/local
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs> set special=/opt/local
zonecfg:my-zone:fs> add option nodevices
zonecfg:my-zone:fs> end
```

`inherit-pkg-dir` `dir`

The lines in the following example specify that `/opt/sfw` is to be loopback mounted from the global zone.

```
zonecfg:my-zone> add inherit-pkg-dir
zonecfg:my-zone:inherit-pkg-dir> set dir=/opt/sfw
zonecfg:my-zone:inherit-pkg-dir> end
```

`net` `address`, `physical`

In the following example, IP address `192.168.0.1` is added to a zone. An `hme0` card is used for the physical interface.

```
zonecfg:my-zone> add net
zonecfg:my-zone:net> set physical=hme0
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> end
```

Note – To determine which physical interface to use, type `ifconfig -a` on your system. Each line of the output, other than loopback driver lines, begins with the name of a card installed on your system. Lines that contain `LOOPBACK` in the descriptions do not apply to cards.

`device` `match`

In the following example, a `/dev/pts` device is included in a zone.

```
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/pts*
zonecfg:my-zone:device> end
```

`rctl` `name, value`

In this release, there is one zone-specific resource control, `zone.cpu-shares`. This resource control sets a limit on the number of fair share scheduler (FSS) CPU shares for a zone. CPU shares are first allocated to the zone, and then further subdivided among projects within the zone in accordance with the `project.cpu-shares` entry.

Note that zone-wide resource control entries in a zone are configured differently than resource control entries in the `project` database. In a configuration for a zone, the `rctl` resource type consists of three name/value pairs. The names are `priv`, `limit`, and `action`. Each of the names takes a simple value.

```
zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone-cpu.shares
zonecfg:my-zone:rctl> add value (priv=privileged,limit=20,action=deny)
zonecfg:my-zone:rctl> end
```

For general information on resource controls and attributes, see Chapter 6 and "Resource Controls Used in Non-Global Zones" on page 240.

`attr` `name, type, value`

In the following example, a comment about a zone is added.

```
zonecfg:my-zone> add attr
zonecfg:my-zone:attr> set name=comment
zonecfg:my-zone:attr> set type=string
zonecfg:my-zone:attr> set value="Production zone"
zonecfg:my-zone:attr> end
```

You can use the `export` subcommand to print a zone configuration to `stdout`. The configuration is saved in a form that can be used in a command file.

Where to Go From Here

To configure a non-global zone for installation on your system, go to Chapter 18.

CHAPTER 18

Planning and Configuring a Non-Global Zone (Tasks)

This chapter describes what you need to do before you configure a zone on your system. It also discusses how to configure a zone and verify that the zone can be installed on a hypothetical Solaris system.

The following topics and tasks are covered.

Defining Configuration Parameters (Task Map)

Before you set up your system to use zones, you must first collect information and make decisions about how you will configure the zones. The following task map summarizes the steps to take as you prepare and create a zone configuration.

Task	Description	For Instructions
Plan your zone strategy.	■ Evaluate the applications running on your system to determine which applications you want to run in the zone. ■ Assess the availability of disk space to hold the files that are unique in the zone. ■ If you are also using resource management features, determine how to align the zone with the resource management boundaries.	Refer to historical usage. See "Disk Space Requirements" on page 193. Also see "Resource Pools Used in Zones" on page 111.
Determine the name for the zone.	Decide what you will call the zone based on the naming conventions.	See "Zone Configuration Data" on page 186.
Obtain or configure IP addresses for the zone.	Depending on your configuration, you must obtain at least one IP address for each non-global zone that you want to have network access.	See "Determine the Zone Hostname and Obtain the Network Address" on page 194 and *System Administration Guide: IP Services*.
Determine which file systems you want to mount in the zone.	Review your application requirements.	TBD
Determine which network interfaces should be plumbed in the zone.	Review your application requirements.	TBD
Determine which devices should be configured in each zone.	Review your application requirements.	TBD
Determine the zone path.	Each zone has a path to its root directory relative to the global zone's root directory.	See "Zone Configuration Data" on page 186.
Create configuration	Use `zonecfg` to create a configuration for the zone.	See "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197.

Task	Description	For Instructions
Verify and commit the configured zone.	Determine whether the resources and properties specified are valid on a hypothetical system.	See "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197.

Evaluate Current System Setup

Zones can be used on any machine that is supported on the Solaris 10 release. These are the primary machine considerations associated with the use of zones.

- The performance requirements of the applications running within each zone.
- The availability of disk space to hold the files that are unique within each zone. See "Disk Space Requirements" on page 193.

Determining Which Applications to Run in a Zone

TBD

Disk Space Requirements

There are no limits on how much disk space can be consumed by a zone. The global administrator is responsible for space restriction. Even a small uniprocessor system can support a number of zones running simultaneously.

The nature of the packages installed in the global zone affects the space requirements of the non-global zones that are created. Number of packages and space requirements are factors.

- As a general guideline, a zone requires about 100 megabytes of free disk space per zone when the global zone has been installed with all of the standard Solaris packages.
- By default, any additional packages installed in the global zone also populate the non-global zones. The amount of disk space required must be increased accordingly. The directory location in the non-global zone for these additional packages is specified through the `inherit-pkg-dir` resource.

An additional 40 megabytes of RAM per zone are suggested, but not required on a machine with sufficient swap space.

Restricting Zone Size

The following options can be used to restrict zone size.

- You can place the zone on a `lofi`-mounted partition. This action will limit the amount of space consumed by the zone to that of the file used by `lofi`. For more information, see the `lofiadm`(1M) and `lofi`(7D) man pages.
- You can use soft partitions to divide disk slices or logical volumes into partitions. You can use these partitions as zone roots, and thus limit per-zone disk consumption. The soft partition limit is 8192 partitions. For more information, see "Soft Partitions (Overview)" in *Solaris Volume Manager Administration Guide*.
- You can use the standard partitions of a disk for zone roots, and thus limit per-zone disk consumption.

Determine the Zone Hostname and Obtain the Network Address

Determine the hostname for the zone. Then, assign an IPv4 address or manually configure and assign an IPv6 address for the zone.

The Zone Hostname

The hostname you select for the zone must be defined either in the `hosts` database or in the `/etc/inet/ipnodes` database, as specified by the `/etc/nsswitch.conf` file in the global zone. The network databases are files that provide information that is needed to configure the network. The `nsswitch.conf`file specifies which name service to use.

If you use local files for the name service, the `hosts` database is maintained in the `/etc/inet/hosts` file. The hostnames for zone interfaces are resolved from the local `hosts` database in `/etc/inet/hosts`. Alternatively, the IP address itself can be specified directly when configuring a zone so that no hostname resolution is required.

For more information, see "TCP/IP Configuration Files" in *System Administration Guide: IP Services* and "Network Databases and File" in *System Administration Guide: IP Services*.

The Zone Network Address

Each zone that requires network connectivity has one or more unique IP addresses. Both IPv4 and IPv6 addresses are supported.

The IPv4 Zone Network Address

If you are using IPv4, obtain and assign an address to the zone.

A prefix length can also be specified with the IP address. The format of this prefix is *address/prefix-length*, for example, `192.168.1.1/24` Thus, the address to use is `192.168.1.1` and the netmask to use is `255.255.255.0`, or the mask where the first 24-bits are 1-bits.

The IPv6 Zone Network Address

If you are using IPv6, you must manually configure the address. Typically, at least the following two types of addresses must be configured.

Link-local address
: Link-local addresses are of the form `fe80::`*<64-bit interface ID>*`/10`. The `/10` indicates that the first 10 bits are the Prefix identifying a link-local address.

Address formed from a global prefix configured on the subnet
: Global unicast addresses are based off a 64–bit prefix that the administrator configures for each subnet, and a 64-bit interface ID. The prefix can also be obtained by running the `ifconfig` command with the `-a6` option on any system on the same subnet that has been configured to use IPv6.

  The 64–bit interface ID is typically derived from a system's MAC address. For zones use, an alternate address that is unique can be derived from the global zone's IPv4 address as follows:

  ```
  <32 bits of zero:32 bits of the non-global zone's IPv4 address>
  ```

  For example, if the global zone's IPv4 address is 192.168.1.1, then a suitable link-local address for a local zone with `zoneid 1` is `fe80::c0a8:0101`. If the global prefix in use on that subnet is `2001:0db8:56bb:9256/64`, then a unique global unicast address for the same non-global zone is `2001:0db8:56bb:9256::c009:c801`. Note that you must always specify a prefix length when configuring an IPv6 address.

For more information on link-local and global unicast addresses, see the `inet6`(7P) manual page.

File System Configuration

You can specify a number of mounts to be performed when the virtual platform is set up. File systems that are loopback-mounted into a zone by using the loopback file system (LOFS) virtual file system should be mounted with the `nodevices` option. For information on the `nodevices` option, see "File Systems and Non-Global Zones" on page 233.

LOFS lets you create a new virtual file system so that you can access files by using an alternative path name. In a non-global zone, a loopback mount makes the file system hierarchy look as though it is duplicated under the zone's root. In the zone, all files will be accessible with a path name that starts from the zone's root. LOFS mounting preserves the file system name space.

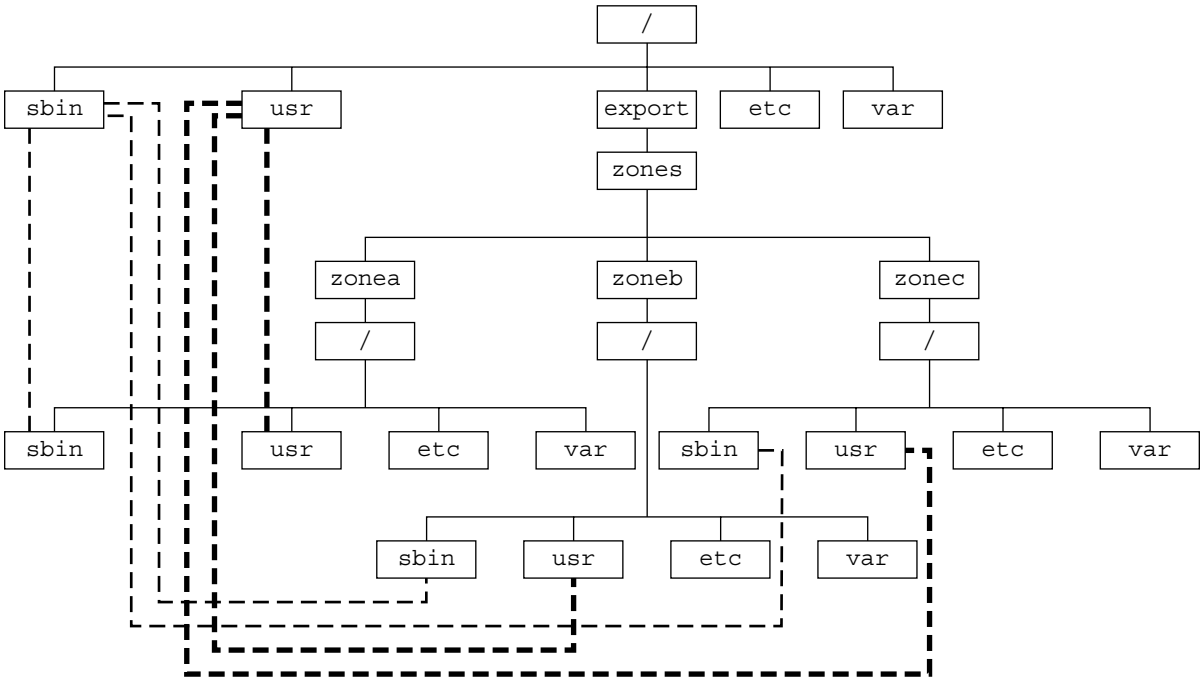


FIGURE 18–1 Loopback-Mounted File Systems

See `lofs`(7S) for more information.

Using the zonecfg Command to Configure, Verify, and Commit a Zone

Use the zonecfg command described in zonecfg(1M) to configure, verify, and commit a non-global zone.

▼ How to Configure the Zone

Create the zone configuration. Then, verify that all required information is present and commit the configuration.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Set up a zone configuration with the zone name you have chosen. The name `my-zone` is used in this example procedure.**

   ```
   global# zonecfg -z my-zone
   ```

   If this is the first time you have configured this zone, you will see the following system message:

   ```
   my-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

3. **Create the new zone configuration. This procedure uses the Sun default settings.**

   ```
   zonecfg:my-zone> create
   ```

4. **Set the zone path. In this example procedure, the path is `/export/home/my-zone`.**

   ```
   zonecfg:my-zone> set zonepath=/export/home/my-zone
   ```

5. **Set the autoboot value.**

   If set to `true`, then the zone will automatically be booted when the global zone is booted. The default value is `false`.

   ```
   zonecfg:my-zone> set autoboot=true
   ```

6. **If resource pools are enabled on your system, associate a pool with the zone. This example uses the default pool, named `pool_default`.**

   ```
   zonecfg:my-zone> set pool=pool_default
   ```

7. **Add a file system. This step can be performed more than once to add more than one file system.**

   ```
   zonecfg:my-zone> add fs
   ```

8. **Set the mount point for the file system. In this procedure, the mount point is `/usr/local`.**

   ```
   zonecfg:my-zone:fs> set dir=/usr/local
   ```

9. **Specify that `/opt/local` in the global zone is to be mounted as `/usr/local` in the zone being configured.**

   ```
   zonecfg:my-zone:fs> set special=/opt/local
   ```

   In the non-global zone, `/usr/local` file system will be readable and writable.

10. **Specify the file system type. In this procedure, the type is `lofs`.**

    ```
    zonecfg:my-zone:fs> set type=lofs
    ```

11. **End the file system specification.**

    ```
    zonecfg:my-zone:fs> end
    ```

12. **Add a shared file system that is loopback-mounted from the global zone. This step can be performed more than once to add more than one shared file system.**

    ```
    zonecfg:my-zone> add inherit-pkg-dir
    ```

13. **Specify that `/opt/sfw` in the global zone is to be mounted in read-only mode in the zone being configured.**

    ```
    zonecfg:my-zone:inherit-pkg-dir> set dir=/opt/sfw
    ```

Note – The zone's packaging database is updated to reflect the packages. These resources cannot be modified or removed after the zone has been installed using `zoneadm`

---

14. **End the `inherit-pkg-dir` specification.**

    ```
    zonecfg:my-zone:inherit-pkg-dir> end
    ```

15. **Add a network virtual interface name. This step can be performed more than once to add more than one network interface.**

    ```
    zonecfg:my-zone> add net
    ```

16. **Set the IP address for the network interface. The address `192.168.0.1` is used here.**

    ```
    zonecfg:my-zone:net> set address=192.168.0.1
    ```

17. **Set the physical device type for the network interface. The `hme` device is used in this procedure.**

```
zonecfg:my-zone:net> set physical=hme0
```

18. **End the specification.**

```
zonecfg:my-zone:net> end
```

19. **Add a device. This step can be performed more than once to add more than once device.**

```
zonecfg:my-zone> add device
```

20. **Set the device match. `/dev/sound/*` is used in this procedure.**

```
zonecfg:my-zone:device> set match=/dev/sound/*
```

21. **End the device specification.**

```
zonecfg:my-zone:device> end
```

22. **Add a zone-wide resource control.**

```
zonecfg:my-zone> add rctl
```

23. **Set the name of the resource control. In this example, `zone.cpu-shares`, is used.**

```
zonecfg:my-zone:rctl> set name=zone.cpu-shares
```

24. **Add values for the privilege, the share limit, and the action to be taken when that threshold is reached.**

```
zonecfg:my-zone:rctl> add value (priv=privileged,limit=20,action=deny)
```

25. **End the `rctl` specification.**

```
zonecfg:my-zone:rctl> end
```

26. **Add a comment by using the `attr` resource type.**

```
zonecfg:my-zone> add attr
```

27. **Set the name to `comment`.**

```
zonecfg:my-zone:attr> set name=comment
```

28. **Set the type to `string`.**

```
zonecfg:my-zone:attr> set type=string
```

29. **Set the `value` to a comment that describes the zone.**

```
zonecfg:my-zone:attr> set value="This is my work zone."
```

30. **End the attr specification.**

```
zonecfg:my-zone:attr> end
```

31. **Verify the zone configuration for the zone.**

```
zonecfg:my-zone> verify
```

32. **Commit the zone configuration for the zone.**

```
zonecfg:my-zone> commit
```

33. **Exit the `zonecfg` command.**

```
zonecfg:my-zone> exit
```

Note that even if you did not explicitly type `commit` at the prompt, a `commit` is automatically attempted when `exit` or an EOF occurs.

See "Installing and Booting Zones" on page 211 to install your commited zone configuration.

Note – The `zonecfg` command also supports multiple subcommands, quoted and separated by semicolons, from the same shell invocation.

```
global# zonecfg -z my-zone "create ; set zonepath=/export/home/my-zone"
```

▼ How to Modify a Resource Type in a Zone Configuration

You can select a resource type and modify the specification for that resource.

Note that the contents of software packages in the `inherit-pkg-dir` directory cannot be modifed or removed after the zone has been installed with `zoneadm`.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Select the zone to be modified. The name `my-zone` is used in this procedure.**

```
global# zonecfg -z my-zone
```

3. **Select the resource type to be changed. This example uses a resource control.**

```
zonecfg:my-zone> select rctl name=zone.cpu-shares
```

4. **Remove the current value.**

```
zonecfg:my-zone:rctl> remove value (priv=privileged,limit=20,action=deny)
```

5. **Add the new value.**

```
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=deny)
```

6. **End the revised `rctl` specification.**

```
zonecfg:my-zone:rctl> end
```

7. **Commit the zone configuration for the zone.**

```
zonecfg:my-zone> commit
```

8. **Exit the `zonecfg` command.**

```
zonecfg:my-zone> exit
```

   Note that even if you did not explicitly type `commit` at the prompt, a `commit` is automatically attempted when `exit` or an EOF occurs.

▼ How to Add a Dedicated Device to a Zone

The following specification places a scanning device in a non-global zone configuration.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Add a device.**

```
zonecfg:myzone> add device
```

3. **Set the device match. `/dev/scsi/scanner/c3t4*` is used in this procedure.**

```
zonecfg:myzone:device> set match=/dev/scsi/scanner/c3t4*
```

4. **End the device specification.**

```
zonecfg:myzone:device> end
```

5. **Exit the `zonecfg` command.**

```
zonecfg:myzone> exit
```

Using the zonecfg Command to Revert or Remove a Zone Configuration

Use the zonecfg command described in zonecfg(1M) to revert a zone's configuration or to delete a zone configuration.

▼ How to Revert a Zone Configuration

While configuring a zone with the zonecfg utility, use the revert subcommand to undo a change made to the zone configuration.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **While configuring a zone called `tmp-zone`, type `info` to view your configuration:**

   ```
   zonecfg:tmp-zone> info
   ```

   The net resource segment of the configuration displays as:

   ```
   .
   .
   .
   fs:
           dir: /tmp
           special: swap
           type: tmpfs
   net:
           address: 192.168.0.1
           physical: eri0
   device
           match: /dev/pts/*
   .
   .
   .
   ```

3. **Remove the net address:**

   ```
   zonecfg:tmp-zone> remove net address=192.168.0.1
   ```

4. **Verify that the `net` entry has been removed by typing `info`.**

   ```
   .
   .
   .
   ```

```
fs:
        dir: /tmp
        special: swap
        type: tmpfs
device
        match: /dev/pts/*
.
.
.
```

5. **Type `revert`.**

```
zonecfg:tmp-zone> revert
```

6. **Respond `yes` to the following question:**

```
Are you sure you want to revert (y/[n])? y
```

7. **Type `info` to verify that the net address is once again present:**

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

▼ How to Delete a Zone Configuration

Use `zonecfg` with the `delete` subcommand to delete a zone configuration from the system.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Delete the zone configuration for the zone `a-zone` by using one of the following two methods.**

   - Use the `-F` option to force the action:

```
global# zonecfg -z a-zone delete -F
```

- Delete the zone interactively by responding y to the system prompt:

```
global# zonecfg -z a-zone delete
Are you sure you want to delete zone a-zone (y/[n])? y
```

CHAPTER 19

About Installing, Halting, and Uninstalling Non-Global Zones (Overview)

This chapter discusses zone installation on your Solaris system. This chapter also describes the two processes that manage the virtual platform and the application environment, `zoneadmd` and `zsched`. Information on halting, rebooting, and uninstalling zones is also provided.

The following topics are addressed:

Zone Installation Basics

The `zoneadm` command described in the `zoneadm`(1M) man page is the primary tool used to install and administer non-global zones. Operations using the `zoneadm` command must be run from the global zone. The following tasks can be performed using the `zoneadm` command:

- Verify a zone
- Install a zone
- Boot a zone
- Display information about a running zone
- Halt a zone
- Reboot a zone
- Uninstall a zone

For zone installation and verification procedures, see Chapter 20 and the `zoneadm`(1M) man page. For zone configuration procedures, see Chapter 18 and the `zonecfg`(1M) man page. Zone states are described in "Non-Global Zone State Model" on page 174.

If you plan to produce Solaris auditing records for zones, read "Using Solaris Auditing in Zones" on page 242 before you install non-global zones.

Zone Construction

After you have configured a non-global zone, you should verify that the zone can be installed safely on your system's configuration. You can then install the zone. The files needed for the zone's root file system are installed by the system under the zone's root path. A successfully installed zone is ready to be booted.

The method used to initially install packages in a Solaris installation is also the method used to populate a non-global zone.

The global zone must contain all the data necessary to populate a non-global zone. Populating a zone includes creating directories, copying files, and providing configuration information.

Only the information or data that was created in the global zone from packages is used to populate zone from the global zone. For more information, see the *Application Packaging Developer's Guide*.

Data from the following:

- Non-installed packages
- Patches
- Data on CDs and DVDs
- Network installation images
- Any prototype or other instance of a zone

are neither referenced nor copied when a zone is installed.

In addition, the following types of information, if present in the global zone, are not copied into a zone that is being installed:

- New or changed users in the `/etc/passwd` file
- New or changed groups in the `/etc/group` file
- Configurations for networking services such as DHCP address assignment, UUCP, or sendmail
- Configurations for network services such as name services
- New or changed crontab, printer, and mail files

- System log, message, and accounting files

If Solaris auditing is used, modifications to auditing files copied from the global zone might be required. For more information, see "Using Solaris Auditing in Zones" on page 242.

The following features cannot be configured in a non-global zone:

- Live Upgrade™ boot environments
- Solaris™ Volume Manager metadevices
- DHCP address assignment

The resources specified in the configuration file are added when the zone transitions from installed to ready. A unique zone ID is assigned by the system. File systems are mounted, network interfaces are plumbed, and devices are configured. Transitioning into the ready state prepares the virtual platform to begin running user processes. In the ready state, the `zsched` and `zoneadmd` processes are started to manage the virtual platform.

- `zsched`, a system scheduling process similar to `sched`, is used to track kernel resources associated with the zone.
- `zoneadmd` is the zones administration daemon.

A zone in the ready state does not have any user processes executing in it. The primary difference between a ready zone and running zone is that at least one process is executing in a running zone. See the `init`(1M) man page for more information.

The `zoneadmd` Daemon

The zones administration daemon, `zoneadmd`, is the primary process for managing the zone's virtual platform. The daemon is also responsible for managing zone booting and shutting down. There is one `zoneadmd` process running for each active (ready, running, or shutting down) zone on the system.

The `zoneadmd` daemon sets up the zone as specified in the zone configuration. This process includes the following actions:

- Allocating the zone ID and starting the `zsched` system process.
- Setting zone-wide resource controls.
- Preparing the zone's devices as specified in the zone configuration. For more information, see the `devfsadmd`(1M) man page.
- Plumbing virtual network interfaces.
- Mounting loopback and conventional file systems.
- Instantiating and initializing the zone console device.

Unless the `zoneadmd` daemon is already running, it is automatically started by `zoneadm` .

The Zone Scheduler

An active zone is a zone that is in the ready state, the running state, or the shutting down state. Every active zone has an associated kernel process, `zsched`. Kernel threads doing work on behalf of the zone are owned by `zsched`. The `zsched` process enables the zones subsystem to keep track of per-zone kernel threads.

The Zone Application Environment

Booting a zone is similar to booting a regular Solaris system. The `zoneadm` command is used to create the zone application environment.

Before a non-global zone is booted either for the first time or after running the `sys-unconfig -R` */zonepath*`/root` command, the internal configuration of the zone must be created. The internal configuration specifies a naming service to use, the default locale and time zone, the zone's root password, and other aspects of the application environment. The application environment is established by responding to a series of prompts that appear on the zone console, as explained in "Internal Zone Configuration" on page 220. Note that the default locale and time zone for a zone can be configured independently of the global settings.

For more information on the `sys-unconfig` command, see the `sys-unconfig`(1M) man page.

About Halting, Rebooting, and Uninstalling Zones

This section provides an overview of these procedures. Troubleshooting tips for zones that fail to halt when requested are also provided.

Halting a Zone

The `zoneadm halt` command is used to remove both the application environment and the virtual platform for a zone. The zone is then brought back to the installed state. All processes are killed, devices are unconfigured, network interfaces are unplumbed, file systems are unmounted, and the kernel data structures are destroyed.

The `halt` command does *not* run any shutdown scripts within the zone. To shut down a zone, see "How to Use `zlogin` to Shut Down a Zone" on page 228.

In the event that the system state associated with the zone cannot be destroyed, the halt operation will fail halfway. This leaves the zone in an intermediate state, somewhere between running and installed. In this state there are no active user processes or kernel threads, and none can be created. When the halt operation fails, you must manually intervene to complete the process.

The most common cause of a failure is the inability of the system to unmount all file systems. Unlike a traditional Solaris shutdown, which destroys the system state, zones must ensure that no mounts performed while booting the zone or during zone operation remain once the zone has been halted. Even though `zoneadm` makes sure there are no processes executing in the zone, the unmounting can fail if processes in the global zone have open files in the zone. Use the tools described in the `proc`(1) (see `pfiles`) and `fuser`(1M) man pages to find these processes and take appropriate action. After these processes have been dealt with, reinvoking `zoneadm halt`will complete the halt of the zone.

Rebooting a Zone

The `zoneadm reboot` command is used to reboot a zone. The zone is halted and then booted again. The zone ID will change when the zone is rebooted.

Zone `autoboot`

If you set the `autoboot` resource property to `true` in a zone's configuration, that zone is automatically booted when the global zone is booted. The default setting is `false`.

Uninstalling a Zone

The `zoneadm uninstall` command is used to uninstall all of the files under the zone's root file system. The command confirms the action with the user before proceeding, unless the `-F` (force) option is also used. The `uninstall` command should be used with caution because the action is irreversible.

CHAPTER 20

Installing, Booting, Halting, and Uninstalling Non-Global Zones (Tasks)

This chapter describes how to install and boot a non-global zone. Other tasks associated with installation such as halting, rebooting, and uninstalling zones are also addressed.

The following procedures are documented.

Installing and Booting Zones

Use the `zoneadm` command described in `zoneadm`(1M) to perform installation tasks for a non-global zone. You must be the global administrator to perform the zone installation. The examples in this chapter use the zone name and zone path established in "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197.

▼ How to Verify a Configured Zone Before It Is Installed (Optional)

You can verify a zone prior to installing it. If you skip this step, the verification is performed automatically when you install the zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Verify the configured zone by using the `-z` option with the name of the zone and the `verify` subcommand. The zone name used here is `my-zone`.**

   ```
   global# zoneadm -z my-zone verify
   ```

If an error message is displayed and the zone fails to verify, make the corrections specified in the message and try the command again.

For example, `zoneadm verify` might issue a warning if the zone path does not exist because of a typing error:

```
Warning: /export/home1/my-zone does not exist, so it cannot be verified.
When 'zoneadm install' is run, 'install' will try to create
/export/home1/my-zone, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /export/home1/my-zone is group- or other-writable
or
/export/home1/my-zone overlaps with any other installed zones.
```

If no error messages are displayed, you can install the zone.

▼ How to Install a Configured Zone

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Install the configured zone by using the `zoneadm` command with the `-z install` option. The zone name used here is `my-zone`.**

   ```
   global# zoneadm -z my-zone install
   ```

   You will see various messages as the files and directories needed for the zone's root file system are installed under the zone's root path.

   If an error message is displayed and the zone fails to install, type the following to get the zone state:

   ```
   global# zoneadm -z my-zone list -v
   ```

   - If the state is listed as configured, make the corrections specified in the message and try the `zoneadm install` command again.
   - If the state is listed as incomplete, first execute:

     ```
     global# zoneadm -z my-zone uninstall
     ```

     Then make the corrections specified in the message and try the `zoneadm install` command again.

3. **When the installation completes, use the `list` subcommand with the `-i` and the `-v` options to list the installed zones and verify the status.**

```
global# zoneadm list -iv
```

You will see a display that is similar to the following:

```
ID  NAME     STATE        PATH
 0  global   running      /
 -  my-zone  installed    /export/home/my-zone
```

Note – If a zone installation is interrupted or fails, the zone is left in the incomplete state. Use `uninstall -F` to reset the zone to the configured state.

▼ How to Transition the Installed Zone to the Ready State (Optional)

Transitioning into the ready state prepares the virtual platform to begin running user processes. Zones in the ready state do not have any user processes executing in them.

You can skip this step if you want to boot the zone and use it immediately. The transition through the ready state is performed automatically when you boot the zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Use the `zoneadm` command with the `-z` option, the name of the zone, and the `ready` subcommand. The zone name used here is `my-zone`.**

   ```
   global# zoneadm -z my-zone ready
   ```

3. **At the prompt, use the `zoneadm list` command with the `-v` option to verify the status.**

   ```
   global# zoneadm list -v
   ```

   You will see a display that is similar to the following:

   ```
   ID  NAME     STATE        PATH
    0  global   running      /
    1  my-zone  ready        /export/home/my-zone
   ```

   Note that the unique zone ID `1` has been assigned by the system.

▼ How to Boot a Zone

Booting a zone places the zone in the running state. A zone can be booted from the ready state or from the installed state. Upon boot, a zone in the installed state transparently transitions through the ready state to the running state.

Note that you should you should log in and perform the internal zone configuration before you boot the zone for the first time. This is described in "Internal Zone Configuration" on page 220.

If you plan to use an `/etc/sysidcfg` file to perform initial zone configuration, as described in "How to Use an `/etc/sysidcfg` File to Perform the Initial Zone Configuration" on page 225, create the `sysidcfg` file and place it the zone's `/etc` directory before you boot the zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Use the `zoneadm` command with the `-z` option, the name of the zone, and the `ready` subcommand. The zone name used here is `my-zone`.**

   ```
   global# zoneadm -z my-zone boot
   ```

3. **When the installation completes, use the `-v` option to verify the status.**

   ```
   global# zoneadm list -v
   ```

   You will see a display that is similar to the following:

   ```
   ID  NAME     STATE        PATH
    0  global   running      /
    1  my-zone  running      /export/home/my-zone
   ```

If you see the following message when you boot the zone:

```
# zoneadm -z my-zone boot
zoneadm: zone 'my-zone': WARNING: hme0:1: no matching subnet
    found in netmasks(4) for 192.168.0.1; using default of
    255.255.255.0.
```

Note that the message is only a warning and the command has succeeded. The message indicates that the system was unable to find the netmask to be used for the IP address specified in the zones's configuration.

To stop the warning from displaying on subsequent reboots, ensure that the correct `netmasks` databases are listed in the `/etc/nsswitch.conf` file in the global zone and that at least one of these databases contains the subnet and netmasks to be used for the zone `my-zone`.

For example, if the `/etc/inet/netmasks` file and the local NIS database are used for resolving netmasks in the global zone, then the appropriate entry in `/etc/nsswitch.conf` is:

```
netmasks: files nis
```

The subnet and corresponding netmask information for the zone `my-zone` can then be added to `/etc/inet/netmasks` for subsequent use.

For more information the `netmasks` command, see the `netmasks`(4) man page.

Where to Go From Here

To log into the zone and perform the initial internal configuration, see Chapter 22.

Halting, Rebooting, and Uninstalling Zones

For more information about these procedures, see Chapter 19.

▼ How to Halt a Zone

The halt procedure is used to remove both the application environment and the virtual platform for a zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **List the zones running on the system.**

   ```
   global# zoneadm list -v
   ```

   You will see a display that is similar to the following:

   ```
   ID  NAME     STATE        PATH
    0  global   running      /
    1  my-zone  running      /export/home/my-zone
   ```

3. **Use the `zoneadm` command with the `-z` option, the *zonename*, and the `halt` subcommand to halt the given zone. The zone name used here is `my-zone`.**

   ```
   global# zoneadm -z my-zone halt
   ```

4. **List the zones on the system again, to verify that `my-zone` has been halted.**

   ```
   global# zoneadm list -iv
   ```

You will see a display that is similar to the following:

```
ID  NAME     STATE        PATH
 0  global   running      /
 -  my-zone  installed    /export/home/my-zone
```

5. **Boot the zone if you want to restart it.**

```
global# zoneadm -z my-zone boot
```

If the zone does not halt properly, see "Halting a Zone" on page 209 for troubleshooting tips.

▼ How to Reboot a Zone

The following procedure is used to reboot a zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **List the zones running on the system.**

```
global# zoneadm list -v
```

   You will see a display that is similar to the following:

```
ID  NAME     STATE        PATH
 0  global   running      /
 1  my-zone  running      /export/home/my-zone
```

3. **Use the `zoneadm` command with the `-z reboot` option to reboot the zone. The zone name used here is `my-zone`.**

```
global# zoneadm -z my-zone reboot
```

4. **List the zones on the system again, to verify that `my-zone` has been rebooted.**

```
global# zoneadm list -v
```

   You will see a display that is similar to the following:

```
ID  NAME     STATE        PATH
 0  global   running      /
 2  my-zone  running      /export/home/my-zone
```

   Note that the zone ID for `my-zone` has changed. The zone ID generally changes after a reboot.

▼ How to Uninstall a Zone

The following procedure is used to uninstall a zone. This procedure should be used with caution since the action of removing all of the files in the zone's root file system is irreversible.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **List the zones on the system.**

   ```
   global# zoneadm list -v
   ```

   You will see a display that is similar to the following:

   ```
   ID  NAME     STATE        PATH
    0  global   running      /
    -  my-zone  installed    /export/home/my-zone
   ```

3. **Use the `zoneadm` command with the `-z uninstall` option to remove the zone. The zone name used here is `my-zone`. You can also use the `-F` option to force the action. If this option is not specified, the system will prompt for confirmation.**

   ```
   global# zoneadm -z my-zone uninstall -F
   ```

4. **List the zones on the system again, to verify that `my-zone` is no longer listed.**

   ```
   global# zoneadm list -v
   ```

   You will see a display that is similar to the following:

   ```
   ID  NAME     STATE        PATH
    0  global   running      /
   ```

Note – If a zone uninstall is interrupted, the zone is left in the incomplete state. Use the `zoneadm uninstall` comand to reset the zone to the configured state. The `uninstall` command should be used with caution because the action is irreversible.

CHAPTER 21

About Non-Global Zone Login (Overview)

This chapter discusses logging into zones from the global zone.

The following topics are covered.

For procedures and usage information, see Chapter 22.

The `zlogin` Command

After you install a zone, you must log in to the zone to complete its application environment. You might log in to the zone to perform administrative tasks as well. Unless the `-C` option is used to to connect to the zone console, logging into a zone using `zlogin` starts a new task. A task cannot span two zones.

The `zlogin` command is used to log in from the global zone to any zone that is in the running or the ready states. This command can only be used by the global administrator operating in the global zone.

See the `zlogin`(1) man page for more information.

Note – Only the `zlogin` command with the `-C` option can be used to log in to a zone that is in the ready state.

Internal Zone Configuration

When a zone is booted for the first time after installation, the zone is in an unconfigured state. The zone does not have an internal configuration for naming services, its locale and timezone have not been set, and various other configuration tasks have not been performed. Therefore, the `sysidtool` programs are run the first time a zone is booted. For more information, see the `sysidtool`(1M) man page.

Two methods are available for performing the required configuration:

- Zone console login, which initiates a series of questions from the system. Be prepared to respond to the following:
  - Language
  - Type of terminal being used
  - Host name
  - Security policy (Kerberos or standard UNIX)
  - Name service type (`None` is a valid response)
  - Name service Domain
  - Name server
  - Default time zone
  - Root password

  The procedure is described in "Performing the Initial Internal Zone Configuration" on page 223.
- An `/etc/sysidcfg` file, which you can create and place inside the zone before you boot the zone for the first time. See the `sysidcfg`(4) man page for more information.

Non-Global Zone Login

This section describes the methods you can use to log in to a zone.

Zone Console Login

Each zone maintains a virtual console, `/dev/console`. The zone console is accessed by using the `zlogin` command with the `-C` option and the *zonename*. Performing actions on console is referred to as console mode. Connections to the console persist across zone reboots.

Processes inside the zone can open and write messages to the console. If the `zlogin -C` process exits, another process can then access the console.

Other Zone Login Methods

To log into the zone with a user name, use the `zlogin` command with the `-l` option, the user name, and the *zonename*.

To log in as user `root`, use `zlogin` without options.

To log in remotely, see "Remote Login" on page 222.

If a login problem occurs and you are unable to use the standard login methods, you can use the `zlogin` command with the `-S` (safe) option. For more information, see "Failsafe Mode" on page 222.

Interactive and Non-Interactive Modes

Two other methods for accessing the zone and for executing commands inside the zone are also provided by the `zlogin` command. These methods are interactive mode and non-interactive mode.

Interactive Mode

In interactive mode, a new pseudo-terminal is allocated for use inside the zone. Unlike console mode, in which exclusive access to the console device is granted, an arbitrary number of `zlogin` sessions can be open at any time in interactive mode. Interactive mode is activated when you do not include a command to be issued. Programs that require a terminal device, such as an editor, operate correctly in this mode.

Non-Interactive Mode

Non-interactive mode is used to run shell-scripts which administer the zone. Non-interactive mode does not allocate a new pseudo-terminal. Non-interactive mode is enabled when you supply a command to be run inside the zone.

Failsafe Mode

If a login problem occurs and you are unable to use either the `zlogin` command or the `zlogin` command with the `-C` option to access the zone, an alternative is provided. You can enter the zone by using the `zlogin` command with the `-S` (safe) option. This mode should only be used to recover a damaged zone when other forms of login are not succeeding. In this minimal environment, it might be possible to diagnose why the zone login is failing.

Remote Login

The ability to remotely log in to a zone is dependent on the selection of network services that you establish. By default, logins through `rlogin`, `ssh`, and `telnet` function normally. For more information on these commands, see `rlogin`(1), `ssh`(1), and `telnet`(1).

CHAPTER 22

Non-Global Zone Login (Tasks)

This chapter provides procedures for completing the configuration of an installed zone. The chapter also describes logging into zones from the global zone.

The following topics are covered:

For an introduction to the zone login process, see Chapter 21.

Performing the Initial Internal Zone Configuration

When a zone is booted for the first time after installation, it must be configured as described in "Internal Zone Configuration" on page 220.

How to Log In to the Zone Console to Perform the Internal Zone Configuration

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Use the `zlogin` command with the `-C` option. The zone name used here is `my-zone`**

```
global# zlogin -C my-zone
```

3. **The first time you log in to the console, you are prompted to answer a series of questions. Your screen will look similar to this:**

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2002 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Select a Language

     0. English
     1. French

Please make a choice (0 - 1), or press h or ? for help:

Select a Locale

      0. English (C - 7-bit ASCII)
      1. Belgium-Flemish (ISO8859-1)
      2. Belgium-Flemish (ISO8859-15 - Euro)
      3. Great Britain (ISO8859-1)
      4. Great Britain (ISO8859-15 - Euro)
      5. Ireland (ISO8859-1)
      6. Ireland (ISO8859-15 - Euro)
      7. Netherlands (ISO8859-1)
      8. Netherlands (ISO8859-15 - Euro)
      9. Go Back to Previous Screen

Please make a choice (0 - 9), or press h or ? for help:

What type of terminal are you using?
      1) ANSI Standard CRT
      2) DEC VT52
      3) DEC VT100
      4) Heathkit 19
      5) Lear Siegler ADM31
      6) PC Console
      7) Sun Command Tool
      8) Sun Workstation
      9) Televideo 910
      10) Televideo 925
      11) Wyse Model 50
      12) X Terminal Emulator (xterms)
      13) CDE Terminal Emulator (dtterm)
      14) Other
Type the number of your choice and press Return:
.
.
.
```

For the full list of questions you must answer, see “Internal Zone Configuration” on page 220.

If you booted this zone before you logged in, you might have missed the initial prompt for configuration information. If you see the following system message at zone login instead of a prompt:

```
[connected to zone zonename console]
```

Press Return to display the prompt again.

If you enter an incorrect response and try to restart the configuration, you might experience difficulty when you attempt the process again. This occurs because the `sysidtools` can store your previous responses.

If this happens, use one of the following workarounds from the global zone to restart the configuration process.

- Use the `sys-unconfig` command with the `-R` option and the path to the zone's root to undo the zone's configuration:

  ```
  global# sys-unconfig -R /zonepath/root
  ```

- Use the `zoneadm` command with the `-z` option and the `boot` subcommand to boot the zone. Then use the `zlogin` command with the `-S` option and the `sys-unconfig` command to undo the zone's configuration:

  ```
  global# zoneadm -z zonename bootglobal# zoneadm -S zonename sys-unconfig
  ```

▼ How to Use an `/etc/sysidcfg` File to Perform the Initial Zone Configuration

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Create the `sysidcfg` file and place it in the non-global zone's `/etc` directory. The file will look similar to the following:**

   ```
   system_locale=C
   terminal=dtterm
   network_interface=primary {
               hostname=my-zone
   }
   security_policy=NONE
   name_service=NIS {
               domain_name=special.example.com
               name_server=bird(192.168.112.3)
   }
   timezone=US/Central
   root_password=m4qtoWN
   ```

3. **Boot the zone.**

Using the `zlogin` Command to Log Into a Zone

Use the `zlogin` command to log in from the global zone to any zone that is running or in the ready state. See the `zlogin`(1) man page for more information.

▼ How to Log in to the Zone Console

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Use the `zlogin` command with the `-C` option. The zone name used here is `my-zone`**

   ```
   global# zlogin -C my-zone
   ```

   Note – If you start the `zlogin` session immediately after issuing the `zoneadm boot` command, boot-up messages from the zone will display:

   ```
   SunOS Release 5.10 Version Generic 64-bit
   Copyright 1983-2004 Sun Microsystems, Inc. All rights reserved.
   Use is subject to license terms.
   starting rpc services: rpcbind done.
   syslog service starting.
   The system is ready.
   ```

3. **When the zone console displays, log in as `root` and press Return, and type the root password when prompted.**

   ```
   my-zone console login: root
   Password:
   ```

▼ How to Use Interactive Mode to Access a Zone

In interactive mode, a new pseudo-terminal is allocated for use inside the zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **At the global zone prompt, type** *tty*.

```
global# tty
```

The pseudo-terminal device name will display:

```
/dev/pts/3
```

3. **From the global zone, log into the zone. This procedure uses the zone name `my-zone`.**

```
global# zlogin my-zone
```

Information similar to the following will display:

```
[Connected to zone 'my-zone' pts/2]
Last login: Wed Jul  3 16:25:00 on console
Sun Microsystems Inc. SunOS 5.10 Generic July 2003
```

4. **At the `my-zone` prompt, type** *tty*.

```
my-zone# tty
```

The name of the pseudo-terminal device displays:

```
/dev/pts/2
my-zone#
```

5. **Type exit to close the connection.**

You will see a messages such as:

```
[Connection to zone 'my-zone' pts/2 closed]
```

▼ How to Use Non-Interactive Mode to Access a Zone

Non-interactive mode is enabled when the user supplies a command to be run inside the zone. Non-interactive mode does not allocate a new pseudo-terminal.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **From the global zone, log into the `my-zone` zone, supplying a command name. The command `zonename` is used here.**

```
global# zlogin my-zone zonename
```

You will see the following output:

```
my-zone
```

▼ How to Use Failsafe Mode to Enter a Zone

When connection to the zone is denied, the `zlogin` command can be used with the `-S` option to enter a minimal environment in the zone.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **From the global zone, use the `zlogin` command with the `-S` option to access the zone. The zone name used here is `my-zone`.**

   ```
   global# zlogin -S my-zone
   ```

▼ How to Use `zlogin` to Shut Down a Zone

This procedure is used to cleanly shut down a zone. To halt a zone without running shutdown scripts, see "How to Halt a Zone" on page 215.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Log into the zone to be shut down. The zone name used here is `my-zone`. Specify `shutdown` as the name of the utility.**

   ```
   global# zlogin my-zone shutdown
   ```

   Your site might have its own shutdown script, tailored for your specific environment.

Note – Running `init 0` in the global zone to cleanly shut down a Solaris system also runs `init 0` in each of the non-global zones on the system. Note that `init 0` does not warn local and remote users to log off before the system is taken down.

Printing the Name of the Current Zone

The `zonename` command described in the `zonename`(1) man page prints the name of the current zone. The following example shows the result when `zonename` is used in the global zone.

```
# zonename
global
```

CHAPTER 23

Zone Administration (Overview)

This chapter covers general administration information.

Global Zone Visibility and Access

The global zone acts as both the default zone for the system and as a zone for system-wide administrative control. There are administrative issues asssociated with this dual role. Since applications within the zone have access to processes and other system objects in other zones, the effect of administrative actions can be wider than expected. For example, service shutdown scripts often use `pkill` to signal processes of a given name to exit. When such a script is run from the global zone, all such processes in the system will be signaled, regardless of zone.

The system-wide scope is often needed. For example, to monitor system-wide resource usage, you must view process statistics for the whole system. A view of just global zone activity would miss relevant information from other zones in the system that might be sharing some or all of the system resources. Such a view is particularly important when system resources such as CPU are not strictly partitioned using resource management facilities.

Thus, processes in the global zone can observe processes and other objects in non-global zones. This allows such processes to have system-wide observability. The ability to control or send signals to processes in other zones is restricted by the

privilege PRIV_PROC_ZONE. The privilege is similar to PRIV_PROC_OWNER because the privilege allows processes to override the restrictions placed on unprivileged processes. In this case, the restriction is that unprivileged processes in the global zone cannot signal or control processes in other zones. This is true even when the user IDs of the processes match or the acting process has the PRIV_PROC_OWNER privilege. The PRIV_PROC_ZONE privilege can be removed from otherwise privileged processes to restrict actions to the global zone.

For information about matching processes by using a zoneidlist, see the pgrep(1) and pkill(1) man pages.

Process ID Visibility in Zones

Only processes in the same zone will be visible through system call interfaces that take process IDs, such as the kill and priocntl commands. For information, see the kill(1) and the priocntl(1) man pages.

System Observability in Zones

A -o option, which specifies output format, has been added to the ps command. This option allows you to print the zone ID of a process or the name of the zone in which the process is running. For more information, see the ps(1) man page.

A -z *zonename* option has been added to the following Solaris utilities. You can use this option to filter the information to include only the zone or zones specified.

- ipcs (see the ipcs(1) man page)
- pgrep (see the pgrep(1) man page)
- ptree (see the ptree(1) man page)
- prstat (see the prstat(1M) man page)

Non-Global Zone Node Name

The node name in /etc/nodename returned by uname -n can be set by the zone administrator. The node name must be unique.

File Systems and Non-Global Zones

This section provides information on file system issues in a zones environment. Each zone has its own section of the file system hierarchy, rooted at a directory known as the zone `root`. Processes in the zone can access only files in the part of the hierarchy that is located under the zone root. The `chroot` utility can be used in a zone, but only to restrict the process to a root path within the zone. For more information on `chroot`, see `chroot`(1M).

The `-o nosuid` Option

The `-o nosuid` option to the `mount` utility provides the following functionality:

- Processes from a `setuid` binary located on a file system that is mounted using the `nosetuid` option do not run with the privileges of the `setuid` binary. The processes run with the privileges of the user that executes the binary.

  For example, if a user executes a `setuid` binary that is owned by `root`, the processes run with the privileges of the user.
- Opening device-special entries in the file system is not allowed. This behavior is equivalent to that of specifying the `nodevices` option.

This filesystem-specific option is available to all Solaris file systems that have `mount` utilities. These file systems are listed in "About Mounting File Systems in Zones" on page 233. Mounting capabilities are also described. For more information on the `-o nosuid` option, see "Accessing Network File Systems (Reference)" in *System Administration Guide: Network Services*.

About Mounting File Systems in Zones

Options for mounting file systems in non-global zones are described in the following table. Procedures for these mounting alternatives are provided in "Using the `zonecfg` Command to Configure, Verify, and Commit a Zone" on page 197 and "Mounting File Systems in Running Non-Global Zones" on page 249.

File System	Mounting Options in a Non-Global Zone
AutoFS	Cannot be mounted using `zonecfg`, cannot be manually mounted from the global zone into a non-global zone. Can be mounted from within the zone.

File System	Mounting Options in a Non-Global Zone
CacheFS™	*Cannot be used in a zone.* Cannot be mounted using `zonecfg`, cannot be manually mounted from the global zone into a non-global zone, cannot be mounted from within the zone.
FDFS	Can be mounted using `zonecfg`, can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
HSFS	Cannot be mounted using `zonecfg` at this time. Can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
LOFS	Can be mounted using `zonecfg`, can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
MNTFS	Cannot be mounted using `zonecfg`, cannot be manually mounted from the global zone into a non-global zone. Can be mounted from within the zone.
NFS	Cannot be mounted using `zonecfg`. V2 and V3, which are the versions currently supported in zones, can be mounted from within the zone.
PCFS	Cannot be specified using `zonecfg` at this time. Can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
PROCFS	Cannot be mounted using `zonecfg`, cannot be manually mounted from the global zone into a non-global zone. Can be mounted from within the zone.
TMPFS	Can be mounted using `zonecfg`, can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
UDFS	Cannot be specified using `zonecfg` at this time. Can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.

File System	Mounting Options in a Non-Global Zone
UFS	Cannot be specified using `zonecfg` at this time. Can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.
XMEMFS	Can be mounted using `zonecfg`, can be manually mounted from the global zone into a non-global zone, can be mounted from within the zone.

For more information, see "How to Configure the Zone" on page 197, "Mounting File Systems in Running Non-Global Zones" on page 249, and the `mount`(1M) man page.

Security Restrictions and File System Behavior

There are security restrictions on mounting certain file systems from within a zone. Other file systems exhibit special behavior when mounted in a zone. The list of modified file systems follows.

AutoFS

Autofs is a client-side service that automatically mounts the appropriate file system. When a client attempts to access a file system that is not presently mounted, the autofs file system intercepts the request and calls `automountd` to mount the requested directory. AutoFS mounts established within a zone are local to that zone. The mounts cannot be accessed from other zones, including the global zone. The mounts are removed when the zone is halted or rebooted. For more information on AutoFS, see "How Autofs Works" in *System Administration Guide: Resource Management and Network Services*.

Each zone runs its own copy of `automountd`. The auto maps and timeouts are controlled by the zone administrator. You cannot trigger a mount in another zone by crossing an AutoFS mount point for a non-global zone from the global zone.

Certain AutoFS mounts are created in the kernel when another mount is triggered. Such mounts can not be removed by using the regular `umount` interface since they must be mounted or unmounted as a group. Note that this functionality is provided for zone shutdown.

MNTFS

MNTFS is a virtual file system that provides read-only access to the table of mounted file systems for the local

	system. The set of file systems visible by using `mnttab` from within a non-global zone is the set of file systems mounted in the zone, plus an entry for root (`/`) . Mount points with a special device that is not accessible from within the zone, such as `/dev/rdsk/c0t0d0s0`, have their special device set to the same as the mount point. All mounts in the system are visible from the global zone's `/etc/mnttab` table. For more information on MNTFS, see "Mounting and Unmounting File Systems (Tasks)" in *System Administration Guide: Devices and File Systems*.
NFS	NFS mounts established within a zone are local to that zone. The mounts cannot be accessed from other zones, including the global zone. The mounts are removed when the zone is halted or rebooted. As documented in the `mount_nfs`(1M) man page, an NFS server should not attempt to mount its own file systems. Thus, a zone cannot NFS mount a file system exported by the global zone. Zones cannot be NFS servers. NFS mounts from within a zone behave as if mounted with the `nodevices` option.
PROCFS	The `/proc` file system, or PROCFS, provides process visibility and access restrictions as well as information about the zone association of processes. Only processes in the same zone will be visible through `/proc`. Processes in the global zone can observe processes and other objects in non-global zones. This allows such processes to have system-wide observability. `procfs` mounts from within a zone behave as if mounted with the `nodevices` option. For more information on `procfs`, see the `proc`(4) man page.
LOFS	The scope of what can be mounted through lofs is limited to the portion of the file system visible to the zone. Hence, there are no restrictions on LOFS mounts in a zone.
UFS, UDFS, HSFS, PCFS	These disk-based file systems cannot be specified for automatic mounting through `zonecfg` at this time. The following alternatives are possible: ■ The global zone administrator can manually mount these file systems from the global zone into a non-global zone.

- The device can be imported into the zone by using the `zonecfg` command. With access to the disk, the zone administrator can create a new file system on the disk. The administrator can then either mount this file system manually, or put it in `/etc/vfstab` so it will be mounted on zone boot.

Procedures for both of these options are provided in "Mounting File Systems in Running Non-Global Zones" on page 249.

Traversing File Systems

A zone's file system namespace is a subset of that accessible from the global zone. Unprivileged processes in the global zone are prevented from traversing a non-global zone's file system hierarchy by specifying that the zone root's parent directory is owned, readable, writable, and executable by root only, and by restricting access to directories exported by `/proc`.

Note that attempting to access AutoFS nodes mounted for another zone will fail. The global administrator must not have auto maps that descend into other zones.

Networking in Non-Global Zones

In a zones environment, the zones can communicate with each other over the network. The zones all have separate bindings, or connections, and the zones can all run their own server daemons. These daemons can listen on the same port numbers without any conflict. The IP stack resolves conflicts by considering the IP addresses for incoming connections. The IP addresses identify the zone.

Zone Partitioning

The IP stack in a system supporting zones implements the separation of network traffic between zones. Applications that receive IP traffic can only receive traffic from the same zone.

Each logical interface on the system belongs to a specific zone, the global zone by default. Logical network interfaces assigned to zones though the `zonecfg` utility are used to communicate over the network. Each stream and connection belongs to the zone of the process that opened it.

Bindings between upper-layer streams and logical interfaces are restricted. A stream can only establish bindings to logical interfaces in the same zone. Likewise, packets from a logical interface can only be passed to upper-layer streams in the same zone as the logical interface.

Each zone has its own set of binds. Each zone can be running the same application listening on the same port number without binds failing because the address is already in use. Each zone can run its own version of the following services:

- Internet services daemon with a full configuration file (see `inetd`(1M))
- `sendmail` (see `sendmail`(1M))
- `apache` (see `apache`(1M))

Zones other than the global zone have restricted access to the network. The standard TCP and UDP socket interfaces are available, but `SOCK_RAW` socket interfaces are restricted to Internet Control Message Protocol (ICMP). ICMP is necessary for detecting and reporting network error conditions or using the `ping` command.

Network Interfaces

Each non-global zone that requires network connectivity has one or more dedicated IP addresses. These addresses are associated with logical network interfaces that can be placed in a zone by using the `ifconfig` command. Zone interfaces configured by `zonecfg` will automatically be plumbed and placed in the zone when it is booted. The `ifconfig` command can be used to add or remove logical interfaces when the zone is running. Only the global zone administrator can modify the interface configuration and the network routes.

Within a non-global zone, only that zone's interfaces will be visible to `ifconfig`.

For more information, see the `ifconfig`(1M) and the `if_tcp`(7P) man pages.

Device Use in Non-Global Zones

The set of devices available within a zone is restricted to prevent a process in one zone from interfering with processes running in other zones. For example, a process in a zone cannot modify kernel memory or modify the contents of the root disk. Thus, by default, only certain pseudo devices considered safe for use in a zone are available. Additional devices can be made available within specific zones be using the `zonecfg` utility.

/dev and /devices Namespace

The devfs file system described in the devfs(7FS) man page is used by Solaris to manage /devices. Each element in this namespace represents the physical path to a hardware device, pseudo device, or nexus device. The namespace is a reflection of the device tree. As such, the file system is populated by a hierarchy of directories and device special files.

The /dev file hierarchy, which is today part of the / (root) file system, consists of symbolic links, or logical paths, to the physical paths present in /devices. Applications reference the logical path to a device presented in /dev. The /dev file system is loopback-mounted into the zone using a read-only mount.

The /dev file hierarchy is managed by a system comprised of the components in the following list:

- devfsadm (seedevfsadm(1M))
- sysevent d (seesysevent d(1M))
- libdevinfo device information library (seelibdevinfo(3LIB))
- devinfo driver (see devinfo(7D))
- The Reconfiguration Coordination Manager (RCM) (see "Reconfiguration Coordination Manager (RCM) Script Overview" in *System Administration Guide: Devices and File Systems*).

Caution – Subsystems that rely on /devices pathnames are not able to run in non-global zones until /dev pathnames are established.

Exclusive-Use Devices

You might have devices that you want to assign to specific zones. Allowing unprivileged users to access specific devices could permit those devices to be used to cause system panic, bus resets, or other adverse effects. Before making such assignments, consider the following issues:

- Before assigning a SCSI tape device to a specific zone, consult the sgen(7D) man page.
- Placing a physical device into more than one zone risks creating a covert channel between zones. Global zone applications using such a device risk compromise or data corruption by a non-global zone.

Device Driver Administration

In a non-global zone, you can use the `modinfo` command described in the `modinfo`(1M) man page to examine the list of loaded kernel modules.

Most operations concerning kernel, device, and platform management will not work inside a non-global zone because modifying platform hardware configurations violates the zone security model. These operations include the following:

- Adding and removing drivers
- Explicitly loading and unloading kernel modules
- Initiating dynamic reconfiguration (DR) operations
- Using facilities that affect the state of the physical platform

Utilities That Do Not Work in Zones

The utilities in the following list do not work in a zone because they rely on devices that are not normally available:

- EEPROM (see `eeprom`(1M))
- `prtconf` (see`prtconf`(1M))
- `prtdiag` (see `prtdiag`(1M))

Resource Controls Used in Non-Global Zones

For additional information on resource management features used in zones, also refer to the chapter in Part 1 of this guide that describes the feature.

Any of the resource controls and attributes described in the resource management chapters can be set in the non-global zone `/etc/project` file, NIS map, or LDAP directory service. The settings for a given non-global zone affect only that zone. A project running autonomously in different zones can have controls set individually in each zone. For example, Project A in the global zone can be set `project.cpu-shares=10` while Project A in a non-global zone can be set `project.cpu-shares=5`. You could have several instances of `rcapd` running, with each instance operating only on its zone.

The resource controls and attributes used in a zone to control projects, tasks, and processes within that zone are subject to the additional requirements regarding pools and the zone-wide resource control.

A “one zone, one pool” rule applies to non-global zones. Processes in the global zone, however, can be bound by a sufficiently privileged process to any pool. The resource controller `poold` only runs in the global zone, where there is more than one pool for it to operate on. The `poolstat` utility run in a non-global zone displays only information about the pool to which the zone is boundassociated with the zone. The `pooladm` command run without arguments in a non-global zone displays only information about the pool associated with the zone.

Zone-wide resource controls do not take effect if set in the `project` file. FSS CPU shares are controlled by the global zone through the zone-wide resource control `zone.cpu-shares`. The `project.cpu-shares` control set for each zone further subdivides the shares set through the zone-wide control. A zone-wide resource control is set through the `zonecfg` utility.

Extended Accounting in a Zones Environment

The extended accounting subsystem collects and reports information for the entire system (including non-global zones) when run in the global zone. The global administrator can also determine resource consumption on a per-zone basis.

The extended accounting subsystem permits different accounting settings and files on a per-zone basis for process- and task-based accounting. The `exacct` records can be tagged with the zone name `EXD PROC ZONENAME` for processes and the zone name `EXD TASK ZONENAME` for tasks. Accounting records are written to the global zone’s accounting files as well as the per-zone accounting files. The `EXD TASK HOSTNAME`, `EXD PROC HOSTNAME`, and `EXD HOSTNAME` records contain the `uname -n` value for the zone in which the process or task executed instead of the global zone’s node name.

For information on IPQoS flow accounting, see “Using Flow Accounting and Statistics Gathering (Tasks)” in *IPQoS Administration Guide*.

Privileges in a Non-Global Zone

Processes are restricted to a subset of privileges. Privilege restriction prevents a zone from performing operations that might affect other zones. The set of privileges limits the capabilities of privileged users within the zone. To display the list of privileges available within a zone, use the `ppriv` utility.

For usage examples, see “Using the ppriv Utility” on page 245. For more information about privileges, see the ppriv(1) man page and *System Administration Guide: Security Services*.

Using IP Security Architecture in Zones

The Internet Protocol Security Architecture (IPsec), which provides IP datagram protection, is described in “IP Security Architecture (Overview)” in *System Administration Guide: IP Services*. The Internet Key Exchange (IKE) protocol is used to manage the required keying material for authentication and encryption automatically.

IPsec can be used in the global zone. However, IPsec in a non-global zone cannot use IKE. Therefore, you must manage the IPsec keys and policy for the non-global zones by running the ipseckey and ipsecconf commands from the global zone. Use the source address that corresponds to the non-global zone that you are configuring.

For more information, see the ipsecconf(1M) and the ipseckey(1M) man pages.

Using Solaris Auditing in Zones

Solaris auditing is described in “Solaris Auditing (Overview)” in *System Administration Guide: Security Services*.

An audit record describes an event, such as a logging in to a system or writing to a file. The record is composed of tokens, which are sets of audit data. By using the zonename token, you can configure Solaris auditing to identify audit events by zone. Use of the zonename token allows you to produce the following information:

- Audit records that are marked with the name of the zone that generated the record
- An audit log for a specific zone that the global administrator can make available to the non-global zone administrator

Configuring Audit in the Global Zone

Solaris audit trails are configured in the global zone. Audit policy is set in the global zone and applies to processes in all zones. The audit records can be marked with the name of the zone in which the event occurred. To include zone names in audit records, you must edit the `/etc/security/audit_startup` file before you install any non-global zones. The zone name selection is case-sensitive.

To configure auditing in the global zone to include all zone audit records, add this line to the `/etc/security/audit_startup` file:

```
/usr/sbin/auditconfig -setpolicy +zonename
```

As the global administrator in the global zone, execute the `auditconfig` utility:

```
global# auditconfig -setpolicy +zonename
```

For additional information, see the `audit_startup`(1M) and `auditconfig`(1M) man pages and "Configuring Audit Files (Task Map)" in *System Administration Guide: Security Services*.

Configuring User Audit Characteristics in the Non-Global Zone

When a non-global zone is installed, the `audit_control` file and the `audit_user` file in the global zone are copied to the zone's `/etc/security` directory. These files might require modification to reflect the zone's audit needs.

For example, each zone can be configured to audit some users differently from others. To apply different per-user preselection criteria, both the `audit_control` and the `audit_user` files must be edited. The `audit_user` file in the non-global zone might also require revisions to reflect the user base for the zone if necessary. Because each zone can be configured differently with regard to auditing users, it is possible for the `audit_user` file to be empty.

For additional information, see the `audit_control`(4) and `audit_user`(4) man pages.

Providing Audit Records for a Specific Non-Global Zone

By including the `zonename` token as described in "Configuring Audit in the Global Zone" on page 243, Solaris audit records can be categorized by zone. Records from different zones can then be collected by using the `auditreduce` command to create logs for a specific zone.

For more information, see the `audit_startup`(1M) and the `auditreduce`(1M) man pages.

Core Files in Zones

The `coreadm` command is used to specify the name and location of core files produced by abnormally-terminating processes. Core file paths that include the *zonename* of the zone in which the process executed can be produced by specifying the `%z` variable. The path name is relative to a zone's root directory.

For more information, see the `coreadm`(1M) and `core`(4) man pages.

CHAPTER 24

Zone Administration (Tasks)

This chapter covers general administration tasks and provides usage examples.

Using the ppriv Utility

Use the ppriv utility to display the zone's privileges.

▼ How to List the Non-Global Zone's Privilege Set

Use the ppriv utility with the -i option to list the zone's privileges.

1. **Log into the non-global zone. This example uses a zone named *my-zone*.**
2. **At the prompt, type `ppriv -i` to report the set of privileges available in the zone.**

   ```
   my-zone# ppriv -z
   ```

   You will see a display similar to the following:

   ```
   file_chown
   file_chown_self
   file_dac_execute
   file_dac_read
   file_dac_search
   file_dac_write
   file_link_any
   file_owner
   file_setdac
   ```

```
file_setid
ipc_dac_read
ipc_dac_write
ipc_owner
net_icmpaccess
net_privaddr
proc_chroot
proc_audit
proc_exec
proc_fork
proc_owner
proc_session
proc_setid
proc_taskid
sys_acct
sys_admin
sys_mount
sys_nfs
sys_resource
```

▼ How to List the Non-Global Zone's Privilege Set With Verbose Output

Use the `ppriv` utility with the `-i` and `-v`options to list the zone's privileges.

1. **Log into the non-global zone. This example uses a zone named *my-zone*.**
2. **At the prompt, type `ppriv -i -v`to report the set of privileges available in the zone, with a description of each privilege.**

   ```
   my-zone# ppriv -z
   ```

   You will see a display similar to the following:

   ```
   file_chown
           Allows a process to change a file's owner user ID.
           Allows a process to change a file's group ID to one other than
           the process' effective group ID or one of the process'
           supplemental group IDs.
   file_chown_self
           Allows a process to give away its files; a process with this
           privilege will run as if {_POSIX_CHOWN_RESTRICTED} is not
           in effect.
   file_dac_execute
           Allows a process to execute an executable file whose permission
           bits or ACL do not allow the process execute permission.
   file_dac_read
           Allows a process to read a file or directory whose permission
           bits or ACL do not allow the process read permission.
   file_dac_search
           Allows a process to search a directory whose permission bits or
           ACL do not allow the process search permission.
   ```

```
file_dac_write
        Allows a process to write a file or directory whose permission
        bits or ACL do not allow the process write permission.
        In order to write files owned by uid 0 in the absence of an
        effective uid of 0 ALL privileges are required.
file_link_any
        Allows a process to create hardlinks to files owned by a uid
        different from the process' effective uid.
file_owner
        Allows a process which is not the owner of a file to modify
        that file's access and modification times.
        Allows a process which is not the owner of a directory to
        modify that directory's access and modification times.
        Allows a process which is not the owner of a file or directory
        to remove or rename a file or directory whose parent directory
        has the ''save text image after execution'' (sticky) bit set.
        Allows a process which is not the owner of a file to mount a
        ''namefs'' upon that file.
        (Does not apply to setting access permission bits or ACLs.)
file_setdac
        Allows a process which is not the owner of a file or directory
        to modify that file's or directory's permission bits or ACL
        except for the set-uid and set-gid bits.
file_setid
        Allows a process to change the ownership of a file or write to
        a file without the set-user-ID and set-group-ID bits being
        cleared.
        Allows a process to set the set-group-ID bit on a file or
        directory whose group is not the process' effective group or
        one of the process' supplemental groups.
        Allows a process to set the set-user-ID bit on a file with
        different ownership in the presence of PRIV_FILE_SETDAC.
        Additional restrictions apply when creating or modifying a
        set-uid 0 file.
ipc_dac_read
        Allows a process to read a System V IPC
        Message Queue, Semaphore Set, or Shared Memory Segment whose
        permission bits do not allow the process read permission.
        Allows a process to read remote shared memory whose
        permission bits do not allow the process read permission.
ipc_dac_write
        Allows a process to write a System V IPC
        Message Queue, Semaphore Set, or Shared Memory Segment whose
        permission bits do not allow the process write permission.
        Allows a process to read remote shared memory whose
        permission bits do not allow the process write permission.
        Additional restrictions apply if the owner of the object has uid 0
        and the effective uid of the current process is not 0.
ipc_owner
        Allows a process which is not the owner of a System
        V IPC Message Queue, Semaphore Set, or Shared Memory Segment to
        remove, change ownership of, or change permission bits of the
        Message Queue, Semaphore Set, or Shared Memory Segment.
        Additional restrictions apply if the owner of the object has uid 0
        and the effective uid of the current process is not 0.
```

```
net_icmpaccess
        Allows a process to send and receive ICMP packets.
net_privaddr
        Allows a process to bind to a privileged port
        number. The privilege port numbers are 1-1023 (the traditional
        UNIX privileged ports) as well as those ports marked as
        "udp/tcp_extra_priv_ports" with the exception of the ports
        reserved for use by NFS.
proc_chroot
        Allows a process to change its root directory.
proc_audit
        Allows a process to generate audit records.
        Allows a process to get its own audit pre-selection information.
proc_exec
        Allows a process to call execve().
proc_fork
        Allows a process to call fork1()/forkall()/vfork()
proc_owner
        Allows a process to send signals to other processes, inspect
        and modify process state to other processes regardless of
        ownership.  When modifying another process, additional
        restrictions apply:  the effective privilege set of the
        attaching process must be a superset of the target process'
        effective, permitted and inheritable sets; the limit set must
        be a superset of the target's limit set; if the target process
        has any uid set to 0 all privilege must be asserted unless the
        effective uid is 0.
        Allows a process to bind arbitrary processes to CPUs.
proc_session
        Allows a process to send signals or trace processes outside its
        session.
proc_setid
        Allows a process to set its uids at will.
        Assuming uid 0 requires all privileges to be asserted.
proc_taskid
        Allows a process to assign a new task ID to the calling process.
sys_acct
        Allows a process to enable and disable and manage accounting through
        acct(2), getacct(2), putacct(2) and wracct(2).
sys_admin
        Allows a process to perform system administration tasks such
        as setting node and domain name and specifying nscd and coreadm
        settings.
sys_mount
        Allows a process to mount and unmount filesystems which would
        otherwise be restricted (i.e., most filesystems except namefs)
        Allows a process to add and remove swap devices.
sys_nfs
        Allows a process to perform Sun private NFS specific system calls.
        Allows a process to bind to ports reserved by NFS: ports 2049 (nfs)
        and port 4045 (lockd).
sys_resource
        Allows a process to modify the resource limits with
        by setrlimit(2) and setrctl(2) without restriction.
        Allows a process to exceed the per-user maximum number of
```

```
processes.
Allows a process to extend or create files on a filesystem that
has less than minfree space in reserve.
```

Mounting File Systems in Running Non-Global Zones

You can mount file systems in a running non-global zone. The following procedures are covered.

- As the global administrator in the global zone, you can import raw and block devices into a non-global zone. After the devices are imported, the zone administrator has access to the disk. The zone administrator can then create a new file system on the disk and perform one of the following actions:
  - Mount the file system manually
  - Place the file system in `/etc/vfstab` so that it will be mounted on zone boot
- As the global administrator, you can also mount a file system from the global zone into the non-global zone.

▼ How to Import Raw and Block Devices Using `zonecfg`

This procedure uses the `lofi` command. For information about `lofi`, see the `lofiadm`(1M) and `lofi`(7D) man pages.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Change directories to /usr/tmp:**

   ```
   global# cd /usr/tmp
   ```

3. **Construct a new UFS file system:**

   ```
   global# mkfile 10m fsfile
   ```

4. **Attach the file as a block device:**

   ```
   global# lofiadm -a `pwd`/fsfile/dev/lofi/1
   ```

   You will also get the required character device.

5. **Import the devices into the zone `my-zone`:**

```
global# zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/rlofi/1
zonecfg:my-zone:device> end
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/lofi/1
zonecfg:my-zone:device> end
```

6. **Reboot the zone:**

```
global# zoneadm -z my-zone boot
```

7. **Log in to the zone and verify that the devices were successfully imported:**

```
my-zone# ls  -l /dev/*lofi/*
```

You will see a display that is similar to the following:

```
brw-------   1 root     sys      147,  1 Jan  7 11:26 /dev/lofi/1
crw-------   1 root     sys      147,  1 Jan  7 11:26 /dev/rlofi/1
```

▼ How to Mount the File System Manually

You must be the zone administrator and have the Zone Management profile to perform this procedure. This procedure uses the `newfs` command, which is described in the `newfs`(1M) man page.

1. **Become superuser, or have the Zone Management rights profile in your list of profiles.**

2. **In the zone `my-zone`, create a new file system on the disk by typing:**

```
my-zone# newfs /dev/lofi/1
```

3. **Respond `yes` at the prompt:**

```
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
```

You will see a display that is similar to the following:

```
/dev/rlofi/1:   20468 sectors in 34 cylinders of 1 tracks, 602 sectors
        10.0MB in 3 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9664, 19296,
```

4. **Check the file system for errors by typing:**

```
my-zone# fsck -F ufs /dev/rlofi/1
```

You will see a display that is similar to the following:

```
** /dev/rlofi/1
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
```

```
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 9320 free (16 frags, 1163 blocks, 0.2% fragmentation)
```

5. **Mount the file system:**

   ```
   my-zone# mount -F ufs /dev/lofi/1 /mnt
   ```

6. **Verify the mount:**

   ```
   my-zone# grep /mnt /etc/mnttab
   ```

   You will see a display similar to the following:

   ```
   /dev/lofi/1     /mnt    ufs
   rw,suid,intr,largefiles,xattr,onerror=panic,zone=foo,dev=24c0001
   1073503869
   ```

▼ How to Place a File System in /etc/vfstab to be Mounted on Zone Boot

This procedure is used to mount the block device `/dev/lofi/1` on the file system path `/mnt`. The block device contains a UFS filesystem. The following options are used:

- `logging` is used as the mount option.
- `yes` tells the system to automatically mount the file system on boot.
- `/dev/rlofi/1` is the character (or raw) device. The `fsck` command is run on the raw device if required.

1. **Become superuser, or have the Zone Management rights profile in your list of profiles.**

2. **In the zone `my-zone`, add the following line to `/etc/vfstab`:**

   ```
   /dev/lofi/1 /dev/rlofi/1  /mnt   ufs  2  yes logging
   ```

▼ How to Mount a File System From the Global Zone Into a Non-Global Zone

Assume that a zone has the `zonepath` `/export/home/my-zone`. You want to mount the disk `/dev/lofi/1` from the global zone into `/mnt` in the non-global zone.

For information about `lofi`, see `lofiadm`(1M) and `lofi`(7D)

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **To mount the disk into `/mnt` in the non-global zone, type the following from the global zone:**

   ```
   global# mount -F ufs /dev/lofi/1 /export/home/my-zone/root/mnt
   ```

Using Rights Profiles in Zone Administration

This section covers various tasks associated with administering non-global zones.

▼ How to Assign the Zone Management Profile

The Zone Management profile grants the power to manage all of the zones on the system to a user.

You must be the global administrator in the global zone to perform this procedure.

1. **Become superuser, or assume the Primary Administrator role.**

   To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2. **Add the rights profile to the `/etc/user_attr` database file.**

   ```
   username::::profiles=Zone Management
   ```

   The fields in the `auth_attr` database are separated by colons.

Example—Using Profile Shells With Zone Commands

You can execute zone commands in a profile using the `pfexec` program. The program executes commands with the attributes specified by the user's profiles in the `exec_attr` database. It is invoked by the profile shells `pfksh`, `pfcsh`, and `pfsh`.

For example, use the `pfexec` profile shell to log in to a zone. This example uses the zone `my-zone`.

```
machine$ pfexec zlogin my-zone
```

Glossary

anon A page of memory allocated within an anon segment or a page allocated on the first write to it within a named, private segment.

application memory set A memory set that has been explicitly created by an application or by an administrator, which contains a subset of the system's physical memory. Processes bound to a memory set can use the memory in the set exclusively.

bless In Perl, the keyword used to create an object.

blessed In Perl, the term used to denote class membership.

cap A limit that is placed on system resource usage.

A memory set's cap is the maximum amount of memory allowed to all processes bound to that set.

capping The process of placing a limit on system resource usage.

default set A system memory set which provides memory for all processes that are not explicitly bound to another set. The initial reservation is equal to all pageable memory on the system minus the amount taken for other system sets. It has a disabled cap.

dynamic pools configuration Representation of the resource pools functionality for a given system at a point in time.

extended accounting A flexible way to record resource consumption on a task or process basis in the Solaris Operating System.

fair share scheduler A scheduling class, also known as FSS, that allows you to allocate CPU time that is based on shares. Shares define the portion of the system's CPU resources that are allocated to a project.

file cache set A system memory set that contains pages used to cache file system data. This set provides control over pages in the kernel's `segmap`

	segment. These pages are used for file system read and write operations. The set has a zero reservation. The cap is set according to platform parameters.
global administrator	An administrator with root privileges or the Primary Administrator role. When logged into the global zone, the global administrator can monitor and control the system as a whole.
global zone	Zone contained on every Solaris system. The global zone is both the default zone for the system and the zone used for system-wide administrative control.
heap	Process-allocated scratch memory.
locked memory	Memory that cannot be paged.
memory cap enforcement threshold	The percentage of physical memory utilization on the system that will trigger cap enforcement by the resource capping daemon.
memory set	A resource set that contains a subset of the system's physical memory.
non-global zone	A virtualized operating system environment created within a single instance of the Solaris Operating System. Zones are a partitioning technology used to virtualize operating system services.
page in	To read data from a file into physical memory one page at a time.
page out	To relocate pages to an area outside of physical memory.
pool	See resource pool.
pool daemon	A system daemon that is active when dynamic resource allocation is required.
project	A network-wide administrative identifier for related work.
resident set size	The size of the resident set. The resident set is the set of pages that are resident in physical memory.
resource	An aspect of the computing system that can be manipulated with the intent to change application behavior.
resource capping daemon	A daemon that regulates the consumption of physical memory by processes running in projects that have resource caps defined.
resource consumer	Fundamentally, a Solaris process. Process model entities such as the project and the task provide ways of discussing resource consumption in terms of aggregated resource consumption.
resource control	A per-process, per-task, or per-project limit on the consumption of a resource.
resource management	A functionality that enables you to control how applications use available system resources.

resource partition An exclusive subset of a resource. All of the partitions of a resource sum to represent the total amount of the resource available in a single executing Solaris instance.

resource pool A configuration mechanism that is used to partition machine resources. A resource pool represents an association between groups of partitionable resources.

resource set A process-bindable resource. Most often used to refer to the objects constructed by a kernel subsystem offering some form of partitioning. Examples of resource sets include scheduling classes and processor sets.

RSS See resident set size.

scanner A kernel thread that identifies infrequently used pages and relocates the pages to an area outside of physical memory.

shared set System memory set that contains pages shared between memory sets.

static pools configuration A representation of the way in which an administrator would like a system to be configured with respect to resource pools functionality.

system memory set A memory set that is automatically created at system initialization time and never removed.

task In resource management, a process collective that represents a set of work over time. Each task is associated with one project.

usage count A dynamic count, in pages, of the amount of memory being used in a memory set.

working set size The size of the working set. The working set is the set of pages that the project workload actively uses during its processing cycle.

WSS See working set size.

zone administrator An administrator having the Zone Management profile. The privileges of a zone administrator are confined to a given zone.

zone scope An attribute specifying the type of zone in which a package can be installed.

Index

S

T

V

Z