Administering Resource Management in Oracle® Solaris 11.4

ORACLE®

Part No: E61042
February 2021

Administering Resource Management in Oracle Solaris 11.4

Part No: E61042

Copyright © 2004, 2021, Oracle and/or its affiliates.

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Revenue Recognition Notice

If this document is in private pre-General Availability status:

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document may change and remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Référence: E61042

Copyright © 2004, 2021, Oracle et/ou ses affiliés.

Restrictions de licence/Avis d'exclusion de responsabilité en cas de dommage indirect et/ou consécutif

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Exonération de garantie

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Avis sur la limitation des droits

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Avis sur les applications dangereuses

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Marques

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Inside sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Epyc, et le logo AMD sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Avis d'exclusion de responsabilité concernant les services, produits et contenu tiers

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Avis sur la reconnaissance du revenu

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère privé :

Les informations contenues dans ce document sont fournies à titre informatif uniquement et doivent être prises en compte en votre qualité de membre du customer advisory board ou conformément à votre contrat d'essai de Version préliminaire de Disponibilité Générale ("Pre-GA") uniquement. Ce document ne constitue en aucun cas un engagement à fournir des composants, du code ou des fonctionnalités et ne doit pas être retenu comme base d'une quelconque décision d'achat. Le développement, la publication, les dates et les tarifs des caractéristiques ou fonctionnalités décrites sont susceptibles d'être modifiés et relèvent de la seule discrétion d'Oracle.

Ce document contient des informations qui sont la propriété exclusive d'Oracle, qu'il s'agisse de la version électronique ou imprimée. Votre accès à ce contenu confidentiel et son utilisation sont soumis aux termes de vos contrats, Contrat-Cadre Oracle (OMA), Contrat de Licence et de Services Oracle (OLSA), Contrat Réseau Partenaires Oracle (OPN), contrat de distribution Oracle ou de tout autre contrat de licence en vigueur que vous avez signé et que vous vous engagez à respecter. Ce document et son contenu ne peuvent en aucun cas être communiqués, copiés, reproduits ou distribués à une personne extérieure à Oracle sans le consentement écrit d'Oracle. Ce document ne fait pas partie de votre contrat de licence. Par ailleurs, il ne peut être intégré à aucun accord contractuel avec Oracle ou ses filiales ou ses affiliés.

Accessibilité de la documentation

Pour plus d'informations sur l'engagement d'Oracle pour l'accessibilité de la documentation, visitez le site Web Oracle Accessibility Program, à l'adresse : `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` ou le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to write applications that partition and manage system resources such as processor sets and scheduling classes.
- **Audience** – Programmers with experience with operating system interfaces.
- **Required knowledge** – Knowledge of C and of the system interfaces of Oracle Solaris or other UNIX systems.

Product Documentation Library

Documentation and resources for this product and related products are available at `http://www.oracle.com/pls/topic/lookup?ctx=E37838-01`.

Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

CHAPTER 1

Introduction to Resource Management

Oracle Solaris resource management functionality 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.

- "Resource Management Overview" on page 15
- "When to Use Resource Management" on page 18
- "Setting Up Resource Management Task Map" on page 20

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 Oracle 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. 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 other

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 Oracle 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(5)` 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 Oracle 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 discussed in Chapter 6, “About Resource Controls” provide 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, “About Fair Share Scheduler” 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, “About Resource Pools” enables you to limit workloads to specific subsets of the system.

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 capability allows the administrator to apply resource management constraints across a collection of systems, rather than on an exclusively per-system 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, “About Projects and Tasks”. The extended accounting facility that links these identifiers with application resource usage is described in Chapter 4, “About Extended Accounting”.

Interaction With Non-Global Zones

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

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 systems 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 Oracle 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 system. 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 system.
- 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 provides a high-level overview of the steps to set up resource management on your system.

Task	Description	For Instructions
Identify the workloads on your system and categorize each workload by project.	Create project entries in either the `/etc/project` file, in the NIS map, or in the LDAP directory service.	"`project` Database" on page 26
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.	`cpustat`(8), `iostat`(8), `mpstat`(8), `prstat`(8), `sar`(1), and `vmstat`(8) man pages
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 Oracle Solaris system. You can update the values from the command line while the task or process is running.	"Available Resource Controls" on page 70 "Global and Local Actions on Resource Control Values" on page 75 "Temporarily Updating Resource Control Values on a Running System" on page 80 `rctladm`(8) and `prctl`(1) man pages
Set resource controls and project attributes for every project entry in the `project` database or naming service project database.	Each project entry in the `/etc/project` file or the naming service project database 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.	"`project` Database" on page 26, "Local `/etc/project` File Format" on page 27 "Available Resource Controls" on page 70 "Global and Local Actions on Resource Control Values" on page 75 Chapter 8, "About Fair Share Scheduler"
Place an upper bound on the resource consumption of physical memory by	The resource cap enforcement daemon will enforce the physical memory resource cap defined for the	"`project` Database" on page 26

Task	Description	For Instructions
collections of processes attached to a project.	project's `rcap.max-rss` attribute in the `/etc/project` file.	Chapter 10, "About Controlling Physical Memory With the Resource Capping Daemon (`rcapd`)"
Create resource pool configurations.	Resource pools provide a way to partition system resources, such as processors, and maintain those partitions across reboots. You can add one `project.pool` attribute to each entry in the `/etc/project` file.	"`project` Database" on page 26 Chapter 12, "About Resource Pools"
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.	"Configuring the FSS" on page 109 `dispadmin`(8) man page
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 to 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 systems.	"How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 58 "How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 58 `acctadm`(8) man page
(Optional) If you need to make additional adjustments to your configuration, 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` file or in the naming service project database.	"Temporarily Updating Resource Control Values on a Running System" on page 80 `rctladm`(8) and `prctl`(1) man pages
(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.	"Perl Interface to the `libexacct` Library" on page 54 `wracct`(8) man page

CHAPTER 2

About Projects and Tasks

This chapter discusses the *project* and *task* facilities of Oracle 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:

- "Project and Task Facilities" on page 23
- "Project Identifiers" on page 24
- "Task Identifiers" on page 29
- "Using Projects to Assign, Modify, and Remove Multi-CPU Binding" on page 31
- "Commands Used With Projects and Tasks" on page 32

To use the projects and tasks facilities, see Chapter 3, "Administering Projects and Tasks".

Project and Task Facilities

An Oracle Solaris project is a collection of processes with predefined attributes. The Multi-CPU binding (MCB) attribute is applied to processes that belong to the project. MCB is managed through Oracle Solaris projects.

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 Oracle 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.

The controls specified in the `project` name service database are set on the process, task, and project. Since process and task controls are inherited across `fork` and `settaskid` system calls, all processes and tasks that are created within the project inherit these controls. For information about these system calls, see the `fork(2)` and `settaskid(2)` man pages.

Based on their project or task membership, running processes can be manipulated with standard Oracle 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 systems through the `project` name service database. Thus, the resource consumption of related workloads that run on (or span) multiple systems can ultimately be analyzed across all of the systems.

Project Identifiers

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 be assigned to a default project, 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. A user is automatically a member of that default project, even if the user is not in the user or group list specified in that 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 `getprojent`(3PROJECT) man page. The system follows ordered steps to determine the default project. 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 the `user_attr(5)` man page.
2. If a project with the name `user.`*user-id* is present in the `project` database, then that project is the default project. See the `project(5)` man page 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 about the passwd file, see the passwd(5) man page.

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 the getprojent(3PROJECT) man page for more information.

Setting User Attributes With the useradd and usermod Commands

You can use the following commands with the -K option and a *key*=*value* pair to set user attributes in local files:

useradd
: Set default project for user

usermod
: Modify user information

Local files can include the following:

- /etc/group
- /etc/passwd
- /etc/project
- /etc/shadow
- /etc/user_attr

If a network naming service such as NIS is being used to supplement the local file with additional entries, these commands cannot change information supplied by the network name service. However, the commands do verify the following against the external *naming service database*:

- Uniqueness of the user name (or role)
- Uniqueness of the user ID
- Existence of any group names specified

For more information, see the useradd(8), usermod(8), and user_attr(5) man pages.

`project` Database

You can store project data in a local file, in the Domain Name System (DNS), in a Network Information Service (NIS) project map, or in a Lightweight Directory Access Protocol (LDAP) directory service. The `/etc/project` file or naming 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 naming service, are not applied to currently active projects. The updates are applied to new tasks that join the project when either the `login` or the `newtask` command is used. For more information, see the `login(1)` and `newtask(1)` man pages.

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 an identity, a set of configurable modules is used to provide authentication, account management, credentials management, and session management.

For an overview of PAM, see Chapter 1, "Using Pluggable Authentication Modules" in *Managing Authentication in Oracle Solaris 11.4*.

Naming Services Configuration

Resource management supports naming service `project` databases. The location where the `project` database is stored is defined in the `/etc/nsswitch.conf` file. 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, and then search subsequent sources.

For more information about the `/etc/nsswitch.conf` file, see Chapter 2, "About the Name Service Switch" in *Working With Oracle Solaris 11.4 Directory and Naming Services: DNS and NIS* and `nsswitch.conf(5)`.

Local `/etc/project` File Format

If you select `files` as your `project` database source in the `nsswitch.conf` file, the login process searches the `/etc/project` file for project information. See the `projects(1)` and `project(5)` man pages for more information.

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

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, underline (_) characters, hyphens (-), and periods (.). The period, which is reserved for projects with special meaning to the operating system, can only be used in the names of default projects for users. The *projname* field cannot contain 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

A description of the project.

user-list

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

Wildcards can be used in this field. An asterisk (*) allows all users to join the project. An exclamation point followed by an asterisk (!*) excludes all users from the project. An 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. An asterisk (*) allows all groups to join the project. An exclamation point followed by an asterisk (!*) excludes all groups from the project. An 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, "About Resource Controls"). *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 periods. A period is conventionally used as a separator between the categories and subcategories of the resource control. 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.

A semicolon is used to separate name-value pairs. A semicolon cannot be used in a value definition. A colon is used to separate project fields. A colon cannot be used in a value definition.

Note - Routines that read this file halt if they encounter a malformed entry. Any projects that are specified after the incorrect entry are not assigned.

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

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

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

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

You can also add resource controls and attributes to the `/etc/project` file:

- To add resource controls for a project, see "Setting Resource Controls" on page 84.
- To define a physical memory resource cap for a project using the resource capping daemon described in `rcapd(8)`, see "Attribute to Limit Physical Memory Usage for Projects" on page 114.
- To add a `project.pool` attribute to a project's entry, see "Creating the Consolidated Configuration" on page 178.

Project Configuration for NIS

If you are using NIS, you can specify in the `/etc/nsswitch.conf` file to search the NIS project 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 Chapter 5, "About the Network Information Service" in *Working With Oracle Solaris 11.4 Directory and Naming Services: DNS and NIS*.

Project 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
```

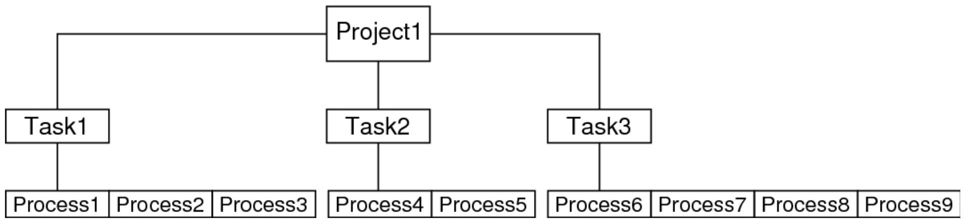
For more information about LDAP, see Chapter 1, "Introduction to the LDAP Naming Service" in *Working With Oracle Solaris 11.4 Directory and Naming Services: LDAP*. For more information about the schema for project entries in an LDAP database, see "Oracle Solaris Schemas" in *Working With Oracle Solaris 11.4 Directory and Naming Services: LDAP*.

Task Identifiers

Each successful login into a project creates a new *task* that contains the login process. 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 task is automatically assigned a task ID.

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

FIGURE 1 Project and Task Tree



All operations on process groups, such as signal delivery, are also supported on tasks. You can also bind a task to a *processor set* and set a scheduling priority and class for a task, which modifies all current and subsequent processes in the task.

A task is created whenever a project is joined. The following actions, commands, and functions create tasks:

- `cron`
- `login`
- `newtask`
- `setproject`
- `su`

You can create a finalized task by using one of the following methods. All further attempts to create new tasks will fail.

- Use the `newtask` command with the `-F` option.
- Set the `task.final` attribute on a project in the `project` naming service database. All tasks created in that project by `setproject` have the `TASK_FINAL` flag.

For more information, see the `login(1)`, `newtask(1)`, `cron(8)`, `su(8)`, and `setproject(3PROJECT)` man pages.

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

Using Projects to Assign, Modify, and Remove Multi-CPU Binding

Use multi-CPU binding (MCB) to bind a project to a specific set of CPUs, but not bind the CPUs exclusively. MCB binding allows other processes to use those CPUs as well.

The resource pools feature requires the hard partitioning of processors in the system. With hard partitioning, you cannot specify, for example, that process *A* runs on CPUs `1` and `2`, and process *B* runs on CPUs `2` and `3`, because these partitions overlap. Use MCB to assign overlapping partitions to processes. When you bind a process to MCB, the set of CPUs bounded by MCB must reside in the pool to which the process is bound. If there is a `project.pool` entry, that is used. If there is no `project.pool` entry, processes are bound to the default pool of the target zone.

To create and modify the project file, use the standard command-line tools `projadd` and `projmod`. Use the following attributes with the `-K` option of the `projadd` and `projmod` commands to configure MCB:

`project.mcb.cpus=`

The values are the set of CPUs bound by MCB.

Note that MCB for projects also supports the following CPU structures:

- `project.mcb.cores`
- `project.mcb.lgroups`
- `project.mcb.pgs`
- `project.mcb.sockets`

`project.mcb.flags=`

The values are `strong` and `weak`. The default is `strong`. Strong binding specifies that processes run only on designated CPUs.

You can also set the project resource pool by using `project.pool`. If not set, the system uses the default pool for the target zone.

The `newtask` command also utilizes projects. When you set MCB attributes for a project in the project configuration file, a user with appropriate privileges can use the `newtask` command described in the `newtask(1)` man page to manipulate the project file.

For MCB usage examples and task information, see Chapter 3, "Administering Projects and Tasks". To obtain information on the CPU structures of a given system, use the following commands.

`lgrpinfo -c`

Information about locality groups (`lgroups`).

`pginfo`

Information about processor groups.

`psrinfo -t`

Information about CPU, core, and socket structure.

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.

Man Page Reference	Description
`projects`(1)	Displays project memberships for users. Lists projects from `project` database. Prints information on given projects. If no project names are supplied, information is displayed for all projects. Use the `projects` command with the `-l` option to print verbose output.
`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. The `newtask` command can also be used to change the task and the project binding for a running process. Use with the `-F` option to create a finalized task.
`projadd`(8)	Adds a new project entry to the `/etc/project` file. The `projadd` command creates a project entry only on the local system. The `projadd` command cannot change information that is supplied by the network naming service. Can be used to edit project files other than the default file, `/etc/project`. Provides syntax checking for the `project` file. Validates and edits project attributes. Supports scaled values.
`projmod`(8)	Modifies information for a project on the local system. The `projmod` command cannot change information that is supplied by the network naming service. However, the command does verify the uniqueness of the project name and project ID against the external naming service. Can be used to edit project files other than the default file, `/etc/project`. Provides syntax checking for the `project` file. Validates and edits project attributes. Can be used to add a new attribute, add values to an attribute, or remove an attribute. Supports scaled values. Use with the `-A` option to apply the resource control values found in the project database to the active project. Existing values that do not match the values defined in the `project` file are removed.

Man Page Reference	Description
projdel(8)	Deletes a project from the local system. The `projdel` command cannot change information that is supplied by the network naming service.
useradd(8)	Adds default project definitions to the local files. Use with the `-K` *key*=*value* option to add or replace user attributes.
userdel(8)	Deletes a user's account from the local file.
usermod(8)	Modifies a user's login information on the system. Use with the `-K` *key*=*value* option to add or replace user attributes.

For additional information on using MCB, see the `pbind`(8) and `processor_affinity`(2) man pages.

♦ ♦ ♦ CHAPTER 3

Administering Projects and Tasks

This chapter describes how to use the project and task facilities of Oracle Solaris resource management, including multi-CPU binding (MCB). The following topics are covered:

For an overview of the projects and tasks facilities, see Chapter 2, "About Projects and Tasks".

Note - If you are using these facilities on an Oracle Solaris system with zones installed, only processes in the same zone are visible through system call interfaces that take process IDs when these commands are run in a non-global zone.

Administering Projects and Tasks Task Map

Task	Description	For Instructions
View examples of commands and options used with projects and tasks.	Display task and project IDs, display various statistics for processes and projects that are currently running on your system.	"Example Commands and Command Options" on page 36
Define a project.	Add a project entry to the `/etc/project` file and alter values for that entry.	"Example: How to Define a Project and View the Current Project" on page 38
Delete a project.	Remove a project entry from the `/etc/project` file.	"Example: How to Delete a Project From the `/etc/project` File" on page 40
Validate the `project` file or project database.	Check the syntax of the `/etc/project` file or verify the uniqueness of the project name and project ID against the external naming service.	"How to Validate the Contents of the `/etc/project` File" on page 41
Obtain project membership information.	Display the current project membership of the invoking process.	"How to Obtain Project Membership Information" on page 42

Task	Description	For Instructions
Create a new task.	Create a new task in a particular project by using the `newtask` command.	"How to Create a New Task" on page 42
Associate a running process with a different task and project.	Associate a process number with a new task ID in a specified project.	"Example: How to Move a Running Process Into a New Task" on page 42
Add and work with project attributes.	Use the project database administration commands to add, edit, validate, and remove project attributes.	"Editing and Validating Project Attributes" on page 43

Example Commands and Command Options

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

Command Options Used With Projects and Tasks

`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, similar to the following:

```
$ 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
 12905 root     4472K 3640K cpu0    59    0   0:00:01 0.4% prstat/1
   829 root       43M   33M sleep   59    0   0:36:23 0.1% Xorg/1
…
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        4 4728K   19M   0.9%   0:00:01 0.4% user.root
     0      111  278M  344M    17%   1:15:02 0.1% system
    10        2 1884K 9132K   0.4%   0:00:00 0.0% group.staff
     3        3 1668K 6680K   0.3%   0:00:00 0.0% default

Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

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
 12907 root     4488K 3588K cpu0    59    0   0:00:00 0.3% prstat/1
   829 root       43M   33M sleep   59    0   0:36:24 0.1% Xorg/1
…
   311 root     3488K 2512K sleep   59    0   0:00:00 0.0% picld/4
   409 root     4356K 2768K sleep   59    0   0:00:00 0.0% hald-addon-cpuf/1
TASKID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
  1401        2 2540K 8120K   0.4%   0:00:00 0.3% user.root
    94       15   84M  162M   7.9%   0:59:37 0.1% system
   561        1   37M   24M   1.2%   0:02:06 0.0% system
     0        2    0K    0K   0.0%   0:02:47 0.0% system
    46        1 4224K 5524K   0.3%   0:00:38 0.0% system
Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

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

`ps` Command

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

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

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

The `su` command joins the target user's default project by creating a new task, as part of simulating a login.

To switch the user's default project, type the following command:

```
$ su - user
```

Administering Projects

▼ Example: 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. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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

```
$ projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

3. **Add a project with the name `booksite`, and 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.**

```
$ projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

```
…
booksite
        projid : 4113
        comment: ""
        users  : mark
        groups : (none)
        attribs:
```

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.**

```
$ projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
…
booksite
        projid : 4113
        comment: "Book Auction Project"
        users  : mark
        groups : (none)
        attribs:
```

See Also To bind projects, tasks, and processes to a pool, see "Setting Pool Attributes and Binding to a Pool" on page 172.

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

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

1. **Assume the `root` role.**
   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Remove the project `booksite`.**

```
$ projdel booksite
```

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

```
$ projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

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

```
$ su - mark
$ projects
default
```

▼ How to Validate the Contents of the `/etc/project` File

- **If no editing options are given, the `projmod` command validates the contents of the `project` file.**

- **To validate a NIS map, type the following:**

  ```
  $ ypcat project | projmod -f -
  ```

- **To check the syntax of the `/etc/project` file, type the following:**

  ```
  $ projmod -n
  ```

How to Obtain Project Membership Information

- **Display 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. **Log in as a member of the destination project, *booksite* in this example.**

2. **Create a new task in the *booksite* project to obtain the system task ID.**

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

   ```
   machine% newtask -v -p booksite
   16
   ```

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

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

   The process is now a member of the new project.

Example: 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.

Before You Begin To perform this action, you must be the root user, have the required rights profile, or be the owner of the process and be a member of the new project.

1. **Obtain the process ID of the `book_catalog` process.**

```
$ pgrep book_catalog
8100
```

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

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

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

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

```
$ pgrep -T 17
8100
```

Editing and Validating Project Attributes

You can use the `projadd` and `projmod` project database administration commands to edit project attributes.

The `-K` option specifies a replacement list of attributes. Attributes are delimited by semicolons (`;`). If the `-K` option is used with the `-a` option, the attribute or attribute value is added. If the `-K` option is used with the `-r` option, the attribute or attribute value is removed. If the `-K` option is used with the `-s` option, the attribute or attribute value is substituted.

▼ Example: How to Add Attributes and Attribute Values to Projects

Use the `projmod` command with the `-a` and `-K` options to add values to a project attribute. If the attribute does not exist, it is created.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Add a `task.max-lwps` resource control attribute with no values in the project `myproject`.**

   A task entering the project has only the system value for the attribute.

   ```
   $ projmod -a -K task.max-lwps myproject
   ```

3. **Add a value to `task.max-lwps` in the project `myproject`.**

   The value consists of a privilege level, a threshold value, and an action associated with reaching the threshold.

   ```
   $ projmod -a -K "task.max-lwps=(priv,100,deny)" myproject
   ```

4. **(Optional) Add another value to the existing list of values by using the same options.**

   Resource controls can have multiple values, separated by commas.

   ```
   $ projmod -a -K "task.max-lwps=(priv,1000,signal=KILL)" myproject
   ```

   The `task.max-lwps` entry now reads:

   ```
   task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
   ```

▼ Example: How to Remove Attribute Values From Projects

This procedure uses the following values:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Remove an attribute value from the resource control `task.max-lwps` in the project `myproject`.**

   ```
   $ projmod -r -K "task.max-lwps=(priv,100,deny)" myproject
   ```

   If `task.max-lwps` has multiple values, such as:

   ```
   task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
   ```

   The first matching value would be removed. The result would then be:

```
task.max-lwps=(priv,1000,signal=KILL)
```

Example: How to Remove a Resource Control Attribute From a Project

This example removes the resource control `task.max-lwps` from the project `myproject`.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Remove the attribute `task.max-lwps` and all of its values from the project `myproject`.**

   ```
   $ projmod -r -K task.max-lwps myproject
   ```

Example: How to Substitute Attributes and Attribute Values for Projects

This example substitutes a different value for the attribute `task.max-lwps` in the project `myproject`. If the attribute does not exist, it is created.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Replace the current `task.max-lwps` values with the new values shown:**

   ```
   $ projmod -s -K "task.max-lwps=(priv,100,none),(priv,120,deny)" myproject
   ```

   The result would be:

   ```
   task.max-lwps=(priv,100,none),(priv,120,deny)
   ```

▼ Example: How to Remove the Existing Values for a Resource Control Attribute

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Remove the current values for `task.max-lwps` from the project `myproject`.**

   ```
   $ projmod -s -K task.max-lwps myproject
   ```

How to Use Projects to Assign, Modify, and Remove Multi-CPU Binding

Use multi-CPU binding (MCB) to bind a project to a specific set of CPUs, but not bind the CPUs exclusively. MCB allows other processes to use those CPUs as well. Use MCB to assign overlapping partitions to processes. When a process is bound to MCB, the set of CPUs bound by MCB must reside in the pool that the process is bound to. If there is a `project.pool` entry, that is used. If there is no `project.pool` entry, processes are bound to the default pool of the target zone.

▼ How to Create a Project With MCB Resource Controls

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Create the project `new-project`.**

   ```
   $ projadd -K project.mcb.cpus=0,3-5,9-11 -K project.mcb.flags=weak -K project.
   pool=pool_default new-project
   ```

3. **View the project.**

   ```
   $ projects -l new-project
   new-project
   ```

```
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: project.mcb.cpus=0,3-5,9-11
                 project.mcb.flags=weak
                 project.pool=pool_default
```

To check the validity of the project file only, use the `projmod` command without options.

4. **Check the binding.**

```
$ pbind -q -i projid 100
pbind(8): pid 4156 weakly bound to processor(s) 0 3 4 5 9 10 11.
pbind(8): pid 4170 weakly bound to processor(s) 0 3 4 5 9 10 11.
pbind(8): pid 4184 weakly bound to processor(s) 0 3 4 5 9 10 11.
```

Example: How to Change the Multi-CPU Binding Type

This example changes the binding type to `strong` bind.

Alternatively, you could instead delete the `project.mcb.flag` key, because the value is set to `strong` by default.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Change the value of `project.mcb.flags` to `strong`.**

```
$ projmod -s -K project.mcb.flags=strong new-project
```

3. **View the project.**

```
$ projects -l new-project
new-project
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: project.mcb.cpus=0,3-5,9-11
                 project.mcb.flags=strong
```

Note - By default, the `projmod` command only modifies the project configuration file. To apply the changes to the processes in the project, use the `-A` option.

```
$ projmod -A new-project
projmod: Updating project new-project succeeded with following warning message.
WARNING: We bind the target project to the default pool of the zone because an MCB entry
 exists.
```

The update was successful. However, at least one of the CPUs described in the `project.mcb.cpus` entry must exist in the system and be online. If a subset of the CPUs do not exist or are not online, these are not bound to, and warnings are printed.

If you try to apply the attributes of the project to processes, an error message is displayed. For example, the following message displays if none of the specified CPUs `17-20` exist:

```
ERROR: All of given multi-CPU binding (MCB) ids are not found in the system:
project.mcb.cpus=17-20
```

▼ Example: How to Clear Multi-CPU Binding From a Project

This example shows how to remove the multi-CPU binding (MCB) attributes.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Set the value of `project.mcb.cpus` to `none` and remove `project.mcb.flags`, if set.**

```
$ projmod -s -K project.mcb.cpus=none new-project
```

3. **View the project.**

```
$ projects -l new-project
new-project
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: project.mcb.cpus=none
        project.pool=pool_default
```

CHAPTER 4

About Extended Accounting

By using the project and task facilities that are described in Chapter 2, "About Projects and Tasks" 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, skip to "How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 58.

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, 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 Oracle Solaris operating system provides a flexible way to record system and network resource consumption for the following:

- Tasks.
- Processes.

- Selectors provided by the IPQoS `flowacct` module. For more information, see the `ipqos`(4IPP) man page.
- Network management. See `dladm`(8) and `flowadm`(8).

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.

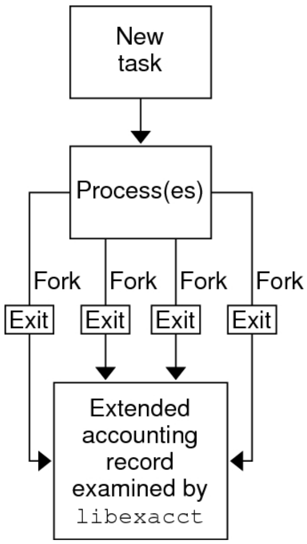
How Extended Accounting Works

The extended accounting facility in the Oracle 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 the `libexacct`(3LIB) man page). 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. The `libexacct` API 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 the `libexacct` Library" on page 54.

For example, 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, "About Projects and Tasks".

FIGURE 2 Task Tracking With Extended Accounting Activated



Extensible Format

The extended accounting format is substantially more extensible than the legacy system accounting software format. 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 the `getacct(2)` man page).

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 on an Oracle Solaris System with Zones Installed

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 or a user granted appropriate authorizations through the zonecfg utility can also determine resource consumption on a per-zone basis. See Chapter 1, "Non-Global Zone Configuration Command and Resources" in *Oracle Solaris Zones Configuration Resources* for more information.

Extended Accounting Configuration

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 the acctadm(8) man page for more information.

Starting and Persistently Enabling Extended Accounting

The acctadm command described in the acctadm(8) man page starts extended accounting through the Oracle Solaris service management facility (SMF) service described in the smf(7) man page.

The extended accounting configuration is stored in the SMF repository. The configuration is restored at boot by a service instance, one for each accounting type. Each of the extended accounting types is represented by a separate instance of the SMF service:

`svc:/system/extended-accounting:flow`
: Flow accounting

`svc:/system/extended-accounting:process`
: Process accounting

`svc:/system/extended-accounting:task`
: Task accounting

`svc:/system/extended-accounting:net`
: Network accounting

Enabling extended accounting by using the `acctadm` command causes the corresponding service instance to be enabled if not currently enabled, so that the extended accounting configuration will be restored at the next boot. Similarly, if the configuration results in accounting being disabled for a service, the service instance will be disabled. The instances are enabled or disabled by the `acctadm` command as needed. For more information, see the `acctadm`(8) man page.

To permanently activate extended accounting for a resource, run the following command:

$ **acctadm -e** *resource_list*

The *resource_list* is a comma-separated list of resources or resource groups.

Accounting Records

The `acctadm` command appends new records to an existing file in `/var/adm/exacct`.

Commands Used With Extended Accounting

Command Reference	Description
`acctadm`(8)	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, flows and network.

Command Reference	Description
wracct(8)	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 about commands that are associated with tasks and projects, see "Example Commands and Command Options" on page 36. For information about IPQoS flow accounting, see the ipqosconf(8) man page.

Perl Interface to the libexacct Library

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 interface allows easier access to the data, and 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 Oracle Solaris 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 library provides operations on exacct format files, catalog tags, and exacct objects. The 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 Name (should not contain spaces)	Function Reference	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)

Module Name (should not contain spaces)	Function Reference	For More Information
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)
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::Group	This module provides object-oriented methods to access an individual exacct accounting file Group. Objects of this type inherit from the Sun::Solaris::Exacct::Object module. 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::Exacct:: Object::Item	This module provides object-oriented methods to access an individual exacct accounting file Item. Objects of this type inherit from the Sun::Solaris::Exacct::Object module.	Exacct::Object::Item(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 the /bin/kstat file, 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 61.

♦ ♦ ♦ CHAPTER 5

Administering Extended Accounting Tasks

This chapter describes how to administer the extended accounting subsystem.

For an overview of the extending accounting subsystem, see Chapter 4, "About Extended Accounting".

Administering the Extended Accounting Facility Task Map

Task	Description	For Instructions
Activate the extended accounting facility.	Use extended accounting to monitor resource consumption by each project running on your system. You can use the *extended accounting* subsystem to capture historical data for tasks, processes, and flows.	"How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 58
Display extended accounting status.	Determine the status of the extended accounting facility.	"How to Display Extended Accounting Status" on page 59
View available accounting resources.	View the accounting resources available on your system.	"How to View Available Accounting Resources" on page 59
Deactivate the flow, process, task, and net accounting instances.	Turn off the extended accounting functionality.	"How to Deactivate Process, Task, Flow, and Network Management Accounting" on page 60
Use the Perl interface to the extended accounting facility.	Use the Perl interface to develop customized reporting and extraction scripts.	"Using the Perl Interface to `libexacct`" on page 61

Using Extended Accounting Functionality

Users can manage extended accounting (start accounting, stop accounting, and change accounting configuration parameters) if they have the appropriate rights profile for the accounting type to be managed:

- Extended Accounting Flow Management

- Process Management
- Task Management
- Network Management

Note - Roles contain authorizations and privileged commands. For information about how to assign user rights, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components

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

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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
   ```

5. **Activate extended accounting for network.**

   ```
   $ acctadm -e extended -f /var/adm/exacct/net net
   ```

   Run the `acctadm` command on links and flows administered by the `dladm` and `flowadm` commands.

See Also See the `acctadm(8)` man page for more information.

How to Display Extended Accounting Status

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

```
machine% 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
                 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(5)`). This data provides substantially 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. Run the `acctadm` command with the `-r` option to view the accounting resource groups available on your system.

```
machine% acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,zone,flag,
memory,mstate     displays as one line
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
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
net:
  extended name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts,saddr,daddr,sport,dport,protocol,dsfield
  basic    name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts
```

▼ How to Deactivate Process, Task, Flow, and Network Management Accounting

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

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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. **Turn off network management accounting.**

   ```
   $ acctadm -x net
   ```

6. **Verify that task accounting, process accounting, flow, and network accounting are turned off.**

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

```
        Process accounting: inactive
   Process accounting file: none
 Tracked process resources: none
Untracked process resources: extended
           Flow accounting: inactive
      Flow accounting file: none
    Tracked flow resources: none
  Untracked flow resources: extended
            Net accounting: inactive
       Net accounting file: none
     Tracked Net resources: none
   Untracked Net resources: extended
```

Using the Perl Interface to `libexacct`

This section provides the following information:

- "How to Recursively Print the Contents of an `exacct` Object" on page 61
- "How to Create a New Group Record and Write It to a File" on page 63
- "How to Print the Contents of an `exacct` File" on page 64
- "Example Output From `Sun::Solaris::Exacct::Object->dump()`" on page 65

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

Use the following code to recursively print the contents of an `exacct` object. 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

The following Perl script creates a new group record and writes it to a file named `/tmp/exacct`.

```
#!/usr/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/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 exacct 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()

The following example output is 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 63.

```
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

About Resource Controls

After you determine the resource consumption of workloads on your system as described in Chapter 4, "About Extended Accounting", 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.

This chapter covers the following topics.

For information about how to administer resource controls, see Chapter 7, "Administering Resource Controls Tasks".

Resource Controls Concepts

In the Oracle Solaris operating system, the concept of a per-process resource limit has been extended to the task and project entities described in Chapter 2, "About Projects and Tasks". These enhancements are provided by the resource controls 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 `zone`, `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.

For a list of the standard resource controls that are available in this release, see "Available Resource Controls" on page 70. See "Available Resource Controls" on page 70 for information about available zone-wide resource controls.

Resource Limits and Resource Controls

UNIX systems have traditionally provided a resource limit facility. This facility enables 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.

Interprocess Communication and Resource Controls

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. The resource controls facility provides resource controls that define the behavior of the kernel's IPC facilities. These resource controls replace the `/etc/system` tunables.

Obsolete parameters that are used to initialize the default resource control values might be included in the `/etc/system` file on this Oracle Solaris system. However, using the obsolete parameters is not recommended.

To observe which IPC objects are contributing to a project's usage, use the `ipcs` command with the `-J` option. See "How to Display IPC Information" on page 93 to view an example display. For more information about the `ipcs` command, see the `ipcs(1)` man page.

For information about Oracle Solaris system tuning, see the *Oracle Solaris 11.4 Tunable Parameters Reference Manual*.

Resource Control Constraint Mechanisms

Resource controls provide a mechanism for the constraint of system resources. Processes, tasks, projects, and zones 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 about the `rcapd` daemon, see Chapter 10, "About Controlling Physical Memory With the Resource Capping Daemon (`rcapd`)" and the `rcapd`(8)man page.

The `project.pool` attribute is used to specify a pool binding for a project. For more information on resource pools, see Chapter 12, "About Resource Pools".

Configuring Resource Controls and Attributes

The resource controls facility is configured through the `project` database. See Chapter 2, "About Projects and Tasks". 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-task lightweight process limit and a per-process maximum CPU time limit on a project entity. The `process.max-cpu-time` value would send a process a SIGTERM after the process ran for 1 hour, and a SIGKILL if the process continued to run for a total of 1 hour and 1 minute. See Table 3, "Signals Available to Resource Control Values," on page 77.

Typed as one line:

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
process.max-cpu-time=(basic,3600,signal=TERM),(priv,3660,signal=KILL)
```

Note - On systems that have zones enabled, zone-wide resource controls are specified in the zone configuration using a slightly different format. See "Setting Zone-Wide Resource Controls" in *Oracle Solaris Zones Configuration Resources* 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*.

For more information, see "Global and Local Actions on Resource Control Values" on page 75 and the `rctladm(8)` and `prctl(1)` man pages.

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

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` resource control specifies the number of shares to which the project is entitled. The `process.max-file-descriptor` resource control specifies the highest file number that can be assigned to a process by the `open` system call.

TABLE 1 Standard Project, Task, and Process Resource Controls

Control Name	Description	Default Unit
`process.max-address-space`	Maximum amount of address space, as summed over segment sizes, that is available to this process.	Size (bytes)
`process.max-core-size`	Maximum size of a core file created by this process.	Size (bytes)
`process.max-cpu-time`	Maximum CPU time that is available to this process.	Time (seconds)
`process.max-data-size`	Maximum heap memory available to this process.	Size (bytes)

Control Name	Description	Default Unit
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-msg-messages	Maximum number of messages on a message queue (value copied from the resource control at msgget() time).	Quantity (number of messages)
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-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)
process.max-stack-size	Maximum stack memory segment available to this process.	Size (bytes)
project.max-adi-metadata-memory	Total amount of memory for storing Silicon Secured Memory (SSM), also known as ADI, metadata of pages that might be written to backing store, expressed as a number of bytes.	Size (bytes)
project.cpu-cap	Absolute limit on the amount of CPU resources that can be consumed by a project. A value of 100 means 100% of one CPU as the project.cpu-cap setting. A value of 125 is 125%, because 100% corresponds to one full CPU on the system when using CPU caps.	Quantity (number of CPUs)
project.cpu-shares	Number of CPU shares granted to this project for use with the fair share scheduler (see FSS(4)).	Quantity (shares)
project.max-contracts	Maximum number of contracts allowed in this project.	Quantity (contracts)
project.max-locked-memory	Total amount of physical locked memory allowed. If priv_proc_lock_memory is assigned to a user, consider setting this resource control as well to prevent that user from locking all memory. Note that this resource control replaced project.max-device-locked-memory, which has been removed.	Size (bytes)
project.max-lwps	Maximum number of LWPs simultaneously available to this project.	Quantity (LWPs)
project.max-msg-ids	Maximum number of message queue IDs allowed for this project.	Quantity (message queue IDs)
project.max-port-ids	Maximum allowable number of event ports.	Quantity (number of event ports)
project.max-processes	Maximum number of process table slots simultaneously available to this project. Note that because both normal processes and zombie processes take up process table slots, the max-processes control thus protects against zombies exhausting the process table. Because zombie processes do not have any LWPs by definition, the max-lwps control cannot protect against this possibility.	Quantity (process table slots)
project.max-sem-ids	Maximum number of semaphore IDs allowed for this project.	Quantity (semaphore IDs)
project.max-shm-ids	Maximum number of shared memory IDs allowed for this project.	Quantity (shared memory IDs)

Control Name	Description	Default Unit
`project.max-shm-memory`	Total amount of System V shared memory allowed for this project.	Size (bytes)
`project.max-tasks`	Maximum number of tasks allowable in this project.	Quantity (number of tasks)
`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)
`task.max-processes`	Maximum number of process table slots simultaneously available to this task's processes.	Quantity (process table slots)

You can display the default values for resource controls on a system that does not have any resource controls set or changed. Such a system contains no non-default entries in the `/etc/system` file or the `project` database. To display values, use the `prctl` command.

Zone-Wide Resource Controls

Zone-wide resource controls limit the total resource usage of all process entities within a zone. Zone-wide resource controls can also be set using global property names as described in "Configurable Resource Types and Global Properties" in *Oracle Solaris Zones Configuration Resources*.

TABLE 2 Zones Resource Controls

Control Name	Description	Default Unit
`zone.cpu-cap`	Absolute limit on the amount of CPU resources that can be consumed by a non-global zone. A value of `100` means 100% of one CPU as the `project.cpu-cap` setting. A value of `125` is 125%, because 100% corresponds to one full CPU on the system when using CPU caps.	Quantity (number of CPUs)
`zone.cpu-shares`	Number of fair share scheduler (FSS) CPU shares for this zone.	Quantity (shares)
`zone.max-adi-metadata-memory`	Total amount of memory for storing Silicon Secured Memory (SSM) metadata of pages that may be written to backing store, expressed as a number of bytes. SSM is also known as ADI.	Size (bytes)
`zone.max-lofi`	Maximum number of `lofi` devices that can be created by a zone. The value limits each zone's usage of the minor node namespace.	Quantity (number of `lofi` devices)
`zone.max-locked-memory`	Total amount of physical locked memory available to a zone.	Size (bytes)

Control Name	Description	Default Unit
	When `priv_proc_lock_memory` is assigned to a zone, consider setting this resource control as well to prevent that zone from locking all memory.	
`zone.max-lwps`	Maximum number of LWPs simultaneously available to this zone	Quantity (LWPs)
`zone.max-msg-ids`	Maximum number of message queue IDs allowed for this zone	Quantity (message queue IDs)
`zone.max-processes`	Maximum number of process table slots simultaneously available to this zone. Because both normal processes and zombie processes take up process table slots, the `max-processes` control thus protects against zombies exhausting the process table. Because zombie processes do not have any LWPs by definition, the `max-lwps` control cannot protect against this possibility.	Quantity (process table slots)
`zone.max-sem-ids`	Maximum number of semaphore IDs allowed for this zone	Quantity (semaphore IDs)
`zone.max-shm-ids`	Maximum number of shared memory IDs allowed for this zone	Quantity (shared memory IDs)
`zone.max-shm-memory`	Total amount of System V shared memory allowed for this zone	Size (bytes)
`zone.max-swap`	Total amount of swap that can be consumed by user process address space mappings and `tmpfs` mounts for this zone.	Size (bytes)

For information about configuring zone-wide resource controls, see "Configuring Resource Controls and Attributes" on page 69 and "Setting Zone-Wide Resource Controls" in *Oracle Solaris Zones Configuration Resources*.

Note that it is possible to apply a zone-wide resource control to the global zone.

Units Support

Global flags that identify resource control types are defined for all resource controls. The flags are used by the system to communicate basic type information to applications such as the `prctl` command. Applications use the information to determine the following:

- The unit strings that are appropriate for each resource control
- The correct scale to use when interpreting scaled values

The following global flags are available:

RCTL_GLOBAL_BYTES

Resource control type string: bytes

Modifier	Scale
B	1
KB	2^{10}
MB	2^{20}
GB	2^{30}
TB	2^{40}
PB	2^{50}
EB	2^{60}

RCTL_GLOBAL_SECONDS

Resource control type string: seconds

Modifier	Scale
s	1
Ks	10^{3}
Ms	10^{6}
Gs	10^{9}
Ts	10^{12}
Ps	10^{15}
Es	10^{18}

RCTL_GLOBAL_COUNT

Resource control type string: count

Modifier	Scale
none	1
K	10^{3}
M	10^{6}
G	10^{9}
T	10^{12}
P	10^{15}
E	10^{18}

Scaled values can be used with resource controls. The following example shows a scaled threshold value:

```
task.max-lwps=(priv,1K,deny)
```

Note - Unit modifiers are accepted by the `prctl`, `projadd`, and `projmod` commands. You cannot use unit modifiers in the `project` database itself.

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 (root) 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 the `setrctl(2)` man page for more information. You can use the `prctl` command to modify values that are associated with basic and privileged levels.

Global and Local Actions on Resource Control Values

There are two categories of actions on resource control values: global and local.

Global Actions on Resource Control Values

Global actions apply to resource control values for every resource control on the system. You can use the `rctladm` command described in the rctladm(8) man page to perform the following actions:

- Display the global state of active system resource controls
- Set global logging actions

You can disable or enable the global logging action on resource controls. You can set the `syslog` action to a specific degree by assigning a severity level, `syslog=`*level*. The possible settings for *level* are as follows:

- `alert`
- `crit`
- `debug`
- `emerg`
- `err`
- `info`
- `notice`
- `warning`

By default, there is no global logging of resource control violations. The level `n/a` indicates resource controls on which no global action can be configured.

Local Actions on Resource Control Values

Local actions are taken on a process that attempts to exceed the control value. For each threshold value that is placed on a resource control, you can associate one or more actions. There are three types of local actions: `deny`, `none`, and `signal=`. These three actions are used as follows:

`deny`

You can deny resource requests for an amount that is greater than the threshold. For example, a `task.max-lwps` resource control with action `deny` causes a `fork` system call to fail if the new process would exceed the control value. See the fork(2) man page.

`none`

No action is taken on resource requests for an amount that is greater than the threshold. This action is useful for monitoring resource usage without affecting the progress of applications. You can also enable a global message that displays when the resource control is exceeded, although the process exceeding the threshold is not affected.

`signal=`

You can enable a global signal message action when the resource control is exceeded. A signal is sent to the process when the threshold value is exceeded. Additional signals are not sent if the process consumes additional resources. Available signals are listed in Table 3, "Signals Available to Resource Control Values," on page 77.

Not all of the actions can be applied to every resource control. For example, a process cannot exceed the number of CPU shares assigned to the project of which it is a member. Therefore, a deny action is not allowed on the `project.cpu-shares` resource control.

Due to implementation restrictions, the global properties of each control can restrict the range of available actions that can be set on the threshold value. (See the `rctladm`(8) man page.) A list of available signal actions is presented in the following table. For additional information about signals, see the `signal`(3HEAD) man page.

TABLE 3 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.	
SIGKILL	Terminate the process and quit the program.	
SIGSTOP	Stop the process. Job control signal.	
SIGTERM	Terminate the process. Termination signal sent by software.	
SIGXCPU	Terminate the process. CPU time limit exceeded.	Available only to resource controls with the RCTL_GLOBAL_CPUTIME property (`process.max-cpu-time`). See the `rctlblk_set_value`(3C) man page for more information.
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 the `rctlblk_set_value`(3C) man page for more information.
SIGXRES	Resource control limit exceeded. Generated by resource control facility.	

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. For more information, see the `setrctl(2)` man page.

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

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

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

The global flags indicate the following.

`bytes`	Unit of size for the resource control.
`count`	A count (integer) value for the resource control.
`cpu-time`	SIGXCPU is available to be sent when threshold values of this resource are reached.
`deny`	Always deny request for resource when threshold values are exceeded.
`lowerable`	Superuser privileges are not required to lower the privileged values for this control.
`no-basic`	Resource control values with the privilege type `basic` cannot be set. Only privileged resource control values are allowed.
`no-deny`	Even when threshold values are exceeded, access to the resource is never denied.
`no-signal`	A local signal action cannot be set on resource control values.
`no-syslog`	The global `syslog` message action may not be set for this resource control.
`seconds`	The time value for the resource control.

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

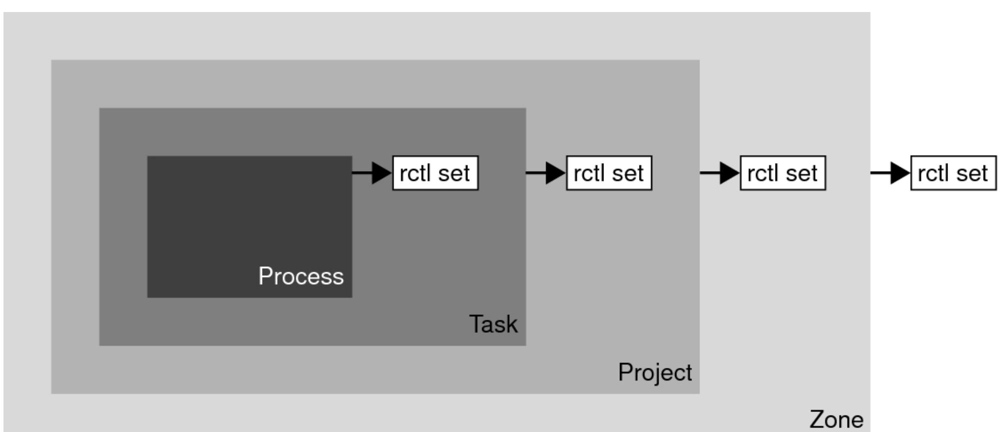
```
$ prctl -n process.max-cpu-time $$
process 353939: -ksh
NAME    PRIVILEGE    VALUE    FLAG   ACTION              RECIPIENT
 process.max-cpu-time
         privileged   18.4Es    inf   signal=XCPU                 -
         system       18.4Es    inf   none
```

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. An `inf` value has an infinite quantity. The value is never enforced. Hence, as configured, both threshold quantities represent infinite values that are never 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 3 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, "Administering Resource Controls Tasks" and the rctladm(8) man page for more information.

Applying Resource Controls

Each resource control listed in Table 1, "Standard Project, Task, and Process Resource Controls," on page 70 can be assigned to a project at login or when newtask, su, 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), cron(8), and su(8) 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 process, task, or project 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 root privileges can also modify privileged resource controls. System resource controls cannot be altered.

Commands Used With Resource Controls

The commands that are used with resource controls are shown in the following table.

Command Reference	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(8)`	Allows you to make runtime interrogations of and modifications to the resource controls facility, with global scope

The `resource-controls(7)` man page describes resource controls available through the project database, including units and scaling factors.

♦ ♦ ♦ CHAPTER 7

Administering Resource Controls Tasks

This chapter describes how to administer the resource controls facility.

For an overview of the resource controls facility, see Chapter 6, "About Resource Controls".

Administering Resource Controls Task Map

Task	Description	For Instructions
Set resource controls.	Set resource controls for a project in the `/etc/project` file.	"Setting Resource Controls" on page 84
Get or revise the resource control values for active processes, tasks, or projects, with local scope.	Make runtime interrogations of and modifications to the resource controls associated with an active process, task, or project on the system.	"Displaying Default Resource Control Values" on page 86
On a running system, view or update the global state of resource controls.	View the global logging state of each resource control on a system-wide basis. Also set up the level of `syslog` logging when controls are exceeded.	"Administering System Resource Controls" on page 92
Report status of active interprocess communication (IPC) facilities.	Display information about active interprocess communication (IPC) facilities. Observe which IPC objects are contributing to a project's usage.	"Displaying IPC Information" on page 92
Determine whether a web server is allocated sufficient CPU capacity.	Set a global action on a resource control. This action enables you to receive notice of any entity that has a resource control value that is set too low.	"How to Determine Whether a Web Server Is Allocated Enough CPU Capacity" on page 93

Setting Resource Controls

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

This procedure adds a project named `x-files` to the `/etc/project` file and sets a maximum number of LWPs for a task created in the project.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Create a project called `x-files` and set the maximum number of LWPs for each task created in the project to `3`.**

   ```
   $ projadd -K 'task.max-lwps=(privileged,3,deny)' x-files
   ```

3. **View the entry in the `/etc/project` file by using one of the following methods:**

   - **Run the `project -l` command.**

     ```
     $ projects -l
     system
             projid : 0
             comment: ""
             users  : (none)
             groups : (none)
             attribs:
     …
     x-files
             projid : 100
             comment: ""
             users  : (none)
             groups : (none)
             attribs: task.max-lwps=(privileged,3,deny)
     ```

   - **Display the `/etc/project` file contents.**

     ```
     $ cat /etc/project
     system:0:System:::
     …
     ```

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

Example 1 Sample Session

After implementing the steps in this procedure, when the user creates a new task in project `x-files` by joining the project with `newtask`, the user 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 $$
process: 111107: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
task.max-lwps
        usage            3
        privileged       3       -   deny                      -
        system       2.15G     max   deny                      -
$ id -p
uid=0(root) gid=1(other) projid=100(x-files)

$ ps -o project,taskid -p $$
 PROJECT TASKID
 x-files    73

$ csh        /* creates second LWP */

$ csh        /* creates third LWP */

$ csh        /* cannot create more LWPs */
Vfork failed
$
```

Example: 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.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Set resource controls on project `x-files`.**

```
$ projmod -s -K 'task.max-lwps=(basic,10,none),(privileged,500,deny);
process.max-file-descriptor=(basic,128,deny)' x-files      one line in file
```

The following controls are set:

- A `basic` control with no action on the maximum LWPs per task.
- 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 "Example: How to Set the Maximum Number of LWPs for Each Task in a Project" on page 84.
- A limit on the maximum file descriptors per process at the `basic` level, which forces the failure of any `open` call that exceeds the maximum.

3. **View the entry in the file by using one of the following methods:**

- **Run the `project -l` command.**

```
$ projects -l
…
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: process.max-file-descriptor=(basic,128,deny)
                 task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in
file
```

- **Display the contents of the `/etc/project` file.**

```
$ cat /etc/project
…
x-files:100::::process.max-file-descriptor=(basic,128,deny);
task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

Displaying Default Resource Control Values

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.

▼ How to Display Default Resource Control Values

Before You Begin This procedure must be used on a system on which no resource controls have been set or changed. There can be only non-default entries in the `/etc/system` file or in the `project` database.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Use the `prctl` command on any process, such as the current shell that is running.**

   Command output is similar to the following:

```
$ prctl $$
process: 3320: bash
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
process.max-port-events
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-msg-messages
        privileged      8.19K       -   deny                                 -
        system          4.29G     max   deny                                 -
process.max-msg-qbytes
        privileged      64.0KB      -   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-sem-ops
        privileged        512       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-sem-nsems
        privileged        512       -   deny                                 -
        system          32.8K     max   deny                                 -
process.max-address-space
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-descriptor
        basic             256       -   deny                              3320
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-core-size
        privileged      8.00EB    max   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-stack-size
        basic           10.0MB      -   deny                              3320
        privileged      32.0TB      -   deny                                 -
        system          32.0TB    max   deny                                 -
```

```
process.max-data-size
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-size
        privileged      8.00EB    max   deny,signal=XFSZ                     -
        system          8.00EB    max   deny                                 -
process.max-cpu-time
        privileged      18.4Es    inf   signal=XCPU                          -
        system          18.4Es    inf   none                                 -
task.max-cpu-time
        usage               0s
        system          18.4Es    inf   none                                 -
task.max-processes
        usage               2
        system          2.15G     max   deny                                 -
task.max-lwps
        usage               3
        system          2.15G     max   deny                                 -
project.max-contracts
        privileged      10.0K       -   deny                                 -
        system          2.15G     max   deny                                 -
project.max-locked-memory
        usage               0B
        system          16.0EB    max   deny                                 -
project.max-port-ids
        privileged      8.19K       -   deny                                 -
        system          65.5K     max   deny                                 -
project.max-shm-memory
        privileged       510MB      -   deny                                 -
        system          16.0EB    max   deny                                 -
project.max-shm-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-msg-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-sem-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-tasks
        usage               2
        system          2.15G     max   deny                                 -
project.max-processes
        usage               4
        system          2.15G     max   deny                                 -
project.max-lwps
        usage              11
        system          2.15G     max   deny                                 -
```

```
project.cpu-cap
        usage             0
        system          4.29G     inf   deny                                 -
project.cpu-shares
        usage             1
        privileged        1         -   none                                 -
        system          65.5K     max   none                                 -
zone.max-lofi
        usage             0
        system          18.4E     max   deny                                 -
zone.max-swap
        usage            180MB
        system          16.0EB    max   deny                                 -
zone.max-locked-memory
        usage             0B
        system          16.0EB    max   deny                                 -
zone.max-shm-memory
        system          16.0EB    max   deny                                 -
zone.max-shm-ids
        system          16.8M     max   deny                                 -
zone.max-sem-ids
        system          16.8M     max   deny                                 -
zone.max-msg-ids
        system          16.8M     max   deny                                 -
zone.max-processes
        usage            73
        system          2.15G     max   deny                                 -
zone.max-lwps
        usage            384
        system          2.15G     max   deny                                 -
zone.cpu-cap
        usage             0
        system          4.29G     inf   deny                                 -
zone.cpu-shares
        usage             1
        privileged        1         -   none                                 -
        system          65.5K     max   none
```

▼ How to Display Information for a Given Resource Control

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Display the maximum file descriptor for the current shell that is running.**

   Command output is similar to the following:

```
$ prctl -n process.max-file-descriptor $$
process: 110453: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION       RECIPIENT
process.max-file-descriptor
        basic             256       -   deny            11731
        privileged      65.5K       -   deny                -
        system          2.15G     max   deny
```

Example: How to Temporarily Change a Privileged Value

This example procedure uses the `prctl` command to temporarily add a new privileged value to deny the use of more than three LWPs per project for the `x-files` project. The result is comparable to the result in "Example: How to Set the Maximum Number of LWPs for Each Task in a Project" on page 84.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Join the `x-files` project.**

```
$ newtask -p x-files
```

3. **Verify that the correct project has been joined.**

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

4. **Add a new privileged value for `project.max-lwps` that limits the number of LWPs to three.**

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

5. **Verify the result.**

```
$ prctl -n project.max-lwps -i project x-files
process: 111108: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
```

```
project.max-lwps
        usage           203
        privileged     1000       -    deny                       -
        system        2.15G     max    deny                       -
```

How to Lower a Resource Control Value

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Change the lowest value of the `process.max-file-descriptor` resource control.**

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

Example: How to Display, Replace, and Verify the Value of a Control on a Project

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Display the value of `project.cpu-shares` in the project `group.staff`.**

   ```
   $ prctl -n project.cpu-shares -i project group.staff
   project: 2: group.staff
   NAME    PRIVILEGE       VALUE    FLAG   ACTION      RECIPIENT
   project.cpu-shares
           usage               1
           privileged          1       -   none                -
           system          65.5K     max   none
   ```

3. **Replace the current `project.cpu-shares` value `1` with the value `10`.**

   ```
   $ prctl -n project.cpu-shares -v 10 -r -i project group.staff
   ```

4. **Display the value of `project.cpu-shares` in the project `group.staff`.**

   ```
   $ prctl -n project.cpu-shares -i project group.staff
   project: 2: group.staff
   ```

```
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage               1
        privileged          1       -   none                               -
        system          65.5K     max   none
```

Administering System Resource Controls

Use the `rctladm` command to make runtime interrogations of and modifications to the global state of the resource controls facility. See the `rctladm(8)` man page for more information.

How to Administer System Resource Controls

For example, you can use `rctladm` with the `-e` option to enable the global `syslog` attribute of a resource control. When the control is exceeded, notification is logged at the specified `syslog` level. To enable the global `syslog` attribute of `process.max-file-descriptor`, type the following:

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

When used without arguments, the `rctladm` command displays the global flags, including the global type flag, for each resource control.

```
$ rctladm
process.max-port-events     syslog=off  [ deny count ]
process.max-msg-messages    syslog=off  [ deny count ]
process.max-msg-qbytes      syslog=off  [ deny bytes ]
process.max-sem-ops         syslog=off  [ deny count ]
process.max-sem-nsems       syslog=off  [ deny count ]
process.max-address-space   syslog=off  [ lowerable deny no-signal bytes ]
process.max-file-descriptor syslog=off  [ lowerable deny count ]
process.max-core-size       syslog=off  [ lowerable deny no-signal bytes ]
process.max-stack-size      syslog=off  [ lowerable deny no-signal bytes ]
…
```

Displaying IPC Information

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

How to Display IPC Information

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

```
$ 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 system's capabilities.

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

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

1. **Assume the `root` role.**

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Place a privileged (root-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`
```

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

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

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

   If the workload trips the resource control, you will see `/var/adm/messages` similar to the following:

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

♦ ♦ ♦ CHAPTER 8

About Fair Share Scheduler

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, "Administering the Fair Share Scheduler Tasks".

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 Oracle 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` and `priocntl` commands. Project shares used by the FSS are specified through the `project.cpu-shares` property in the `project` database.

Note - If you are using the `project.cpu-shares` resource control on an Oracle Solaris system with zones installed, see "Setting Zone-Wide Resource Controls" in *Oracle Solaris Zones Configuration Resources* and "Using the Fair Share Scheduler on a System With Zones Installed" in *Creating and Using Oracle Solaris Zones*.

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 Oracle 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.

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.

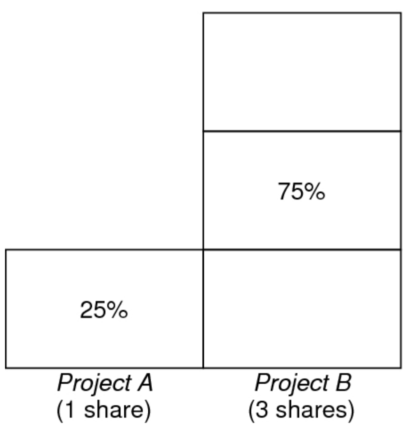
When run under the control of the FSS scheduler with $S_A=S_B$, these projects are also given approximately the same amounts of CPU resources. However, if the projects are given different numbers of shares, their CPU resource allocations are different.

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.

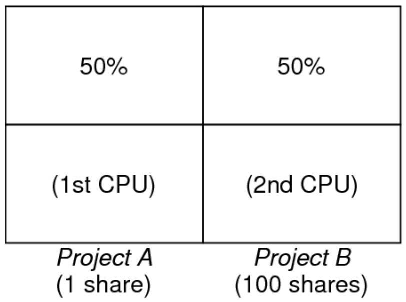
FIGURE 4 CPU Resource Allocation by Percentages



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.

FIGURE 5 CPU Resource Allocation for Projects With No Competition



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 will never be able to run because processes in project *A* always have higher priorities.

FIGURE 6 CPU Resource Allocation When a Project Is Not Assigned Shares

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 about the `project` database and name services, see “`project` Database” on page 26 and the `project(5)` man page.

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` 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 to `3` the value of the `project.cpu-shares` resource control for project `x-files` 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 about the options used in this example.

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.

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.

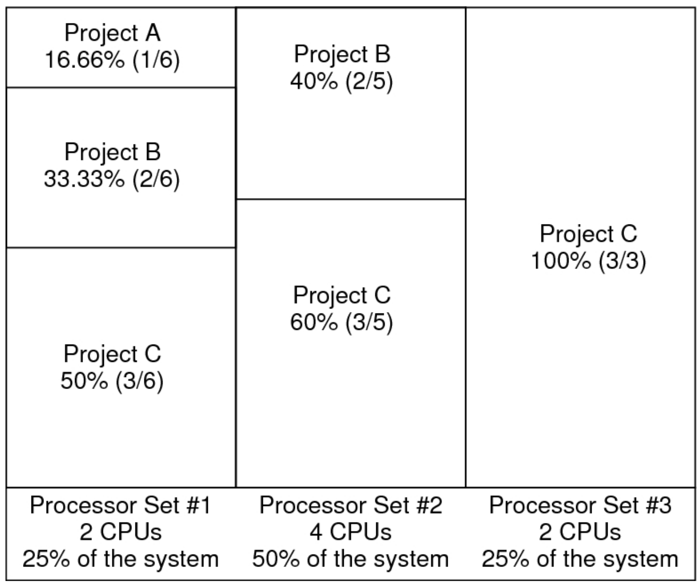
Empty processor sets (sets without processors in them) or processor sets without processes bound to them do not have any impact on the FSS scheduler behavior.

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.

FIGURE 7 CPU Resource Allocations for Multiple Processor Sets Using FSS



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

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

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	Allocation
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 Oracle Solaris system also offers a real-time (RT) scheduler to users with root 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*, or non-overlapping, 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
```

Setting the Scheduling Class for the System

To set the default scheduling class for the system, see "How to Make FSS the Default Scheduler Class" on page 109, "Using the Fair Share Scheduler on a System With Zones Installed" in *Creating and Using Oracle Solaris Zones*, and the `dispadmin(8)` man page. To move running processes into a different scheduling class, see "Configuring the FSS" on page 109 and the `priocntl(1)` man page.

Scheduling Class on a System with Zones Installed

Non-global zones use the default scheduling class for the system. If the system is updated with a new default scheduling class setting, non-global zones obtain the new setting when booted or rebooted.

The preferred way to use FSS in this case is to set FSS to be the system default scheduling class with the `dispadmin` command. All zones then benefit from getting a fair share of the system CPU resources. See "Using the Fair Share Scheduler on a System With Zones Installed" in *Creating and Using Oracle Solaris Zones* for more information on scheduling class when zones are in use.

For information about moving running processes into a different scheduling class without changing the default scheduling class and rebooting, see the `priocntl(1)` man page.

Commands Used With FSS

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

Command Reference	Description
`dispadmin(8)`	Lists the available schedulers on the system. Sets the default scheduler for the system. Also used to examine and tune the FSS scheduler's time quantum value.
`FSS(4)`	Describes the fair share scheduler (FSS).
`priocntl(1)`	Displays or sets scheduling parameters of specified processes, moves running processes into a different scheduling class.

Command Reference	Description
ps(1)	Lists information about running processes, identifies in which scheduling classes processor sets are running.

◆ ◆ ◆ CHAPTER 9

Administering the Fair Share Scheduler Tasks

This chapter describes how to use the fair share scheduler (FSS).

For an overview of the FSS, see Chapter 8, "About Fair Share Scheduler". For information about scheduling class when zones are in use, see "Fair Share Scheduler on a System With Zones Installed" in *Creating and Using Oracle Solaris Zones*.

Administering the Fair Share Scheduler Task Map

Task	Description	For Information
Monitor CPU usage.	Monitor the CPU usage of projects, and projects in processor sets.	"Monitoring the FSS" on page 108
Set the default scheduler class.	Make a scheduler such as the FSS the default scheduler for the system.	"How to Make FSS the Default Scheduler Class" on page 109
Move running processes from one scheduler class to a different scheduling class, such as the FSS class.	Manually move processes from one scheduling class to another scheduling class without changing the default scheduling class and rebooting.	"How to Manually Move Processes From the TS Class Into the FSS Class" on page 110
Move all running processes from all scheduling classes to a different scheduling class, such as the FSS class.	Manually move processes in all scheduling classes to another scheduling class without changing the default scheduling class and rebooting.	"How to Manually Move Processes From All User Classes Into the FSS Class" on page 111
Move a project's processes into a different scheduling class, such as the FSS class.	Manually move a project's processes from their current scheduling class to a different scheduling class.	"How to Manually Move a Project's Processes Into the FSS Class" on page 111
Examine and tune FSS parameters.	Tune the 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.	"How to Tune Scheduler Parameters" on page 112

Monitoring the FSS

You can use the `prstat` command described in the prstat(8) man page 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, "About Extended Accounting" for more information.

▼ How to Monitor System CPU Usage by Projects

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Monitor the CPU usage of projects that run on the system.**

   Command output is similar to the following:

```
$ prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  5107 root     4556K 3268K cpu0    59    0   0:00:00 0.0% prstat/1
  4570 root       83M   47M sleep   59    0   0:00:25 0.0% java/13
…
    10 root       15M   13M sleep   59    0   0:00:05 0.0% svc.startd/19
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        6   25M   18M   0.9%   0:00:00 0.0% user.root
     0       73  479M  284M    14%   0:02:31 0.0% system
     3        4   28M   24M   1.1%   0:00:26 0.0% default
    10        2   14M 7288K   0.3%   0:00:00 0.0% group.staff

Total: 85 processes, 553 lwps, load averages: 0.00, 0.00, 0.00
```

▼ How to Monitor CPU Usage by Projects in Processor Sets

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Monitor the CPU usage of projects on a list of processor sets.**

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

pset-list

Specifies a comma-delimited list of processor set IDs.

Configuring the FSS

The same commands that you use with other scheduling classes in the Oracle 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.

Note that you can use the `svcadm restart` command to restart the scheduler service. See the `svcadm`(8) man page for more information.

Listing the Scheduler Classes on the System

To display the scheduler classes on the system, use the `dispadmin` command with the `-l` option.

```
$ dispadmin -l
CONFIGURED CLASSES
==================

SYS     (System Class)
TS      (Time Sharing)
SDC     (System Duty-Cycle Class)
FSS     (Fair Share)
FX      (Fixed Priority)
IA      (Interactive)
```

▼ How to Make FSS the Default Scheduler Class

The FSS must be the default scheduler on your system to have CPU shares assignment take effect.

Using a combination of the `priocntl` and `dispadmin` commands ensures that the FSS becomes the default scheduler immediately and also after reboot.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Set the default scheduler for the system to be the FSS.**

   ```
   $ dispadmin -d FSS
   ```

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

3. **Make this configuration take effect immediately, without rebooting.**

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

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 procedure shows how to manually move processes from the TS scheduling class into the FSS scheduling class.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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
   ```

   Note - 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. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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
   ```

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

How to Manually 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. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Move processes that run in project ID *N* to the FSS scheduling class.**

   ```
   $ priocntl -s -c FSS -i projid N
   ```

   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 display or change process scheduler parameters while the system is running. For example, 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 while the system is running, type the following command:

```
$ 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.

```
$ 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

About Controlling Physical Memory With the Resource Capping Daemon (rcapd)

The resource capping daemon rcapd enables you to regulate physical memory consumption by processes running in projects that have resource caps defined.

If you are running zones on your system, you can use the rcapd daemon from the global zone to regulate physical memory consumption in non-global zones. See "Capped Memory and Physical Memory Control" in *Oracle Solaris Zones Configuration Resources*.

The following topics are covered in this chapter.

For procedures using the rcapd utility, see Chapter 11, "Administering the Resource Capping Daemon Tasks".

Introduction to the Resource Capping Daemon

The resource capping daemon rcapd and its associated utilities rcapadm and rcapstat provide mechanisms for defining, managing, and providing statistics on physical memory resource cap enforcement and administration.

A *resource cap* is an upper bound placed on the consumption of a resource, such as physical memory. Resource caps are supported on collections of system processes that can be grouped together, such as projects and zones.

Like the resource control, the resource cap can be defined by using attributes of project entries in the project database.

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(8) man page. For information about projects and the project database, see Chapter 2, "About Projects and Tasks" and the project(5) man page. For information about resource controls, see Chapter 6, "About Resource Controls".

How Resource Capping Works

The rcapd daemon repeatedly samples the resource utilization of projects and zones that have physical memory caps. 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 and zones with memory 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 Oracle 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 rcapd daemon can *page out*, or relocate, infrequently used pages to a swap device, which is an area outside of physical memory.

The rcapd 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 for Projects

To define a physical memory resource cap for a project, establish a resident set size (RSS) cap by adding the rcap.max-rss attribute to the project database entry. This attribute sets 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` file sets an RSS cap of 10 Gbytes 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.

You can also use the `projmod` command to set the `rcap.max-rss` attribute in the `/etc/project` file.

For more information, see "Setting the Resident Set Size Cap" on page 121.

Managing the Resource Capping Daemon

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

- Enable or disable resource capping
- Display the current status of the configured resource capping daemon

You must be the root user or have the required administrative rights to use `rcapadm`.

The `rcapadm` command can set the following parameters:

`rcapd mode`

Defines whether `rcapd` must enforce the caps or log the scan results. `rcapd` can support the following operation modes:

`log-only`

`rcapd` prints log messages for every `rss sample` interval for each collection. No further action is taken.

`pageout`

The default mode. `rcapd` prints log messages for `rss sample intervals` for each collection that exceeds its RSS cap. `rcapd` then enforces caps on the collection.

`rss sample interval`

The interval used for sampling resident set size (RSS) for each process collection. For each `rss sample interval` in seconds, the RSS of each collection is updated. This measurement is used by `rcapd` to enforce caps on a collection.

Configuration changes can be incorporated into the `rcapd` daemon on demand by sending a `SIGHUP` (see the `kill(1)` man page).

If used without arguments, the `rcapadm` command displays the current status of the resource capping daemon.

The following subsections discuss cap enforcement and cap values.

Using the Resource Capping Daemon on a System With Zones Installed

You can control resident set size (RSS) usage of a zone by setting the `capped-memory` resource when you configure the zone. For more information, see "Capped Memory and Physical Memory Control" in *Oracle Solaris Zones Configuration Resources*. To use the `capped-memory` resource, the `resource-cap` package must be installed in the global zone. You can run the `rcapd` daemon *within* a zone, including the global zone, to enforce memory caps on projects in that zone.

You can set a temporary cap for the maximum amount of memory that can be consumed by a specified zone, until the next reboot. See "Example: How to Specify a Temporary Resource Cap for a Zone" on page 124.

If you are using the `rcapd` daemon on a zone to regulate physical memory consumption by processes running in projects that have resource caps defined, you must configure the daemon in those zones.

When choosing memory caps for applications in different zones, you generally do not have to consider that the applications reside in different zones. The exception is per-zone services. Per-zone services consume memory. This memory consumption must be considered when determining the amount of physical memory for a system, as well as memory caps.

Unenforced Caps and `rcapd` Daemon Operations

If a process collection is found to exceed its cap for an extended period of time, the `rcapd` daemon can decide to stop or to resume enforcing caps for that collection. If the daemon stops enforcing caps, a detailed error message is logged reporting the cause and suggesting setting appropriate caps.

If at a later time the daemon determines that the `rss` of a collection can be reduced to match its cap set, the daemon can resume enforcing the caps.

You can use the following methods to force the `rcapd` daemon to resume enforcing caps:

- Set an appropriate cap on the collection. See "Determining Cap Values" on page 117.
- Examine the process log and restart the `rcapd` daemon.

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 the `rcapd` daemon.

In determining caps on projects, consider the following 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 system 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 of Capped Projects" on page 118 and the `rcapstat(1)` man page for more information.

Reporting on shared memory

The rcapd daemon reports the RSS of pages of memory that are shared with other processes or mapped multiple times within the same process as a reasonably accurate estimate. If processes in different projects share the same memory, then that memory is counted towards the RSS total for all projects sharing the memory.

The estimate is usable with workloads such as databases, which utilize shared memory extensively. For database workloads, you can also sample a project's regular usage to determine a suitable initial cap value by using output from the -J or -Z options of the prstat command. For more information, see the prstat(8) man page.

Monitoring Resource Utilization of Capped Projects

Use the rcapstat command to monitor the resource utilization of capped projects. To view an example rcapstat report, see "Reporting Resource Capping Statistics" on page 124.

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

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.

interval

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

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
at	The total amount of memory that rcapd attempted to page out since the last rcapstat sample.
avgat	The average amount of memory that rcapd attempted to page out during each sample cycle that occurred 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.
cap	The RSS cap defined for the project. See "Attribute to Limit Physical Memory Usage for Projects" on page 114 or the rcapd(8) man page for information about how to specify memory caps.

rcapstat Column Headings	Description
cappd	Informs if caps are enforced on the project.
id	The project ID of the capped project.
nproc	The number of processes in the project.
pg	The total amount of memory that rcapd successfully paged out since the last rcapstat sample.
project	The project name.
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.
vm	The total amount of virtual memory size used by processes in the project, including all mapped files and devices, in kilobytes (K), megabytes (M), or gigabytes (G).

Commands Used With the rcapd Daemon

Command Reference	Description
rcapadm(8)	Configures the resource capping daemon, displays the current status of the resource capping daemon if it has been configured, and enables or disables resource capping. Also used to set a temporary memory cap.
rcapd(8)	The resource capping daemon.
rcapstat(1)	Monitors the resource utilization of capped projects.

♦ ♦ ♦ CHAPTER 11

Administering the Resource Capping Daemon Tasks

This chapter contains procedures for configuring and using the resource capping daemon `rcapd`.

For an overview of `rcapd`, see Chapter 10, "About Controlling Physical Memory With the Resource Capping Daemon (`rcapd`)".

Setting the Resident Set Size Cap

Define a physical memory resource resident set size (RSS) cap for a project by adding an `rcap.max-rss` attribute to the `project` database entry.

▼ How to Add an `rcap.max-rss` Attribute for a Project

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Add the `rcap.max-rss` attribute to the `/etc/project` file.**

   `rcap.max-rss=`*value*

Example 2 RSS Project Cap

The following line in the `/etc/project` file sets an RSS cap of 10 GBytes for a project named `db`.

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

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

Configuring and Using the Resource Capping Daemon Task Map

Task	Description	For Instructions
Enable resource capping.	Activate resource capping on your system.	"How to Enable Resource Capping" on page 122
Disable resource capping.	Deactivate resource capping on your system.	"How to Disable Resource Capping" on page 123
Report cap and project information.	View example commands for producing reports.	Example 3, "Reporting Cap and Project Information," on page 125
Monitor a project's resident set size.	Produce a report on the resident set size of a project.	Example 4, "Monitoring the RSS of a Project," on page 125
Determine a project's working set size.	Produce a report on the working set size of a project.	Example 5, "Determining the Working Set Size of a Project," on page 126

Administering the Resource Capping Daemon

This section contains procedures for configuring the resource capping daemon with the `rcapadm` command. See "Managing the Resource Capping Daemon" on page 115 and the `rcapadm(8)` man page for more information. Using the `rcapadm` to specify a temporary resource cap for a zone is also covered.

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

▼ How to Enable Resource Capping

There are three ways to enable resource capping on your system. Enabling resource capping also sets the `/etc/rcap.conf` file with default values.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Enable the resource capping daemon.**

Use one of the following methods.

- **Turn on resource capping by using the `svcadm` command.**

  ```
  $ svcadm enable rcap
  ```

- **Enable the resource capping daemon so it starts now and also starts each time the system is booted.**

  ```
  $ rcapadm -E
  ```

- **Enable the resource capping daemon at boot only.**

  The `-n` option prevents the daemon from starting until the system is booted.

  ```
  $ rcapadm -n -E
  ```

▼ How to Disable Resource Capping

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

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Disable the resource capping daemon.**

   Use one of the following methods.

   - **Turn off resource capping by using the `svcadm` command.**

     ```
     $ svcadm disable rcap
     ```

   - **Disable the resource capping daemon so that it is stopped now and will not be started when the system is booted.**

     ```
     $ rcapadm -D
     ```

   - **Disable the resource capping daemon without stopping it now.**

     The `-n` option delays stopping the daemon until the system is booted.

     ```
     $ rcapadm -n -D
     ```

Tip - Use the `rcapadm -D` option to safely disable `rcapd`. If the daemon is terminated (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.

▼ Example: How to Specify a Temporary Resource Cap for a Zone

This example procedure allocates the maximum amount of memory that can be consumed by the specified zone. This value lasts only until the next reboot.

To set a persistent cap, instead use the `zonecfg` command.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Set a maximum memory value of 512 megabytes for the zone.**

   ```
   $ rcapadm -z my-zone -m 512M
   ```

Reporting Resource Capping Statistics

Use the `rcapstat` command to report resource capping statistics. "Monitoring Resource Utilization of Capped Projects" on page 118 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.

- Example 3, "Reporting Cap and Project Information," on page 125
- Example 4, "Monitoring the RSS of a Project," on page 125
- Example 5, "Determining the Working Set Size of a Project," on page 126

EXAMPLE 3 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  cappd  nproc     vm    rss   cap    at   avgat    pg  avgpg
112270   user1   Yes      24   123M    35M   50M   50M      0K 3312K     0K
 78194   user2   Yes       1  2368K    1856K 10M    0K      0K    0K     0K
    id project  cappd  nproc     vm    rss   cap    at   avgat    pg    avgpg
112270   user1    Yes     24   123M    35M   50M    0K      0K    0K       0K
 78194   user2    Yes      1  2368K  1856K   10M    0K      0K    0K       0K
    id project  cappd  nproc     vm    rss   cap    at    avgat   pg    avgpg
112270   user1    Yes     24   123M    35M   50M    0K      0K    0K       0K
 78194   user2    Yes      1  2368K  1928K   10M    0K      0K    0K       0K
    id project  cappd  nproc     vm    rss   cap    at    avgat   pg    avgpg
112270   user1    Yes     24   123M    35M   50M    0K      0K    0K       0K
 78194   user2    Yes      1  2368K  1928K   10M    0K      0K    0K       0K
    id project  cappd  nproc     vm    rss   cap    at    avgat   pg    avgpg
112270   user1    Yes     24   123M    35M   50M    0K      0K    0K       0K
 78194   user2    Yes      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.

EXAMPLE 4 Monitoring the RSS of a Project

The following example uses 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   cappd  nproc       vm       rss       cap    at     avgat     pg  avgpg
376565   user1   Yes        3    6249M     6144M     6144M  690M      220M  5528K  2764K
376565   user1   Yes        3    6249M     6144M     6144M    0M      131M  4912K  1637K
376565   user1   Yes        3    6249M     6171M     6144M   27M      147M  6048K  2016K
376565   user1   Yes        3    6249M     6146M     6144M 4872M      174M  4368K  1456K
376565   user1   Yes        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 that a page fault occurs when either a new page must be created or the system must copy in a page from a swap device.

EXAMPLE 5 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  cappd  nproc    vm   rss   cap    at  avgat    pg  avgpg
376565   user1    Yes      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1    Yes      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1    Yes      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1    Yes      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1    Yes      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1    Yes      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1    Yes      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1    Yes      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1    Yes      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1    Yes      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 system. 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 continuously incurring page faults.

While the cap on `user1` is 6 gigabytes, in every 5–second sample interval the RSS decreases and I/O increases as `rcapd` pages out some of the workload's memory. Shortly after a 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. The RSS then 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. Thus, the project required 6.1 gigabytes to perform the work it was doing at the time it was being observed.

Also see the `vmstat(8)` and `iostat(8)` man pages.

CHAPTER 12

About Resource Pools

This chapter discusses the following technologies:

- Resource pools, which are used for partitioning system resources
- Dynamic resource pools (DRPs), which dynamically adjust each resource pool's resource allocation to meet established system goals

To assign CPUs to processes by using processor affinity, or Multi-CPU binding, through the projects feature, see Chapter 2, "About Projects and Tasks" and Chapter 3, "Administering Projects and Tasks",

Resource pools and dynamic resource pools are services in the Oracle Solaris service management facility (SMF). Each of these services is enabled separately.

The following topics are covered in this chapter:

- "How Dynamic Resource Allocation Works" on page 148
- "Monitoring the Pools Facility and Resource Utilization" on page 151
- "Commands Used With the Resource Pools Facility" on page 153

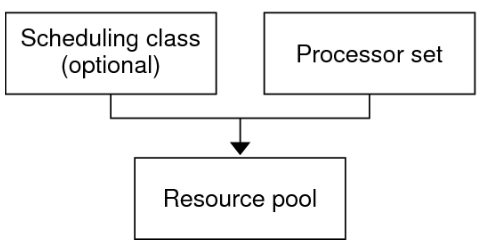
For procedures using this functionality, see Chapter 13, "Creating and Administering Resource Pools Tasks".

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 configuration and, optionally, scheduling class assignment.

FIGURE 8 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` file can have a single pool associated with that entry, which is specified using the `project.pool` attribute.

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. A CPU can only belong to one processor set. 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 processor set has the `pset.default` property set to `true`. Thus, both the default pool and the default processor set can be identified even if their names have been changed.

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

Introduction to Dynamic Resource Pools

Dynamic resource pools (DRPs) 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. The changes made to the configuration are logged. These capabilities are primarily enacted through the resource controller `poold`, a system daemon that should always be active when dynamic resource allocation is required. Periodically, the `poold` daemon 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 about the `poold` daemon, see the `poold`(8) man page.

About Enabling and Disabling Resource Pools and Dynamic Resource Pools

To enable and disable resource pools and dynamic resource pools, see "Enabling and Disabling the Pools Facility" on page 156.

Resource Pools Used in Zones

As an alternative to associating a zone with a configured resource pool on your system, you can use the `zonecfg` command to create a temporary pool that is in effect while the zone is running. See *Creating and Using Oracle Solaris Zones* for more information.

On a system that has zones enabled, a non-global zone can be associated with one resource pool, although the pool need not be exclusively assigned to a particular zone. Moreover, you cannot bind individual processes in non-global zones to a different pool by using the `poolbind` command from the global zone. To associate a non-global zone with a pool, see *Creating and Using Oracle Solaris Zones*.

Note that if you set a scheduling class for a pool and you associate a non-global zone with that pool, the zone uses that scheduling class by default.

If you are using dynamic resource pools, the scope of an executing instance of the `poold` daemon is limited to the global zone.

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

For information about resource pool commands, see "Commands Used With the Resource Pools Facility" on page 153.

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 system that is running only a fraction of the services that the system is ultimately expected to deliver. User difficulties can occur if reservation-based resource management mechanisms are not established when the system comes online.

For example, the fair share scheduler optimizes CPU utilization. The response times for a system 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` job. (See "Resource Pools Framework" on page 133.)

Real-time applications

Create a real-time pool by using the RT scheduler and designated processor resources.

System utilization

Enforce system goals that you establish.

Use the automated pools daemon functionality 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.

When the service management facility (SMF) or the `pooladm -e` command is used to enable the resource pools framework, then, if an `/etc/pooladm.conf` file exists, the configuration contained in the file is applied to the system.

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 in the following ways:

- Indirectly, by applying a static configuration file
- Directly, by using the `poolcfg` command with the `-d` option

More than one static pools configuration file can exist, for activation at different times. You can alternate between multiple pools configurations by invoking `pooladm` from a `cron` job. See the `cron`(8) man page for more information on the `cron` utility.

By default, the resource pools framework is not active. Resource pools must be enabled to create or modify the dynamic configuration. 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 on the configuration file, see "Creating Pools Configurations" on page 137.

The commands used with resource pools and the `poold` system daemon are described in the following man pages:

- `pooladm`(8)
- `poolbind`(8)
- `poolcfg`(8)
- `poold`(8)
- `poolstat`(8)
- `libpool`(3LIB)

`/etc/pooladm.conf` Contents

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

`cpu`

A processor definition

`pool`

A resource pool definition

`pset`

A processor set definition

`system`

Properties affecting the total behavior of the system

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` element is associated with a particular `Datatree` database.

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

For additional information about property types, see "`poold` Daemon Properties" on page 144.

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

Implementing Pools on a System

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

- When the Oracle Solaris operating system boots, an `init` script checks to see if the `/etc/pooladm.conf` file exists. If this file is found and pools are enabled, 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 system's resources are partitioned accordingly.

- When the Oracle Solaris operating 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, the `pooladm` command operates on the `/etc/pooladm.conf` file. However, you can optionally specify an alternate location and file name, and use that file to update the pools configuration.

For information about enabling and disabling resource pools, see "Enabling and Disabling the Pools Facility" on page 156. The pools facility cannot be disabled when there are user-defined pools or resources in use.

To configure resource pools, you must have root privileges or have the required rights profile.

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

`project.pool` Attribute

The `project.pool` attribute can be added to a project entry in the `/etc/project` file to associate a single pool with that entry. New work that is started on a project is bound to the appropriate pool. See Chapter 2, "About Projects and Tasks" for more information.

For example, the following `projmod` command sets the `project.pool` attribute for the project `sales` in the `/etc/project` file:

```
$ projmod -a -K project.pool=mypool sales
```

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 about the DR process and the force option, see the dynamic reconfiguration user guide for your Sun hardware.

If you are using dynamic resource pools, note that it is possible for a partition to move out of `poold` daemon control while the daemon is active. For more information, see "Identifying a Resource Shortage" on page 149.

Creating Pools Configurations

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

- `cpu`
- `pool`
- `pset`
- `system`

See the `poolcfg`(8) man page for more information about elements that be manipulated.

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

- 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 is the current method to use.

  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.
- If necessary for backward compatibility with previous releases, use the `poolcfg` command with the `-c` option and the `discover` or `create system` *name* subcommands to create a new pools configuration.

Note - Only use the `poolcfg` or `libpool` command to modify the `/etc/pooladm.conf` file. Do not directly edit this file.

Specific Assignment of CPUs, Cores, and Sockets

Use the subcommands `assign` and `unassign` assign specific CPUs, cores, and sockets.

The `assign` and `unassign` subcommands are applicable to both the persistent and runtime configurations of the pools. Using the `assign` subcommand and setting `pset.min` and `pset.max` directly are mutually exclusive. Each method overwrites the configuration set by the other.

CPUs configured to processor sets using the `pset.min` and `pset.max` properties are considered *allocated* rather than assigned. The `assign` and `unassign` subcommands add and remove specific CPUs to or from a `pset`, respectively. The first `assign` clears any `pset` element configuration set up by a previous allocation. Use the `unassign` subcommand only after a successful `assign` operation.

The `unassign` subcommand cannot be used to manipulate CPUs from allocated processor sets.

Also see "dedicated-cpu Resource Type" in *Oracle Solaris Zones Configuration Resources*.

Directly Manipulating the Dynamic Configuration

You can 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 170.

Note - If Dynamic resource pools are in use, the resource transfer might trigger action from the `poold` daemon. See "`poold` Daemon Overview" on page 138 for more information.

`poold` Daemon 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. The poold daemon 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, the daemon logs that user-specified objectives can no longer be achieved. Following a reconfiguration, the daemon resumes monitoring workload objectives.

The poold daemon 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.

Managing Dynamic Resource Pools

The dynamic resource pools service is managed by the service management facility (SMF) under the service identifier svc:/system/pools/dynamic.

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using the svcadm command. The service's status can be queried using the svcs command. See the svcs(1) andsvcadm(8) man pages for more information.

The SMF interface is the preferred method for controlling dynamic resource pools, but for backward compatibility, the following methods can also be used:

- If dynamic resource allocation is not required, the poold daemon can be stopped with the SIGQUIT or the SIGTERM signal. Either of these signals causes the daemon to terminate gracefully.
- Although the poold daemon will automatically detect changes in the resource or pools configuration, you can also force a reconfiguration to occur by using the SIGHUP signal.

Configuration Constraints and Objectives

When making changes to a configuration, the poold daemon acts on directions that you provide. You specify these directions as a series of constraints and objectives. The poold daemon uses your specifications to determine the relative value of different configuration possibilities in relation to the existing configuration. The daemon 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. See "`cpu.pinned` Property Constraint" on page 140.
- The importance factor of the pool. See "`pool.importance` Property Constraint" on page 140.
- Pinned components that are not available to be moved from a set. See "`pset.min` Property and `pset.max` Property Constraints" on page 140.

See the `libpool`(3LIB) man page and "Pools Properties" on page 135 for more information about pools properties. See "How to Set Pools Configuration Constraints" on page 165 for usage instructions.

`cpu.pinned` Property Constraint

The `cpu-pinned` property indicates that a particular CPU should not be moved by dynamic resource pools 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 4, "Defined Property Names," on page 144 for more details about this property.

`pool.importance` Property Constraint

The `pool.importance` property describes the relative importance of a pool as defined by the administrator.

`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 4, "Defined Property Names," on page 144 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 Oracle 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.

Configuration Objectives

Objectives are specified similarly to constraints. The full set of objectives is documented in Table 4, "Defined Property Names," on page 144.

Objectives fall into two categories:

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 workload objectives.

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

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

- `pset.poold.objectives`
- `system.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 Pools Configuration Objectives" on page 166 for usage examples.

`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 Oracle Solaris system to determine the distance between resources, using time as the measurement..

This objective can take one of the following three values:

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

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

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

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

`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 as follows:

<

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` element 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.

`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. The `wt-load` objective name means "weighted load".

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.

Configuration Objectives Example

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

- The `utilization` should be kept between 30 percent and 80 percent.

- The `locality` should be maximized for the processor set.
- The objectives should take the default importance of `1`.

EXAMPLE 6 `poold` Daemon Objectives Example

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

See "How to Define Pools Configuration Objectives" on page 166 for additional usage examples.

`poold` Daemon Properties

The `poold` daemon has four categories of properties:

- Configuration
- Constraint
- Objective
- Objective Parameter

TABLE 4 Defined Property Names

Property Name	Type	Category	Description
`cpu.pinned`	`bool`	Constraint	CPUs pinned to this processor set
`pool.importance`	`int64`	Objective parameter	User-assigned importance
`pset.max`	`uint64`	Constraint	Maximum number of CPUs for this processor set
`pset.min`	`uint64`	Constraint	Minimum number of CPUs for this processor set
`pset.poold.objectives`	`string`	Objective	Formatted string following the `poold` daemon's expression syntax
`system.poold.history-file`	`string`	Configuration	Decision history location
`system.poold.log-level`	`string`	Configuration	Logging level
`system.poold.log-location`	`string`	Configuration	Logging location
`system.poold.monitor-interval`	`uint64`	Configuration	Monitoring sample interval
`system.poold.objectives`	`string`	Objective	Formatted string following the `poold` daemon's objective expression syntax

poold Daemon Functionality That Can Be Configured

You can configure these aspects of the poold 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 the poold daemon.

poold Daemon Monitoring Interval

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

poold Daemon Logging Information

Three categories of information are provided through logging. These categories are identified in the following 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 the poold daemon to log all defined messages. The INFO level 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 to logging are available:

- ALERT
- CRIT
- DEBUG
- ERR
- INFO

- NOTICE
- WARNING

The parameter levels map directly onto their syslog equivalents. See "Logging Location" on page 148 for more information about using he syslog function.

For more information about how to configure poold daemon logging, see "How to Set the poold Daemon Logging Level" on page 168.

Configuration Information Logging

The following messages can be generated about libpool configuration:

ALERT

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

CRIT

Problems due to unanticipated failures. Causes the daemon to exit and requires immediate administrative attention.

DEBUG

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

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. The poold daemon 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 resources as pinned, which means that the poold daemon cannot move CPU resources between processor sets.

Monitoring Information Logging

The following types of monitoring messages can be generated:

`CRIT`

Problems due to unanticipated monitoring failures. Causes the daemon to exit and requires immediate administrative attention.

`DEBUG`

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

`ERR`

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

`INFO`

Messages about resource utilization statistics.

`NOTICE`

Messages about resource control region transitions.

Optimization Information Logging

The following message optimizations can be logged:

`DEBUG`

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

`INFO`

Messages about alternate configurations considered could be displayed.

`NOTICE`

Messages about usable configurations or configurations that will not be implemented due to overriding decision histories could be displayed.

`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 unforeseen limitations. Examples could include removing the last processor from a processor set which contains a bound resource consumer.

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 the `syslog`(3C) man page).

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

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

Log Management With the `logadm` Command

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

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

For more information, see the `logadm`(8) and `logadm.conf`(5) man pages.

How Dynamic Resource Allocation Works

This section explains the process and the factors that the `poold` daemon 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 Oracle Solaris instance.

On a system that has zones enabled, the scope of an executing instance of the `poold` daemon 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 Oracle 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 system 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 system cannot have the following setup because CPU 1 can appear in only one processor set at a time:

```
pset 0: 0 1
pset 1: 1 2 3
```

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

To discover the available resources, the poold daemon 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 the poold daemon uses in its operations. However, there are constraints upon this figure that limit the flexibility that the daemon has to make allocations. For information about available constraints, see "Configuration Constraints" on page 140.

Identifying a Resource Shortage

The control scope for the poold daemon is defined as the set of available resources for which the daemon 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 the daemon is active, the daemon tries to restore control through the judicious manipulation of available resources. If the daemon cannot locate additional resources within its scope, then the daemon logs information about the resource shortage.

Determining Resource Utilization

The poold daemon 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 shows the proportion of time that the resource is in use over the sample interval. Resource utilization is displayed as a percentage from 0 to 100.

Identifying Control Violations

The directives described in "Configuration Constraints and Objectives" on page 139 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 of an objective is detected by the daemon in the course of its workload monitoring.
- An asynchronous violation of an objective 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 that meet the objectives specified in the configuration for the scope of control are examined and evaluated.

This process is refined over time as the results of shifting resources 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 `poold` daemon cannot take corrective action, the condition is logged. For more information, see "`poold` Daemon Logging Information" on page 145.

Monitoring 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 pressure for resources.

The `poolstat` utility includes options that can be used to examine specific 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 is displayed.

For more information about the `poolstat` utility, see the `poolstat`(8) man page.
For `poolstat` task and usage information, see "Reporting Statistics for Pool-Related Resources" on page 174.

`poolstat` Utility Output

In default output format, the `poolstat` utility 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 for the `poolstat` utility are as follows:

`id` — Pool ID.

`load` — Absolute representation of the load that is put on the resource set.

For more information about this property, see the `libpool`(3LIB) man page.

`max`	Maximum resource set size.
`min`	Minimum resource set size.
`pool`	Pool name.
`rid`	Resource set ID.
`rset`	Resource set name.
`size`	Current resource set size.
`type`	Resource set type.
`used`	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 (`-`).

You can specify the following in `poolstat` output:

- The order of the columns
- The headings that appear

Tuning `poolstat` Operation Intervals

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

count

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

interval

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

If *interval* and *count* are not specified, 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 following commands, listed in man page format, provide the primary administrative interface to the pools facility. For information about using these commands on a system that has zones enabled, see "Resource Pools Used in Zones" on page 132.

`pooladm`(8)

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, the `pooladm` command prints out the current dynamic pools configuration.

`poolbind`(8)

Enables the manual binding of projects, tasks, and processes to a resource pool.

`poolcfg`(8)

Provides configuration operations on pools and sets. Configurations created using this tool are instantiated on a target host by using the `pooladm` command.

If run with the `info` subcommand argument to the `-c` option, the `poolcfg` command 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, the command `poolcfg -c info /tmp/newconfig` displays information about the static configuration contained in the file `/tmp/newconfig`.

`poold`(8)

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, the `poold` daemon logs the condition.

`poolstat`(8)

Displays statistics for pool-related resources. Simplifies performance analysis and provides information that supports system administrators in resource partitioning and repartitioning tasks. Options are provided for examining specified pools and reporting resource set-specific statistics.

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

♦ ♦ ♦ CHAPTER 13

Creating and Administering Resource Pools Tasks

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

For background information about resource pools, see Chapter 12, "About Resource Pools".

To assign CPUs to processes by using processor affinity, or Multi-CPU binding, through the projects feature, see Chapter 2, "About Projects and Tasks" and Chapter 3, "Administering Projects and Tasks",

Administering Resource Pools Task Map

Task	Description	For Instructions
Enable or disable resource pools and dynamic resource pools.	Activate or disable resource pools on your system.	"Enabling and Disabling the Pools Facility" on page 156
Create a static resource pools configuration.	Create a static configuration file that matches the current dynamic configuration. For more information, see "Resource Pools Framework" on page 133.	"How to Create a Static Pools Configuration" on page 159
Modify a resource pools configuration.	Revise a pools configuration on your system, for example, by creating additional pools.	"How to Modify a Pools Configuration" on page 161
Associate a resource pool with a scheduling class.	Associate a pool with a scheduling class so that all processes bound to the pool use the specified scheduler.	"How to Associate a Pool With a Scheduling Class" on page 163
Set configuration constraints and define configuration objectives.	Specify objectives for the `poold` daemon to consider when taking corrective action. For more information on configuration objectives, see "`poold` Daemon Overview" on page 138.	"How to Set Pools Configuration Constraints" on page 165 and "How to Define Pools Configuration Objectives" on page 166
Set the logging level.	Specify the level of logging information that the `poold` daemon generates.	"How to Set the `poold` Daemon Logging Level" on page 168
Use a text file with the `poolcfg` command.	Script the configuration that you pass to the `poolcfg` command.	"How to Use Command Files With the `poolcfg` Command" on page 169

Task	Description	For Instructions
Transfer resources in the kernel.	Transfer resources in the kernel. For example, transfer resources with specific IDs to a target set.	"Transferring Resources" on page 170
Activate a pools configuration.	Activate the configuration in the default configuration file.	"How to Activate a Pools Configuration" on page 171
Validate a pools configuration before you commit the configuration.	Validate a pools configuration to test what will happen when the validation occurs.	"How to Validate a Pools Configuration Before Committing the Configuration" on page 171
Remove a pools configuration from your system.	Return all associated resources, such as processor sets, to their default status.	"How to Remove a Pools Configuration" on page 171
Bind processes to a pool.	Manually associate a running process on your system with a resource pool.	"How to Bind Processes to a Pool" on page 172
Bind tasks or projects to a pool.	Associate tasks or projects with a resource pool.	"How to Bind Tasks or Projects to a Pool" on page 173
Bind new processes to a resource pool.	Add an attribute to each entry in the `project` database.	"How to Set the `project.pool` Attribute for a Project" on page 173
Use `project` attributes to bind a process to a different pool.	Modify the pool binding for new processes that are started.	"Example: How to Use `project` Attributes to Bind a Process to a Different Pool" on page 174
Use the `poolstat` utility to produce reports.	Produce multiple reports at specified intervals.	"Producing Multiple Reports at Specific Intervals" on page 175
Report resource set statistics.	Use the `poolstat` utility to report statistics for a processor set.	"Reporting Resource Set Statistics" on page 175

Enabling and Disabling the Pools Facility

You can enable and disable the resource pools and dynamic resource pools services on your system by using the `svcadm` command described in the `svcadm(8)` man page. The dynamic resource pools services are dependent on the resource pools service being enabled.

You can also use the `pooladm` command described in the `pooladm(8)` man page to perform the following tasks:

- Enable the pools facility so that pools can be manipulated
- Disable the pools facility so that pools cannot be manipulated

---

Note - When a system is upgraded, if the resource pools framework is enabled and an `/etc/pooladm.conf` file exists, the pools service is enabled and the configuration contained in the file is applied to the system.

▼ How to Enable and Disable the Resource Pools Services

1. **Become an administrator who is assigned the Service Configuration rights profile.**

   The `root` role has all service configuration rights. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Enable the resource pools service.**

   ```
   $ svcadm enable system/pools:default
   ```

3. **Enable the dynamic resource pools (DRP) service.**

   Perform Step 2 before performing this step.

   ```
   $ svcadm enable system/pools/dynamic:default
   ```

4. **(Optional) Disable the resource pools services.**

   To stop the dynamic pools service only, run the following command.

   ```
   $ svcadm disable system/pools/dynamic:default
   ```

   To stop all resource pools services, run the preceding and the following command.

   ```
   $ svcadm disable system/pools:default
   ```

Example 7 Effect on Dynamic Resource Pools When the Resource Pools Service Is Disabled

In this example, both resource pool services are online.

1. The administrator disables the resource pools service:

   ```
   $ svcadm disable svc:/system/pools:default
   ```

   The DRP service eventually moves to `offline` because the resource pools service has been disabled:

   ```
   $ svcs "*pool*"
   STATE          STIME    FMRI
   disabled       2017     svc:/system/pools:default
   offline        2017     svc:/system/pools/dynamic:default
   ```

2. The administrator determines why the DRP service is offline.

   ```
   $ svcs -x "*pool*"
   svc:/system/pools:default (resource pools framework)
   ```

```
 State: disabled since Sun Feb 04 02:36:15 2017
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(8)
   See: poolbind(8)
   See: poolcfg(8)
   See: poolstat(8)
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Sun Feb 04 02:36:15 2017
Reason: Service svc:/system/pools:default is disabled.
   See: http://support.oracle.com/msg/SMF-8000-GE
   See: poold(8)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

Resource pools must be online for DRP to be online.

3. So the administrator enables the resource pools service.

```
$ svcadm enable system/pools:default
```

After the resource pools service is re-enabled, the dynamic resource pools are again online:

```
$ svcs "*pool*"
STATE          STIME    FMRI
online         2017     svc:/system/pools:default
online         2017     svc:/system/pools/dynamic:default
```

Specific CPU Assignment

You can `assign` and `unassign` CPUs, cores, and sockets.

`cpus=`

List of CPUs assigned to zone.

`cores=`

List of cores assigned to zone.

`sockets=`

List of sockets assigned to zone.

EXAMPLE 8 Assigning Cores

This example assigns cores to the processor set `new`.

```
$ poolcfg -dc 'assign to pset new (core 0 ; core 1)'
```

EXAMPLE 9 Updating the Running Pool to Match the Persistent Static Configuration

This example assigns cores to the processor set `new` and makes the running pools match the static configuration.

```
$ poolcfg -c 'assign to pset new (core 0 ; core 1)'
$ pooladm -c
```

Configuring Pools

▼ How to Create a Static Pools Configuration

Use the `-s` option to `/usr/sbin/pooladm` to create a static configuration file that matches the current dynamic configuration, preserving changes across reboots. Unless a different file name is 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.

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

Before You Begin Enable pools on your system.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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. **Commit the configuration at `/etc/pooladm.conf`.**

```
$ pooladm -c
```

5. **(Optional) Copy the dynamic configuration to a static configuration file called `/tmp/backup`.**

```
$ pooladm -s /tmp/backup
```

How to Modify a Pools Configuration

To enhance your 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. **Assume the `root` role.**
   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Create the processor set.**

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

3. **Create the pool.**

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

4. **Join the pool and the processor set 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. **Commit the configuration at `/etc/pooladm.conf`.**

```
$ pooladm -c
```

7. **(Optional) Copy the dynamic configuration to a static configuration file named `/tmp/backup`.**

```
$ 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. This example associates the pool `pool_batch` with the fair share scheduler (FSS).

1. **Assume the `root` role.**
   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Modify the pool 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. **Commit the configuration at `/etc/pooladm.conf`.**

```
$ pooladm -c
```

5. **(Optional) Copy the dynamic configuration to a static configuration file.**

```
$ pooladm -s /tmp/backup
```

▼ How to Set Pools Configuration Constraints

Constraints affect the range of possible pools configurations by eliminating some of the potential changes that could be made to a configuration. This procedure shows how to set the `cpu.pinned` property.

In the following examples, `cpuid` is an integer.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Modify the `cpu.pinned` property in the static or dynamic configuration.**

   Use either of the following methods:

- **Modify the boot-time (static) configuration.**

  ```
  $ poolcfg -c 'modify cpu CPU-ID (boolean cpu.pinned = true)'
  ```

- **Modify the running (dynamic) configuration without modifying the boot-time configuration.**

  ```
  $ poolcfg -dc 'modify cpu CPU-ID (boolean cpu.pinned = true)'
  ```

How to Define Pools Configuration Objectives

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

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

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

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

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

3. **Disable the `locality` objective for the default processor set.**

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

4. **Disable the `locality` objective for the `pset_batch` processor set.**

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

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. **Commit the configuration at `/etc/pooladm.conf`.**

```
$ pooladm -c
```

7. **(Optional) Copy the dynamic configuration to a static configuration file called `/tmp/backup`.**

```
$ pooladm -s /tmp/backup
```

▼ How to Set the `poold` Daemon Logging Level

To specify the level of logging information that the `poold` daemon 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` Daemon Logging Information" on page 145 and the `poolcfg(8)` and `libpool(3LIB)` man pages.

You can also use the `poold` command at the command line to specify the level of logging information that the `poold` daemon generates.

1. **Assume the `root` role.**

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Set the logging level.**

```
$ /usr/lib/pool/poold -l level
```

For information about available parameters, see "`poold` Daemon Logging Information" on page 145. The default logging level is `NOTICE`.

How to Use Command Files With the `poolcfg` Command

The `poolcfg` command with the `-f` option can take input from a text file that contains `poolcfg` subcommand arguments to the `-c` option. This method is appropriate when you want a set of operations to be performed. 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.

Note that in command files, the `#` character acts as a comment mark for the rest of the line.

1. **Create the input file.**

```
$ 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. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

3. **Execute the command.**

```
$ /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 procedure moves two CPUs from processor set `pset1` to processor set `pset2` in the kernel.

▼ How to Move CPUs Between Processor Sets

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Move two CPUs from *pset1* to *pset2*.**

   The `from` and `to` subclauses can be used in any order. Only one `to` and `from` subclause is supported per command.

```
$ poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

Example 10 Alternative Method to Move CPUs Between Processor Sets

If specific known IDs of a resource type are to be transferred, an alternative syntax is provided. For example, the following command assigns two CPUs with IDs `0` and `2` to the `pset_large` processor set:

```
$ poolcfg -dc 'transfer to pset pset_large (cpu 0; cpu 2)'
```

Troubleshooting If a transfer fails because there are not enough resources to match the request or because the specified IDs cannot be located, the system displays an error message.

Activating and Removing Pool Configurations

Use the `pooladm` command to make a particular pool configuration active or to remove the currently active pool configuration. See the `pooladm(8)` man page 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".

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Commit the configuration at `/etc/pooladm.conf`.**

   ```
   $ pooladm -c
   ```

3. **(Optional) Copy the dynamic configuration to a static configuration file.**

   ```
   $ pooladm -s /tmp/backup
   ```

▼ How to Validate a Pools Configuration Before Committing the Configuration

This procedure attempts to validate a specified configuration file, to test what will happen when the validation occurs. The configuration will not actually be committed .Any error conditions encountered are displayed, but the configuration itself is not modified.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Test the validity of the configuration before committing it.**

   ```
   $ pooladm -n -c /path/configuration-file
   ```

▼ 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. **Assume the `root` role.**

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Remove the current active configuration.**

```
$ 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.

Mixing Scheduling Classes Within a Processor Set

You can safely mix processes in the TS and IA classes in the same processor set. Mixing other 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, use the `priocntl` command to move running processes into a different scheduling class. See "How to Manually Move Processes From the TS Class Into the FSS Class" on page 110. Also see the `priocntl(1)` man page.

Setting Pool Attributes and Binding to a Pool

You can set a `project.pool` attribute to associate a resource pool with a project.

You can bind a running process to a pool in two ways:

- You can use the `poolbind` command described in `poolbind(8)` command to bind a specific process to a named resource pool.
- You can use the `project.pool` attribute in the `project` database to identify the pool binding for a new login session or a task that is launched through the `newtask` command. See the `newtask(1)`, `projmod(8)`, and `project(5)` man pages.

▼ How to Bind Processes to a Pool

This procedure manually binds a process to a specified pool.

1. **Assume the `root` role.**

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Manually bind the process to the pool.**

```
$ poolbind -p pool-name $$
```

3. **Verify the pool binding for the process.**

```
$ poolbind -q $$
155509 pool-name
```

The system displays the process ID and the pool binding.

▼ How to Bind Tasks or Projects to a Pool

This procedure binds all processes in the *my-project* project to the *my-pool* pool.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Bind all processes in the *my-project* project to the *my-pool* pool.**

```
$ poolbind -i project -p my-pool my-project
```

▼ How to Set the `project.pool` Attribute for a Project

This procedure sets the `project.pool` attribute to bind a project's processes to a resource pool.

1. **Assume the `root` role.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Add a `project.pool` attribute to each entry in the `project` database.**

```
$ projmod -a -K project.pool=my-pool my-project
```

Example: How to Use `project` Attributes to Bind a Process to a Different Pool

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.

User `paul` can modify the pool binding for processes he starts. `paul` 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. **Verify the pool binding for the process.**

   Use a double dollar sign (`$$`) to pass the process number of the parent shell to the command.

   ```
   $ poolbind -q $$
   6384  pool backstage
   ```

   The system displays the process ID and the pool binding.

Reporting Statistics for Pool-Related Resources

The `poolstat` command is used to display statistics for pool-related resources. See "Monitoring the Pools Facility and Resource Utilization" on page 151 and the `poolstat`(8) man page for more information.

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

Displaying 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 uses the poolstat command with the -r option to report statistics for the processor set resource set. Note that the resource set pset_default is attached to more than one pool, so this processor set 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
```

♦ ♦ ♦ C H A P T E R 1 4

Resource Management Consolidated Configuration Example

This chapter reviews the resource management framework and describes a hypothetical server consolidation project.

Application Configurations to Be Consolidated Onto One Server

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 consolidation is done by a user assuming the `root` role.

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
Standalone database instance	Processes a large number of transactions and serves multiple time zones

Goals of Consolidated Configuration

The following configuration is used to consolidate the applications onto a single system that has the resource pools and the dynamic resource pools facilities enabled.

- 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 processor 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 Consolidated Configuration

Edit the `/etc/project` database file. Add entries to implement the required resource controls and to map users to resource pools, 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="wt-load")
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.

Enable DRP.

```
# svcadm enable pools/dynamic:default
```

Viewing the Consolidated Configuration Example

To view the framework configuration, which also contains default elements created by the system, type the `pooladm` command. Command output is similar to the following:

```
# pooladm
```

```
system host
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        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
```

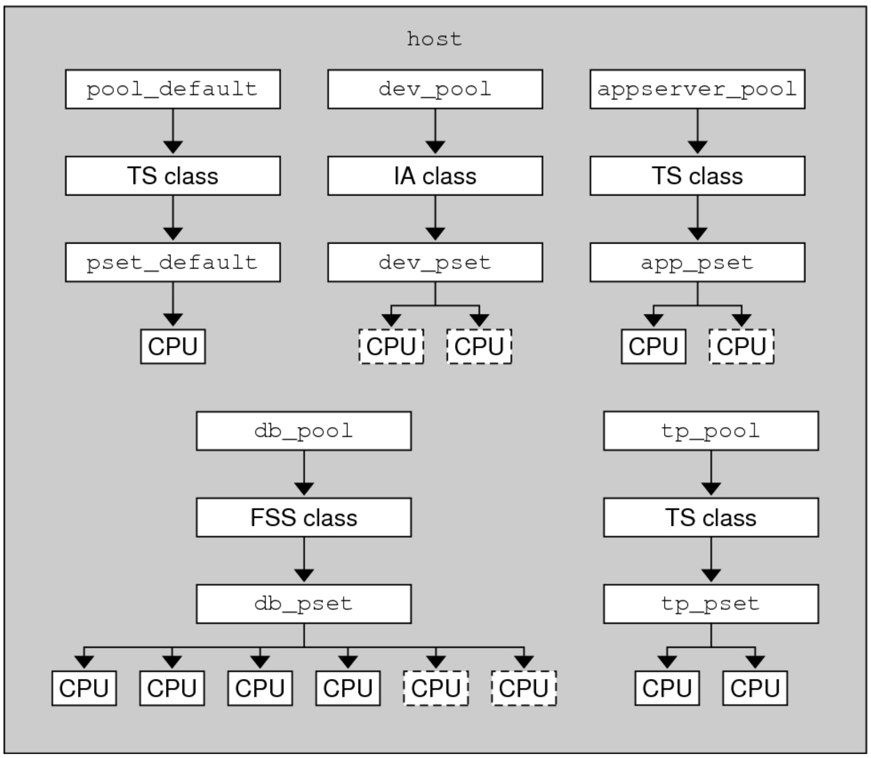
```
        string  pool.comment
        string  pool.scheduler TS
        pset    pset_default

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 9 Server Consolidation Configuration



Note - In the pool `db_pool`, the standalone database instance is guaranteed 75 percent of the CPU resource.

Index