

Comprehensive System Accounting for Linux

007-4315-001

CONTRIBUTORS

Written by Terry Schultz

Edited by Rick Thompson

Illustrated by Chris Wengelski

Production by Judi Holin

Engineering contributions by Marlys Kohnke

COPYRIGHT

© 2001 Silicon Graphics, Inc. All rights reserved; provided portions may be copyright in third parties, as indicated elsewhere herein. No permission is granted to copy, distribute, or create derivative works from the contents of this electronic documentation in any manner, in whole or in part, without the prior written permission of Silicon Graphics, Inc.

LIMITED RIGHTS LEGEND

The electronic (software) version of this document was developed at private expense; if acquired under an agreement with the USA government or any contractor thereto, it is acquired as "commercial computer software" subject to the provisions of its applicable license agreement, as specified in (a) 48 CFR 12.212 of the FAR; or, if acquired for Department of Defense units, (b) 48 CFR 227-7202 of the DoD FAR Supplement; or sections succeeding thereto. Contractor/manufacturer is Silicon Graphics, Inc., 1600 Amphitheatre Pkwy 2E, Mountain View, CA 94043-1351

TRADEMARKS AND ATTRIBUTIONS

Silicon Graphics is a registered trademark and CXFS, IRIS, IRIS FailSafe, IRIS InSight, IRIX, Origin, SGI, and the SGI logo are trademarks of Silicon Graphics, Inc.

Netscape is a trademark of Netscape Communications Corporation. Sun is a trademark of Sun Microsystems, Inc. PBS is a trademark of Veridian Corporation. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited. Windows is a trademark of Microsoft Corporation.

Cover Design By Sarah Bolles, Sarah Bolles Design, and Dany Galgani, SGI Technical Publications.

Record of Revision

Version	Description
001	January 2001 Original publication. Supports the Comprehensive System Accounting for Linux release 1.0.

Contents

About This Manual	xiii
Related Publications	xiii
Obtaining Publications	xiii
Conventions	xiv
Reader Comments	xiv
1. Comprehensive System Accounting	1
CSA Overview	3
Concepts and Terminology	6
Enabling or Disabling CSA	7
CSA Files and Directories	8
Files in the /var/csa Directory	8
Files in the /var/csa/ Directory	9
Files in the /var/csa/day Directory	10
Files in the /var/csa/work Directory	10
Files in the /var/csa/sum Directory	11
Files in the /var/csa/fiscal Directory	11
Files in the /var/csa/nite Directory	11
/usr/local/sbin Directory	14
/etc Directory	15
/etc/rc.d Directory	15
Comprehensive System Accounting Expanded Description	15
Daily Operation Overview	16
Setting Up CSA	17
007-4315-001	v

The <code>csarun</code> Command	21
Daily Invocation	22
Error and Status Messages	22
States	22
Restarting <code>csarun</code>	24
Verifying and Editing Data Files	25
CSA Data Processing	26
Data Recycling	30
How Jobs Are Terminated	30
Why Recycled Sessions Should Be Scrutinized	31
How to Remove Recycled Data	31
Adverse Effects of Removing Recycled Data	33
Workload Management Requests and Recycled Data	35
Tailoring CSA	36
System Billing Units (SBUs)	36
Process SBUs	37
Workload Management SBUs	39
Tape SBUs (deferred)	40
Daemon Accounting	40
Setting up User Exits	41
Charging for Workload Management Jobs	42
Tailoring CSA Shell Scripts and Commands	42
Using <code>at</code> to Execute <code>csarun</code>	43
Using an Alternate Configuration File	43
CSA Reports	43
CSA Daily Report	44
Consolidated Information Report	44

Unfinished Job Information Report	45
Disk Usage Report	45
Command Summary Report	46
Last Login Report	46
Daemon Usage Report	47
Periodic Report	48
Consolidated accounting report	48
Command summary report	49
CSA Man Pages	50
User-Level Man Pages	50
Administrator Man Pages	50
Index	53

Figures

Figure 1-1	Point of Entry Processes	1
Figure 1-2	The /var/csa Directory	9
Figure 1-3	CSA Data Processing	27

Tables

Table 1-1	Possible Effects of Removing Recycled Data	34
------------------	--	----

About This Manual

This manual describes the installation and administration of Comprehensive System Accounting (CSA) for Linux.

This publication was prepared in conjunction with Release of CSA for Linux product.

This manual contains the following sections:

- "CSA Overview", page 3
- "Concepts and Terminology", page 6
- "Enabling or Disabling CSA", page 7
- "CSA Files and Directories", page 8
- "Comprehensive System Accounting Expanded Description", page 15
- "CSA Reports", page 43
- "CSA Man Pages", page 50

Related Publications

For a list of CSA man pages, see "CSA Man Pages", page 50.

Obtaining Publications

To obtain SGI documentation, go to the SGI Technical Publications Library at:

<http://techpubs.sgi.com>.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
<code>command</code>	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
<i>variable</i>	Italic typeface denotes variable entries and words or concepts being defined.
user input	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number is located at the bottom of each page.)

You can contact us in any of the following ways:

- Send e-mail to the following address:
`techpubs@sgi.com`
- Use the Feedback option on the Technical Publications Library World Wide Web page:
`http://techpubs.sgi.com`
- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

Technical Publications
SGI
1600 Amphitheatre Pkwy., M/S 535
Mountain View, California 94043-1351

- Send a fax to the attention of "Technical Publications" at +1 650 932 0801.

We value your comments and will respond to them promptly.

Comprehensive System Accounting

Comprehensive System Accounting (CSA) provides detailed, accurate accounting data per job. It also provides data from some daemons. CSA is dependent on the concept of a Linux kernel job which is described in detail in the following paragraphs.

Work on a machine is submitted in a variety of ways, such as an interactive login, a submission from a workload management system, a cron job, or a remote access such as rsh, rcp, or array services. Each of these points of entry create an original shell process and multiple processes flow from that original point of entry. The Linux kernel job provides a means to limit the resource usage of all the processes resulting from a point of entry. A job is a group of related processes all descended from a point of entry process and identified by a unique job ID. A job can contain multiple process groups, sessions, or array sessions and all processes in one of these subgroups are always contained within one job. Figure 1-1, page 1, shows the point of entry processes that initiate the creation of jobs.

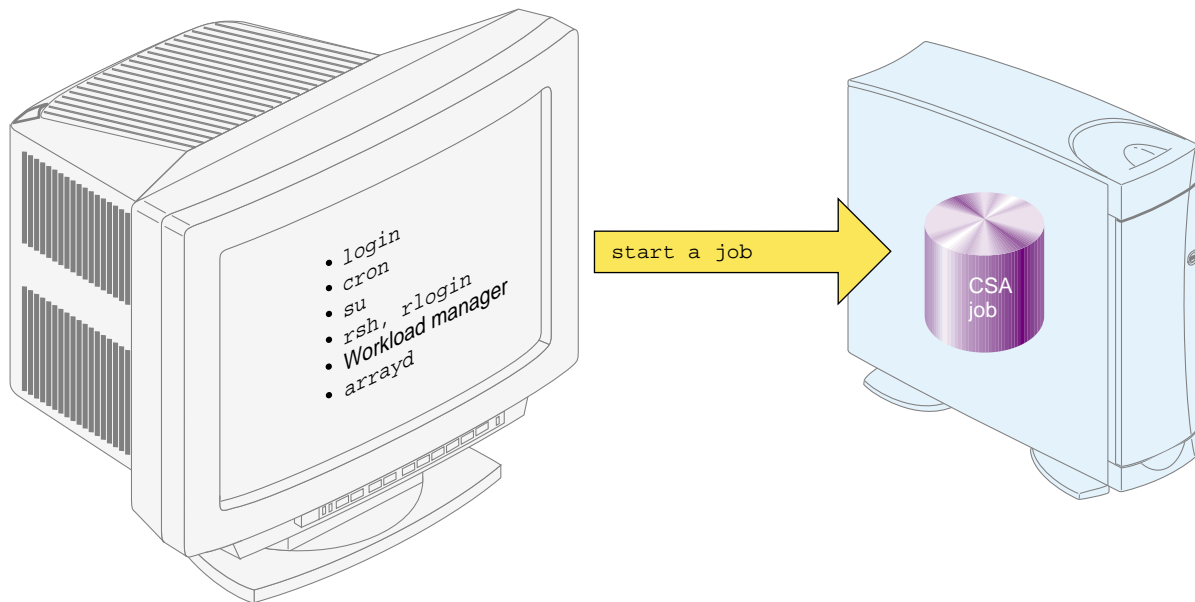


Figure 1-1 Point of Entry Processes

A Linux job has the following characteristics:

- A job is an inescapable container. A process cannot leave the job nor can a new process be created outside the job without explicit action, that is, a system call with root privilege.
- Each new process inherits the job ID from its parent process.
- All point of entry processes (job initiators) create a new job.
- The job initiator performs authentication and security checks.

The process control initialization process (`init(8)`) and startup scripts called by `init` are not part of a job and have a job ID of zero.

The `csarun(8)` command, usually initiated by the `cron(8)` command, directs the processing of the CSA daily accounting files. The `csarun(8)` command processes accounting records written into the CSA accounting data file.

Using accounting data, you can determine how system resources were used and if a particular user has used more than a reasonable share; trace significant system events, such as security breaches, by examining the list of all processes invoked by a particular user at a particular time; and set up billing systems to charge login accounts for using system resources.

This chapter contains the following sections:

- "CSA Overview", page 3
- "Concepts and Terminology", page 6
- "Enabling or Disabling CSA", page 7
- "CSA Files and Directories", page 8
- "Comprehensive System Accounting Expanded Description", page 15
- "CSA Reports", page 43
- "CSA Man Pages", page 50

CSA Overview

Comprehensive System Accounting (CSA) is a set of C programs and shell scripts that, like the other accounting packages, provide methods for collecting per-process resource usage data, monitoring disk usage, and charging fees to specific login accounts. CSA provides:

- Per-job accounting
- Daemon accounting (workload management systems and tape systems; note that tape daemon accounting is deferred)
- Flexible accounting periods (daily and periodic (monthly) accounting reports can be generated as often as desired and are not restricted to once per day or once per month)
- Flexible system billing units (SBUs)
- Offline archiving of accounting data
- User exits for site specific customizing of daily and periodic (monthly) accounting
- Configurable parameters within the `/etc/csa.conf` file
- User job accounting (`ja(1)` command)

CSA takes this per-process accounting information and combines it by job identifier (`jid`) within system boot uptime periods. CSA accounting for a job consists of all accounting data for a given job identifier during a single system boot period.

However, since workload management jobs may span multiple reboots and thereby consist of multiple job identifiers, CSA accounting for these jobs includes the accounting data associated with the workload management identifier.

Daemon accounting records are written at the completion of daemon specific events. These records are combined with per-process accounting records associated with the same job.

By default, CSA only reports accounting data for terminated jobs. Interactive jobs, `cron` jobs and `at` jobs terminate when the last process in the job exits, which is normally the login shell. A workload management job is recognized as terminated by CSA based upon daemon accounting records and an end-of-job record for that job. Jobs which are still active are recycled into the next accounting period. This behavior can be changed through use of the `csarun` command `-A` option.

A system billing unit (SBU) is a unit of measure that reflects use of machine resources. SBUs are defined in the CSA configuration file `/etc/csa.conf` and are set to `0.0` by default. The weighting factor associated with each field in the CSA accounting records can be altered to obtain an SBU value suitable for your site. For more information on SBUs, see "System Billing Units (SBUs)", page 36.

The CSA accounting records are written into a separate CSA `/var/csa/day/pacct` file. The CSA commands can only be used with CSA generated accounting records.

There are four user exits available with the `csarun(8)` daily accounting script. There is one user exit available with the `csaperiod(8)` monthly accounting script. These user exits allow sites to tailor the daily and monthly run of accounting to their specific needs by creating user exit scripts to perform any additional processing and to allow archiving of accounting data. See the `csarun(8)` and `csaperiod(8)` man pages for further information.

CSA provides two user accounting commands, `csacom(1)` and `ja(1)`. The `csacom` command reads the CSA `pacct` file and writes selected accounting records to standard output. The `ja` command provides job accounting information for the current job of the caller. This information is obtained from a separate user job accounting file to which the kernel writes. See the `csacom(1)` and `ja(1)` man pages for further information.

The `/etc/csa.conf` file contains CSA configuration variables. These variables are used by the CSA commands.

Like any accounting or monitoring package, the CSA features do contribute to overall system overhead. For this reason, CSA is disabled in the kernel by default. To enable CSA, see "Enabling or Disabling CSA", page 7.

Concepts and Terminology

The following concepts and terms are important to understand when using the accounting features:

Term	Description
Daily accounting	<p>Daily accounting is the processing, organizing, and reporting of the raw accounting data, generally performed once per day.</p> <p>In CSA, daily accounting can be run as many times as necessary during a day; however, this feature is still referred to as daily accounting.</p>
Job	<p>A job is a grouping of processes that the system treats as a single entity and is identified by a unique job identifier (job ID).</p> <p>CSA is the only accounting type to organize accounting data by jobs and boot times and then place the data into a sorted <code>pacct</code> file.</p> <p>For non-workload management jobs, a job consists of all accounting data for a given job ID during a single boot period.</p> <p>A workload management job consists of the accounting data for all job IDs associated with the job's workload management request ID. Workload management jobs may span multiple boot periods. If a job is restarted, it has the same job ID associated with it during all boot periods in which it runs. Rerun workload management jobs have multiple job IDs. CSA treats all phases of a workload management job as being in the same job.</p>
Periodic accounting	<p>Periodic (monthly) accounting further processes, reports, and summarizes the daily accounting reports to give a higher level view of how the system is being used.</p> <p>CSA lets system administrators specify the time periods for which monthly or cumulative accounting is to be run. Thus, periodic accounting can be run more than</p>

	once a month, but sometimes is still referred to as monthly accounting.
Daemon accounting	Daemon accounting is the processing, organizing, and reporting of the raw accounting data, performed at the completion of daemon specific events.
Recycled data	Recycled data is data left in the raw accounting data file, saved for the next accounting report run. By default, accounting data for active jobs is recycled until the job terminates. CSA reports only data for terminated jobs unless <code>csarun</code> is invoked with the <code>-A</code> option. <code>csarun</code> places recycled data into the <code>/var/csa/day/pacct0</code> data file.

The following abbreviations and definitions are used throughout this chapter:

Abbreviation	Definition
<i>MMDD</i>	Month, day
<i>hhmm</i>	Hour, minute

Enabling or Disabling CSA

The following steps are required to set up CSA job accounting:

Note: Because Linux CSA come preinstalled on your system, you can skip step 1 in the following procedure.

1. Install Linux CSA using the instructions in the `README.csa` file from the download link at:

`http://oss.sgi.com/projects/csa`
2. Configure CSA on across system reboots by using the `chkconfig(8)` utility as follows:

`chkconfig --add csaacct`
3. Modify the CSA configuration variables in `/etc/csa.conf` as desired.

4. Use the `csaswitch(8)` command to configure on the accounting record types and thresholds defined in `/etc/csa.conf` as follows:

```
csaswitch -c on
```

This step will be done automatically for subsequent system reboots when CSA is configured on via the `chkconfig(8)` utility.

For information on adding entries to the `crontabs` file so that the `cron(1M)` command automatically runs daily accounting, see "Setting Up CSA", page 17.

The following steps are required to disable CSA job accounting:

1. To turn off CSA, use the `csaswitch(8)` command:

```
csaswitch -c halt
```

2. To stop CSA from initiating after a system reboot, use the `chkconfig(8)` command:

```
chkconfig --del csaacct
```

CSA Files and Directories

The following sections describe the CSA files and directories.

Files in the `/var/csa` Directory

The `/var/csa` directory contains CSA data and report files within various subdirectories. `/var/csa` contains data collection files used by CSA. CSA accesses `pacct` files to process system accounting data.. The following diagram shows the directory and file layout for CSA:

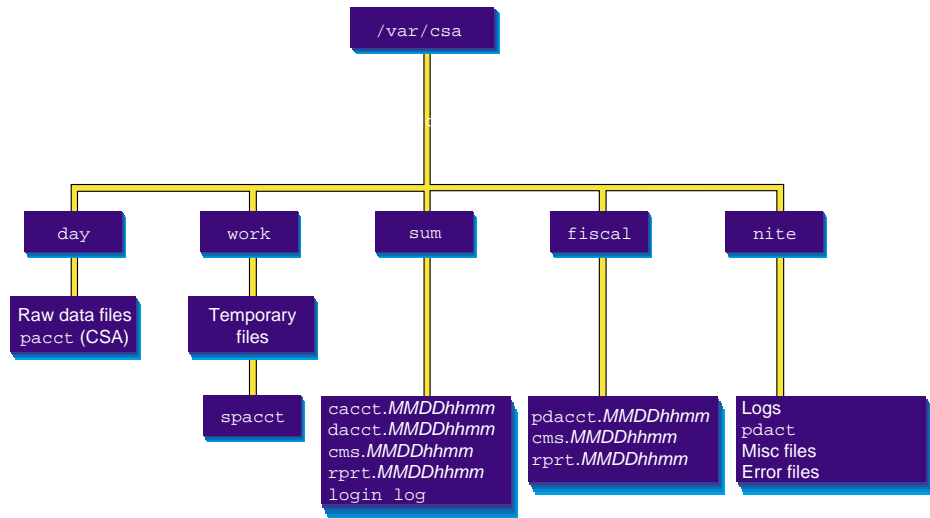


Figure 1-2 The /var/csa Directory

Each data and report file for CSA has a month-day-hour-minute suffix.

Files in the /var/csa/ Directory

The /var/csa directory contains the following directories:

Directory	Description
day	Contains the current raw accounting data files in pacct format.
work	Used by CSA as a temporary work area. Contains raw files that were moved from /var/csa/day at the start of an CSA daily accounting run and the spacct file.
sum	Contains the cumulative daily accounting summary files and reports created by csarun(8). The ASCII format is in /var/csa/sum/rprt.MMDDhhmm. The binary data is in /var/csa/sum/cacct.MMDDhhmm, /var/csa/sum/cms.MMDDhhmm, and /var/csa/sum/dacct.MMDDhhmm.

fiscal	Contains periodic accounting summary files and reports created by <code>csaperiod(8)</code> . The ASCII format is in <code>/var/csa/fiscal/csa/rprt.MMDDhhmm</code> . The binary data is in <code>/usr/csa/fiscal/cms.MMDDhhmm</code> and <code>/usr/csa/fiscal/pdacct.MMDDhhmm</code> .
nite	Contains log files, <code>csarun</code> state, and execution times files.

Files in the `/var/csa/day` Directory

The following files are located in the `/var/csa/day` directory:

File	Description
<code>dodiskerr</code>	Disk accounting error file.
<code>pacct</code>	Process and daemon accounting data.
<code>pacct0</code>	Recycled process and daemon accounting data.
<code>dtmp</code>	Disk accounting data (ASCII) created by <code>dodisk</code> .

Files in the `/var/csa/work` Directory

The following files are located in the `/var/csa/work/MMDD/hhmm` directory:

File	Description
<code>BAD.Wpacct*</code>	Unprocessed accounting data containing invalid records (verified by <code>csaverify(8)</code>).
<code>Ever.tmp1</code>	Data verification work file.
<code>Ever.tmp2</code>	Data verification work file.
<code>Rpacct0</code>	Process and daemon accounting data to be recycled in the next accounting run.
<code>Wdiskcacct</code>	Disk accounting data (<code>cacct.h</code> format) created by <code>dodisk(8)</code> (See the <code>dodisk(8)</code> man page).
<code>Wdtmp</code>	Disk accounting data (ASCII) created by <code>dodisk(8)</code> .
<code>Wpacct*</code>	Raw process and daemon accounting data.

spacct sorted pacct file

Files in the `/var/csa/sum` Directory

The following data files are located in the `/var/csa/sum` directory:

File	Description
<code>cacct.MMDDhhmm</code>	Consolidated daily data in <code>cacct.h</code> format. This file is deleted by <code>csaperiod</code> if the <code>-r</code> option is specified.
<code>cms.MMDDhhmm</code>	Daily command usage data in command summary (<code>cms</code>) record format. This file is deleted by <code>csaperiod</code> if the <code>-r</code> option is specified.
<code>dacct.MMDDhhmm</code>	Daily disk usage data in <code>cacct.h</code> format. This file is deleted by <code>csaperiod</code> if the <code>-r</code> option is specified.
<code>loginlog</code>	Login record file created by <code>lastlogin</code> .
<code>rprrt.MMDDhhmm</code>	Daily accounting report.

Files in the `/var/csa/fiscal` Directory

The following files are located in the `/var/csa/fiscal` directory:

File	Description
<code>cms.MMDDhhmm</code>	Periodic command usage data in command summary (<code>cms</code>) record format.
<code>pdacct.MMDDhhmm</code>	Consolidated periodic data.
<code>rprrt.MMDDhhmm</code>	Periodic accounting report.

Files in the `/var/csa/nite` Directory

The following files are located in the `/var/csa/nite` directory:

File	Description
<code>active</code>	Used by the <code>csarun(8)</code> command to record progress and print warning and error messages.
<code>activeMMDDhhmm</code>	<code>activeMMDDhhmm</code> is the same as <code>active</code> after <code>csarun</code> detects an error.

<code>clastdate</code>	Last two times <code>csarun</code> was executed; in <i>MMDDhhmm</i> format.
<code>dk2log</code>	Diagnostic output created during execution of <code>dodisk</code> (see the cron entry for <code>dodisk</code> in "Setting Up CSA", page 17).
<code>diskcacct</code>	Disk accounting records in <code>cacct.h</code> format, created by <code>dodisk</code> .
<code>EaddcMMDDhhmm</code>	Error/warning messages from the <code>csaaddc(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Earc1MMDDhhmm</code>	Error/warning messages from the <code>csa.archive1(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Earc2MMDDhhmm</code>	Error/warning messages from the <code>csa.archive2(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Ebld.MMDDhhmm</code>	Error/warning messages from the <code>csabuild(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Ecnd.MMDDhhmm</code>	Error/warning messages from the <code>csacms(8)</code> command when generating an ASCII report for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Ecms.MMDDhhmm</code>	Error/warning messages from the <code>csacms(8)</code> command when generating binary data for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Econ.MMDDhhmm</code>	Error/warning messages from the <code>csacon(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Ecrep.MMDDhhmm</code>	Error/warning messages from the <code>csacrep(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Ecrpt.MMDDhhmm</code>	Error/warning messages from the <code>csacrep(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .
<code>Edrpt.MMDDhhmm</code>	Error/warning messages from the <code>csadrep(8)</code> command for an accounting run done on <i>MMDD</i> at <i>hhmm</i> .

<code>Erec.MMDDhhmm</code>	Error/warning messages from the <code>csarecy(8)</code> command for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>Euser.MMDDhhmm</code>	Error/warning messages from the <code>csa.user(8)</code> user exit for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>Epuser.MMDDhhmm</code>	Error/warning messages from the <code>csa.puser(8)</code> user exit for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>Ever.tmp1MMDDhhmm</code>	Output file from invalid record offsets from the <code>csaverify(8)</code> command for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>Ever.tmp2MMDDhhmm</code>	Error/warning messages from the <code>csaverify(8)</code> command for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>Ever.MMDDhhmm</code>	Error/warning messages from the <code>csaedit(8)</code> and <code>csaverify(8)</code> command (from the <code>Ever.tmp2</code> file) for an accounting run done on <code>MMDD</code> at <code>hhmm</code> .
<code>fd2log</code>	Diagnostic output created during execution of <code>csarun</code> (see cron entry for <code>csarun</code> in "Setting Up CSA", page 17).
<code>lock lock1</code>	Used to control serial use of the <code>csarun(8)</code> comand.
<code>pd2log</code>	Diagnostic output created during execution of <code>csaperiod</code> (see cron entry for <code>csaperiod</code> in "Setting Up CSA", page 17).
<code>pdact</code>	Progress and status of <code>csaperiod</code> . <code>pdact.MMDDhhmm</code> is the same as <code>pdact</code> after <code>csaperiod</code> detects an error.
<code>statefile</code>	Used to record current state during execution of the <code>csarun</code> command.

/usr/local/sbin Directory

The `/usr/local/sbin` directory contains the following commands and shell scripts used by CSA:

Command	Description
<code>csaaddc</code>	Combines <i>cacct</i> records.
<code>csabuild</code>	Organizes accounting records into job records.
<code>csachargefee</code>	Charges a fee to a user.
<code>csackpacct</code>	Checks the size of the CSA process accounting file.
<code>csacms</code>	Summarizes command usage from per-process accounting records.
<code>csacon</code>	Condenses records from the <code>sorted pacct</code> file.
<code>csacrep</code>	Reports on consolidated accounting data.
<code>csadrep</code>	Reports daemon usage.
<code>csaedit</code>	Displays and edits the accounting information.
<code>csagetconfig</code>	Searches the accounting configuration file for the specified argument.
<code>csajrep</code>	Prints a job report from the <code>sorted pacct</code> file.
<code>csaperiod</code>	Runs periodic accounting.
<code>csarecy</code>	Recycles unfinished job records into next accounting run.
<code>csarun</code>	Processes the daily accounting files and generates reports.
<code>csaswitch</code>	Checks the status of, enables or disables the different types of Comprehensive System Accounting (CSA), and switches accounting files for maintainability.
<code>csaverify</code>	Verifies that the accounting records are valid.

The `/usr/local/bin` directory contains user commands associated with CSA:

Command	Description
<code>ja</code>	Starts and stops user job accounting information.
<code>csacom</code>	Searches and prints the CSA process accounting files.

The `/usr/local/sbin` directory may also contain the following scripts if your site uses the accounting user exits:

Script	Description
<code>csa.archive1</code>	Site-generated user exit for <code>csarun</code> .
<code>csa.archive2</code>	Site-generated user exit for <code>csarun</code> .
<code>csa.fef</code>	Site-generated user exit for <code>csarun</code> .
<code>csa.user</code>	Site-generated user exit for <code>csarun</code> .
<code>csa.puser</code>	Site-generated user exit for <code>csaperiod</code> .

`/etc` Directory

The `/etc` directory is the location of the `csa.conf` file that contains the parameter labels and values used by CSA software.

`/etc/rc.d` Directory

The `/etc/rc.d/init.d` directory is the location of the `csaacct` file used by the `chkconfig(8)` command. Use a text editor to add any `csaswitch(8)` options to be passed to `csaswitch` during system startup only.

Comprehensive System Accounting Expanded Description

This section contains detailed information about CSA and covers the following topics:

- "Daily Operation Overview", page 16
- "Setting Up CSA", page 17
- "The `csarun` Command", page 21
- "Verifying and Editing Data Files", page 25

- "CSA Data Processing", page 26
- "Data Recycling", page 30
- "Tailoring CSA", page 36

Daily Operation Overview

When the Linux operating system is run in multiuser mode, accounting behaves in a manner similar to the following process. However, because sites may customize CSA, the following may not reflect the actual process at a particular site:

1. When CSA accounting is enabled and the system is switched to multiuser mode, the `/usr/local/sbin/csaswitch` (see the `csaswitch(8)` man page) command is called by `local/sbin/rc.d/init.d/CSAAcct`.
2. By default, `csa`, memory, and I/O record types are enabled in `/etc/csa.conf`. However, to run workload management and tape daemon accounting, you must modify the `/etc/csa.conf` and the appropriate subsystem. For more information, see "Setting Up CSA", page 17.
3. The amount of disk space used by each user is determined periodically. The `/usr/local/sbin/dodisk` command (see `dodisk(8)`) is run periodically by the `cron` command to generate a snapshot of the amount of disk space being used by each user. The `dodisk` command should be run at most once for each time `/usr/local/sbin/csarun` is run (see `csarun(8)`). Multiple invocations of `dodisk` during the same accounting period write over previous `dodisk` output.
4. A fee file is created. Sites desiring to charge fees to certain users can do so by invoking `/usr/local/sbin/csachargefee` (see `csachargefee(8)`). Each accounting period's fee file (`/var/csa/day/fee`) is merged into the consolidated accounting records by `/usr/local/sbin/csaperiod` (see `csaperiod(8)`).
5. Daily accounting is run. At specified times during the day, `csarun` is executed by the `cron` command to process the current accounting data. The output from `csarun` is daily accounting files and an ASCII report.
6. Periodic (monthly) accounting is run. At a specific time during the day, or on certain days of the month, `/usr/local/sbin/csaperiod` (see `csaperiod`) is executed by the `cron` command to process consolidated accounting data from previous accounting periods. The output from `csaperiod` is periodic (monthly) accounting files and an ASCII report.

7. Accounting is disabled. When the system is shut down gracefully, the `csaswitch(8)` command is executed to halt all CSA process and daemon accounting.

Setting Up CSA

The following is a brief description of setting up CSA. Site-specific modifications are discussed in detail in "Tailoring CSA", page 36. As described in this section, CSA is run by a person with superuser permissions.

1. Change the default system billing unit (SBU) weighting factors, if necessary. By default, no SBUs are calculated. If your site wants to report SBUs, you must modify the configuration file `/etc/csa.conf`.
2. Modify any necessary parameters in the `/etc/csa.conf` file, which contains configurable parameters for the accounting system.
3. If you want daemon accounting, you must enable daemon accounting at system startup time by performing the following steps:
 - a. Ensure that the variables in `/etc/csa.conf` for the subsystems for which you want to enable daemon accounting are set to on. Set `WKMG_START` to on to enable workload management.
4. As root, use the `crontab(1)` command with the `-e` option to add entries similar to the following:

Note: If you do not use the `crontab(1)` command to update the `crontab` file (for example, using the `vi(1)` editor to update the file), you must signal `cron(8)` after updating the file. The `crontab` command automatically updates the `crontab` file and signals `cron(8)` when you save the file and exit the editor. For more information on the `crontab` command, see the `crontab(1)` man page.

```
0 4 * * 1-6 if /etc/chkconfig csaacct; then /usr/local/sbin/csarun 2> /var/csa/nite/fd2log; fi
0 2 * * 4   if /etc/chkconfig csaacct; then /usr/local/sbin/dodisk -c > /var/csa/nite/dk2log; fi
5 * * * 1-6 if /etc/chkconfig csaacct; then /usr/local/sbin/csackpacct; fi
0 5 1 * *   if /etc/chkconfig csaacct; then /usr/local/sbin/csaperiod -r \
2> /var/csa/nite/pd2log; fi
```

These entries are described in the following steps:

- a. For most installations, entries similar to the following should be made in `/var/spool/cron/root` so that `cron(8)` automatically runs daily accounting:

```
0 4 * * 1-6 if /etc/chkconfig csaacct; then /usr/local/sbin/csarun 2> /var/csa/nite/fd2log; fi
0 2 * * 4    if /etc/chkconfig csaacct; then /usr/local/sbin/dodisk -c > /var/csa/nite/dk2log; fi
```

The `csarun(8)` command should be executed at such a time that `dodisk` has sufficient time to complete. If `dodisk` does not complete before `csarun` executes, disk accounting information may be missing or incomplete.

The `dodisk` command must be invoked with the `-c` option. For more information, see the `dodisk(8)` man page.

- b. Periodically check the size of the `pacct` files. An entry similar to the following should be made in `/var/spool/cron/root`:

```
5 * * * 1-6 if /etc/chkconfig csaacct; then /usr/local/sbin/csackpacct; fi
```

The `cron` command should periodically execute the `csackpacct(8)` shell script. If the `pacct` file grows larger than 4000 1K blocks (default), `csackpacct` calls the command `/usr/local/sbin/csaswitch -c switch` to start a new `pacct` file. The `csackpacct` command also makes sure that there are at least 2000 1K blocks free on the file system containing `/var/csa`. If there are not enough blocks, CSA accounting is turned off. The next time `csackpacct` is executed, it turns CSA accounting back on if there are enough free blocks.

Ensure that the `ACCT_FS` and `MIN_BLKs` variables have been set correctly in the `/etc/csa.conf` configuration file. `ACCT_FS` is the file system containing `/var/csa`. `MIN_BLKs` is the minimum number of free 1K blocks needed in the `ACCT_FS` file system. The default is 2000.

It is very important that `csackpacct` be run periodically so that an administrator is notified when the accounting file system (located in the `/var/csa` directory by default) runs out of disk space. After the file system is cleaned up, the next invocation of `csackpacct` enables process and daemon accounting. You can manually re-enable accounting by invoking `csaswitch -c on`.

If `csackpacct` is not run periodically, and the accounting file system runs out of space, an error message is written to the console stating that a write error occurred and that accounting is disabled. If you do not free disk space as soon as possible, a vast amount of accounting data can be lost unnecessarily. Additionally, lost accounting data can cause `csarun` to abort or report erroneous information.

- c. To run monthly accounting, an entry similar to the command shown below should be made in `/var/spool/cron/root`. This command generates a monthly report on all consolidated data files found in `/var/csa/sum/*` and then deletes those data files:

```
0 5 1 * *    if /etc/chkconfig csaacct; then /usr/local/sbin/csaperiod -r \
2> /var/csa/nite/pd2log; fi
```

This entry is executed at such a time that `csarun` has sufficient time to complete. This example results in the creation of a periodic accounting file and report on the first day of each month. These files contain information about the previous month's accounting.

5. Update the `holidays` file. The file `/usr/local/etc/holidays` contains the prime/nonprime table for the accounting system. The table should be edited to reflect your location's holiday schedule for the year. By default, the `holidays` file is located in the `/usr/local/etc` directory. You can change this location by modifying the `HOLIDAY_FILE` variable in `/etc/csa.conf`. If necessary, modify the `NUM_HOLIDAYS` variable (also located in `/etc/csa.conf`), which sets the upper limit on the number of holidays that can be defined in `HOLIDAY_FILE`. The format of this file is composed of the following types of entries:

- Comment lines: These lines may appear anywhere in the file as long as the first character in the line is an asterisk (*).
- Version line: This line must be the first uncommented line in the file and must only appear once. It denotes that the new holidays file format is being used. This line should not be changed by the site.
- Year designation line: This line must be the second uncommented line in the file and must only appear once. The line consists of two fields. The first field is the keyword `YEAR`. The second field must be either the current year or the wildcard character, asterisk (*). If the year is wildcarded, the current year is

automatically substituted for the year. The following are examples of two valid entries:

```
YEAR      1995
YEAR      *
```

- Prime/nonprime time designation lines: These must be uncommented lines 3, 4, and 5 in the file. The format of these lines is:

```
period  prime_time_start  nonprime_time_start
```

The variable, *period*, is one of the following: WEEKDAY, SATURDAY, or SUNDAY. The *period* can be specified in either uppercase or lowercase.

The prime and nonprime start time can be one of two formats:

- Both start times are 4–digit numeric values between 0000 and 2359. The *nonprime_time_start* value must be greater than the *prime_time_start* value. For example, it is incorrect to have prime time start at 07:30 A.M. and nonprime time start at 1 minute after midnight. Therefore, the following entry is wrong and can cause incorrect accounting values to be reported.

```
WEEKDAY  0730  0001
```

It is correct to specify prime time to start at 07:30 A.M. and nonprime time to start at 5:30 P.M. on weekdays. You would enter the following in the holiday file:

```
WEEKDAY  0730  1730
```

- NONE/ALL or ALL/NONE. These start times specify that the entire period is to be either all prime time or all nonprime time. To specify that the entire period is to be considered prime time, set *prime_time_start* to ALL and *nonprime_time_start* to NONE. If the period is to be considered all nonprime time, set *prime_time_start* to NONE and *nonprime_time_start* to ALL. For example, to specify Monday through Friday as all prime time, you would enter the following:

```
WEEKDAY ALL NONE
```

To specify all of Sunday to be nonprime time, you would enter the following:

```
SUNDAY NONE ALL
```

- Company holidays lines: These entries follow the year designation line and have the following general format:

day-of-year Month Day Description of Holiday

The *day-of-year* field is either a number in the range of 1 through 366, indicating the day for a given holiday (leading white space is ignored), or it is the month and day in the *mm/dd* format. The other three fields are commentary and are not currently used by other programs. Each holiday is considered all nonprime time.

If the `holidays` file does not exist or there is an error in the year designation line, the default values for all lines are used.

If there is an error in a prime/nonprime time designation line, the entry for the erroneous line is set to a default value. All other lines in the `holidays` file are ignored and default values are used.

If there is an error in a company holidays line, all holidays are ignored.

The defaults values are as follows:

YEAR	The current year
WEEKDAY	Monday through Friday is all prime time
SATURDAY	Saturday is all nonprime time
SUNDAY	Sunday is all nonprime time
	No holidays are specified

The `csarun` Command

The `/usr/local/sbin/csarun` command, usually initiated by `cron(1)`, directs the processing of the daily accounting files. `csarun` processes accounting records written into the `pacct` file. It is normally initiated by `cron` during nonprime hours.

The `csarun` command also contains four user-exit points, allowing sites to tailor the daily run of accounting to their specific needs.

The `csarun` command does not damage files in the event of errors. It contains a series of protection mechanisms that attempt to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that `csarun` can be restarted with minimal intervention.

Daily Invocation

The `csarun` command is invoked periodically by `cron`. It is very important that you ensure that the previous invocation of `csarun` completed successfully before invoking `csarun` for a new accounting period. If this is not done, information about unfinished jobs will be inaccurate.

Data for a new accounting period can also be interactively processed by executing the following:

```
nohup csarun 2> /var/csa/nite/fd2log &
```

Before executing `csarun` in this manner, ensure that the previous invocation completed successfully. To do this, look at the files `active` and `statefile` in `/var/csa/nite`. Both files should specify that the last invocation completed successfully. See "Restarting `csarun`", page 24.

Error and Status Messages

The `csarun` error and status messages are placed in the `/var/csa/nite` directory. The progress of a run is tracked by writing descriptive messages to the file `active`. Diagnostic output during the execution of `csarun` is written to `fd2log`. The `lock` and `lock1` files prevent concurrent invocations of `csarun`; `csarun` will abort if these two files exist when it is invoked. The `clastdate` file contains the month, day, and time of the last two executions of `csarun`.

Errors and warning messages from programs called by `csarun` are written to files that have names beginning with `E` and ending with the current date and time. For example, `Eb1d.11121400` is an error file from `csabuild` for a `csarun` invocation on November 12, at 14:00.

If `csarun` detects an error, it writes a message to the `var/log/messages` file, removes the locks, saves the diagnostic files, and terminates execution. When `csarun` detects an error, it will send mail either to `MAIL_LIST` if it is a fatal error, or to `WMAIL_LIST` if it is a warning message, as defined in the configuration file `/etc/csa.conf`.

States

Processing is broken down into separate reentrant states so that `csarun` can be restarted. As each state completes, `/var/csa/nite/statefile` is updated to reflect the next state. When `csarun` reaches the `CLEANUP` state, it removes various data files and the locks, and then terminates.

The following describes the events that occur in each state. *MMDD* refers to the month and day *csarun* was invoked. *hhmm* refers to the hour and minute of invocation.

State	Description
SETUP	The current accounting file is switched via <i>csaswitch</i> . The accounting file is then moved to the <i>/var/csa/work/MMDD/hhmm</i> directory. File names are prefaced with <i>W</i> . <i>/var/csa/nite/diskcacct</i> is also moved to this directory.
VERIFY	The accounting files are checked for valid data. Records with invalid data are removed. Names of bad data files are prefixed with <i>BAD</i> . in the <i>/var/csa/work/MMDD/hhmm</i> directory. The corrected files do not have this prefix.
ARCHIVE1	First user exit of the <i>csarun</i> script. If a script named <i>/usr/local/sbin/csa.archive1</i> exists, it will be executed through the shell <i>.</i> (<i>dot</i>) command. The <i>.</i> (<i>dot</i>) command will not execute a compiled program, but the user exit script can. You might use this user exit to archive the accounting files in <i>/\${WORK}</i> .
BUILD	The <i>pacct</i> accounting data is organized into a sorted <i>pacct</i> file.
ARCHIVE2	Second user exit of the <i>csarun</i> script. If a script named <i>/usr/local/sbin/csa.archive2</i> exists, it will be executed through the shell <i>.</i> (<i>dot</i>) command. The <i>.</i> (<i>dot</i>) command will not execute a compiled program, but the user exit script can. You might use this exit to archive the sorted <i>pacct</i> file.
CMS	Produces a command summary file in <i>cms.h</i> format. The <i>cms</i> file is written to <i>/var/csa/sum/cms.MMDDhhmm</i> for use by <i>csaperiod</i> .
REPORT	Generates the daily accounting report and puts it into <i>/var/csa/sum/rprt.MMDDhhmm</i> . A consolidated data file, <i>/var/csa/sum/cacct.MMDDhhmm</i> , is also produced from the sorted <i>pacct</i> file. In addition, accounting data for unfinished jobs is recycled.
DREP	Generates a daemon usage report based on the sorted <i>pacct</i> file. This report is appended to the daily accounting report, <i>/var/csa/sum/rprt.MMDDhhmm</i> .
FEF	Third user exit of the <i>csarun</i> script. If a script named <i>/var/local/sbin/csa.fef</i> exists, it will be executed through the

shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. The `csarun` variables are available, without being exported, to the user exit script. You might use this exit to convert the `sorted pacct` file to a format suitable for a front-end system.

USEREXIT Fourth user exit of the `csarun` script. If a script named `/usr/local/sbin/csa.user` exists, it will be executed through the shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. The `csarun` variables are available, without being exported, to the user exit script. You might use this exit to run local accounting programs.

CLEANUP Cleans up temporary files, removes the locks, and then exits.

Restarting `csarun`

If `csarun` is executed without arguments, the previous invocation is assumed to have completed successfully.

The following operands are required with `csarun` if it is being restarted:

```
csarun [MMDD [hhmm [state]]]
```

`MMDD` is month and day, `hhmm` is hour and minute, and `state` is the `csarun` entry state.

To restart `csarun`, follow these steps:

1. Remove all lock files, by using the following command line:

```
rm -f /var/csa/nite/lock*
```

2. Execute the appropriate `csarun` restart command, using the following examples as guides:

- a. To restart `csarun` using the time and the state specified in `clastdate` and `statefile`, execute the following command:

```
nohup csarun 0601 2> /var/csa/nite/fd2log &
```

In this example, `csarun` will be rerun for June 1, using the time and state specified in `clastdate` and `statefile`.

- b. To restart `csarun` using the state specified in `statefile`, execute the following command:

```
nohup csarun 0601 0400 2> /var/csa/nite/fd2log &
```

In this example, `csarun` will be rerun for the June 1 invocation that started at 4:00 A.M., using the state found in `statefile`.

- c. To restart `csarun` using the specified date, time, and state, execute the following command:

```
nohup csarun 0601 0400 BUILD 2> /var/csa/nite/fd2log &
```

In this example, `csarun` will be restarted for the June 1 invocation that started at 4:00 A.M., beginning with state `BUILD`.

Before `csarun` is restarted, the appropriate directories must be restored. If the directories are not restored, further processing is impossible. These directories are as follows:

```
/var/csa/work/MMDD/hhmm  
/var/csa/sum
```

If you are restarting at state `ARCHIVE2`, `CMS`, `REPORT`, `DREP`, or `FEF`, the sorted `pacct` file must be in `/var/csa/work/MMDD/hhmm`. If the file does not exist, `csarun` automatically will restart at the `BUILD` state. Depending on the tasks performed during the site-specific `USEREXIT` state, [the sorted `pacct` file may or may not need to exist.] This may or may not be acceptable.

Verifying and Editing Data Files

This section describes how to remove bad data from various accounting files.

The `csaverify(8)` command verifies that the accounting records are valid and identifies invalid records. The accounting file can be a `pacct` or sorted `pacct` file. When `csaverify` finds an invalid record, it reports the starting byte offset and length of the record. This information can be written to a file in addition to standard output. A length of `-1` indicates the end of file. The resulting output file can be used as input to `csaedit(8)` to delete `pacct` or sorted `pacct` records.

1. The `pacct` file is verified with the following command line, and the following output is received:

```
$ /usr/local/sbin/cverify -P pacct -o offsetfile
/usr/local/sbin/cverify: CAUTION
readacctent(): An error was returned from the 'readpacct()' routine.
```

2. The file `offsetfile` from `cverify` is used as input to `csaedit` to delete the invalid records as follows (remaining valid records are written to `pacct.NEW`):

```
/usr/local/sbin/csaedit -b offsetfile -P pacct -o pacct.NEW
```

3. The new `pacct` file is reverified as follows to ensure that all the bad records have been deleted:

```
/usr/local/sbin/cverify -P pacct.NEW
```

You can use the `csaedit -A` option to produce an abbreviated ASCII version of `pacct` or sorted `pacct` files.

CSA Data Processing

The flow of data among the various CSA programs is explained in this section and is illustrated in Figure 1-3.

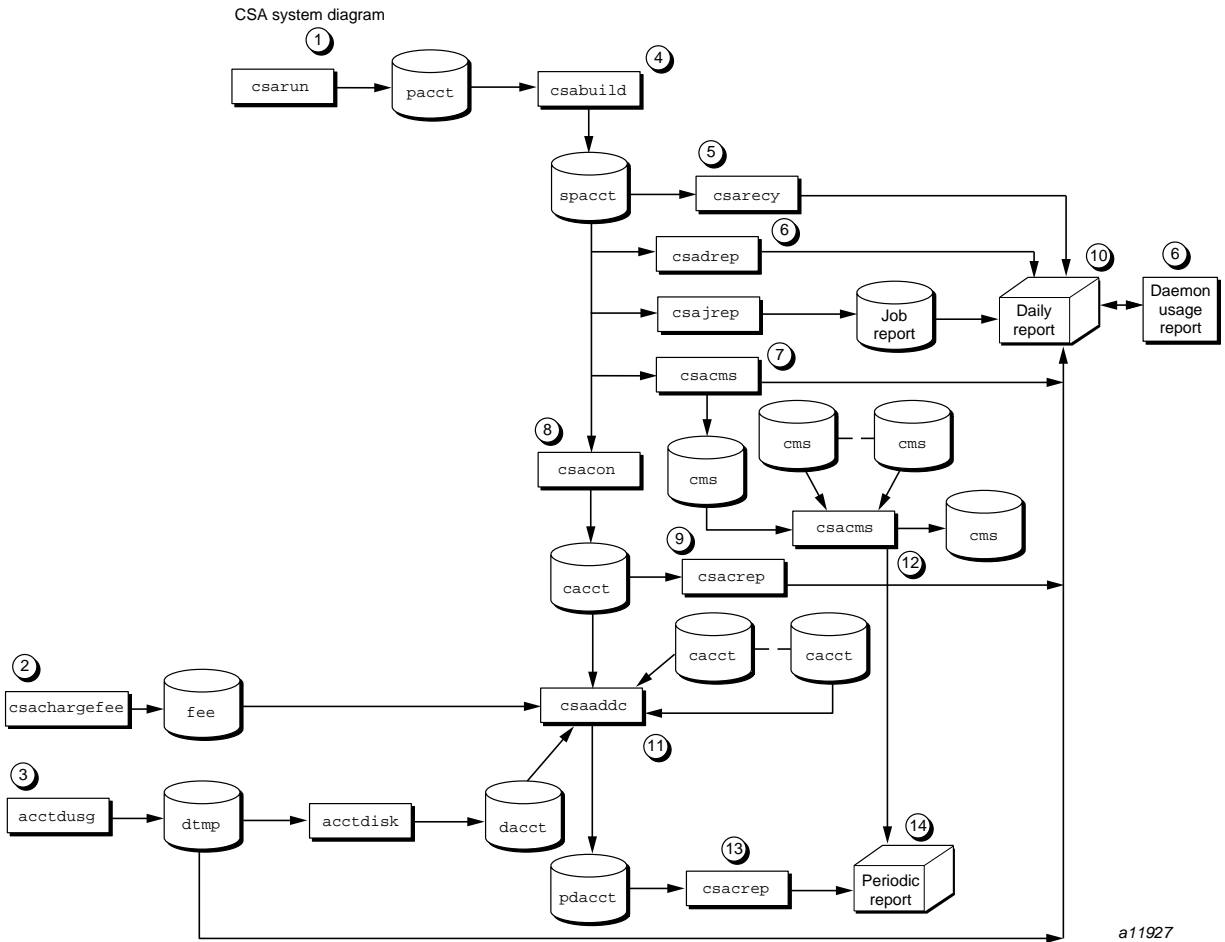


Figure 1-3 CSA Data Processing

1. Generate raw accounting files. Various daemons and system processes write to the raw `pacct` accounting files.
2. Create a fee file. Sites that want to charge fees to certain users can do so with the `csachargefee`(8) command. The `csachargefee` command creates a fee file that is processed by `csaaddc`(8).

3. Produce disk usage statistics. The `dodisk(8)` shell script allows sites to take snapshots of disk usage. `dodisk` does not report dynamic usage; it only reports the disk usage at the time the command was run. Disk usage is processed by `csaaddc`.
4. Organize accounting records into job records. The `csabuild(8)` command reads accounting records from the CSA `pacct` file and organizes them into job records by job ID and boot times. It writes these job records into the `sorted pacct` file. This `sorted pacct` file contains all of the accounting data available for each job. The configuration records in the `pacct` files are associated with the job ID 0 job record within each boot period. The information in the `sorted pacct` file is used by other commands to generate reports and for billing.
5. Recycle information about unfinished jobs. The `csarecy(8)` command retrieves job information from the `sorted pacct` file of the current accounting period and writes the records for unfinished jobs into a `pacct0` file for recycling into the next accounting period. `csabuild(8)` marks unfinished accounting jobs (those are jobs without an end-of-job record). `csarecy` takes these records from the `sorted pacct` file and puts them into the next period's accounting files directory. This process is repeated until the job finishes.

Sometimes data for terminated jobs are continually recycled. This can occur when accounting data is lost. To prevent data from recycling forever, edit `csarun` so that `csabuild` is executed with the `-o nday` option, which causes all jobs older than `nday` days to terminate. Select an appropriate `nday` value (see the `csabuild` man page for more information and "Data Recycling", page 30).

6. Generate the daemon usage report, which is appended to the daily report. `csadrep(8)` reports usage of the workload management and tape (tape is deferred) daemons. Input is either from a `sorted pacct` file created by `csabuild(8)` or from a binary file created by `csadrep` with the `-o` option. The `files` operand specifies the binary files.
7. Summarize command usage from per-process accounting records. The `csacms(8)` command reads the `sorted pacct` files. It adds all records for processes that executed identically named commands, and it sorts and writes them to `var/csa/sum/cms.MMDDhhmm`, using the `cms` format. The `csacms(8)` command can also create an ASCII file.
8. Condense records from the `sorted pacct` file. The `csacon(8)` command condenses records from the `sorted pacct` file and writes consolidated records in `cacct` format to `var/csa/sum/cacct.MMDDhhmm`.

9. Generate an accounting report based on the consolidated data. The `csacrep(8)` command generates reports from data in `cacct` format, such as output from the `csacon(8)` command. The report format is determined by the value of `CSACREP` in the `/etc/csa.conf` file. Unless modified, it will report the CPU time, total `KCORE` minutes total `KVIRTUAL` minutes, block I/O wait time, and raw I/O wait time. The report will be sorted first by user ID and then by the secondary key of project ID (project ID is deferred) and the headers will be printed.
10. Create the daily accounting report. The daily accounting report includes the following:
 - Consolidated information report (step 11)
 - Unfinished recycled jobs (step 5)
 - Disk usage report (step 3)
 - Daily command summary (step 7)
 - Last login information
 - Daemon usage report (step 6)
11. Combine `cacct` records. The `csaaddc(8)` command combines `cacct` records by specified consolidation options and writes out a consolidated record in `cacct` format.
12. Summarize command usage from per-process accounting records. The `csacms(8)` command reads the `cms` files created in step 7. Both an ASCII and a binary file are created.
13. Produce a consolidated accounting report. `csacrep(8)` is used to generate a report based on a periodic accounting file.
14. The periodic accounting report layout is as follows:
 - Consolidated information report
 - Command summary report

Steps 4 through 11 are performed during each accounting period by `csarun(8)`. Periodic (monthly) accounting (steps 12 through 14) is initiated by the `csaperiod(8)` command. Daily and periodic accounting, as well as fee and disk usage generation (steps 2 through 3), can be scheduled by `cron(8)` to execute regularly. See "Setting Up CSA", page 17, for more information.

Data Recycling

A system administrator must correctly maintain recycled data to ensure accurate accounting reports. The following sections discuss data recycling and describe how an administrator can purge unwanted recycled accounting data.

Data recycling allows CSA to properly bill jobs that are active during multiple accounting periods. By default, `csarun` reports data only for jobs that terminate during the current accounting period. Through data recycling, CSA preserves data for active jobs until the jobs terminate.

In the sorted `pacct` file, `csabuild` flags each job as being either active or terminated. `csarecy` reads the sorted `pacct` file and recycles data for the active jobs. `csacon` consolidates the data for the terminated jobs, which `csaperiod` uses later. `csabuild`, `csarecy`, and `csacon` are all invoked by `csarun`.

`csarun` puts recycled data in the `/var/csa/day/pacct0` file.

Normally, an administrator should not have to manually purge the recycled accounting data. This purge should only be necessary if accounting data is missing. Missing data can cause jobs to recycle forever and consume valuable CPU cycles and disk space.

How Jobs Are Terminated

Interactive jobs, `cron` jobs, and `at` jobs terminate when the last process in the job exits. Normally, the last process to terminate is the login shell. The kernel writes an end-of-job (EOJ) record to the `pacct` file when the job terminates.

When the workload management daemon delivers a workload management request's output, the request terminates. The daemon then writes an `NQ_DISP` record type to the `pacct` accounting file, while the kernel writes an EOJ record to the `pacct` file.

Unlike interactive jobs, workload management requests can have multiple EOJ records associated with them. In addition to the request's EOJ record, there can be EOJ records for net clients and checkpointed portions of the request. The net client perform workload management processing on behalf of the request.

The `csabuild` command flags jobs in the sorted `pacct` file as being terminated if they meet one of the following conditions:

- The job is an interactive, `cron`, or `at` job, and there is an EOJ record for the job in the `pacct` file.

- The job is a workload management request, and there is both an EOJ record for the request and an NQ_DISP record type in the `pacct` file.
- The job is an interactive, `cron`, or `at` job and is active at the time of a system crash.
- The job is manually terminated by the administrator using one of the methods described in "How to Remove Recycled Data", page 31.

Why Recycled Sessions Should Be Scrutinized

Recycling unnecessary data can consume large amounts of disk space and CPU time. The sorted `pacct` file and recycled data can occupy a vast amount of disk space on the file system containing `/var/csa/day`. Sites that archive data also require additional offline media. Wasted CPU cycles are used by `csarun` to reexamine and recycle the data. Therefore, to conserve disk space and CPU cycles, unnecessary recycled data should be purged from the accounting system.

Any of the following situations can cause CSA erroneously to recycle terminated jobs:

- Kernel or daemon accounting is turned off.
The kernel or `csackpacct(8)` command can turn off accounting when there is not enough space on the file system containing `/var/csa/day`.
- Accounting files are corrupt. Accounting data can be lost or corrupted during a system or disk crash.
- Recycled data is erroneously deleted in a previous accounting period.

How to Remove Recycled Data

Before choosing to delete recycled data, you should understand the repercussions, as described in "Adverse Effects of Removing Recycled Data", page 33. Data removal can affect billing and can alter the contents of the consolidated data file, which is used by `csaperiod`.

You can remove recycled data from CSA in the following ways:

- Interactively execute the `csarecy -A` command. Administrators can select the active jobs that are to be recycled by running `csarecy` with the `-A` option. Users are not billed for the resources used in the jobs terminated in this manner. Deleted data is also not included in the consolidated data file.

The following example is one way to execute `csarecy -A` (which generates two accounting reports and two consolidated files):

1. Run `csarun` at the regularly scheduled time.
2. Edit a copy of `/usr/local/sbin/csarun`. Change the `-r` option on the `csarecy` invocation line to `-A`. Also, do not redirect standard output to `/${SUM_DIR}/recyrpt`. The result should be similar to the following:

```
csarecy -A -s ${SPACCT} -P ${WTIME_DIR}/Rpacct \ 2> ${NITE_DIR}/Erec.${DTIME}
```

Since both the `-A` and `-r` options write output to `stdout`, the `-r` option is not invoked and `stdout` is not redirected to a file. As a result, the recycled job report is not generated.

3. Execute the `jstat` command, as follows, to display a list of currently active jobs:

```
jstat -a > jstat.out
```

4. Execute the `qstat` command to display a list of workload management requests. The `qstat` command is used for seeing whether there are requests that are not currently running. This includes requests that are checkpointed, held, queued, or waiting.

To list all workload management requests, execute the `qstat` command, as follows, using a login that has either workload management manager or workload management operator privilege:

```
qstat -a > qstat.out
```

5. Interactively run the modified version of `csarun`. If you execute the modified `csarun` soon after the first step is complete, little data is lost because not very much data exists.

For each active job, `csarecy` asks you if you want to preserve the job. Preserve the active and nonrunning workload management jobs found in the third and fourth steps. All other jobs are candidates for removal.

- Execute `csabuild` with the `-o ndays` option, which terminates all active jobs older than the specified number of days. Resource usage for these terminated jobs is reported by `csarun`, and users are billed for the jobs. The consolidated data file also includes this resource usage.

To execute `csabuild` with the `-o` option, edit a copy of `/usr/local/sbin/csarun`. Add the `-o ndays` option to the `csabuild` invocation line. Specify for `ndays` an appropriate value for your site.

Recycled data for currently active jobs will be removed if you specify an inappropriate value for `ndays`.

- Execute `csarun` with the `-A` option. It reports resource usage for both active and terminated jobs, so users are billed for recycled sessions. This data is also included in the consolidated data file.

None of the data for the active jobs, including the currently active jobs, is recycled. No recycled data file is generated in the `/var/csa/day` directory.

- Remove the recycled data file from the `/var/csa/day` directory. You can delete data for all of the recycled jobs, both terminated and active, by executing the following command:

```
rm /var/csa/day/pacct0
```

The next time `csarun` is executed, it will not find data for any recycled jobs. Thus, users are not billed for the resources used in the recycled jobs, and this data is not included in the consolidated data file. `csarun` recycles the data for currently active jobs.

Adverse Effects of Removing Recycled Data

CSA assumes that all necessary accounting information is available to it, which means that CSA expects kernel and daemon accounting to be enabled and recycled data not to have been mistakenly removed. If some data is unavailable, CSA may provide erroneous billing information. Sites should be aware of the following facts before removing data:

- Users may or may not be billed for terminated recycled jobs. Administrators must understand which of the previously described methods cause the user to be billed for the terminated recycled jobs. It is up to the site to decide whether or not it is valid for the user to be billed for these jobs.

For those methods that cause the user to be billed, both `csarun` and `csaperiod` report the resource usage.

- It may be impossible to reconstruct a terminated recycled job. If a recycled job is terminated by the administrator, but the job actually terminates in a later accounting period, information about the job is lost. If a user questions the

resource billing, it may be extremely difficult or impossible for the administrator to correctly reassemble all accounting information for the job in question.

- Manually terminated recycled jobs may be improperly billed in a future billing period. If the accounting data for the first portion of a job has been deleted, CSA may be unable to correctly identify the remaining portion of the job. Errors may occur, such as workload management requests being flagged as interactive jobs, or workload management requests being billed at the wrong queue rate. This is explained in detail in "Workload Management Requests and Recycled Data", page 35.
- CSA programs may detect data inconsistencies. When accounting data is missing, CSA programs may detect errors and abort.

The following table summarizes the effects of using the methods described in "How to Remove Recycled Data", page 31.

Table 1-1 Possible Effects of Removing Recycled Data

Method	Underbilling?	Incorrect billing?	Consolidated data file
<code>csarecy -A</code>	Yes. Users are not billed for the portion of the job that was terminated by <code>csarecy -A</code> .	Possible. Manually terminated recycled jobs may be billed improperly in a future billing period.	Does not include data for jobs terminated by <code>csarecy -A</code> .
<code>csabuild -o</code>	No. Users are billed for the portion of the job that was terminated by <code>csabuild -o</code> .	Possible. Manually terminated recycled jobs may be billed improperly in a future billing period.	Includes data for jobs terminated by <code>csabuild -o</code> .
<code>csarun -A</code>	No. All active and recycled jobs are billed.	Possible. All active and recycled jobs that eventually terminate may be billed improperly in a future billing period, because no data is recycled.	Includes data for all active and recycled jobs.
<code>rm</code>	Yes. All users are not billed for the portion of the job that was recycled.	Possible. All recycled jobs that eventually terminate may be billed improperly in a future billing period.	Does not include data for any recycled job.

By default, the consolidated data file contains data only for terminated jobs. Manual termination of recycled data may cause some of the recycled data to be included in the consolidated file.

Workload Management Requests and Recycled Data

For CSA to identify all workload management requests, data must be properly recycled. When an administrator manually purges recycled data for a workload management request, errors such as the following can occur:

- CSA fails to flag the job as a workload management job. This causes the request to be billed at standard rates instead of a workload management queue rate (see "Workload Management SBUs", page 39 or "Workload Management SBUs", page 39).
- The request is billed at the wrong queue rate.
- The wrong queue wait time is associated with the request.

These errors occur because valuable workload management accounting information was purged by the administrator. Only a few workload management accounting records are written by the workload management daemon, and all of the records are needed for CSA to properly bill workload management requests.

Workload management accounting records are only written under the following circumstances:

- The workload management daemon receives a request.
- A request executes. This includes executing a request for the first time, restarting, and rerunning a request.
- A request terminates. A workload management request can terminate because it is completed, requeued, held, rerun, or migrated.
- Output is delivered.

Thus, for long running requests that span days, there can be days when no workload management data is written. Consequently, it is extremely important that accounting data be recycled. If the site administrator manually terminates recycled jobs, care must be taken to be sure that only nonexistent workload management requests are terminated.

Tailoring CSA

This section describes the following actions in CSA:

- Setting up SBUs
- Setting up daemon accounting
- Setting up user exits
- Modifying the charging of workload management jobs based on workload management termination status
- Tailoring CSA shell scripts
- Using `at(1)` instead of `cron(8)` to periodically execute `csarun`
- Allowing users without superuser permissions to run CSA
- Using an alternate configuration file

System Billing Units (SBUs)

A *system billing unit* (SBU) is a unit of measure that reflects use of machine resources. You can alter the weighting factors associated with each field in each accounting record to obtain an SBU value suitable for your site. SBUs are defined in the accounting configuration file, `/etc/csa.conf`. By default, all SBUs are set to 0.0.

Accounting allows different periods of time to be designated either prime or nonprime time (the time periods are specified in `/usr/local/sbin/holidays`).

Following is an example of how the prime/nonprime algorithm works:

Assume a user uses 10 seconds of CPU time, and executes for 100 seconds of prime wall-clock time, and pauses for 100 seconds of nonprime wall-clock time. Therefore, elapsed time is 200 seconds (100+100). If

```
prime = prime time / elapsed time
nonprime = nonprime time / elapsed time
cputime[PRIME] = prime * CPU time
cputime[NONPRIME] = nonprime * CPU time
```

then

```
cputime[PRIME] == 5 seconds
cputime[NONPRIME] == 5 seconds
```

Under CSA, an SBU value is associated with each record in the `sorted pacct` file when that file is assembled by `csabuild`. Final summation of the SBU values is done by `csacon` during the creation of the `cacct` record file.

The following examples show how a site can bill different NQS or workload management queues at differing rates:

$$\text{Total SBU} = (\text{Workload management queue SBU value}) * (\text{sum of all process record SBUs} \\ + \text{sum of all tape record SBUs})$$

Process SBUs

The SBUs for process data are separated into prime and nonprime values. Prime and nonprime use is calculated by a ratio of elapsed time. If you do not want to make a distinction between prime and nonprime time, set the nonprime time SBUs and the prime time SBUs to the same value. Prime time is defined in `/usr/local/etc/holidays`. By default, Saturday and Sunday are considered nonprime time.

The following is a list of prime time process SBU weights. Descriptions and factor units for the nonprime time SBU weights are similar to those listed here. SBU weights are defined in `/etc/csa.conf`.

Value	Description
P_BASIC	Prime-time weight factor. P_BASIC is multiplied by the sum of prime time SBU values to get the final SBU factor for the process record.
P_TIME	General-time weight factor. P_TIME is multiplied by the time SBUs (made up of P_STIME, P_UTIME, P_QTIME, P_BWTIME, and P_RWTIME) to get the time contribution to the process record SBU value.
P_STIME	System CPU-time weight factor. The unit used for this weight is <i>billing units</i> per second. P_STIME is multiplied by the system CPU time.
P_UTIME	User CPU-time weight factor. The unit used for this weight is <i>billing units</i> per second. P_UTIME is multiplied by the user CPU time.

P_BWTIME	Block I/O wait time weight factor. The unit used for this weight is <i>billing units</i> per second. P_BWTIME is multiplied by the block I/O wait time.
P_RWTIME	Raw I/O wait time weight factor. The unit used for this weight is <i>billing units</i> per second. P_RWTIME is multiplied by the raw I/O wait time.
P_MEM	General-memory-integral weight factor. P_MEM is multiplied by the memory SBUs (made up of P_XMEM and P_VMEM) to get the memory contribution to the process record SBU value.
P_XMEM	CPU-time-core-physical memory-integral weight factor. The unit used for this weight is <i>billing units</i> per Mbyte-minute. P_XMEM is multiplied by the core-memory integral.
P_VMEM	CPU-time-virtual-memory-integral weight factor. The unit used for this weight is <i>billing units</i> per Mbyte-minute. P_VMEM is multiplied by the virtual memory integral.
P_IO	General-I/O weight factor. P_IO is multiplied by the I/O SBUs (made up of P_BIO, P_CIO, and P_LIO) to get the I/O contribution to the process record SBU value.
P_BIO	Blocks-transferred weight factor. The unit used for this weight is <i>billing units</i> per block transferred. P_BIO is multiplied by the number of I/O blocks transferred.
P_CIO	Characters-transferred weight factor. The unit used for this weight is <i>billing units</i> per character transferred. P_CIO is multiplied by the number of I/O characters transferred.

`P_LIO` Logical-I/O-request weight factor. The unit used for this weight is *billing units* per logical I/O request. `P_LIO` is multiplied by the number of logical I/O requests made. The number of logical I/O requests is total number of `read` and `write` system calls.

The formula for calculating the whole process record SBU is as follows:

$$\text{PSBU} = (\text{P_TIME} * (\text{P_STIME} * \text{stime} + \text{P_UTIME} * \text{utime} + \text{P_BWTIME} * \text{bwtime} + \text{P_RWTIME} * \text{rwtime})) + (\text{P_MEM} * (\text{P_XMEM} * \text{coremem} + \text{P_VMEM} * \text{virtmem})) + (\text{P_IO} * (\text{P_BIO} * \text{bio} + \text{P_CIO} * \text{cio} + \text{P_LIO} * \text{lio}));$$

$$\text{NSBU} = (\text{NP_TIME} * (\text{NP_STIME} * \text{stime} + \text{NP_UTIME} * \text{utime} + \text{NP_BWTIME} * \text{bwtime} + \text{NP_RWTIME} * \text{rwtime})) + (\text{NP_MEM} * (\text{NP_XMEM} * \text{coremem} + \text{NP_VMEM} * \text{virtmem})) + (\text{NP_IO} * (\text{NP_BIO} * \text{bio} + \text{NP_CIO} * \text{cio} + \text{NP_LIO} * \text{lio}));$$

$$\text{SBU} = \text{P_BASIC} * \text{PSBU} + \text{NP_BASIC} * \text{NSBU};$$

The variables in this formula are described as follows:

Variable	Description
<i>stime</i>	System CPU time in seconds
<i>utime</i>	User CPU time in seconds
<i>bwtime</i>	Block I/O wait time in seconds
<i>rwtime</i>	Raw I/O wait time in seconds
<i>coremem</i>	Core (physical) memory integral in Mbyte-minutes
<i>virtmem</i>	Virtual memory integral in Mbyte-minutes
<i>bio</i>	Number of blocks of data transferred
<i>cio</i>	Number of characters of data transferred
<i>lio</i>	Number of logical I/O requests

Workload Management SBUs

The `/etc/csa.conf` file contains the configurable parameters that pertain to workload management SBUs.

The `WKMG_NUM_QUEUES` parameter sets the number of queues for which you want to set SBUs (the value must be set to at least 1). Each `WKMG_QUEUE` x variable in the configuration file has a queue name and an SBU pair associated with it (the total

number of queue/SBU pairs must equal `WKMG_NUM_QUEUES`). The queue/SBU pairs define weights for the queues. If an SBU value is less than 1.0, there is an incentive to run jobs in the associated queue; if the value is 1.0, jobs are charged as though they are non-workload management jobs; and if the SBU is 0.0, there is no charge for jobs running in the associated queue. SBUs for queues not found in the configuration file are automatically set to 1.0.

The `WKMG_NUM_MACHINES` parameter sets the number of originating machines for which you want to set SBUs (the value must be at least 1). Each `WKMG_MACHINE x` variable in the configuration file has an originating machine and an SBU pair associated with it (the total number of machine/SBU pairs must equal `WKMG_NUM_MACHINES`). SBUs for originating machines not specified in `/etc/csa.conf` are automatically set to 1.0.

Tape SBUs (deferred)

There is a set of weighting factors for each group of tape devices. By default, there are only two groups, `tape` and `cart`. The `TAPE_SBU i` parameters in `/etc/csa.conf` define the weighting factors for each group. There are SBUs associated with the following:

- Number of mounts
- Device reservation time (seconds)
- Number of bytes read
- Number of bytes written

Note: Tape support is deferred.

Daemon Accounting

Accounting information is available from the workload management daemon. Data is written to the `pacct` file in the `/var/csa/day` directory.

In most cases, daemon accounting must be enabled by both the CSA subsystem and the daemon. "Setting Up CSA", page 17, describes how to enable daemon accounting at system startup time. You can also enable daemon accounting after the system has booted.

You can enable accounting for a specified daemon by using the `csaswitch` command. For example, to start tape accounting, you should do the following:

```
/usr/local/sbin/csaswitch -c on -n tape
```

Daemon accounting is disabled at system shutdown (see "Setting Up CSA", page 17). It can also be disabled at any time by the `csaswitch` command when used with the `off` operand. For example, to disable workload management accounting, execute the following command:

```
/usr/local/sbin/csaswitch -c off -n wkmng
```

These dynamic changes using `csaswitch` are not saved across a system reboot.

Setting up User Exits

CSA accommodates the following user exits, which can be called from certain `csarun` states:

csarun state	User exit
ARCHIVE1	/usr/local/sbin/csa.archive1
ARCHIVE2	/usr/local/sbin/csa.archive2
FEF	/var/local/sbin/csa.fef
USEREXIT	/usr/local/sbin/csa.user

CSA accommodates the following user exit, which can be called from certain `csaperiod` states:

csaperiod state	User exit
USEREXIT	/usr/local/sbin/csa.puser

These exits allow an administrator to tailor the `csarun` procedure (or `csaperiod` procedure) to the individual site's needs by creating scripts to perform additional site-specific processing during daily accounting. (Note that the following comments also apply to `csaperiod`).

While executing, `csarun` checks in the `ARCHIVE1`, `ARCHIVE2`, `FEF` and `USEREXIT` states for a shell script with the appropriate name.

If the script exists, it is executed via the shell `.` (`dot`) command. If the script does not exist, the user exit is ignored. The `.` (`dot`) command will not execute a compiled program, but the user exit script can. `csarun` variables are available, without being

exported, to the user exit script. `csarun` checks the return status from the user exit and if it is nonzero, the execution of `csarun` is terminated.

Charging for Workload Management Jobs

By default, SBUs are calculated for all workload management jobs regardless of the workload management termination code of the job. If you do not want to bill portions of a workload management request, set the appropriate `WKMG_TERM_xxxx` variable (termination code) in the `/etc/csa.conf` file to 0, which sets the SBU for this portion to 0.0. This sets the SBU for this portion to 0.0. By default, all portions of a request are billed.

The following table describes the termination codes:

Code	Description
<code>WKMG_TERM_EXIT</code>	Generated when the request finishes running and is no longer in a queued state.
<code>WKMG_TERM_REQUEUE</code>	Written for a request that is requeued.
<code>WKMG_TERM_HOLD</code>	Written for a request that is checkpointed and held.
<code>WKMG_TERM_RERUN</code>	Written when a request is rerun.
<code>WKMG_TERM_MIGRATE</code>	Written when a request is migrated.

Note: The above descriptions of the termination codes are very generic. Different workload managers will tailor the meaning of these codes to suit their products. LSF currently only uses the `WKMG_TERM_EXIT` termination code.

Tailoring CSA Shell Scripts and Commands

Modify the following variables in `/etc/csa.conf` if necessary:

Variable	Description
<code>ACCT_FS</code>	File system on which <code>/var/csa</code> resides. The default is <code>/var</code> .
<code>MAIL_LIST</code>	List of users to whom mail is sent if fatal errors are detected in the accounting shell scripts. The default is <code>root</code> and <code>adm</code> .

WMAIL_LIST	List of users to whom mail is sent if warning errors are detected by the accounting scripts at cleanup time. The default is <code>root</code> and <code>adm</code> .
MIN_BLKs	Minimum number of free blocks needed in <code>#{ACCT_FS}</code> to run <code>csarun</code> or <code>csaperiod</code> . The default is 2000 free blocks. Block size is 1024 bytes.

Using `at` to Execute `csarun`

You can use the `at` command instead of `cron` to execute `csarun` periodically. If your system is down when `csarun` is scheduled to run via `cron`, `csarun` will not be executed until the next scheduled time. On the other hand, `at` jobs execute when the machine reboots if their scheduled execution time was during a down period.

You can execute `csarun` by using `at` in several ways. For example, a separate script can be written to execute `csarun` and then resubmit the job at a specified time. Also, an `at` invocation of `csarun` could be placed in a user exit script, `/usr/local/sbin/csa.user`, that is executed from the `USEREXIT` section of `csarun`. For more information, see "Setting up User Exits", page 41.

Using an Alternate Configuration File

By default, the `/etc/csa.conf` configuration file is used when any of the CSA commands are executed. You can specify a different file by setting the shell variable `CSACONFIG` to another configuration file, and then executing the CSA commands.

For example, you would execute the following commands to use the configuration file `/tmp/myconfig` while executing `csarun`:

```
CSACONFIG=/tmp/myconfig
/usr/local/sbin/csarun 2> /var/csa/nite/fd2log
```

CSA Reports

You can use CSA to create accounting reports. The reports can be used to help track system usage, monitor performance, and charge users for their time on the system.

The CSA daily reports are located in the `/var/csa/sum` directory; periodic reports are located in the `/var/csa/fiscal` directory. To view the reports, go to the ASCII file `rprt.MMDDhhmm` in the report directories.

The CSA reports contain more detailed data than the other accounting reports. For CSA accounting, daily reports are generated by the `csarun` command. The daily report includes the following:

- disk usage statistics
- unfinished job information
- command summary data
- consolidated accounting report
- last login information
- daemon usage report

Periodic reports are generated by the `csaperiod` command. You can also create a disk usage report using the `diskusg` command.

This section describes the following reports:

CSA Daily Report

This section describes the following reports:

- "Consolidated Information Report", page 44
- "Unfinished Job Information Report", page 45
- "Disk Usage Report", page 45
- "Command Summary Report", page 46
- "Last Login Report", page 46
- "Daemon Usage Report", page 47

Consolidated Information Report

The Consolidated Information Report is sorted by user ID and then project ID (project ID is deferred). The following usage values are the total amount of resources used by all processes for the specified user and project during the reporting period.

Heading	Description
PROJECT NAME	Project associated with this resource usage information (deferred)
USER ID	User identifier
LOGIN NAME	Login name for the user identifier
CPU_TIME	Total accumulated CPU time in seconds
KCORE * CPU-MIN	Total accumulated amount of Kbytes of core (physical) memory used per minute of CPU time
KVIRT * CPU-MIN	Total accumulated amount of Kbytes of virtual memory used per minute of CPU time
IOWAIT BLOCK	Total accumulated block I/O wait time in seconds
IOWAIT RAW	Total accumulated raw I/O wait time in seconds

Unfinished Job Information Report

The Unfinished Job Information Report describes jobs which have not terminated and are recycled into the next accounting period.

Heading	Description
JOB ID	Job identifier
USERS	Login name of the owner of this job
PROJECT ID	Project identifier associated with this job (deferred)
STARTED	Beginning time of this job

Disk Usage Report

The Disk Usage Report describes the amount of disk resource consumption by login name.

There are no column headings for this report. The first column gives the user identifier. The second column gives the login name associated with the user identifier. The third column gives the number of disk blocks used by this user.

Command Summary Report

The Command Summary Report summarizes command usage during this reporting period. The usage values are the total amount of resources used by all invocations of the specified command. Commands which were run only once are combined together in the "***other" entry. Only the first 44 command entries are displayed in the daily report. The periodic report displays all command entries.

Heading	Description
COMMAND NAME	Name of the command (program)
NUMBER OF COMMANDS	Number of times this command was executed
TOTAL KCORE-MINUTES	Total amount of Kbytes of core (physical) memory used per minute of CPU time
TOTAL KVIRT-MINUTES	Total amount of Kbytes of virtual memory used per minute of CPU time
TOTAL CPU	Total amount of CPU time used in minutes
TOTAL REAL	Total amount of real (wall clock) time used in minutes
MEAN SIZE KCORE	Average amount of core (physical) memory used in Kbytes
MEAN SIZE KVIRT	Average amount of virtual memory used in Kbytes
MEAN CPU	Average amount of CPU time used in minutes
HOG FACTOR	Total CPU time used divided by the total real time (elapsed time)
K-CHARS READ	Total number of characters read in Kbytes
K-CHARS WRITTEN	Total number of characters written in Kbytes
BLOCKS READ	Total number of blocks read
BLOCKS WRITTEN	Total number of blocks written

Last Login Report

The Last Login Report shows the last login date for each login account listed.

There are no column headings for this report. The first column is the last login date. The second column is the login account name.

Daemon Usage Report

Daemon Usage Report shows reports usage of the workload management and tape daemons (tape is deferred). This report has several individual reports depending upon if there was workload management or tape daemon activity within this reporting period.

The Job Type Report gives the workload management and interactive job usage count.

Heading	Description
Job Type	Type of job (interactive or workload management)
Total Job Count	Number and percentage of jobs per job type
Tape Jobs	Number and percentage of tape jobs associated with these interactive and workload management job (deferred)

The CPU Usage Report gives the workload management and interactive job usage related to CPU usage.

Heading	Description
Job Type	Type of job (interactive or workload management)
Total CPU Time	Total amount of CPU time used in seconds and percentage of CPU time
System CPU Time	Amount of system CPU time used of the total and the percentage of the total time which was system CPU time usage
User CPU Time	Amount of user CPU time used of the total and the percentage of the total time which was user CPU time usage

The workload management Queue Report gives the following information for each workload management queue.

Queue Name	Name of the workload management queue
Number of Jobs	Number of jobs initiated from this queue
CPU Time	Amount of system and user CPU times used by jobs from this queue and percentage of CPU time used
Used Tapes	How many jobs from this queue used tapes
Ave Queue Wait	Average queue wait time before initiation in seconds

Periodic Report

This section describes two periodic reports as follows:

- "Consolidated accounting report", page 48
- "Command summary report", page 49

Consolidated accounting report

The following usage values for the Consolidated accounting report are the total amount of resources used by all processes for the specified user and project during the reporting period.

Heading	Description
PROJECT NAME	Project associated with this resource usage information
USER ID	User identifier
LOGIN NAME	Login name for the user identifier
CPU_TIME	Total accumulated CPU time in seconds
KCORE * CPU-MIN	Total accumulated amount of Kbytes of core (physical) memory used per minute of CPU time of processes
KVIRT * CPU-MIN	Total accumulated amount of Kbytes of virtual memory used per minute of CPU time
IOWAIT BLOCK	Total accumulated block I/O wait time in seconds
IOWAIT RAW	Total accumulated raw I/O wait time in seconds
DISK BLOCKS	Total number of disk blocks used

DISK SAMPLES	Number of times disk accounting was run to obtain the disk blocks used value
FEE	Total fees charged to this user from csachargefee(8)
SBU\$	System billing units charged to this user and project

Command summary report

The following information summarizes command usage during the defined reporting period. The usage values are the total amount of resources used by all invocations of the specified command. Unlike the daily command summary report, the periodic command summary report displays all command entries. Commands executed only once are not combined together into an "***other" entry but are listed individually in the periodic command summary report.

Heading	Description
COMMAND NAME	Name of the command (program)
NUMBER OF COMMANDS	Number of times this command was executed
TOTAL KCORE-MINUTES	Total amount of Kbytes of core (physical) memory used per minute of CPU time
TOTAL KVIRT-MINUTES	Total amount of Kbytes of virtual memory used per minute of CPU time
TOTAL CPU	Total amount of CPU time used in minutes
TOTAL REAL	Total amount of real (wall clock) time used in minutes
MEAN SIZE KCORE	Average amount of core (physical) memory used in Kbytes
MEAN SIZE KVIRT	Average amount of virtual memory used in Kbytes
MEAN CPU	Average amount of CPU time used in minutes
HOG FACTOR	Total CPU time used divided by the total real time (elapsed time)
K-CHARS READ	Total number of characters read in Kbytes
K-CHARS WRITTEN	Total number of characters written in Kbytes

BLOCKS READ	Total number of blocks read
BLOCKS WRITTEN	Total number of blocks written

CSA Man Pages

The man command provides online help on all resource management commands. To view a man page online, type `mancommandname`.

User-Level Man Pages

The following user-level man pages are provided with CSA software:

User-level man page	Description
<code>csacom(1)</code>	Searches and prints the CSA process accounting files.
<code>ja(1)</code>	Starts and stops user job accounting information.

Administrator Man Pages

The following administrator man page is provided with CSA software:

Administrator man page	Description
<code>csaaddc(8)</code>	Combines <code>cacct</code> records.
<code>csabuild(8)</code>	Organizes accounting records into job records.
<code>csachargefee(8)</code>	Charges a fee to a user.
<code>csackpacct(8)</code>	Checks the size of the CSA process accounting file.
<code>csacms(8)</code>	Summarizes command usage from per-process accounting records
<code>csacon(8)</code>	Condenses records from the sorted <code>pacct</code> file.

<code>csacrep(8)</code>	Reports on consolidated accounting data.
<code>csadrep(8)</code>	Reports daemon usage.
<code>csaedit(8)</code>	Displays and edits the accounting information.
<code>csagetconfig(8)</code>	Searches the accounting configuration file for the specified argument.
<code>csajrep(8)</code>	Prints a job report from the sorted <code>pacct</code> file.
<code>csarecy(8)</code>	Recycles unfinished jobs into the next accounting run.
<code>csaswitch(8)</code>	Checks the status of, enables or disables the different types of CSA, and switches accounting files for maintainability.
<code>csaverify(8)</code>	Verifies that the accounting records are valid.

Index

A

- accounting
 - concepts, 6
 - daily accounting, 6
 - job, 6
 - jobs, 6
 - terminology, 6

C

- Comprehensive System Accounting
 - accounting commands, 50
 - administrator commands, 14
 - charging for workload management jobs, 42
 - commands
 - csaaddc, 28
 - csachargefee, 16, 27
 - csackpact, 18
 - csacms, 28
 - csacon, 28
 - csadrep, 28
 - csaedit, 25, 28
 - csaperiod, 4, 16
 - csarecy, 28
 - csarun, 4, 16, 21
 - csaswitch, 16, 17
 - csaverify, 25
 - dodisk, 16
 - ja, 4
 - configuration file
 - See also `"/etc/csa.conf"`, 4, 17
 - configuration variables
 - See also `"/etc/csa.conf"`, 5
 - daemon accounting, 40
 - daily operation overview, 16

- data processing, 26
- data recycling, 30
- enabling or disabling, 7
- `/etc/csa.conf`
 - See also "configuration file", 4
- files and directories, 8
- overview, 3
- recycled data
 - workload management requests, 35
- recycled sessions, 31
- removing recycled data, 31
- reports
 - daily, 44
 - periodic, 48
- SBU's
 - process
 - See also "system billing units", 37
 - See "system billing units", 36
 - tape
 - See also "system billing units", 40
 - workload management
 - See also "system billing units", 39
- setting up CSA, 17
- system billing units, 36
- tailoring CSA, 36
 - commands, 42
 - shell scripts, 42
- terminating jobs, 30
- user commands, 15
- user exits, 41
- verifying and editing data files, 25

F

- Files
 - holidays file (accounting) updating, 19

H

holidays file (accounting) updating, 19

J

Job

job characteristics, 2

job initiators

See also "point of entry processes", 2

Job Limits

point of entry processes

See also "job initiators", 1

jobs

accounting, in, 6