



SunSHIELD Basic Security Module Guide

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043-1100
U.S.A.

Part No: 802-5757
August 1997

Copyright 1997 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, SunSoft, SunDocs, SunExpress, SunSHIELD, SHIELD, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1997 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunSoft, SunDocs, SunExpress, SunSHIELD, SHIELD, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Contents

	Preface	ix
1.	Installation	1
	Enabling BSM	1
	Disabling BSM	2
	BSM and Client-Server Relationships	3
2.	Administering Auditing	5
	More on Auditing	6
	Audit Startup	6
	Audit Classes and Events	7
	Kernel Events	7
	User-Level Events	7
	Audit Records	8
	Audit Flags	8
	Definitions of Audit Flags	8
	Audit Flag Syntax	10
	Prefixes to Modify Previously Set Audit Flags	11
	The <code>audit_control</code> File	11
	Sample <code>audit_control</code> File	12
	User Audit Fields in the <code>audit_user</code> File	13

Process Audit Characteristics	14
Process Preselection Mask	14
Audit ID	15
Audit Session ID	15
Terminal ID	15
How the Audit Trail Is Created	15
audit_data File	16
Audit Daemon's Role	16
What Makes a Directory Suitable	17
Keeping Audit Files Manageable	17
The audit_warn Script	17
Using the auditreduce Command	19
Controlling Audit Costs	21
Cost of Increased Processing Time	21
Cost of Analysis	21
Cost of Storage	21
Auditing Normal Users	23
Auditing Efficiently	23
▼ How to Combine and Reduce audit Files	24
Learning About the Audit Trail	24
More About the Audit Files	25
Handling Nonactive Files Marked not_terminated	27
▼ How to Create Audit Partitions and Export Them	27
▼ How to Configure Auditing	29
▼ How to Plan Audit Configuration	30
Preventing Audit Trail Overflow	33
▼ How to Prevent Audit Trail Overflow	33
The auditconfig Command	34

Setting Audit Policies	36
▼ How to Change Which Events Are in Which Audit Classes	37
Changing Class Definitions	37
3. Audit Trail Analysis	39
Auditing Features	39
Audit User ID	40
Audit Session ID	40
Self-Contained Audit Records	40
Tools for Merging, Selecting, Viewing, and Interpreting Audit Records	40
Audit Record Format	41
Order of Audit Tokens	42
Human-Readable Audit Record Format	42
header Token	43
trailer Token	44
arbitrary Token	44
arg Token	44
attr Token	45
exit Token	45
file Token	45
groups Token	46
in_addr Token	46
ip Token	46
ipc Token	46
ipc_perm Token	47
iport Token	47
opaque Token	47
path Token	47
process Token	48

return Token	48
seq Token	49
socket Token	49
subject Token	49
text Token	50
Using the auditreduce Command	50
How auditreduce Helps in a Distributed System	50
Using auditreduce	51
Other Useful auditreduce Options	52
Using praudit	53
4. Device Allocation	55
Risks Associated With Device Use	55
Components of the Device-Allocation Mechanism	56
Using the Device-Allocation Utilities	57
The Allocate Error State	58
The device_maps File	58
The device_allocate File	59
Device-Clean Scripts	61
Object Reuse	61
Writing New Device-Clean Scripts	63
Setting Up Lock Files	63
▼ How to Set Up Lock Files for a Device to Be Made Allocatable	63
Managing and Adding Devices	66
▼ How to Manage Devices	66
▼ How to Add a New Allocatable Device	66
Using Device Allocations	67
▼ How to Allocate a Device	68
▼ How to Deallocate a Device	68

A.	Audit Record Descriptions	69
	Audit Record Structure	69
	Audit Token Structure	70
	arbitrary Token	71
	arg Token	73
	attr Token	73
	exec_args Token	73
	exec_env Token	74
	exit Token	74
	file Token	75
	groups Token (Obsolete)	75
	header Token	76
	in_addr Token	76
	ip Token	76
	ipc Token	77
	ipc_perm Token	78
	iport Token	78
	newgroups Token	78
	opaque Token	79
	path Token	79
	process Token	80
	return Token	80
	seq Token	81
	socket Token	81
	socket-inet Token	81
	subject Token	82
	text Token	82
	trailer Token	82

Audit Records	83
General Audit Record Structure	83
Kernel-Level Generated Audit Records	84
User-Level Generated Audit Records	156
Event-to-System Call Translation	172
B. BSM Reference	185
Index	191

Preface

The Solaris™ SHIELD™ Basic Security Module (BSM) provides additional security features defined as C2 in the Trusted Computer System Evaluation Criteria (TCSEC) that are not supplied in standard UNIX®. The features provided by the BSM are the security auditing subsystem and a device-allocation mechanism that provides the required object-reuse characteristics for removable or assignable devices. C2 discretionary-access control, as well as C2 identification and authentication features are provided by the standard Solaris system.

Who Should Use This Book

SunSHIELD Basic Security Module Guide is intended for the system administrator whose duties include setting up and maintaining BSM. Familiarity with basic system administration concepts and with a text editor are helpful.

How This Book Is Organized

Chapter 1 describes enabling and disabling the BSM. Topics covered include how to enable the Solaris system to use these additional security features, and how clients and servers interact in an enabled environment.

Chapter 2 explains the system management and configuration of the auditing subsystem. Topics discussed include managing audit trail storage, determining global and per-user preselection, and setting site-specific configuration options.

Chapter 3 details processes for audit trail analysis and postprocessing. Topics discussed include overall audit record structure and formats, the audit trail printing utility, and the audit record selection and merging utility.

Chapter 4 describes the allocation mechanism for removable or assignable devices. Topics discussed include setting up and administering allocatable device files and use of the allocation mechanism by nonprivileged users.

Appendix A describes in detail the content of the audit records generated.

Appendix B lists and describes the man pages added for the Solaris SunSHIELD™ Basic Security Module.

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at <http://www.sun.com/sunexpress>.

What Typographic Changes Mean

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

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name % You have mail.</code>
AaBbCc123	What you type, contrasted with on-screen computer output	<code>machine_name% su</code> Password:

TABLE P-1 Typographic Conventions (continued)

Typeface or Symbol	Meaning	Example
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

Shell Prompts in Command Examples

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

TABLE P-2 Shell Prompts

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

Installation

Starting with the Solaris 2.3 release, BSM has been included in the full release and is part of the release media. You do not need to install BSM separately because BSM is now enabled or disabled by running one of two simple scripts. All of the BSM software is included in the initial system installation, provided you install the following packages:

- SUNWcar – Core architecture
- SUNWcsr – Core SPARC
- SUNWcsu – Core SPARC
- SUNWhea – Header files
- SUNWman – Online manual pages

The following procedures should be performed only by root. Additionally, the commands should be run only on a server or standalone system and never on a diskless client.

- “Enabling BSM” on page 1
- “Disabling BSM” on page 2
- “BSM and Client-Server Relationships” on page 3

Enabling BSM

After becoming root, bring the system into the single-user mode using `telinit` (see the `init(1M)` man page).

```
# /etc/telinit 1
```

In single-user mode, change directories to the `/etc/security` directory, and execute the `bsmconv` script located there. The script sets up a standard Solaris machine to run BSM after a reboot.

```
# cd /etc/security
# ./bsmconv
```

After the script finishes, halt the system with the `telinit` command. Then reboot the system to bring it up as a multiuser BSM system.

```
# /etc/telinit 6
```

Note - The `bsmconv` script adds a line to `/etc/system` to disable the ability to abort the system using the Stop-a keyboard sequence. If you want to retain the ability to abort the system using the Stop-a keyboard sequence, you must comment out the line that reads “`set abort_enable = 0`” in `/etc/system`.

Disabling BSM

If at some point BSM is no longer required, you can disable it by running `bsmunconv` (see the `bsmconv(1M)` man page). Again, first bring the system into the single-user mode using `telinit`, then change to the `/etc/security` directory and run `bsmunconv`.

```
# /etc/telinit 1
# cd /etc/security
# ./bsmunconv
```

After unconvverting the system, reboot it to run as a multiuser Solaris machine.

```
# /etc/telinit 6
```

Note - The `bsmunconv` script removes the line in `/etc/system` which disables the ability to abort the system using the Stop-a keyboard sequence. If you want to continue to disable the ability to abort the system using the Stop-a keyboard sequence after running the `bsmunconv` script, you must reenter a line that reads “`set abort_enable = 0`” in `/etc/system`.

BSM and Client-Server Relationships

The Solaris 2.1 release required two additional procedures for adding and deleting diskless clients from a BSM-enabled system. With the inclusion of BSM in the Solaris 2.3 and later releases, those procedures are no longer necessary. Enabling BSM on a server now automatically enables the BSM features on all of that server's clients.

Administering Auditing

This chapter describes how to set up and administer auditing. Auditing allows system administrators to monitor the actions of the users. The auditing mechanism allows an administrator to detect potential security breaches. Auditing can reveal suspicious or abnormal patterns of system usage and provide the means to trace suspect actions back to a specific user. Auditing can serve as a deterrent: if users know that their actions are likely to be audited, they might be less likely to attempt malicious activities.

- “More on Auditing” on page 6
- “Audit Startup” on page 6
- “Audit Classes and Events” on page 7
- “Audit Flags” on page 8
- “Process Audit Characteristics” on page 14
- “How the Audit Trail Is Created” on page 15
- “Controlling Audit Costs” on page 21
- “Auditing Normal Users” on page 23
- “Auditing Efficiently” on page 23
- “Learning About the Audit Trail” on page 24
- “Preventing Audit Trail Overflow” on page 33
- “Setting Audit Policies” on page 36
- “Changing Class Definitions” on page 37

More on Auditing

Successful auditing depends on two other security features: identification and authentication. At login, after a user supplies a user name and password, a unique audit ID is associated with the user's process. The audit ID is inherited by every process started during the login session. Even if a user changes identity (see the `su(1M)` man page), all actions performed are tracked with the same audit ID.

Auditing makes it possible to:

- Monitor security-relevant events that take place on the system
- Record the events in an audit trail
- Detect misuse or unauthorized activity (by analyzing the audit trail)

During system configuration, the system administrator selects which activities to monitor. The administrator may also fine-tune the degree of auditing that is done for individual users.

After audit data is collected, audit-reduction and interpretation tools allow the examination of interesting parts of the audit trail. For example, you may choose to look at audit records for individual users or groups, look at all records for a certain type of event on a specific day, or select records that were generated at a certain time of day.

The rest of this chapter describes how to set up and administer auditing. Chapter 4 describes how to interpret the audit data.

Audit Startup

Auditing is enabled by starting up the audit daemon, (see the `auditd(1M)` man page). This can be done manually by executing `/usr/sbin/auditd` as root.

The existence of a file with the path name `/etc/security/audit_startup` causes the audit daemon to be run automatically when the system enters multiuser mode. The file is actually an executable script that is invoked as part of the startup sequence just prior to the execution of the audit daemon (see the `audit_startup(1M)` man page). A default `audit_startup` script that automatically configures the event to class mappings and sets the audit policies is set up during the BSM package installation.

Audit Classes and Events

Security-relevant actions may be audited. The system actions that are auditable are defined as *audit events* in the `/etc/security/audit_event` file. Each auditable event is defined in the file by a symbolic name, an event number, a set of preselection classes, and a short description (see the `audit_event(4)` man page).

Most events are attributable to an individual user. However, some events are nonattributable because they occur at the kernel-interrupt level or before a user is identified and authenticated. Nonattributable events are auditable as well.

Each audit event is also defined as belonging to an *audit class* or classes. By assigning events into classes, an administrator can more easily deal with large numbers of events. When naming a class, one simultaneously addresses all of the events in that class. The mapping of audit events to classes is configurable and the classes themselves are configurable. These configuration changes can be made in the `audit_event` file.

Whether or not an auditable event is recorded in the audit trail depends on whether the administrator preselects a class for auditing that includes the specific event. Out of 32 possible audit classes, 18 are defined. The 18 classes include the two global classes; `all` and `no`.

Kernel Events

Events generated by the kernel (system calls) have event numbers between 1 and 2047. The event names for kernel events begin with `AUE_`, followed by an uppercase mnemonic for the event. For example, the event number for the `creat()` system call is 4 and the event name is `AUE_CREAT`.

User-Level Events

Events generated by application software outside the kernel range from 2048 to 65535. The event names begin with `AUE_`, followed by a lowercase mnemonic for the event. Check the file, `/etc/security/audit_event`, for exact numbers of individual events. Table 2-1 shows general categories of user-related events.

TABLE 2-1 Audit Event Categories

Number Range	Type of Event
2048-65535	User-level audit events
2048-32767	Reserved for SunOS user-level programs
32768-65536	Available for third-party applications

Audit Records

Each *audit record* describes the occurrence of a single audited event and includes such information as who did the action, which files were affected, what action was attempted, and where and when it occurred.

The type of information saved for each audit event is defined as a set of *audit tokens*. Each time an audit record is created for an event, the record contains some or all of the tokens defined for it, depending on the nature of the event. The audit record descriptions in Appendix A list all the audit tokens defined for each event and what each token means.

Audit records are collected in a trail (see the `audit.log(4)` man page) and may be converted to a human readable format by `praudit` (see the `praudit(1M)` man page). See Chapter 3 for details.

Audit Flags

Audit flags indicate classes of events to audit. Machine-wide defaults for auditing are specified for all users on each machine by flags in the `audit_control` file, which is described in “The `audit_control` File” on page 11.

The system administrator can modify what gets audited for individual users by putting audit flags in a user’s entry in the `audit_user` file. The audit flags are also used as arguments to `auditconfig` (see the `auditconfig(1M)` man page).

Definitions of Audit Flags

Each predefined audit class is shown in Table 2-2 with the audit flag (which is the short name that stands for the class), the long name, a short description, and a longer

definition. The system administrator uses the audit flags in the auditing configuration files to specify which classes of events to audit. Additional classes can be defined and existing classes can be renamed by modifying the `audit_class` file (see the `audit_class(4)` man page).

TABLE 2-2 Audit Classes

Short Name	Long Name	Short Description
no	no_class	Null value for turning off event preselection
fr	file_read	Read of data, open for reading, and so forth
fw	file_write	Write of data, open for writing, and so forth
fa	file_attr_acc	Access of object attributes: <code>stat</code> , <code>pathconf</code> , and so forth
fm	file_attr_mod	Change of object attributes: <code>chown</code> , <code>flock</code> , and so forth
fc	file_creation	Creation of object
fd	file_deletion	Deletion of object
cl	file_close	<code>close</code> system call
pc	process	Process operations: <code>fork</code> , <code>exec</code> , <code>exit</code> , and so forth
nt	network	Network events: <code>bind</code> , <code>connect</code> , <code>accept</code> , and so forth
ip	ipc	System V IPC operations
na	non_attrib	Nonattributable events
ad	administrative	Administrative actions
lo	login_logout	Login and logout events
ap	application	Application-defined event
io	ioctl	<code>ioctl</code> system call
ex	exec	Program execution

TABLE 2-2 Audit Classes (continued)

Short Name	Long Name	Short Description
ot	other	Miscellaneous
all	all	All flags set

Audit Flag Syntax

Depending on the prefixes, a class of events can be audited whether it succeeds or fails, or only if it succeeds, or only if it fails. The format of the audit flag is shown here.

prefixflag

Table 2-3 shows prefixes that specify whether the audit class is audited for success or failure or both.

TABLE 2-3 Prefixes Used in Audit Flags

Prefix	Definition
none	Audit for both success and failure
+	Audit for success only
-	Audit for failure only

To give an example of how these work together, the audit flag `lo` means “all successful attempts to log in and log out and all failed attempts to log in.” (You cannot fail an attempt to logout.) For another example, the `-all` flag refers to all failed attempts of any kind, and the `+all` flag refers to all successful attempts of any kind.



Caution - The `-all` flag can generate large amounts of data and fill up audit file systems quickly, so use it only if you have extraordinary reasons to audit everything.

Prefixes to Modify Previously Set Audit Flags

Use the following prefixes in any of three ways: in the `flags` line in the `audit_control` file to modify already-specified flags, in `flags` in the user's entry in the `audit_user` file, or with `auditconfig` (see the `auditconfig(1M)` man page).

The prefixes in the following table, along with the short names of audit classes, turn on or turn off previously specified audit classes. These prefixes turn on or off previously specified flags only.

TABLE 2-4 Prefixes Used to Modify Already-Specified Audit Flags

Prefix	Definition
<code>^-</code>	Turn off for failed attempts
<code>^+</code>	Turn off for successful attempts
<code>^</code>	Turn off for both failed and successful attempts

The `^-` prefix is used in the `flags` line in the following example from an `audit_control` file.

In the sample screen below, the `lo` and `ad` flags specify that all logins and administrative operations are to be audited when they succeed and when they fail. The `-all` means audit "all failed events." Because the `^-` prefix means "turn off auditing for the specified class for failed attempts," the `^-fc` flag modifies the previous flag that specified auditing of all failed events; the two fields together mean "audit all failed events, except failed attempts to create file system objects."

```
flags:lo,ad,-all,^-fc
```

The `audit_control` File

An `audit_control` file on each machine is read by the audit daemon (see the `audit_control(4)` man page). The `audit_control` file is located in the `/etc/security` directory. A separate `audit_control` file is maintained on each machine because machines in the distributed system may be mounting their audit file systems from different locations or specifying them in a different order. For

example, the primary audit file system for machineA might be the secondary audit file system for machineB.

You specify four kinds of information in four kinds of lines in the `audit_control` file:

- The *audit flags* line (`flags:`) contains the audit flags that define what classes of events are audited for all users on the machine. The audit flags specified here are referred to as the *machine-wide audit flags* or the *machine-wide audit preselection mask*. Audit flags are separated by commas, with no spaces.
- The *nonattributable flags* line (`naflags:`) contains the audit flags that define what classes of events are audited when an action cannot be attributed to a specific user. The flags are separated by commas, with no spaces.
- The *audit threshold* line (`minfree:`) defines the minimum free-space level for all audit file systems. See “What Makes a Directory Suitable” on page 17. The `minfree` percentage must be greater than or equal to 0. The default is 20 percent.
- The *directory definition* lines (`dir:`) define which audit file systems and directories the machine will use to store its audit trail files. There may be one or more directory definition lines. The order of the `dir:` lines is significant, because `auditd` opens audit files in the directories in the order specified (see the `audit(1M)` man page). The first audit directory specified is the primary audit directory for the machine, the second is the secondary audit directory where the audit daemon puts audit trail files when the first one fills, and so forth.

The administrator creates an `audit_control` file during the configuration process on each machine.

After the `audit_control` file is created during system configuration, the administrator can edit it. After a change, the administrator runs `audit -s` to instruct the audit daemon to reread the `audit_control` file.

Note - The `audit -s` command does not change the preselection mask for existing processes. Use `autoconfig`, `setaudit` (see the `getaudit(2)` man page), or `auditon` for existing processes.

Sample `audit_control` File

Following is a sample `audit_control` file for the machine `dopey`. `dopey` uses two audit file systems on the audit server `blinken`, and a third audit file system mounted from the second audit server `winken`, which is used only when the audit file systems on `blinken` fill up or become unavailable. The `minfree` value of 20 percent specifies that the warning script (see the `audit_warn(1M)` man page) is run when the file systems are 80 percent filled and the audit data for the current machine will be stored in the next available audit directory, if any. The flags specify that all logins and administrative operations are to be audited (whether or not they succeed),

and that failures of all types except failures to create a file system object are to be audited.

```
flags:lo,ad,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blinken/files
dir:/etc/security/audit/blinken.1/files
#
# Audit filesystem used when blinken fills up
#
dir: /etc/security/audit/winken
```

User Audit Fields in the `audit_user` File

If it is desirable to audit some users differently from others, the administrator may edit the `audit_user` file to add audit flags for individual users. If specified, these flags are combined with the system-wide flags specified in the audit control file to determine which classes of events to audit for that user. The flags the administrator adds to the user's entry in the `audit_user` file modify the defaults from the `audit_control` file in two ways: by specifying a set of event classes that are never to be audited for this user or by specifying a set of event classes that are always to be audited.

In the `audit_user` file entry for each user, there are three fields. The first field is the *username*, the second field is the *always-audit* field, and the third is the *never-audit* field. The two auditing fields are processed in sequence, so auditing is enabled by the first field and turned off by the second.

Note - Avoid the common mistake of leaving the `all` set in the *never-audit* field. This causes all auditing to be turned off for that user, overriding the flags set in the *always-audit* field.

Using the *never-audit* flags for a user is not the same as removing classes from the *always-audit* set. For example, suppose (as shown in the examples below), you have a user `fred` for whom you want to audit everything except successful reads of file system objects. (This is a good way to audit almost everything for a user while generating only about three-quarters of the audit data that would be produced if all data reads were also audited.) You also want to apply the system defaults to `fred`. Here are two possible `audit_user` entries.

The correct entry:

```
fred:all,^+fr:
```

The incorrect entry:

```
fred:all:+fr
```

The first example says, “always audit everything except successful file-reads.” The second example says “always audit everything, but never audit successful file-reads.” The second example is incorrect because it overrides the system default. The first example achieves the desired effect: any earlier default applies, as well as what’s specified in the `audit_user` entry.

Note - Successful events and failed events are treated separately, so a process can (for example) generate more audit records when an error occurs than when the event is successful.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- Process preselection mask
- Audit ID
- Audit Session ID
- Terminal ID (port ID, machine ID)

Process Preselection Mask

When a user logs in, `login` combines the machine-wide audit flags from the `audit_control` file with the user-specific audit flags (if any) from the `audit_user` file, to establish the *process preselection mask* for the user’s processes. The process preselection mask specifies whether events in each audit event class are to generate audit records.

The algorithm for obtaining the process preselection mask is as follows: the audit flags from the `flags:` line in the `audit_control` file are added to the flags from the *always-audit* field in the user’s entry in the `audit_user` file. The flags from the *never-audit* field from the user’s entry in the `audit_user` file are then subtracted from the total:

user’s process preselection mask = (flags: line + always audit flags) - never audit flags

Audit ID

A process also acquires its audit ID when the user logs in, and this audit ID is inherited by all child processes started by the user's initial process. The audit ID helps enforce accountability. Even after a user becomes root, the audit ID remains the same. The audit ID that is saved in each audit record allows the administrator to always trace actions back to the original user who had logged in.

Audit Session ID

The audit session ID is assigned at login and inherited by all descendant processes.

Terminal ID

The terminal ID consists of the host name and the Internet address, followed by a unique number that identifies the physical device on which the user logged in. Most of the time the login will be through the console and the number that corresponds to the console device is 0.

How the Audit Trail Is Created

The *audit trail* is created by the audit daemon (see the `auditd(1M)` man page). The audit daemon starts on each machine when the machine is brought up. After `auditd` starts at boot time, it is responsible for collecting the audit trail data and writing the audit records into *audit files*, which are also called *audit log files*. See the `audit.log(4)` man page for a description of the file format.

The audit daemon runs as root. All files it creates are owned by root. Even when `auditd` has no classes to audit, `auditd` continuously operates, looking for a place to put audit records. The `auditd` operations continue even if the rest of the machine's activities are suspended because the kernel's audit buffers are full. The audit operations can continue because `auditd` is not audited.

Only one audit daemon may run at a time. An attempt to start a second one results in an error message, and the new one exits. If there is a problem with the audit daemon, you should try using `audit -t` to terminate `auditd` gracefully, then restart it manually.

The `audit_warn` script is run by `auditd` whenever the daemon switches audit directories or encounters difficulty (such as a lack of storage). As distributed, the `audit_warn` script sends mail to an `audit_warn` alias and sends a message to the

console. Your site should customize `audit_warn` to suit your needs. Customizing the `audit_warn` script is described in “The `audit_warn` Script” on page 17.

audit_data File

When `auditd` starts on each machine, it creates the file `/etc/security/audit_data`. The format of the file consists of a single entry with the two fields separated by a colon (see the `audit_data(4)` man page). The first field is the audit daemon’s process ID, and the second field is the path name of the audit file to which the audit daemon is currently writing audit records. Here is an example:

```
# cat /etc/security/audit_data
116:/etc/security/audit/blinken.1/files/19910320100002.not_terminated.lazy
```

Audit Daemon’s Role

The following list summarizes what the audit daemon, `auditd`, does.

- `auditd` opens and closes audit log files in the directories specified in the `audit_control` file in the order in which they are specified.
- `auditd` reads audit data from the kernel and writes it to an audit file.
- `auditd` executes the `audit_warn` script when the audit directories fill past limits specified in the `audit_control` file. The script, by default, sends warnings to the `audit_warn` alias and to the console.
- With the system default configuration, when all audit directories are full, processes that generate audit records are suspended. In addition, `auditd` writes a message to the console and to the `audit_warn` alias. (The auditing policy can be reconfigured with `autoconfig`.) At this point only the system administrator can log in to write audit files to tape, delete audit files from the system, or do other cleanup.

When the audit daemon starts as the machine is brought up to multiuser mode, or when the audit daemon is instructed by the `audit -s` command to reread the file after the file has been edited, `auditd` determines the amount of free space necessary and reads the list of directories from the `audit_control` file. It then uses those directories as possible locations for creating audit files.

The audit daemon maintains a pointer into this list of directories, starting with the first. Every time the audit daemon needs to create an audit file, it puts the file into the first available directory in the list, starting at the audit daemon’s current pointer. The pointer may be reset to the beginning of the list if the administrator enters the `audit -s` command. When you use the `audit -n` command to instruct the

daemon to switch to a new audit file, the new file is created in the same directory as the current file.

What Makes a Directory Suitable

A directory is *suitable* to the audit daemon if it is accessible to the audit daemon, which means that it must be mounted, that the network connection (if remote) permits successful access, and that the permissions on the directory allow access. Also, in order for a directory to be suitable for audit files, it must have sufficient free space remaining. You can edit the `minfree:` line in the `audit_control` file to change the default of 20 percent. To give an example of how the `minfree` percentage is applied, if the default minimum free space of 20 percent is accepted, an email notice is sent to the `audit_warn` alias whenever a file system becomes more than 80 percent full.

When no directories on the list have enough free space left, the daemon starts over from the beginning of the list and picks the first accessible directory that has any space available until the hard limit is reached. In the default configuration, if no directories are suitable, the daemon stops processing audit records, and they accumulate within the kernel until all processes generating audit records are suspended.

Keeping Audit Files Manageable

To keep audit files at a manageable size, a `cron` job can be set up that periodically switches audit files (see the `cron(1M)` man page). Intervals might range from once per hour to twice per day, depending on the amount of audit data being collected. The data can then be filtered to remove unnecessary information, and then compressed.

The `audit_warn` Script

Whenever the audit daemon encounters an unusual condition while writing audit records, it invokes the `/etc/security/audit_warn` script. See the `audit_warn(1M)` man page. This script can be customized by your site to warn of conditions that might require manual intervention or to handle them automatically. For all error conditions, `audit_warn` writes a message to the console and sends a message to the `audit_warn` alias. This alias should be set up by the administrator after enabling BSM.

When the following conditions are detected by the audit daemon, it invokes `audit_warn`.

- An audit directory has become more full than the `minfree` value allows. (The `minfree` or `soft` limit is a percentage of the space available on an audit file system.)

The `audit_warn` script is invoked with the string `soft` and the name of the directory whose space available has gone below the minimum. The audit daemon switches automatically to the next suitable directory and writes the audit files there until this new directory reaches its `minfree` limit. The audit daemon then goes to each of the remaining directories in the order listed in `audit_control`, and writes audit records until each is at its `minfree` limit.

- All the audit directories are more full than the `minfree` threshold.

The `audit_warn` script is invoked with the string `allsoft` as an argument. A message is written to the console and mail is sent to the `audit_warn` alias.

When all audit directories listed in `audit_control` are at their `minfree` limits, the audit daemon switches back to the first one, and writes audit records until the directory completely fills.

- An audit directory has become completely full with no space remaining.

The `audit_warn` script is invoked with the string `hard` and the name of the directory as arguments. A message is written to the console and mail is sent to the `audit_warn` alias.

The audit daemon switches automatically to the next suitable directory with any space available, if any. The audit daemon goes to each of the remaining directories in the order listed in `audit_control` and writes audit records until each is full.

- All the audit directories are completely full. The `audit_warn` script is invoked with the string `allhard` as an argument.

In the default configuration, a message is written to the console and mail is sent to the `audit_warn` alias. The processes generating audit records are suspended. The audit daemon goes into a loop, waiting for space to become available, and resumes processing audit records when that happens. While audit records are not being processed, no auditable activities take place—every process that attempts to generate an audit record is suspended. This is one reason why you would want to set up a separate audit administration account that could operate without any auditing enabled. The administrator could then operate without being suspended.

- An internal error occurs: another audit daemon process is already running (string `ebusy`), a temporary file cannot be used (string `tmpfile`), the `auditsvc()` system call fails (string `auditsvc`), or a signal was received during auditing shutdown (string `postsigterm`).

Mail is sent to the `audit_warn` alias.

- A problem is discovered with the `audit_control` file's contents. By default, mail is sent to the `audit_warn` alias and a message is sent to the console.

Using the `auditreduce` Command

Use `auditreduce` to merge together audit records from one or more input audit files or to perform a post selection of audit records. See the `auditreduce(1M)` man page. To merge the entire audit trail, the system administrator enters the command on the machine on which all the audit file systems for the distributed system are mounted.

When multiple machines running BSM are administered as part of a distributed system, each machine performs auditable events, and each machine writes audit records to its own machine-specific audit file. This procedure simplifies software and is robust in the face of machine failures. However, without `auditreduce`, you would have to look at every one of the files to determine what a particular user did because each machine produces its own set of audit files.

The `auditreduce` command makes the job of maintaining the whole audit trail practical. Using `auditreduce` (or shell scripts you write yourself to provide a higher-level interface), you can read the logical combination of all audit files in the system as a single audit trail without regard to how the records were generated or where they are stored.

The `auditreduce` program operates on the audit records produced by the audit daemon. Records from one or more audit files are selected and merged into a single, chronologically ordered output file. The merging and selecting functions of `auditreduce` are logically independent. `auditreduce` selects messages from the input files as the records are read, before the files are merged and written to disk.

Without options, `auditreduce` merges the entire audit trail (which consists of all of the audit files in all of the subdirectories in the audit root directory `/etc/security/audit`) and sends all the audit records to standard output. Making the records human-readable is done by the `praudit` command.

Following are some of the actions performed by some of the options to the `auditreduce` command.

- You can request that the output contain audit records generated by only certain audit flags.
- You can request audit records generated by one particular user.
- You can request audit records generated on specific dates.

With no arguments, `auditreduce` looks in all subdirectories below `/etc/security/audit`, the default audit root directory, for a `files` directory in which the `date.date.hostname` files reside. The `auditreduce` command is very useful when the audit data for different hosts (Figure 2-1) or for different audit servers (Figure 2-2) resides in separate directories.

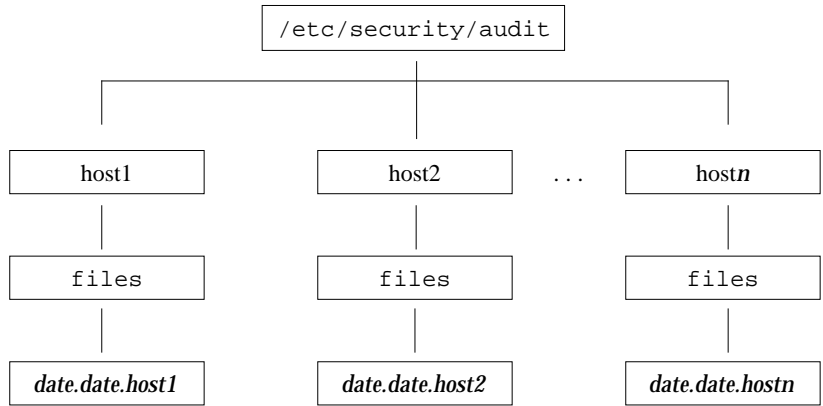


Figure 2-1 Audit Trail Separated by Host

The audit data may not be in the default directory; perhaps because the partition for `/etc/security/audit` is very small or because you want to store audit data on another partition without symbolically linking that partition to `/etc/security/audit`. You can give `auditreduce` another directory (`-R`) to substitute for `/etc/security/audit`, or you can specify one particular subdirectory (`-S`):

```
# auditreduce -R /var/audit-alt
# auditreduce -S /var/audit-alt/host1
```

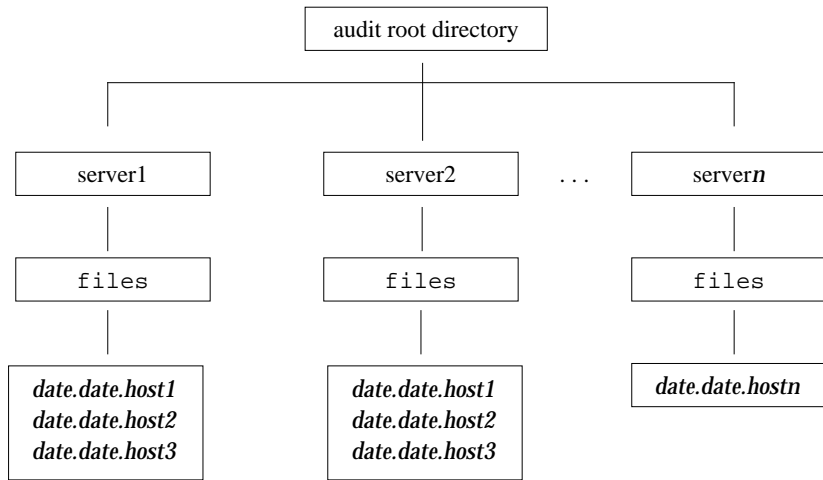


Figure 2-2 Audit Trail Separated by Server

You can direct `auditreduce` to treat only certain files by specifying them as command arguments:


```
# auditreduce /var/audit/bongos/files/1993*.1993*.bongos
```

The `auditreduce(1M)` man page for `auditreduce` lists other options and provides additional examples for using the command.

Controlling Audit Costs

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following three costs of auditing:

- Costs of increased processing time
- Costs of analysis of audit data
- Costs of storage of audit data

Cost of Increased Processing Time

The cost of increased processing time is the least significant of the three costs of auditing. The first reason is that auditing generally does not occur during computational-intensive tasks—image processing, complex calculations, and so forth. The other reason that processing cost is usually insignificant is that single-user workstations have plenty of extra CPU cycles.

Cost of Analysis

The cost of analysis is roughly proportional to the amount of audit data collected. The cost of analysis includes the time it takes to merge and review audit records, and the time it takes to archive them and keep them in a safe place.

The fewer records you generate, the less time it takes to analyze them, so upcoming sections describe how you can reduce the amount of data collected, while still providing enough coverage to achieve your site's security goals.

Cost of Storage

Storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users

- Number of machines
- Amount of use
- Degree of security required

Because the factors vary from one situation to the next, no formula can determine in advance the amount of disk space to set aside for audit data storage.

Full auditing (with the `all` flag) can fill up a disk in no time. Even a simple task like compiling a program of modest size (for example, 5 files, 5000 lines total) in less than a minute could generate thousands of audit records, occupying many megabytes of disk space. Therefore, it is very important to use the preselection features to reduce the volume of records generated. For example, omitting the `fr` class instead of all classes can reduce the audit volume by more than two-thirds. Efficient audit file management is also important after the audit records are created, to reduce the amount of storage required.

The following sections give some ideas on how to reduce the costs of storage by auditing selectively to reduce the amount of audit data collected, while still meeting your site's security needs. Also discussed are how to set up audit file storage and archiving procedures to reduce storage requirements.

Before configuring auditing, understand the audit flags and the types of events they flag. Develop a philosophy of auditing for your organization that is based on the amount of security your site requires, and the types of users you administer.

Unless the process audit preselection mask is modified dynamically, the audit characteristics in place when a user logs in are inherited by all processes during the login session, and, unless the databases are modified, the process preselection mask applies in all subsequent login sessions.

Dynamic controls refer to controls put in place by the administrator while processes are running. These persist only while the affected processes (and any of their children) exist, but will not continue in effect at the next login. Dynamic controls apply to one machine at a time, since the `audit` command only applies to the current machine where you are logged in. However, if you make dynamic changes on one machine, you should make them on all machines at the same time.

Each process has two sets of one-bit flags for audit classes. One set controls whether the process is audited when an event in the class is requested successfully; the other set, when an event is requested but fails (for any reason). It is common for processes to be more heavily audited for failures than for successes, since this can be used to detect attempts at browsing and other types of attempts at violating system security.

In addition to supplying the per-user audit control information in the static databases, you may dynamically adjust the state of auditing while a user's processes are active on a single machine.

To change the audit flags for a specific user to a supplied value, use the `auditconfig` command with the `-setpmask`, `-setsmask`, or `-setumask` options. The command changes the process audit flags for one process, one audit session ID,

or one audit user ID respectively. See the `auditconfig(1M)` man page and “The `auditconfig` Command” on page 34.

Auditing Normal Users

The administrator sets up auditing for the default configuration. You may want all users and administrators to be audited according to the system-wide audit flags you specified in the `audit_control` file. To fine-tune auditing for individual users, you modify the users entries in the `audit_user` file. See the `audit_control(4)` and `audit_user(4)` man pages. You may also choose to add audit flags to users’ entries at the time you add new users, and you should probably set up auditing for the new user just after you unlock the account and configure the security attributes for that user.

Auditing Efficiently

Techniques in this section can allow you to achieve your organization’s security goals while auditing more efficiently:

- Random auditing of only a certain percentage of users at any one time
- Real-time monitoring of the audit data for unusual behaviors. (You set up procedures to monitor the audit trail as it is generated for certain activities and to trigger higher levels of auditing of particular users or machines when suspicious events occur.)
- Reducing the disk-storage requirements for audit files by combining, reducing, and compressing them, and developing procedures for storing them offline

Another technique is to monitor the audit trail in real time. You can write a script to trigger an automatic increase in the auditing of certain users or certain machines in response to detection of unusual events.

To monitor the audit trail in real time and watch for unusual events, write a script that monitors creation of audit files on all the audit file servers and processes them with the `tail` command (see the `tail(1)` man page). The output of `tail -of`, piped through `praudit`, yields a stream of audit records as soon as they are generated. This stream can be analyzed for unusual message types or other indicators and delivered to the auditor or used to trigger automatic responses. The script should be written to constantly watch the audit directories for the appearance of new `not_terminated` audit files, and also the termination of outstanding `tail` processes when their files are no longer being written to (that is, have been replaced by new ones).

▼ How to Combine and Reduce `audit` Files

- ◆ Use `auditreduce` with the `-O` option to combine several audit files into one and save them in a specified output file.

Although `auditreduce` can do this type of combination and deletion automatically (see the `-C` and `-D` options in the `auditreduce(1M)` man page), it is often easier to select the files manually (perhaps with `find`) and use `auditreduce` to combine just the named set of files. When `auditreduce` is used this way, it merges all the records from its input files into a single output file. The input files should then be deleted, and the output file kept in a directory named `/etc/security/audit/server-name/files` so that `auditreduce` can find it.

```
# auditreduce -O combined-filename
```

The `auditreduce` program can also reduce the number of records in its output file by eliminating the less interesting ones as it combines the input files. You might use `auditreduce` to eliminate all except the login/logout events in audit files over a month old, assuming that if you needed to retrieve the complete audit trail, you could recover it from backup tapes.

```
# auditreduce -O daily.summary -b 19930513 -c lo; compress *daily.summary
# mv *daily.summary /etc/security/summary.dir
```

Learning About the Audit Trail

This section describes where audit files are stored, how they are named, and how to manage audit file storage throughout a distributed system.

The audit trail is created when the audit daemon, `auditd`, is started, and is closed when a new audit trail file is created, or when the audit daemon is terminated. The audit trail may consist of audit files in several audit directories, or an audit directory may contain several audit trails.

Most often the audit directories will be separate audit file system partitions. Even though they can be included in other file systems, this is not recommended.

As a rule, locate primary audit directories in dedicated audit file systems mounted on separate partitions. Normally, all audit file systems are subdirectories of `/etc/security/audit`. These should be dedicated audit file systems to ensure

that normal use of the partition is not interrupted, if the audit directories become filled with audit files.

Even though you can physically locate audit directories within other file systems that are not dedicated to auditing, do not do this except for directories of last resort. Directories of last resort are directories where audit files are written only when there is no other suitable directory available. One other scenario where locating audit directories outside of dedicated audit file systems could be acceptable is in a software development environment where auditing is optional, and where it is more important to make full use of disk space than to keep an audit trail. Putting audit directories within other file systems would never be acceptable in a security-conscious production environment.

A diskfull machine should have at least one local audit directory, which it can use as a directory of last resort, if unable to communicate with the audit server.

Mount audit directories with the read-write (*rw*) option. When mounting audit directories remotely (using NFS software), also use the *intr* option.

List the audit file systems on the audit server where they reside. The export list should include all machines in the configuration.

More About the Audit Files

Each audit file is a self-contained collection of records; the file's name identifies the time span during which the records were generated and the machine that generated them.

Audit File Naming

Audit files that are complete have names of the following form:

start-time.finish-time.machine

where *start-time* is the time of the first audit record in the audit file, *finish-time* is the time of the last record, and *machine* is the name of the machine that generated the file. An example of these names can be found in "Example of a Closed Audit File Name" on page 26.

If the audit log file is still active, it has a name of the following form:

start-time.not_terminated.machine

How Audit File Names Are Used

The file name time stamps are used by `auditreduce` to locate files containing records for the specific time range that has been requested; this is important because there may be a month's supply or more of audit files on line, and searching them all for records generated in the last 24 hours would be unacceptably expensive.

Time-Stamp Format and Interpretation

The *start-time* and *end-time* are time stamps with one-second resolution; they are specified in Greenwich mean time. The format is four digits for the year, followed by two for each month, day, hour, minute, and second, as shown below.

```
YYYYMMDDHHMMSS
```

The time stamps are in GMT to ensure that they will sort in proper order even across a daylight savings time boundary. Because they are in GMT, the date and hour must be translated to the current time zone to be meaningful; beware of this whenever manipulating these files with standard file commands rather than with `auditreduce`.

Example of a File Name for a Still-Active File

The format of a file name of a still-active file is shown below:

```
YYYYMMDDHHMMSS.not_terminated.hostname
```

Here is an example:

```
19900327225243.not_terminated.lazy
```

The audit log files are named by the beginning date, so the example above was started in 1990, on March 27, at 10:52:43 p.m, GMT. The `not_terminated` in the file name means either that the file is still active or that `auditd` was unexpectedly interrupted. The name `lazy` at the end is the host name whose audit data is being collected.

Example of a Closed Audit File Name

The format of the name of a closed audit log file is shown below:

```
YYYYMMDDHHMMSS.YYYYMMDDHHMMSS.hostname
```

Here is an example:

```
19900320005243.19900327225351.lazy
```

The example above was started in 1990, on March 20, at 12:52:43 a.m., GMT. The file was closed March 27, at 10:53:51 p.m., GMT. The name `lazy` at the end is the host name of the machine whose audit data is being collected.

Whenever `auditd` is unexpectedly interrupted, the audit file open at the time gets the `not_terminated` end file name time stamp. Also, when a machine is writing to a remotely mounted audit file and the file server crashes or becomes inaccessible, the `not_terminated` end time stamp remains in the current file's name. The audit daemon opens a new audit file and keeps the old name intact.

Handling Nonactive Files Marked `not_terminated`

The `auditreduce` command processes files marked `not_terminated`, but because such files may contain incomplete records at the end, future processing may generate errors. To avoid errors, clean the files of any incomplete records. Before cleaning the files, make sure that `auditd` is not currently writing to the files you want to clean. To check, look at the `audit_data` file to determine the current process number of `auditd`. If that process is still running, and if the file name in `audit_data` is the same as the file in question, do not clean the file.

You can clean a file with the `-O` option of `auditreduce`. This creates a new file containing all the records that were in the old one, but with a proper file name time stamp. This operation loses the previous file pointer that's kept at the beginning of each audit file.

Or you can write a program to read through the file, locate the last record, rename the file, and clear out any incomplete records. A program can also keep the previous file pointer intact and determine which file to use next.

▼ How to Create Audit Partitions and Export Them

1. Assign at least one *primary audit directory* to each machine.

The primary audit directory is the directory where a machine places its audit files under normal conditions.

2. Assign at least one *secondary audit directory* to each machine that is located on a different audit file server than the primary directory.

The secondary audit directory is where a machine places audit files if the primary directory is full or inaccessible, because of network failure, NFS server crash, or some other reason.

3. On every diskfull machine create a local audit directory of last resort (preferably a dedicated audit file system) that is used when the network is inaccessible or the primary and secondary directories are unusable.
4. Spread the directories used as primary and secondary destinations evenly over the set of audit servers in the system.
5. Create audit file systems according to the requirements discussed in this section.

The `/etc/security` directory contains subdirectories with all the audit files and also contains several other files related to audit control. Because the `/etc/security` directory contains the per-machine `audit_data` file, which must be available for successful startup of the audit daemon at boot time, the `/etc/security` directory must be part of the root file system.

The audit post-selection tools look in directories under `/etc/security/audit` by default. For this reason, the path name of the mount point for the first audit file system on an audit server is in the form:

`/etc/security/audit/server-name` (where *server-name* is the name of the audit server). If more than one audit partition is on an audit server, the name of the second mount point is: `/etc/security/audit/server-name.1`, the third is `/etc/security/audit/server-name.2`, and so forth.

For example, the names of the audit file systems available on the audit server `winken` are `/etc/security/audit/winken` and `/etc/security/audit/winken.1`.

On the audit server, each audit file system must also have a subdirectory named `files`. This `files` subdirectory is where the audit files are located and where the `auditreduce` commands look for them. For example, the audit file system on audit server `winken` should have a `files` subdirectory whose full path name is: `/etc/security/audit/winken/files`.

You should make sure that the local `audit_control` file on each machine tells the audit daemon to put the audit files in the `files` subdirectory. Here is the `dir:` line for the `audit_control` file on a machine mounting the audit file system from `eagle`:

```
dir: /etc/security/audit/eagle/files
```

The extra level of hierarchy is required to prevent a machine's local root file system from filling with audit files when (for whatever reason) the `/etc/security/audit/server-name[.suffix]` directory is not available on the audit server. Because the `files` subdirectory is present on the audit server and there should be no `files` subdirectory on any of the clients, audit files cannot be created unintentionally in the local mount-point directory if the mount fails.

Make sure that each audit directory contains nothing except audit files.

6. Assign the required permissions to the audit file systems.

The permissions that must appear on the `/etc/security/audit/server-name` directory and the `files` directory that must be created beneath it on the audit server are shown in Table 2-5.

TABLE 2-5 Audit File Permissions

Owner	Group	Permissions
root	staff	2750

Example `audit_control` File Entries

When you add the `dir:` entries in the `audit_control` file, make sure the full path down to the `files` subdirectory is specified. The following example shows an `audit_control` file `dir:` entry for the server `blinken`, which will be storing its audit files on its own local disk.

```
# cat /etc/security/audit_control
dir:/etc/security/audit/blinken.1/files
dir:/etc/security/audit/blinken.2/files
```

▼ How to Configure Auditing

The following steps are included here to provide an overview of what is required to set up audit directories and specify which audit classes will be audited.

1. Format and partition the disks to create the dedicated audit partitions.

A rule of thumb is to assign 100 MBytes of space for each machine that will be on the distributed system; but remember that the disk space requirements at your site will be based on how much auditing you perform and may be far greater than this figure per machine.

2. Assign the audit file systems to the dedicated partitions.

Each diskfull machine should have a backup audit directory on the local machine in case its NFS-mounted audit file systems are not available.

3. While each machine is in single-user mode, run `tunefs -m 0` on each dedicated audit partition to reduce reserved file system space to 0 percent.

A reserved space percentage (called the `minfree` limit) is specified for audit partitions in the `audit_control` file. The default is 20 percent, and this percentage is tunable. Because this value is set by each site in the `audit_control` file, you should remove the automatically reserved file system space that is set aside by default for all file systems.

- 4. Set the required permissions on each of the audit directories on the audit servers, and make a subdirectory in each audit directory called `files`.**
Use `chown` and `chmod` to assign each audit directory and each `files` subdirectory the required permissions.
- 5. If using audit servers, export the audit directories with the `/etc/dfs/dfstab` file.**
- 6. Create the `audit_control` file entries for all the audit directories in the `audit_control` file on each machine, specifying the `files` subdirectory.**
- 7. On each audit client, create the entries for the audit file systems in the `/etc/vfstab` file.**
- 8. On each audit client, create the mount point directories and use `chmod` and `chown` to set the correct permissions.**

▼ How to Plan Audit Configuration

First, plan for audit trail storage.

- 1. In the `/etc/security/audit_class` file, define the classes needed at your site.**
If the default classes are suitable, you do not need to define new ones. See the `audit_class(4)` man page.
- 2. Set up event-to-class mapping in `/etc/security/audit_event`.**
This step is not needed if the default mapping suits your site's needs. See the `audit_event(4)` man page.
- 3. Determine how much auditing your site needs to do.**
Balance your site's security needs against the availability of disk space for audit trail storage.
See "Controlling Audit Costs" on page 21, "Auditing Efficiently" on page 23, and "Learning About the Audit Trail" on page 24 for guidance on how to reduce storage requirements while still maintaining site security as well as how to design audit storage.

4. **Determine which machines will be audit servers and which will be clients of the audit servers.**
5. **Determine the names and locations of audit file systems.**
6. **Plan which machines will use which audit file systems on the audit servers.**

After dealing with storage, decide who and what to audit.

1. **Determine which audit classes you want to be audited system-wide and which flags to use to select the audit classes.**

2. **Determine if some users will be audited more than others, then decide which flags to use to modify user's audit characteristics.**

See "Process Audit Characteristics" on page 14.

3. **Determine the minimum free space (`minfree`), also called the soft limit, that should be on an audit file system before a warning is sent.**

When the amount of space available goes below the `minfree` percentage, the audit daemon switches to the next *suitable* audit file system and send a notice that the soft limit has been exceeded. (What makes an audit file system suitable is defined in "What Makes a Directory Suitable" on page 17.)

A certain amount of auditing is configured by default on each machine. The default `audit_control` file contains the lines shown in Table 2-6, which set the audit directory as `/var/audit`, one system-wide audit flag (`lo`), a `minfree` threshold of 20 percent, and one nonattributable flag.

TABLE 2-6 `audit_control` File Entries

```
dir:/var/audit
flags:lo
minfree:20
naflags:ad
```

4. **Edit the `/etc/security/audit_control` file.**
 - a. **Specify which audit file systems to use for audit trail storage on this machine.**

Make a `dir:` entry for each audit directory available to the current machine. See "Learning About the Audit Trail" on page 24 for how to set up the audit directory scheme for the distributed system.

- b. **Specify the system-wide audit flags that will apply to all users' processes in the `flags:` field.**

The system-wide audit flags in the `flags:` field will apply to all users' processes, and you should set the flag the same on every machine.

- c. **Change the `minfree` percentage, if desired, to reduce or enlarge the audit threshold.**

- d. **Specify the `naflags:` that will apply to events that cannot be attributed to a particular user.**

- 5. **Use `auditconfig` to modify the audit policy if modification is desired.**

See the `auditconfig(1M)` man page or "The `auditconfig` Command" on page 34. The policy variable is a dynamic kernel variable, so its value is not saved when the system is brought down. Therefore, you should set the desired policy using the appropriate startup script.

- 6. **Set the `cnt` policy or set up an audit administration account.**

In the event of an audit trail overflow, either the `cnt` policy must be enabled, which allows further system functioning, or an account must be available that can work without being audited. To set up such an account:

- a. **In the `/etc/passwd` file, add the following entry.**

```
audit::0:1:::/sbin/sh
```

Note - This entry must be placed below the entry for the root user for processes owned by root to function properly.

- b. **To add a corresponding entry into the `/etc/shadow` file, type the following.**

```
# pwconv
pwconv: WARNING user audit has no password
```

The password for the audit account will be established in Step d.

- c. **In the `/etc/security/audit_user` file, add the following entry to turn off auditing for the account.**

```
audit:no:all
```

- d. Set a password for the new account using `passwd`.

```
# passwd audit
```

Remember that actions taken through this account are not audited. To protect system integrity, choose a password that is not easily compromised. This example uses an account name of `audit`. Choose a name more appropriate for your site if you set up such an account.

Preventing Audit Trail Overflow

If all audit file systems fill up, the `audit_warn` script sends a message to the console that the hard limit has been exceeded on all audit file systems and also sends mail to the alias. By default, the audit daemon remains in a loop sleeping and checking for space until some space is freed. All auditable actions are suspended.

A site's security requirements may be such that the loss of some audit data is preferable to having system activities suspended due to audit trail overflow. In that case, you can build automatic deletion or moving of audit files into the `audit_warn` script or set the `auditconfig` policy to drop records.

▼ How to Prevent Audit Trail Overflow

If your security policy requires that all audit data be saved, do the following:

1. **Set up a schedule to regularly archive audit files and to delete the archived audit files from the audit file system.**
2. **Manually archive audit files by backing them up on tape or moving them to an archive file system.**
3. **Store context-sensitive information that will be needed to interpret audit records along with the audit trail.**
4. **Keep records of what audit files are moved off line.**
5. **Store the archived tapes appropriately.**
6. **Reduce the volume of audit data you store by creating summary files.**

You can extract summary files from the audit trail using options to `auditreduce`, so that the summary files contain only records for certain specified types of audit events. An example of this is a summary file containing only the audit records for all logins and logouts. See Chapter 3.

The `auditconfig` Command

The `autoconfig` command provides a command line interface to get and set audit configuration parameters. See the `auditconfig(1M)` man page. Some of the options to `auditconfig` are:

`-chkconf`

Check the configuration of kernel audit event to class mappings and report any inconsistencies.

`-conf`

Reconfigure kernel event to class mappings at runtime to match the current mappings in the `audit_event` file.

`-getcond`

Get the machine-auditing condition. Table 2-7 shows the possible responses.

TABLE 2-7 Possible Auditing Conditions

Response	Meaning
<code>auditing</code>	Auditing is enabled and turned on.
<code>no audit</code>	Auditing is enabled but turned off.
<code>disabled</code>	The audit module is not enabled.

`-setcond condition`

Set the machine-auditing condition: `auditing` or `noaudit`.

`-getclass event_number`

Get the preselection classes to which the specified event is mapped.

`-setclass event_number audit_flags`

Set the preselection classes to which the specified event is mapped.

`-lsevent`

Display the currently configured (runtime) kernel and user audit event information.

`-getpinfo pid`

Get the audit ID, preselection mask, terminal ID, and audit session ID of the specified process.

`-setpmask pid flags`

Set the preselection mask of the specified process.

`-setsmask asid flags`

Set the preselection mask of all processes with the specified audit session ID.

`-setumask asid flags`

Set the preselection mask of all processes with the specified user audit ID.

`-lspolicy`

Display the list of audit policies with a short description of each one.

`-getpolicy`

Get the current audit policy flags.

`-setpolicy policy_flag[,policy_flag]`

Set the audit policy flags to the specified policies (see “Setting Audit Policies” on page 36).

Setting Audit Policies

You can use `auditconfig` with the `-setpolicy` flag to change the default Solaris-BSM audit policies. The `auditconfig` command with the `-lspolicy` argument shows the audit policies that you can change. The policy flags are described below.

`arge`

Record the environment and arguments on `execv` (see the `exec(2)` man page). The default is not to record these.

`argv`

Record command-line arguments to `execv`. The default is not to record these.

`cnt`

Do not suspend auditable actions when the queue is full; just count how many audit records are dropped. The default is suspend.

`group`

Include the supplementary groups token in audit records. The default is that group token is not included.

`path`

Add secondary `path` tokens to audit record. These secondary paths are typically the path names of dynamically linked shared libraries or command interpreters for shell scripts. By default they are not included.

`trail`

Include the `trailer` token in all records. The default is that the `trailer` token is not recorded.

`seq`

Include a sequence number in every audit record. The default is to not include. (The sequence number could be used to analyze a crash dump to find out whether any audit records are lost.)

▼ How to Change Which Events Are in Which Audit Classes

This procedure describes how to modify the default event to class mappings.

1. **Edit the `/etc/security/audit_event` file to change the class mapping for each event to be changed.**
2. **Reboot the system or run `auditconfig -conf` to change the runtime kernel event-to-class mappings.**

Changing Class Definitions

The file `/etc/security/audit_class` stores class definitions. Site-specific definitions can be added and default definitions can be changed. Each entry in the file has the form:

mask:name:description

Each class is represented as a bit in the mask, which is an unsigned integer, giving 32 different available classes plus two meta-classes of `all` and `no`; `all` is a conjunction of all allowed classes; `no` is the invalid class. Events mapped to this class are not audited. Events mapped solely to the `no` class are not audited, even if the `all` class is turned on. Below is a sample `audit_class` file:

```
0x00000000:no:invalid class
0x00000001:fr:file read
0x00000002:fw:file write
0x00000004:fa:file attribute access
0x00000008:fm:file attribute modify
0x00000010:fc:file create
0x00000020:fd:file delete
0x00000040:cl:file close
0xffffffff:all:all classes
```

If the `no` class is turned on in the system kernel, the audit trail is flooded with records for the audit event `AUE_NULL`.

Audit Trail Analysis

Using the tools described in this chapter, you can develop shell scripts to manage and report on the audit files and then run these scripts periodically. Audit management tasks might include compressing files, combining multiple audit files into one, moving files to different locations on disks in the distributed system, or archiving old files to tape. The scripts may also monitor storage usage, although the audit daemon does some of that automatically.

Another auditing task is to examine the audit trail, which is the logical combination of all the audit files. You can use the auditing tools to interactively query the audit data files for specific information.

- “Auditing Features” on page 39
- “Tools for Merging, Selecting, Viewing, and Interpreting Audit Records” on page 40
- “Audit Record Format” on page 41
- “Using the `auditreduce` Command” on page 50
- “Using `praudit`” on page 53

Auditing Features

The following features of Solaris BSM auditing are provided to interpret the audit records:

- The audit ID assigned to a user’s processes stays the same even when the user ID changes
- Each session has an audit session ID
- Full path names are saved in audit records

Because each audit record contains an audit ID that identifies the user who generated the event, and because full path names are recorded in audit records, you can look at individual audit records and get meaningful information without looking back through the audit trail.

Audit User ID

Solaris BSM processes have an additional user identification attribute not associated with processes in the standard Solaris release: the *audit ID*. A process acquires its audit ID at login time, and this audit ID is inherited by all child processes.

Audit Session ID

Solaris BSM processes have an audit session ID assigned at login time. The ID is inherited by all child processes.

Self-Contained Audit Records

The Solaris BSM audit records contain all the relevant information about an event and do not require you to refer to other audit records to interpret what occurred. For example, an audit record describing a file event contains the file's full path name starting at the root directory and a time and date stamp of the file's opening or closing.

Tools for Merging, Selecting, Viewing, and Interpreting Audit Records

Solaris BSM provides two tools that allow you to merge, select, view, and interpret audit records. The tools can be used directly or in conjunction with third-party application programs.

- The `auditreduce` command allows you to choose sets of records to examine. For instance, you can select all records from the past 24 hours to generate a daily report; you can select all records generated by a specific user to examine that user's activities; or you can select all records caused by a specific event type to see how often that type occurs.
- The `praudit` command allows you to display audit records interactively and create very basic reports. `praudit` displays records in one of several

human-readable but otherwise non-interpreted forms. You may accomplish more sophisticated display and reporting by postprocessing the output from `praudit` (with `sed` or `awk`, for instance) or by writing programs that interpret and process the binary audit records.

The following sections describe the audit record format, the `praudit`, and `auditreduce` commands in more detail, and provide some hints and procedures for using the tools.

Audit Record Format

A Solaris BSM *audit record* consists of a sequence of *audit tokens*, each of which describes an attribute of the system.

Appendix A gives a detailed description of each audit token. The appendix also lists all the audit records generated by Solaris BSM auditing. The definitions are sorted in order of the short descriptions, and a cross-reference table translates event names to event descriptions.

Binary Format

Audit records are stored and manipulated in binary form; however, the byte order and size of data is predetermined to simplify compatibility between different machines.

Audit Event Type

Each auditable event in the system generates a particular type of audit record. The audit record for each event has certain tokens within the record that describe the event. An audit record does not describe the audit event class to which the event belongs; that mapping is determined by an external table, the `/etc/security/audit_event` file.

Audit Token Types

Each token starts with a one-byte token type, followed by one or more data elements in an order determined by the type. The different audit records are distinguished by event type and different sets of tokens within the record. Some tokens, such as the `text` token, contain only a single data element, while others, such as the `process` token, contain several (including the audit user ID, real user ID, and effective user ID).

Order of Audit Tokens

Each audit record begins with a `header` token and ends (optionally) with a `trailer` token. One or more tokens between the header and trailer describe the event. For user-level and kernel events, the tokens describe the process that performed the event, the objects on which it was performed, and the objects' tokens, such as the owner or mode.

Each user-level and kernel event typically has at least the following tokens:

- `header`
- `subject`
- `return`

Many events also include a `trailer` token, but it is optional.

Human-Readable Audit Record Format

This section shows each audit record format as it appears in the output produced by the `praudit` command. This section also gives a short description of each audit token. For a complete description of each field in each token, see Appendix A.

The following token examples show the form that `praudit` produces by default. Examples are also provided of raw (`-r`) and short (`-s`) options. When `praudit` displays an audit token, it begins with the token type, followed by the data from the token. Each data field from the token is separated from other fields by a comma. However, if a field (such as a path name) contains a comma, this cannot be distinguished from a field-separating comma. Use a different field separator or the output will contain commas. The token type is displayed by default as a name, like `header`, or in `-r` format as a decimal number.

The individual tokens are described in the following order:

- “`header` Token” on page 43
- “`trailer` Token” on page 44
- “`arbitrary` Token” on page 44
- “`arg` Token” on page 44
- “`attr` Token” on page 45
- “`exit` Token” on page 45
- “`file` Token” on page 45
- “`groups` Token” on page 46
- “`in_addr` Token” on page 46
- “`ip` Token” on page 46
- “`ipc` Token” on page 46

- “ipc_perm Token” on page 47
- “iport Token” on page 47
- “opaque Token” on page 47
- “path Token” on page 47
- “process Token” on page 48
- “return Token” on page 48
- “seq Token” on page 49
- “socket Token” on page 49
- “subject Token” on page 49
- “text Token” on page 50

header Token

Every audit record begins with a header token. The header token gives information common to all audit records. The fields are:

- A token ID
- The record length in bytes, including the header and trailer tokens
- An audit record structure version number
- An event ID identifying the type of audit event
- An event ID modifier with descriptive information about the event type
- The time and date the record was created

When displayed by `praudit` in default format, a header token looks like the following example from `ioctl`:

```
header,240,1,ioctl(2),es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

Using `praudit -s`, the event description (`ioctl(2)` in the default `praudit` example above) is replaced with the event name (`AUE_IOCTL`), like this:

```
header,240,1,AUE_IOCTL,es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

Using `praudit -r`, all fields are displayed as numbers (that may be decimal, octal, or hex), where 158 is the event number for this event.

```
20,240,1,158,0003,699754304, + 270000 msec
```

Note that `praudit` displays the time to millisecond resolution.

trailer Token

This token marks the end of an audit record and allows backward seeks of the audit trail. The fields are:

- A token ID
- A pad number that marks the end of the record (does not show)
- The total number of audit record characters including the header and trailer tokens

A trailer token is displayed by `praudit` as follows:

```
trailer,136
```

arbitrary Token

This token encapsulates data for the audit trail. The item array may have a number of items. The fields are:

- A token ID
- A suggested format, such as decimal
- A size of encapsulated data, such as int
- A count of the data array items
- An item array

An arbitrary token is displayed by `praudit` as follows:

```
arbitrary,decimal,int,1  
42
```

arg Token

This token contains system call argument information. A 32-bit integer system call argument is allowed in an audit record. The fields are:

- A token ID
- An argument ID of the relevant system call argument
- The argument value
- The length of an optional descriptive text string (does not show)
- An optional text string

An `arg` token is displayed by `praudit` as follows:

```
argument,1,0x00000000,addr
```


attr Token

This token contains information from the file vnode. The `attr` token is usually produced during path searches and accompanies a `path` token, but is not included in the event of a path-search error. The fields are:

- A token ID
- The file access mode and type
- The owner user ID
- The owner group ID
- The file system ID
- The inode ID
- The device ID that the file might represent

An `attr` token is displayed by `praudit` as follows:

```
attribute,100555,root,staff,1805,13871,-4288
```

exit Token

An `exit` token records the exit status of a program. The fields are:

- A token ID
- A program exit status as passed to the `exit()` system call
- A return value that describes the exit status or indicates a system error number

An `exit` token is displayed by `praudit` as follows:

```
exit,Error 0,0
```

file Token

This token is generated by the audit daemon to mark the beginning of a new audit trail file and the end of an old file as the old file is deactivated. The audit record containing this token links together successive audit files into one audit trail. The fields are:

- A token ID
- A time and date stamp of a file opening or closing
- A byte count of the file name (does not show)
- The file name

A `file` token is displayed by `praudit` as follows:

```
file,Tue Sep 1 13:32:42 1992, + 79249 msec,  
/baudit/localhost/files/19920901202558.19920901203241.quisp
```

groups Token

A `groups` token records the `groups` entries from a process's credential. The fields are:

- A token ID
- An array of `groups` entries of size `NGROUPS_MAX` (16)

A `groups` token is displayed by `praudit` as follows:

```
group,staff,wheel,daemon,kmem,bin,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

in_addr Token

An `in_addr` token gives a machine Internet Protocol address. The fields are:

- A token ID
- An Internet address

An `in_addr` token is displayed by `praudit` as follows:

```
ip addr,129.150.113.7
```

ip Token

The `ip` token contains a copy of an Internet Protocol header. The fields are:

- A token ID
- A 20-byte copy of an IP header

An `ip` token is displayed by `praudit` as follows:

```
ip address,0.0.0.0
```

ipc Token

This token contains the System V IPC message/semaphore/shared-memory handle used by a caller to identify a particular IPC object. The fields are:

- A token ID
- An IPC object type identifier
- The IPC object handle

An `ipc` token is displayed by `praudit` as follows:

```
IPC,msg,3
```

ipc_perm Token

An `ipc_perm` token contains a copy of the System V IPC access information. Audit records for shared memory, semaphore, and message IPCs have this token added. The fields are:

- A token ID
- The IPC owner's user ID
- The IPC owner's group ID
- The IPC creator's user ID
- The IPC creator's group ID
- The IPC access modes
- The IPC sequence number
- The IPC key value

An `ipc_perm` token is displayed by `praudit` as follows:

```
IPC_perm,root,wheel,root,wheel,0,0,0x00000000
```

iport Token

This token contains a TCP (or UDP) address. The fields are:

- A token ID
- A TCP/UDP address

An `iport` token is displayed by `praudit` as follows:

```
ip_port,0xf6d6
```

opaque Token

The `opaque` token contains unformatted data as a sequence of bytes. The fields are:

- A token ID
- A byte count of the data array
- An array of byte data

An `opaque` token is displayed by `praudit` as follows:

```
opaque,12,0x4f5041515545204441544100
```

path Token

A `path` token contains access path information for an object. The fields are:

- A token ID
- A byte count of the path length (does not show)
- An absolute path

A path token is displayed by `praudit` as follows:

```
path,/an/anchored/path/name/to/test/auditwrite/AW_PATH
```

process Token

The `process` token contains information describing a process. The fields are:

- A token ID
- The user audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The session ID
- A terminal ID made up of:
 - A device ID
 - A machine ID

A `process` token is displayed by `praudit` as follows:

```
process,root,root,wheel,root,wheel,0,0,0,0.0.0.0
```

return Token

A `return` token gives the return status of the system call and the process return value. This token is always returned as part of kernel-generated audit records for system calls. The fields are:

- A token ID
- The system call error status
- The system call return value

A `return` token is displayed by `praudit` as follows:

```
return,success,0
```

seq Token

This token is optional and contains an increasing sequence number used for debugging. The token is added to each audit record when the `seq` policy is active. The fields are:

- A token ID
- A 32-bit unsigned long-sequence number

A `seq` token is displayed by `praudit` as follows:

```
sequence,1292
```

socket Token

A `socket` token describes an Internet socket. The fields are:

- A token ID
- A socket type field (TCP/UDP/UNIX)
- The local port address
- The local Internet address
- The remote port address
- The remote Internet address

A `socket` token is displayed by `praudit` as follows:

```
socket,0x0000,0x0000,0.0.0.0,0x0000,0.0.0.0
```

subject Token

This token describes a subject (process). The fields are:

- A token ID
- The user audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The session ID
- A terminal ID made up of:
 - A device ID
 - A machine ID

A subject token is displayed by `praudit` as follows:

```
subject,cjc,cjc,staff,cjc,staff,424,223,0 0 quisp
```

text Token

A text token contains a text string. The fields are:

- A token ID
- The length of the text string (does not show)
- A text string

A text token is displayed by `praudit` as follows:

```
text,aw_test_token
```

Using the `auditreduce` Command

The `auditreduce` command merges audit records from one or more input audit files. You would usually enter this command from the machine on which all the audit trail files for the entire distributed system are mounted.

Without options, `auditreduce` merges the entire audit trail (all of the audit files in all of the subdirectories in the `audit /etc/security/audit` directory) and sends the merged file to standard output.

The `praudit` command, described in “Using `praudit`” on page 53 makes the records human-readable.

These are some of the capabilities provided by options to the `auditreduce` command:

- Giving output containing audit records generated only by certain audit flags
- Showing audit records generated by one particular user
- Collecting audit records generated on specific dates

How `auditreduce` Helps in a Distributed System

When multiple machines running Solaris BSM are administered as part of a distributed system, each machine performs auditable events, and each machine writes audit records to its own machine-specific audit file. This simplifies software and is robust in the face of machine failures.

The `auditreduce` command makes the job of maintaining the whole audit trail practical. Using `auditreduce` (or shell scripts you write yourself to provide a higher-level interface), you can read the logical combination of all audit files in the system as a single audit trail, without regard to how the records were generated or where they are stored.

The `auditreduce` program operates on the audit records produced by the audit daemon. Records from one or more audit files are selected and merged into a single, chronologically ordered output file. The merging and selecting functions of `auditreduce` are logically independent. `auditreduce` selects messages from the input files as the records are read, before the files are merged and written to disk. Refer to the `auditreduce(1M)` man page.

Using `auditreduce`

This section describes a few common uses of `auditreduce` to analyze and manage data.

How to Display the Whole Audit Log

To display the whole audit trail at once, pipe the output of `auditreduce` into `praudit`.

```
#auditreduce | praudit
```

How to Print the Whole Audit Log

With a pipe to `lp`, the output goes to the printer.

```
# auditreduce | praudit | lp
```

How to Display User Activity on a Selected Data

In the following example, the system administrator checks to see when a user named `fred` logged in and logged out on April 13, 1990, by requesting the `lo` event class. The short-form date is in the form `yymmdd`. (The long form is described in the `auditreduce(1M)` man page.)

```
# auditreduce -d 900413 -u fred -c lo | praudit
```

How to Copy Login/Logout Messages to a Single File

In this example, login/logout messages for a particular day are summarized in a file. The target file is written in a directory other than the normal audit root.

```
# auditreduce -c lo -d 870413 -O /usr/audit_summary/logins
```

The `-O` option creates an audit file with 14-character timestamps for both start-time and end-time, and the suffix `logins`:

```
/usr/audit_summary/19870413000000.19870413235959.logins
```

How to Clean Up a `not_terminated` Audit File

Occasionally, if an audit daemon dies while its audit file is still open, or a server becomes inaccessible and forces the machine to switch to a new server, an audit file remains in which the end-time in the file name remains the string `not_terminated`, even though the file is no longer used for audit records. When such a file is found, you can manually verify that the file is no longer in use and clean it up by specifying the name of the file with the correct options.

```
# auditreduce -O machine 19870413120429.not_terminated.machine
```

This creates a new audit file with the correct name (both time stamps), the correct suffix (*machine*, explicitly specified), and copies all the messages into it.

Other Useful `auditreduce` Options

`auditreduce` has many additional options described in the man page. Notice that the uppercase options select operations or parameters for *files*, and the lowercase options select parameters for *records*. This subsection shows how to utilize two more useful options.

The *date-time* options `-b` and `-a` allow you to specify records before or after a particular day and time. A day begins at `yyyymmdd00:00:00` and ends at `yyyymmdd23:59:59`. The six parameters of a day are: year, month, day, hour, minute, and second. The digits (19) of the year are assumed and need not be specified.

If `-a` is not specified, `auditreduce` defaults to `00:00:00`, January 1, 1970. If `-b` is not specified, `auditreduce` defaults to the current time of day (GMT). The `-d` option selects a particular 24-hour period, as shown in “How to Copy Login/Logout Messages to a Single File” on page 51.

The `auditreduce -a` command with the date shown in the following screen example sends all audit records created after midnight on July 15, 1991, to `praudit`.

```
# auditreduce -a 91071500:00:00 | praudit
```

The `auditreduce -b` command with the same date shown above sends all audit records created before midnight on July 15, 1991 to `praudit`.

```
# auditreduce -b 91071500:00:00 | praudit
```


The message type selection for `auditreduce` (`-m` option) accepts either numeric message identifiers or `AUE_XXXX` codes. `auditreduce` rejects an incorrect format, but does not describe the correct format.

Using `praudit`

The `praudit` command reads audit records from standard input and displays them on standard output in human-readable form. Usually, the input is either piped from `auditreduce` or a single audit file. Input may also be produced with `cat` to concatenate several files or `tail` for a current audit file.

`praudit` can generate three output formats: default, short (`-s` option), and raw (`-r` option). By default, output is produced with one token per line. The `-l` option requests a whole record on each line. The `-d` option changes the delimiter used between token fields, and between tokens, if `-l` is also specified.

In `-s` format, the type is the audit event table name for the event (such as `AUE_IOCTL`), and in `-r` format, it is the event number (in this case, 158). That is the only distinction between `-s` and default format. In `-r` format, all numeric values (user IDs, group IDs, and so forth) are displayed numerically (in decimal, except for Internet addresses, which are in hex, and for modes, which are in octal). Here is the output from `praudit` for a header token:

```
header,240,1,ioctl(2),es,Tue Sept  1 16:11:44 1992, + 270000 msec
```

And here is the output from `praudit -r` for the same header token:

```
20,240,1,158,0003,699754304, + 270000 msec
```

It is sometimes useful to manipulate the output as lines of text; for example to perform selections that cannot be done with `auditreduce`. A simple shell script can process the output of `praudit`. The following example is called `praudit_grep`:

```
#!/bin/sh
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \
| tr '\012\001' '\002\012' \
| grep "$1" \
| tr '\002' '\012'
```

The example script marks the header tokens by prefixing them with Control-A. (Note that the `^a` is Control-a, not the two characters `^` and `a`. Prefixing is necessary to distinguish them from the string header that might appear as text.) The script then combines all the tokens for a record onto one line while preserving the line breaks as Control-a, runs `grep`, and restores the original new lines.

In the default output format of `praudit`, each record can always be identified unambiguously as a sequence of tokens (each on a separate line) beginning with a header token. Each record, therefore, is easily identified and processed with `awk`, for example.

Device Allocation

The Trusted Computer System Evaluation Criteria's (TCSEC) object-reuse requirement for computing systems at C2 level and above is fulfilled by the device-allocation mechanism. This chapter describes what you need to know about managing devices.

You must decide whether any devices should be allocatable, and if so, which devices should be allocatable, if the defaults are not appropriate for your site's security policy.

- "Risks Associated With Device Use" on page 55
- "Components of the Device-Allocation Mechanism" on page 56
- "Using the Device-Allocation Utilities" on page 57
- "The Allocate Error State" on page 58
- "The `device_maps` File" on page 58
- "The `device_allocate` File" on page 59
- "Device-Clean Scripts" on page 61
- "Setting Up Lock Files" on page 63
- "Managing and Adding Devices" on page 66
- "Using Device Allocations" on page 67

Risks Associated With Device Use

For one example of the security risks associated with the use of various I/O devices, consider how cartridge devices are typically used. Often several users share a single tape drive, which may be located in an office or lab away from where an individual user's own machine is located. This means that, after the user loads a tape into the tape drive, some length of time may elapse before the user can return to the machine

to invoke the command that reads or writes data to or from the tape. Then another time lapse occurs before the user is able to take the tape out of the drive. Because tape devices are typically accessible to all users, during the time when the tape is unattended an unauthorized user can access or overwrite data on the tape. The device-allocation mechanism makes it possible to assign certain devices to one user at a time, so that the device can only be accessed by that user while it is assigned to that user's name.

The device-allocation mechanism ensures the following for tape devices and provides related security services for other allocatable devices:

- Prevents simultaneous access to a device
- Prevents a user from reading a tape just written to by another user before the first user has removed the tape from the tape drive
- Prevents a user from gleaning any information from the device's or the driver's internal storage after another user is finished with the device

Components of the Device-Allocation Mechanism

The components of the allocation mechanism that you must understand in order to manage device allocation are:

- The `allocate`, `deallocate`, `dminfo`, and `list_devices` commands
- The `/etc/security/device_allocate` file (see the `device_allocate(4)` man page)
- The `/etc/security/device_maps` file (see the `device_maps(4)` man page)
- The lock files that must exist for each allocatable device in `/etc/security/dev`
- The changed attributes of the *device-special files* that are associated with each allocatable device
- Device-clean scripts for each allocatable device

How any user invokes the `allocate`, `deallocate`, `dminfo`, and `list_devices` commands is described in "Using the Device-Allocation Utilities" on page 57. All of the options and other descriptions are defined in the man pages.

The `device_allocate` file, the `device_map` file, and the lock files are specific to each machine. The configuration files are not administered as NIS databases because tape drives, diskette drives, and the printers are all connected to specific machines.

Using the Device-Allocation Utilities

This section describes what the administrator can do with the options to `allocate`, `deallocate`, and `list_devices` that are usable only by `root`. The commands are detailed on their respective man pages.

```
allocate -F device_special_filename
```

Reallocates the specified device. This option is often used with the `-U` option to reallocate the specified device to the specified user. Without the `-U` option, the device is allocated to `root`.

```
allocate -U username
```

Causes the device to be allocated to the user specified rather than to the current user. This option allows you to allocate a device for another user while you are `root`, without having to assume that user's identity.

```
deallocate -F device_special_filename
```

Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. When a user forgets to deallocate a tape drive, you can force deallocation using the `-F` option while you are `root`.

```
deallocate -I
```

Forces deallocation of all allocatable devices. This option should only be used at system initialization.

```
list_devices
```

Run `list_devices` to get a listing of all the device-special files that are associated with any device listed in the `device_maps` file.

```
list_devices -U username
```

List the devices that are allocatable or allocated to the user ID associated with the specified user name. This allows you to check which devices are allocatable or allocated to another user while you are `root`.

The Allocate Error State

The allocate error state is mentioned in the man pages for the allocate components. An allocatable device is in the *allocate error state* if it is owned by user `bin` and group `bin` with a device-special file mode of `0100`. If a user wants to allocate a device that is in the allocate error state, you should try to force the deallocation of the device, using the `deallocate` command with the `-F` option, or use `allocate -U` to assign it to the user, then investigate any error messages that display. When the problems with the device are corrected, you must rerun the `deallocate -F` or `allocate -F` commands to clear the allocate error state from the device.

The `device_maps` File

You can look at the `/etc/security/device_maps` file to determine the device names, device types, and device-special files that are associated with each allocatable device. See the `device_maps(4)` man page. Device maps are created by the system administrator when setting up device allocation. A rudimentary file is created by `bsmconv` when the BSM is enabled. This initial map file should be used only as a starting point. The system administrator is expected to augment and customize `device_maps` for the individual site.

This file defines the device-special file mappings for each device, which in many cases is not intuitive. This file allows various programs to discover which device-special files map to which devices. You can use the `dminfo` command, for example, to get the device name, the device type, and the device-special files to specify when setting up an allocatable device; `dminfo` uses the `device_maps` file.

Each device is represented by a one-line entry of the form:

device-name:device-type:device-list

Lines in the file can end with a `\` to continue an entry on the next line. Comments may also be included. A `#` makes a comment of all further text until the next newline not immediately preceded by a `\`. Leading and trailing blanks are allowed in any of the fields.

device-name

The name of the device, for example `st0`, `fd0`, or `audio`. The device name specified here must correspond to the name of the lockfile used in the `/etc/security/dev` directory.

device-type

The generic device type (the name for the class of devices, such as `st`, `fd`, `audio`). The `device-type` logically groups related devices.

device-list

A list of the device-special files associated with the physical device. The *device-list* must contain *all* of the special files that allow access to a particular device. If the list is incomplete, a malevolent user can still obtain or modify private information. Also, as in the example below, either the real device files located under `/devices` or the symbolic links in `/dev`, provided for binary compatibility, are valid entries for the *device-list* field.

For an example of entries for SCSI tape `st0` and diskette `fd0` in a `device_maps` file, see the following screen.

```
fd0:\
fd:\
/dev/fd0 /dev/fd0a /dev/fd0b /dev/rfd0 /dev/rfd0a /dev/rfd0b:\
.
.
.
st0:\
st:\
/dev/rst0 /dev/rst8 /dev/rst16 /dev/nrst0 /dev/nrst8 /dev/nrst16:\
```

The `device_allocate` File

Modify the `device_allocate` file to change devices from allocatable to non-allocatable or to add new devices. Table 4-1 shows a sample `device_allocate` file.

TABLE 4-1 Sample `device_allocate` File

```
st0;st;;;/etc/security/lib/st_clean
fd0;fd;;;/etc/security/lib/fd_clean
sr0;sr;;;/etc/security/lib/sr_clean
audio;audio;;;*/etc/security/lib/audio_clean
```

The administrator defines which devices should be allocatable during initial configuration of the Basic Security Module. You may decide to accept the default devices and their defined characteristics, as shown in Table 4-1. Whenever you add a device to any machine after the system is up and running, you must decide whether to make the new device allocatable.

The entries for devices in the `device_allocate` file may be modified by the administrator after installation. Any device that needs to be allocated before use must be defined in the `device_allocate` file on each machine. Currently, cartridge tape drives, diskette drives, CD-ROM devices, and audio chips are considered allocatable and have device-clean scripts.

Note - If you add a Xylogics tape drive or an Archive tape drive, they can also use the `st_clean` script supplied for SCSI devices. Other devices that you can make allocatable are modems, terminals, graphics tablets, and the like, but you need to create your own device-clean scripts for such devices, and the script must fulfill object-reuse requirements for that type of device.

An entry in the `device_allocate` file does not mean the device is allocatable, unless the entry specifically states the device is allocatable. Notice in Table 4-1 an asterisk (*) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable; that is, the system administrator does not require a user to allocate the device before it is used nor to deallocate it afterward. Any other string placed in this field indicates that the device is allocatable.

In the `device_allocate` file, represent each device by a one-line entry of the form

```
device-name; device-type; reserved; reserved; alloc; device-clean
```

For example, the following line shows the entry for device name `st0`:

```
st0;st;;;;;/etc/security/lib/st_clean
```

Lines in `device_allocate` can end with a `\` to continue an entry on the next line. Comments may also be included. A `#` makes a comment of all further text until the next newline not immediately preceded by a `\`. Leading and trailing blanks are allowed in any of the fields.

The following paragraphs describe each field in the `device_allocate` file in detail.

device-name

Specify the name of the device; for example, `st0`, `fd0`, or `sr0`. When making a new allocatable device, look up the *device-name* from the *device-name* field in the `device_maps` file or use the `dminfo` command. (The name is also the DAC file name for the device.)

device-type

Specify the generic device type (the name for the class of devices, such as `st`, `fd`, and `sr`). This field groups related devices. When making a new allocatable device, look up the *device-type* from the *device-type* field in the `device_maps` file or use the `dminfo` command.

reserved

These fields are reserved for future use.

alloc

Specify whether or not the device is allocatable. An asterisk (*) in this field indicates that the device is *not* allocatable. Any other string, or an empty field, indicates that the device is allocatable.

device-clean

Supply the path name of a program to be invoked for special handling, such as cleanup and object-reuse protection during the allocation process. The *device-clean* program is run any time the device is acted on by `deallocate`, such as when a device is forcibly deallocated with `deallocate -F`.

Device-Clean Scripts

The *device-clean* scripts address the security requirement that all usable data is purged from a physical device before reuse. By default, cartridge tape drives, diskette drives, CD-ROM devices, and audio devices require device-clean scripts, which are provided. This section describes what the device-clean scripts do.

Object Reuse

Device allocation satisfies part of the object-reuse requirement. The device-clean scripts make sure that data left on a device by one user is cleared before the device is allocatable by another user.

Device-Clean Script for Tapes

The three supported tape devices and the device-clean script for each are shown in Table 4-2.

TABLE 4-2 Device-Clean Script for the Three Supported Tape Devices

Tape Device Type	Device-Clean Script
SCSI 1/4-inch tape	st_clean
Archive 1/4-inch tape	st_clean
Open-reel 1/2-inch tape	st_clean

The script uses the `rewoffl` option to `mt` to affect the device cleanup. See the `mt(1)` man page. If the script runs during system boot, it queries the device to see if the device is online and has media in it.

The 1/4-inch tape devices that have media remaining are placed in the allocate error state to force the administrator to clean up the device manually.

During the normal system operation, when `allocate` or `deallocate` is executed in the interactive mode, the user is prompted to remove the media from the device being deallocated. The script pauses until the media is removed from the device.

Device-Clean Scripts for Diskettes and CD-ROM

The device-clean scripts for the diskettes and CD-ROM devices are shown in Table 4-3.

TABLE 4-3 Device-Clean Scripts for the Diskette and CD-ROM Device

Disk Device Type	Device-Clean Script
diskette	fd_clean
CD-ROM	sr_clean

The scripts use the `eject` command to remove the media from the drive. See the `eject(1)` man page. If `eject` fails, the device is placed in the allocate error state.

Device-Clean Script for Audio

The audio device is cleaned up with an audio-clean script. The script performs an `AUDIO_DRAIN ioctl` system call to flush the device, then an `AUDIO_SETINFO`

`ioctl` system call to reset the device configuration to default. In addition, the script retrieves the audio chip registers using the `AUDIOGETREG` `ioctl` system call. Any registers deviating from default are reset using the `AUDIOSETREG` `ioctl` system call.

Writing New Device-Clean Scripts

If you add more allocatable devices to the system, you might need to create your own device-clean scripts. The `deallocate` command passes a parameter to the device-clean scripts. The parameter, shown here, is a string that contains the device name (see the `device_allocate(4)` man page):

```
st_clean -[I|F|S] device-name
```

Device-clean scripts must return 0 for success and greater than 0 for failure. The options `-I`, `-F`, and `-S` help the script determine its running mode.

`-I` is needed during system boot only. All output must go to the system console. Failure or inability to forcibly eject the media must put the device in the `allocate` error state.

`-F` is for forced cleanup. This option is interactive and assumes that the user is there to respond to prompts. A script with this option must attempt to complete the cleanup if one part of the cleanup fails.

`-S` is for standard cleanup. This option is interactive and assumes that the user is there to respond to prompts.

Setting Up Lock Files

The lock files are zero-length files created in `/etc/security/dev` — one for each allocatable device.

If no lock file exists for an allocatable device, the device cannot be allocated, and no one can access the device.

▼ How to Set Up Lock Files for a Device to Be Made Allocatable

1. Use the `dminfo` command to get the device name for the device from its entry in the `device_maps` file.

See “The `device_maps` File” on page 58 and the `dminfo(1M)` and `device_maps(4)` man pages. For example, the device name for device type `st` is `st0`. Use the device name as the name of the lock file.

2. Use the `touch` command to create an empty lock file for the device, using the device name.

```
untouchable# cd /etc/security/dev
untouchable# touch device-name
untouchable# chmod 600 device-name
untouchable# chown bin device-name
untouchable# chgrp bin device-name
```

How the Allocate Mechanism Works

This section shows an example of how the allocate mechanism works.

The `allocate` command first checks for the presence of a lock file under the device name for the specified device in the `/etc/security/dev` directory. If the file is owned by `allocate`, then the ownership of the lock file is changed to the name of the user entering the `allocate` command.

The `allocate` command then checks for an entry for the device in the `device_allocate` file, and checks whether the entry shows the device as allocatable.

The first listing in the screen example below shows that a lock file exists with owner `bin`, group `bin`, and mode `600` for the `st0` device in `/etc/security/dev`. The second listing shows that the associated device-special files are set up properly, with owner `bin`, group `bin`, and mode `000`:

```
untouchable% ls -lg /etc/security/dev/st0
-rw----- 1 bin bin          0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -lg /devices/sbus@1,f8000000/esp@0,800000
c----- 1 bin bin          18,  4 May 12 13:11 st@4,0:
c----- 1 bin bin          18, 20 May 12 13:11 st@4,0:b
c----- 1 bin bin          18, 28 May 12 13:11 st@4,0:bn
c----- 1 bin bin          18, 12 May 12 13:11 st@4,0:c
.
.
.
c----- 1 bin bin          18,  0 May 12 13:11 st@4,0:u
c----- 1 bin bin          18, 16 May 12 13:11 st@4,0:ub
c----- 1 bin bin          18, 24 May 12 13:11 st@4,0:ubn
c----- 1 bin bin          18,  8 May 12 13:11 st@4,0:un
```

In this screen, user `vanessa` allocates device `st0`.

```
untouchable% whoami
vanessa
untouchable% allocate st0
```

When the user `vanessa` enters the `allocate` command to allocate the tape `st0`, `allocate` first checks for the existence of an `/etc/security/dev/st0` file. If no lock file existed or if the lock file were owned by a user other than `allocate`, then the device would not be allocatable by `vanessa`.

If it finds the lock file for the device with the correct ownership and permissions, the `allocate` command then checks to make sure the device has an entry in the `device_allocate` file and that the entry specifies that the device is allocatable.

In this example, the default `device_allocate` entry for the `st0` device specifies that the device is allocatable. Because the `allocate` command finds that all the above conditions are met, the device is allocated to `vanessa`.

The `allocate` command changes the ownership and permissions of the device-special files associated with the device in the `/dev` directory. To allocate the `st0` device to `vanessa`, the mode on its associated device-special files is changed to `600` and the owner is changed to `vanessa`.

The `allocate` command also changes the ownership of the lock file associated with the device in the `/etc/security/dev` directory. To allocate the `st0` device to `vanessa`, the owner of `/etc/security/dev/st0` is changed to `vanessa`.

After the user `vanessa` executes the `allocate` command using the device name `st0`, the following screen example shows that the owner of `/etc/security/dev` is changed to `vanessa` and that the owner of the associated device-special files is now `vanessa` as well, and that `vanessa` now has permission to read and write the files.

```
untouchable% whoami
vanessa
untouchable% allocate st0
untouchable% ls -lg /etc/security/dev/st0
-rw----- 1 vanessa staff          0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -la /devices/sbus@1,f8000000/esp@0,800000
.
.
.
crw----- 1 vanessa 18,  4 May 12 13:11 st@4,0:
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:b
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:bn
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:c
.
.
.
crw----- 1 vanessa 18,  4 May 12 13:11 st@4,0:u
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:ub
```

(continued)

```
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:ubn  
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:un
```

Managing and Adding Devices

The procedures in this section show how to manage devices and how to add devices.

▼ How to Manage Devices

1. **Determine which devices are listed in the `device_allocate` file and which devices can be made allocatable.**
2. **Define which devices, if any, should be made allocatable.**
3. **Decide which normal users, if any, should be allowed to allocate devices.**
4. **Edit the `device_allocate` file and add the new device.**

▼ How to Add a New Allocatable Device

1. **Create an entry for any new allocatable device on the machine in the `device_allocate` file.**
This procedure is described in “The `device_allocate` File” on page 59.
2. **Create an empty lock file for each allocatable device in the `/etc/security/dev` directory.**
This procedure is described in “Setting Up Lock Files” on page 63.
3. **Create a device-clean script if needed, for each new device.**
If you add a Xylogics or an Archive tape drive, you can use the `st_clean` script; otherwise, create your own. How to create a device-handling script is described in “Device-Clean Scripts” on page 61.

4. **Make all device-special files for the device to be owned by user `bin`, group `bin`, and mode `000`.**

You can run the `dminfo` command to get a listing from the `device_maps` file of all the device-special files that are associated with the device you are making allocatable.

Using Device Allocations

The procedures and commands in this section show how to manage devices and how to add devices. The device-allocation and device-deallocation commands are entered from the command line in a Command Tool or Shell Tool window:

- `allocate` assigns a device to a user.

You can specify the device in either of the two ways shown in Table 4-4.

TABLE 4-4 Device-Specification Options for `allocate`

Option	Action
<i>device-name</i>	Allocate the device that matches the device name.
<code>-g device-type</code>	Allocate the device that matches the device group type.

- `deallocate` releases a previously allocated device.
- `list_devices` allows you to see a list of all allocatable devices, devices currently allocated, and allocatable devices not currently allocated.

The `list_devices` command requires one of the three options shown in Table 4-5.

TABLE 4-5 Options for the `list_devices` Command

Option	Action
<code>-l</code>	List all allocatable devices or information about the device.
<code>-n</code>	List devices not currently allocated or information about the device.
<code>-u</code>	List devices currently allocated or information about the device.

▼ How to Allocate a Device

- ◆ Use the `allocate` command with a device specified by name, as in the example, or by type, with `-g` switch.

```
sar1% allocate st0
```

If the command cannot allocate the device, an error message displays in the console window. A list of all error messages appears in the `allocate(1M)` man page.

▼ How to Deallocate a Device

- ◆ Deallocate a tape drive by using the `deallocate` command followed by the device file name.

```
sar1% deallocate st0
```

Deallocation allows other users to allocate the device when you are finished.

Audit Record Descriptions

This appendix has two parts. The first part describes each component of an audit record structure and each audit token structure. The second part defines all of the audit records generated by the Basic Security Module by event description.

- “Audit Record Structure” on page 69
- “Audit Token Structure” on page 70
- “Kernel-Level Generated Audit Records” on page 84
- “User-Level Generated Audit Records” on page 156
- “Event-to-System Call Translation” on page 172

Audit Record Structure

An audit record is a sequence of audit tokens. Each token contains event information such as user ID, time, and date. A header token begins an audit record, and an optional trailer concludes the record. Other audit tokens contain audit-relevant information. Figure A-1 shows a typical audit record.

header token
arg token
data token
subject token
return token

Figure A-1 Typical Audit Record

Audit Token Structure

Logically, each token has a token type identifier followed by data specific to the token. Each token type has its own format and structure. The current tokens are shown in Table A-1. The token scheme can be extended.

TABLE A-1 Basic Security Module Audit Tokens

Token Name	Description
arbitrary	Data with format and type information
arg	System call argument value
attr	Vnode tokens
exec_args	Exec system call arguments
exec_env	Exec system call environment variables
exit	Program exit information
file	Audit file information
groups	Process groups information (obsolete)
header	Indicates start of record
in_addr	Internet address
ip	IP header information

TABLE A-1 Basic Security Module Audit Tokens *(continued)*

Token Name	Description
<code>ipc</code>	System V IPC information
<code>ipc_perm</code>	System V IPC object tokens
<code>ipport</code>	Internet port address
<code>newgroups</code>	Process groups information
<code>opaque</code>	Unstructured data (unspecified format)
<code>path</code>	Path information (path)
<code>process</code>	Process token information
<code>return</code>	Status of system call
<code>seq</code>	Sequence number token
<code>socket</code>	Socket type and addresses
<code>socket-inet</code>	Socket port and address
<code>subject</code>	Subject token information (same structure as <code>process</code> token)
<code>text</code>	ASCII string
<code>trailer</code>	Indicates end of record

An audit record always contains a `header` token. The `header` token indicates where the audit record begins in the audit trail. Every audit record contains a `subject` token, except for audit records from some nonattributable events. In the case of attributable events, these two tokens refer to the values of the process that caused the event. In the case of asynchronous events, the `process` tokens refer to the system.

arbitrary Token

The `arbitrary` token encapsulates data for the audit trail. It consists of four fixed fields and an array of data. The fixed fields are: a token ID that identifies this token as an `arbitrary` token, a suggested format field (for example, hexadecimal), a size

field that specifies the size of data encapsulated (for example, short), and a count field that gives the number of following items. The remainder of the token is composed of one or more items of the specified type. The arbitrary token appears as follows:

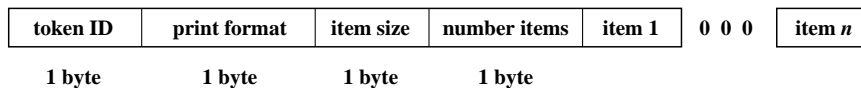


Figure A-2 arbitrary Token Format

The print format field can take the values shown in Table A-2.

TABLE A-2 arbitrary Token Print Format Field Values

Value	Action
AUP_BINARY	Print date in binary
AUP_OCTAL	Print date in octal
AUP_DECIMAL	Print date in decimal
AUP_HEX	Print date in hex
AUP_STRING	Print date as a string

The item size field can take the values shown in Table A-3.

TABLE A-3 arbitrary Token Item Size Field Values

Value	Action
AUR_BYTE	Data is in units of bytes (1 byte)
AUR_SHORT	Data is in units of shorts (2 bytes)
AUR_LONG	Data is in units of longs (4 bytes)

arg Token

The `arg` token contains system call argument information: the argument number of the system call, the argument value, and an optional descriptive text string. This token allows a 32-bit integer system-call argument in an audit record. The `arg` token has 5 fields: a token ID that identifies this token as an `arg` token, an argument ID that tells which system call argument the token refers to, the argument value, the length of a descriptive text string, and the text string. Figure A-3 shows the token form.

token ID	argument #	argument value	text length	text
1 byte	1 byte	4 bytes	2 bytes	<i>n</i> bytes

Figure A-3 `arg` Token Format

attr Token

The `attr` token contains information from the file `vnode`. This token has 7 fields: a token ID that identifies this as an `attr` token, the file access mode and type, the owner user ID, the owner group ID, the file system ID, the inode ID, and device ID the file might represent. See the `statvfs(2)` man page for further information about the file system ID and the device ID. This token usually accompanies a `path` token and is produced during path searches. In the event of a path-search error, this token is not included as part of the audit record since there is no `vnode` available to obtain the necessary file information. Figure A-4 shows the `attr` token format.

token ID	file mode	owner UID	owner GID	file system ID	file inode ID	device ID
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Figure A-4 `attr` Token Format

exec_args Token

The `exec_args` token records the arguments to an `exec` system call. The `exec_args` record has two fixed fields: a token ID field that identifies this as an `exec_args` token, and a count that represents the number of arguments passed to the `exec` call. The remainder of the token is composed of zero or more null-terminated strings. Figure A-5 shows an `exec_args` token.

token ID	count	env_args
----------	-------	----------

1 byte 4 bytes *count* null-terminated strings

Figure A-5 `exec_args` Token Format

Note - The `exec_args` token is output only when the audit policy `argv` is active. See “Setting Audit Policies” on page 36 for more information.

exec_env Token

The `exec_env` token records the current environment variables to an `exec` system call. The `exec_env` record has two fixed fields: a token ID field that identifies this as an `exec_env` token, and a count that represents the number of arguments passed to the `exec` call. The remainder of the token is composed of zero or more null-terminated strings. Figure A-6 shows an `exec_env` token.

token ID	count	env_args
----------	-------	----------

1 byte 4 bytes *count* null-terminated strings

Figure A-6 `exec_env` Token Format

Note - The `exec_env` token is output only when the audit policy `argv` is active. See “Setting Audit Policies” on page 36 for more information.

exit Token

The `exit` token records the exit status of a program. The `exit` token contains the exit status of the program and a return value. The status field is the same as that passed to the `exit` system call. The return value field indicates a system error number or a return value to further describe the exit status. Figure A-7 shows an `exit` token.

token ID	status	return value
----------	--------	--------------

1 byte 4 bytes 4 bytes

Figure A-7 `exit` Token Format

file Token

The `file` token is a special token generated by the audit daemon to mark the beginning of a new audit trail file and the end of an old file as it is deactivated. The audit daemon builds a special audit record containing this token to “link” together successive audit files into one audit trail. The `file` token has four fields: a token ID that identifies this token as a `file` token, a time and date stamp that identifies the time the file was created or closed, a byte count of the file name including a null terminator, and a field holding the file null-terminated name. Figure A-8 shows a `file` token.

token ID	date & time	name length	previous/next file name
1 byte	8 bytes	2 bytes	<i>n</i> bytes

Figure A-8 `file` Token Format

groups Token (Obsolete)

This token has been replaced by the `newgroups` token, which provides the same type of information but requires less space. A description of the `groups` token is provided here for completeness, but the application designer should use the `newgroups` token. Notice that `praudit` does not distinguish between the two tokens, as both token IDs are labelled `groups` when ASCII style output is displayed.

The `groups` token records the `groups` entries from the process’s credential. The `groups` token has two fixed fields: a token ID field that identifies this as a `groups` token, and a count that represents the number of `groups` contained in this audit record. The remainder of the token is composed of zero or more `group` entries. Figure A-9 shows a `groups` token.

token ID	groups
1 byte	<i>n groups</i> x 4 bytes

Figure A-9 `groups` Token Format

Note - The `groups` token is output only when the audit policy `group` is active. See “The `auditconfig` Command” on page 34 for more information.

header Token

The `header` token is special in that it marks the beginning of an audit record and combines with the `trailer` token to bracket all the other tokens in the record. The `header` token has six fields: a token ID field that identifies this as a `header` token, a byte count of the total length of the audit record including both `header` and `trailer`, a version number that identifies the version of the audit record structure, the audit event ID that identifies the type of audit event the record represents, an event ID modifier that contains ancillary descriptive information concerning the type of the event, and the time and date the record was created. Figure A-10 shows a `header` token.

token ID	byte count	version #	event ID	ID modifier	date and time
1 byte	4 bytes	1 byte	2 bytes	2 bytes	8 bytes

Figure A-10 `header` Token Format

The event modifier field has the following flags defined:

0x4000	PAD_NOTATTR	nonattributable event
0x8000	PAD_FAILURE	fail audit event

in_addr Token

The `in_addr` token contains an Internet address. This 4-byte value is an Internet Protocol address. The token has two fields: a token ID that identifies this token as an `in_addr` token and an Internet address. Figure A-11 shows an `in_addr` token.

token ID	Internet Address
1 byte	4 bytes

Figure A-11 `in_addr` Token Format

ip Token

The `ip` token contains a copy of an Internet Protocol header but does not include any IP options. The IP options may be added by including more of the IP header in the token. The token has two fields: a token ID that identifies this as an `ip` token and a copy of the IP header (all 20 bytes). The IP header structure is defined in `/usr/include/netinet/ip.h`. Figure A-12 shows an `ip` token.

token ID	IP header
-----------------	------------------

1 byte **20 bytes**

Figure A-12 `ip` Token Format

`ipc` Token

The `ipc` token contains the System V IPC message/semaphore/shared-memory handle used by the caller to identify a particular IPC object. This token has three fields: a token ID that identifies this as an `ipc` token, a type field that specifies the type of the IPC object, and the handle that identifies the IPC object. Figure A-13 shows an `ipc` token.

token ID	IPC object type	IPC object ID
-----------------	------------------------	----------------------

1 byte **1 byte** **4 bytes**

Figure A-13 `ipc` Token Format

Note - The IPC object identifiers violate the context-free nature of the Solaris CMW audit tokens. No global “name” uniquely identifies IPC objects; instead, they are identified by their handles, which are valid only during the time the IPC objects are active. The identification should not be a problem since the System V IPC mechanisms are seldom used and they all share the same audit class.

The IPC object type field may have the values shown in Table A-4. The values are defined in `/usr/include/bsm/audit.h`.

TABLE A-4 `IPC` Object Type Field

Name	Value	Description
<code>AU_IPC_MSG</code>	1	IPC message object
<code>AU_IPC_SEM</code>	2	IPC semaphore object
<code>AU_IPC_SHM</code>	3	IPC shared memory object

ipc_perm Token

The `ipc_perm` token contains a copy of the System V IPC access information. This token is added to audit records generated by shared memory, semaphore, and message IPC events. The token has eight fields: a token ID that identifies this token as an `ipc_perm` token, the user ID of the IPC owner, the group ID of the IPC owner, the user ID of the IPC creator, the group ID of the IPC creator, the access modes of the IPC, the sequence number of the IPC, and the IPC key value. The values are taken from the `ipc_perm` structure associated with the IPC object. Figure A-14 shows an `ipc_perm` token format.

token ID	owner uid	owner gid	creator uid	creator gid	ipc mode	sequence ID	IPC key
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Figure A-14 `ipc_perm` Token Format

iport Token

The `iport` token contains the TCP (or UDP) port address. The token has two fields: a token ID that identifies this as an `iport` token and the TCP/UDP port address. Figure A-15 shows an `iport` token.

token ID	port ID
1 byte	2 bytes

Figure A-15 `iport` Token Format

newgroups Token

This token is the replacement for the `groups` token. Notice that `praudit` does not distinguish between the two tokens, as both token IDs are labelled `groups` when ASCII output is displayed.

The `newgroups` token records the `groups` entries from the process's credential. The `newgroups` token has two fixed fields: a token ID field that identifies this as a `newgroups` token, and a count that represents the number of groups contained in this audit record. The remainder of the token is composed of zero or more group entries. Figure A-16 shows a `newgroups` token.

token ID	count	groups
1 byte	2 bytes	<i>count</i> * 4 bytes

Figure A-16 newgroups Token Format

Note - The `newgroups` token is output only when the audit policy `group` is active. See “The `auditconfig` Command” on page 34 for more information.

opaque Token

The `opaque` token contains unformatted data as a sequence of bytes. The token has three fields: a token ID that identifies this as an `opaque` token, a byte count of the amount of data, and an array of byte data. Figure A-17 shows an `opaque` token.

token ID	data length	data bytes
1 byte	2 bytes	<i>n</i> bytes

Figure A-17 opaque Token Format

path Token

The `path` token contains access path information for an object. The token contains a token ID and the absolute path to the object based on the real root of the system. The path has the following structure: a byte count of the path length and the path. Figure A-18 shows a `path` token.

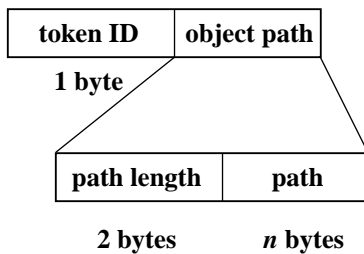


Figure A-18 path Token Format

process Token

The `process` token contains information describing a process as an object such as the recipient of a signal. The token has 9 fields: a token ID that identifies this token as a `process` token, the invariant audit ID, the effective user ID, the effective group ID, the real user ID, the real group ID, the process ID, the audit session ID, and a terminal ID. Figure A-19 shows a `process` token.

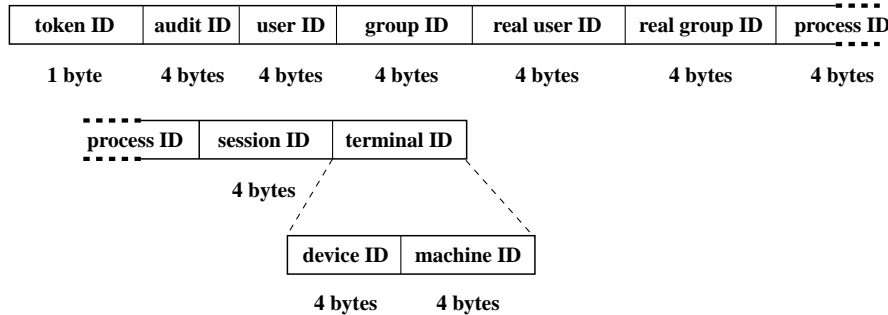


Figure A-19 `process` Token Format

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note - The `process` token fields for the session ID, the real user ID, or the real group ID may be unavailable. The entry is then set to -1.

return Token

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). The token has three fields: a token ID that identifies this token as a `return` token, the error status of the system call, and the system call return value. This token is always returned as part of kernel-generated audit records for system calls. The token indicates exit status and other return values in application auditing. Figure A-20 shows a `return` token.

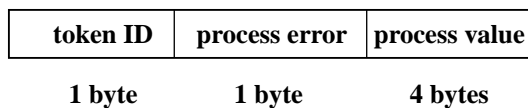


Figure A-20 `return` Token Format

seq Token

The `seq` token (sequence token) is an optional token that contains an increasing sequence number. This token is for debugging. The token is added to each audit record when the `AUDIT_SEQ` policy is active. The `seq` token has 2 fields: a token ID that identifies this token as a `seq` token, and a 32-bit unsigned long field that contains the sequence number. The sequence number is incremented every time an audit record is generated and put onto the audit trail. Figure A-21 shows a `seq` token.

token ID	sequence number
1 byte	4 bytes

Figure A-21 `seq` Token Format

socket Token

The `socket` token contains information describing an Internet socket. The `socket` token has 6 fields: a token ID that identifies this token as a `socket` token, a socket type field that indicates the type of socket referenced (TCP/UDP/UNIX), the local port address, the local Internet address, the remote port address, and the remote Internet address. Figure A-22 shows a `socket` token.

token ID	socket type	local port	local Internet address	remote port	remote Internet address
1 byte	2 bytes	2 bytes	4 bytes	2 bytes	4 bytes

Figure A-22 `socket` Token Format

socket-inet Token

The `socket-inet` token describes a socket connection to a local port, which is used to represent the socket information in the Internet namespace. The `socket-inet` token has 4 fields: a token ID that identifies this token as a `socket-inet` token, a socket family field that indicates the Internet family (`AF_INET`, `AF_OSI`, and so on), the address of the local port, and the address of the socket. Figure A-23 shows a `socket-inet` token.

token ID	socket family	local port	socket address
1 byte	2 bytes	2 bytes	4 bytes

Figure A-23 `socket-inet` Token Format

subject Token

The `subject` token describes a subject (process). The structure is the same as the `process` token. The token has 9 fields: an ID that identifies this as a `subject` token, the invariant audit ID, the effective user ID, the effective group ID, the real user ID, the real group ID, the process ID, the audit session ID, and a terminal ID. This token is always returned as part of kernel-generated audit records for system calls. Figure A-24 shows the token.

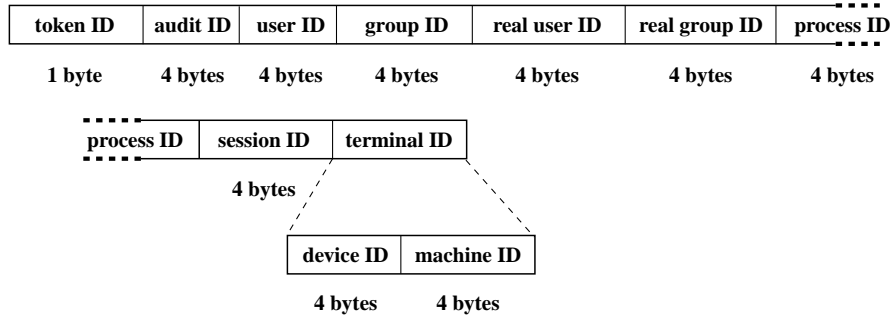


Figure A-24 subject Token Format

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note - The `subject` token fields for the session ID, the real user ID, or the real group ID may be unavailable. The entry is then set to -1.

text Token

The `text` token contains a text string. The token has three fields: a token ID that identifies this token as a `text` token, the length of the text string, and the text string itself. Figure A-25 shows a `text` token.

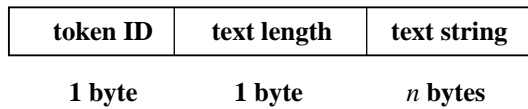


Figure A-25 text Token Format

trailer Token

The two tokens, `header` and `trailer`, are special in that they distinguish the endpoints of an audit record and bracket all the other tokens. A `header` token

begins an audit record. A trailer token ends an audit record. It is an optional token that is added as the last token of each record only when the `AUDIT_TRAIL` audit policy has been set.

The trailer token is special in that it marks the termination of an audit record. Together with the header token, the trailer token delimits an audit record. The trailer token supports backward seeks of the audit trail. The trailer token has three fields: a token ID that identifies this token as a trailer token, a pad number to aid in marking the end of the record, and the total number of characters in the audit record, including both the header and trailer tokens. Figure A-26 shows a trailer token.

token ID	pad number	byte count
1 byte	2 bytes	4 bytes

Figure A-26 trailer Token Format

The audit trail analysis software ensures that each record contains both header and trailer. In the case of a write error, as when a file system becomes full, an audit record can be incomplete and truncated. `auditsvc`, the system call responsible for writing data to the audit trail, attempts to put out complete audit records. See the `auditsvc(2)` man page. If file system space has run out, the call terminates without releasing the current audit record. When the call resumes, it can then repeat the truncated record.

Audit Records

This section presents all of the audit records. The audit records generated by kernel events are described first (see “Kernel-Level Generated Audit Records” on page 84). The audit records generated by user-level events are described next (see “User-Level Generated Audit Records” on page 156).

“Event-to-System Call Translation” on page 172 includes two tables that include all possible audit events and identifies which kernel or user event created the audit event. Table A-173 maps audit events to system calls. Table A-174 maps audit events to an application or command.

General Audit Record Structure

The audit records produced by Basic Security Module have a sequence of tokens. Certain tokens are optional within an audit record, according to the current audit policy. The `group`, `sequence`, and `trailer` tokens fall into this category. The

administrator can determine if these are included in an audit record with the `auditconfig` command `-getpolicy` option.

Kernel-Level Generated Audit Records

These audit records are created by system calls that are used by the kernel. The records are sorted alphabetically by system call. The description of each record includes:

- The name of the system call
- A man page reference (if appropriate)
- The audit event number
- The audit event name
- The audit event class
- The mask for the event class
- The audit record structure

TABLE A-5 `access(2)`

Event Name	Event ID	Event Class	Mask
AUE_ACCESS	14	fa	0x00000004

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-6 acct(2)

Event Name	Event ID	Event Class	Mask
AUE_ACCT	18	ad	0x00000800

Format (zero path):

header-token

argument-token (1, "accounting off", 0)

subject-token

return-token

Format (non-zero path):

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-7 adjtime(2)

Event Name	Event ID	Event Class	Mask
AUE_ADJTIME	50	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-8 audit(2)

Event Name	Event ID	Event Class	Mask
AUE_AUDIT	211	no	0x00000000

Format:

header-token

subject-token

return-token

TABLE A-9 auditon(2) - get car

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETCAR	224	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-10 auditon(2) - get event class

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETCLASS	231	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-11 auditon(2) - get audit state

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETCOND	229	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-12 `auditon(2)` - `get cwd`

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETCWD	223	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-13 `auditon(2)` - `get kernal mask`

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETKMASK	221	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-14 auditon(2) - get audit statistics

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GETSTAT	225	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-15 auditon(2) - GPOLICY command

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GPOLICY	114	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-16 `auditon(2)` - `GQCTRL` command

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_GQCTRL	145	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-17 `auditon(2)` - `set event class`

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETCLASS	232	ad	0x00000800

Format:

header-token

[*argument-token*] (2, "setclass:ec_event", event number)

[*argument-token*] (3, "setclass:ec_class", class mask)

subject-token

return-token

TABLE A-18 auditon(2) - set audit state

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETCOND	230	ad	0x00000800

Format:

header-token

[*argument-token*] (3, "setcond", audit state)

subject-token

return-token

TABLE A-19 auditon(2) - set kernel mask

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETKMASK	222	ad	0x00000800

Format:

header-token

[*argument-token*] (2, "setkmask:as_success", kernel mask)

[*argument-token*] (2, "setkmask:as_failure", kernel mask)

return-token

TABLE A-20 auditon(2) - set mask per session ID

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETSMASK	228	ad	0x00000800

Format:

header-token

[*argument-token*] (3, "setsmask:as_success", session ID mask)

[*argument-token*] (3, "setsmask:as_failure", session ID mask)

subject-token

return-token

TABLE A-21 auditon(2) - reset audit statistics

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETSTAT	226	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-22 auditon(2) - set mask per uid

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SETUMASK	227	ad	0x00000800

Format:

header-token

[*argument-token*] (3, "setumask:as_success", audit ID mask)

[*argument-token*] (3, "setumask:as_failure", audit ID mask)

subject-token

return-token

TABLE A-23 auditon(2) - SPOLICY command

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SPOLICY	147	ad	0x00000800

Format:

header-token

[*argument-token*] (1, "policy", audit policy flags)

subject-token

return-token

TABLE A-24 auditon(2) - SQCTRL command

Event Name	Event ID	Event Class	Mask
AUE_AUDITON_SQCTRL	146	ad	0x00000800

Format:

header-token

[*argument-token*] (3,"setqctrl:aq_hiwater",queue control param.)

[*argument-token*] (3,"setqctrl:aq_lowater",queue control param.)

[*argument-token*] (3,"setqctrl:aq_bufsz",queue control param.)

[*argument-token*] (3,"setqctrl:aq_delay",queue control param.)

subject-token

return-token

TABLE A-25 auditsvc(2)

Event Name	Event ID	Event Class	Mask
AUE_AUDITSVC	136	ad	0x00000800

Format (valid file descriptor):

header-token

[*path-token*]

[*attr-token*]

subject-token

return-token

Format (not valid file descriptor):

header-token

argument-token (1, "no path: fd", fd)

subject-token

return-token

TABLE A-26 chdir(2)

Event Name	Event ID	Event Class	Mask
AUE_CHDIR	8	pc	0x00000080

Format:
header-token
path-token
[*attr-token*]
subject-token
return-token

TABLE A-27 chmod(2)

Event Name	Event ID	Event Class	Mask
AUE_CHMOD	10	fm	0x00000008

Format:
header-token
argument-token (2, "new file mode", mode)
path-token
[*attr-token*]
subject-token
return-token

TABLE A-28 chown(2)

Event Name	Event ID	Event Class	Mask
AUE_CHOWN	11	fm	0x00000008

Format:

header-token

argument-token (2, "new file uid", uid)

argument-token (3, "new file gid", gid)

path-token

[*attr-token*]

subject-token

return-token

TABLE A-29 chroot(2)

Event Name	Event ID	Event Class	Mask
AUE_CHROOT	24	pc	0x00000080

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-30 close(2)

Event Name	Event ID	Event Class	Mask
AUE_CLOSE	112	cl	0x00000040

Format:

<file system object>

header-token

argument-token (1, "fd", file descriptor)

[*path-token*]

[*attr-token*]

subject-token

return-token

TABLE A-31 creat(2)

Event Name	Event ID	Event Class	Mask
AUE_CREAT	4	fc	0x00000010

Format

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-32 enter prom

Event Name	Event ID	Event Class	Mask
AUE_ENTERPROM	153	na	0x00000400

Format:

header-token

text-token (addr, "monitor PROM" | "kadb")

subject-token

return-token

TABLE A-33 exec(2)

Event Name	Event ID	Event Class	Mask
AUE_EXEC	7	pc,ex	0x40000080

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-34 `execve(2)`

Event Name	Event ID	Event Class	Mask
AUE_EXECVE	23	pc,ex	0x40000080

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-35 `exit prom`

Event Name	Event ID	Event Class	Mask
AUE_EXITPROM	154	na	0x00000400

Format:
header-token
text-token (addr, "monitor PROM"|"kadb")
subject-token
return-token

TABLE A-36 exit(2)

Event Name	Event ID	Event Class	Mask
AUE_EXIT	1	pc	0x00000080

Format:

header-token

subject-token

return-token

TABLE A-37 fchdir(2)

Event Name	Event ID	Event Class	Mask
AUE_FCHDIR	68	pc	0x00000080

Format:

header-token

[*path-token*]

[*attr-token*]

subject-token

return-token

TABLE A-38 fchmod(2)

Event Name	Event ID	Event Class	Mask
AUE_FCHMOD	39	fm	0x00000008

Format (valid file descriptor):

header-token

argument-token (2, "new file mode", mode)

[*path-token*]

[*attr-token*]

subject-token

return-token

Format (not valid file descriptor):

header-token

argument-token (2, "new file mode", mode)

argument-token (1, "no path: fd", fd)

subject-token

return-token

TABLE A-39 fchown(2)

Event Name	Event ID	Event Class	Mask
AUE_FCHOWN	38	fm	0x00000008

Format (valid file descriptor):

header-token (2, "new file uid", uid)
argument-token (3, "new file gid", gid)
 [*path-token*]
 [*attr-token*]
subject-token
return-token

Format (non-file descriptor):

header-token
argument-token (2, "new file uid", uid)
argument-token (3, "new file gid", gid)
argument-token (1, "no path: fd", fd)
subject-token
return-token

TABLE A-40 fchroot(2)

Event Name	Event ID	Event Class	Mask
AUE_FCHROOT	69	pc	0x00000080

Format:

header-token
 [*path-token*]
 [*attr-token*]
subject-token
return-token

TABLE A-41 fcntl(2)

Event Name	Event ID	Event Class	Mask
AUE_FCNTL (cmd=F_GETLK, F_SETLK, F_SETLKW)	30	fm	0x00000008

Format (file descriptor):

header-token

argument-token (2, "cmd", cmd)

path-token

attr-token

subject-token

return-token

Format (bad file descriptor):

header-token

argument-token (2, "cmd", cmd)

argument-token (1, "no path: fd", fd)

subject-token

return-token

TABLE A-42 fork(2)

Event Name	Event ID	Event Class	Mask
AUE_FORK	2	pc	0x00000080

Format:

header-token

[*argument-token*] (0, "child PID", pid)

subject-token

return-token

The `fork()` return values are undefined because the audit record is produced at the point that the child process is spawned.

TABLE A-43 fork1(2)

Event Name	Event ID	Event Class	Mask
AUE_FORK1	241	pc	0x00000080

Format:

header-token

[*argument-token*] (0, "child PID", pid)

subject-token

return-token

The `fork1()` return values are undefined because the audit record is produced at the point that the child process is spawned.

TABLE A-44 fstatfs(2)

Event Name	Event ID	Event Class	Mask
AUE_FSTATFS	55	fa	0x00000004

Format (file descriptor):

header-token
[path-token]
[attr-token]
subject-token
return-token

Format (non-file descriptor):

header-token
argument-token (1, "no path: fd", fd)
subject-token
return-token

TABLE A-45 getaudit(2)

Event Name	Event ID	Event Class	Mask
AUE_GETAUDIT	132	ad	0x00000800

Format:

header-token
subject-token
return-token

TABLE A-46 getauid(2)

Event Name	Event ID	Event Class	Mask
AUE_GETAUID	130	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-47 getmsg(2)

Event Name	Event ID	Event Class	Mask
AUE_GETMSG	217	nt	0x00000100

Format:

header-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

TABLE A-48 getmsg - accept

Event Name	Event ID	Event Class	Mask
AUE_SOCKACCEPT	247	nt	0x00000100

Format:*header-token**socket-inet-token**argument-token* (1, "fd", file descriptor)*argument-token* (4, "pri", priority)*subject-token**return-token***TABLE A-49** getmsg - receive

Event Name	Event ID	Event Class	Mask
AUE_SOCKRECEIVE	250	nt	0x00000100

Format:*header-token**socket-inet-token**argument-token* (1, "fd", file descriptor)*argument-token* (4, "pri", priority)*subject-token**return-token*

TABLE A-50 getpmsg(2)

Event Name	Event ID	Event Class	Mask
AUE_GETPMSG	219	nt	0x00000100

Format:

header-token

argument-token (1, "fd", file descriptor)

subject-token

return-token

TABLE A-51 getportaudit(2)

Event Name	Event ID	Event Class	Mask
AUE_GETPORTAUDIT	149	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-52 `ioctl(2)`

Event Name	Event ID	Event Class	Mask
AUE_IOCTL	158	io	0x20000000

Format (good file descriptor):

header-token

path-token

[*attr-token*]

argument-token (2, "cmd" ioctl cmd)

argument-token (3, "arg" ioctl arg)

subject-token

return-token

Format (socket):

header-token

[*socket-token*]

argument-token (2, "cmd" ioctl cmd)

argument-token (3, "arg" ioctl arg)

subject-token

return-token

Format (non-file file descriptor):

header-token

argument-token (1, "fd", file descriptor)

argument-token (2, "cmd", ioctl cmd)

argument-token (3, "arg", ioctl arg)

subject-token

return-token

Format (bad file name):

header-token

argument-token (1, "no path: fd", fd)

argument-token (2, "cmd", ioctl cmd)

argument-token (3, "arg", ioctl arg)

subject-token

return-token

TABLE A-53 kill(2)

Event Name	Event ID	Event Class	Mask
AUE_KILL	15	pc	0x00000080

Format (valid process):

header-token

argument-token (2, "signal", signo)

[*process-token*]

subject-token

return-token

Format (zero or negative process):

header-token

argument-token (2, "signal", signo)

argument-token (1, "process", pid)

subject-token

return-token

TABLE A-54 lchown(2)

Event Name	Event ID	Event Class	Mask
AUE_LCHOWN	237	fm	0x00000008

Format:

header-token

argument-token (2, "new file uid", uid)

argument-token (3, "new file gid", gid)

path-token

[*attr-token*]

subject-token

return-token

TABLE A-55 link(2)

Event Name	Event ID	Event Class	Mask
AUE_LINK	5	fc	0x00000010

Format:

header-token

path-token (from path)

[*attr-token*] (from path)

path-token (to path)

subject-token

return-token

TABLE A-56 lstat(2)

Event Name	Event ID	Event Class	Mask
AUE_LSTAT	17	fa	0x00000004

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-57 lxstat(2)

Event Name	Event ID	Event Class	Mask
AUE_LXSTAT	236	fa	0x00000004

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-58 memcntl(2)

Event Name	Event ID	Event Class	Mask
AUE_MEMCNTL	238	ot	0x80000000

Format:

header-token

argument-token (1, "base", base address)

argument-token (2, "len", length)

argument-token (3, "cmd", command)

argument-token (4, "arg", command args)

argument-token (5, "attr", command attributes)

argument-token (6, "mask", 0)

subject-token

return-token

TABLE A-59 mkdir(2)

Event Name	Event ID	Event Class	Mask
AUE_MKDIR	47	fc	0x00000010

Format:

header-token

argument-token (2, "mode", mode)

path-token

[*attr-token*]

subject-token

return-token

TABLE A-60 mknod(2)

Event Name	Event ID	Event Class	Mask
AUE_MKNOD	9	fc	0x00000010

Format:

header-token

argument-token (2, "mode", mode)

argument-token (3, "dev", dev)

path-token

[*attr-token*]

subject-token

return-token

TABLE A-61 mmap (2)

Event Name	Event ID	Event Class	Mask
AUE_MMAP	210	no	0x00000000

Format (valid file descriptor):

header-token

argument-token (1, "addr", segment address)

argument-token (2, "len", segment length)

[*path-token*]

[*attr-token*]

subject-token

return-token

Format (not valid file descriptor):

header-token

argument-token (1, "addr", segment address)

argument-token (2, "len", segment length)

argument-token (1, "no path: fd", fd)

subject-token

return-token

TABLE A-62 modctl(2) - bind module

Event Name	Event ID	Event Class	Mask
AUE_MODADDMAJ	246	ad	0x00000800

Format:

header-token

[*text-token*] driver major number)

[*text-token*] (driver name)

text-token (root dir. | "no rootdir")

text-token (driver major number | "no drvname")

argument-token (5, "", number of aliases)

(0..n)[*text-token*] (aliases)

subject-token

return-token

TABLE A-63 modctl(2) - configure module

Event Name	Event ID	Event Class	Mask
AUE_MODCONFIG	245	ad	0x00000800

Format:

header-token

text-token (root dir. | "no rootdir")

text-token (driver major number | "no drvname")

subject-token

return-token

TABLE A-64 modctl(2) - load module

Event Name	Event ID	Event Class	Mask
AUE_MODLOAD	243	ad	0x00000800

Format:

header-token

[*text-token*] (default path)

text-token (filename path)

subject-token

return-token

TABLE A-65 modctl(2) - unload module

Event Name	Event ID	Event Class	Mask
AUE_MODUNLOAD	244	ad	0x00000800

Format:

header-token

argument-token (1, "id", module ID)

subject-token

return-token

TABLE A-66 mount(2)

Event Name	Event ID	Event Class	Mask
AUE_MOUNT	62	ad	0x00000800

Format (UNIX file system):

header-token

argument-token (3, "flags", flags)

text-token (filesystem type)

path-token

[*attr-token*]

subject-token

return-token

Format (NFS file system):

header-token

argument-token (3, "flags", flags)

text-token (filesystem type)

text-token (host name)

argument-token (3, "internal flags", flags)

TABLE A-67 msgctl(2) - IPC_RMID command

Event Name	Event ID	Event Class	Mask
AUE_MSGCTL_RMID	85	ip	0x00000200

Format:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the msg ID is not valid.

TABLE A-68 msgctl(2) - IPC_SET command

Event Name	Event ID	Event Class	Mask
AUE_MSGCTL_SET	86	ip	0x00000200

Format:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the msg ID is not valid.

TABLE A-69 msgctl(2) - IPC_STAT command

Event Name	Event ID	Event Class	Mask
AUE_MSGCTL_STAT	87	ip	0x00000200

Format:

header-token

argument-token (1, "msg ID", message ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the msg ID is not valid.

TABLE A-70 msgget(2)

Event Name	Event ID	Event Class	Mask
AUE_MSGGET	88	ip	0x00000200

Format:

header-token

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the msg ID is not valid.

TABLE A-71 msgrcv(2)

Event Name	Event ID	Event Class	Mask
AUE_MSGRCV	89	ip	0x00000200

Format:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the msg ID is not valid.

TABLE A-72 msgsnd(2)

Event Name	Event ID	Event Class	Mask
AUE_MSGSND	90	ip	0x00000200

Format:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the msg ID is not valid.

TABLE A-73 munmap(2)

Event Name	Event ID	Event Class	Mask
AUE_MUNMAP	214	cl	0x00000040

Format:

header-token

argument-token (1, "addr", address of memory)

argument-token (2, "len", memory segment size)

subject-token

return-token

TABLE A-74 old nice(2)

Event Name	Event ID	Event Class	Mask
AUE_NICE	203	pc	0x00000080

Format:

header-token

subject-token

return-token

TABLE A-75 open(2) - read

Event Name	Event ID	Event Class	Mask
AUE_OPEN_R	72	fr	0x00000001

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-76 open(2) - read,creat

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RC	73	fc,fr	0x00000011

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-77 open(2) - read,creat,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RTC	75	fc,fd,fr	0x00000031

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-78 open(2) - read,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RT	74	fd,fr	0x00000021

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-79 open(2) - read,write

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RW	80	fr, fw	0x00000003

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-80 open(2) - read,write,creat

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RWC	81	fr, fw, fc	0x00000013

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-81 open(2) - read,write,create,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RWTC	83	fr, fw, fc, fd	0x00000033

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-82 open(2) - read,write,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RWT	82	fr, fw, fd	0x00000023

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-83 open(2) - write

Event Name	Event ID	Event Class	Mask
AUE_OPEN_W	76	fw	0x00000002

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-84 open(2) - write,creat

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WC	77	fw,fc	0x00000012

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-85 open(2) - write,creat,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WTC	79	fw,fc,fd	0x00000032

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-86 open(2) - write,trunc

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WT	78	fw,fd	0x00000022

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-87 pathconf(2)

Event Name	Event ID	Event Class	Mask
AUE_PATHCONF	71	fa	0x00000004

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-88 pipe(2)

Event Name	Event ID	Event Class	Mask
AUE_PIPE	185	no	0x00000000

Format:
header-token
subject-token
return-token

TABLE A-89 priocntlsys(2)

Event Name	Event ID	Event Class	Mask
AUE_PRIOCNTLSYS	212	pc	0x0000080

Format:

header-token

argument-token (1, "pc_version", priocntl version num.)

argument-token (3, "cmd", command)

subject-token

return-token

TABLE A-90 process dumped core

Event Name	Event ID	Event Class	Mask
AUE_CORE	111	fc	0x0000010

Format:

header-token

path-token

[*attr-token*]

argument-token (1, "signal", signal)

subject-token

return-token

TABLE A-91 putmsg(2)

Event Name	Event ID	Event Class	Mask
AUE_PUTMSG	216	nt	0x00000100

Format:

header-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

TABLE A-92 putmsg-connect

Event Name	Event ID	Event Class	Mask
AUE_SOCKCONNECT	248	nt	0x00000100

Format:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

TABLE A-93 putmsg-send

Event Name	Event ID	EventClass	Mask
AUE SOCKSEND	249	nt	0x00000100

Format:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

TABLE A-94 putpmsg(2)

Event Name	Event ID	Event Class	Mask
AUE_PUTPMSG	218	nt	0x00000100

Format:

header-token

argument-token (1, "fd", file descriptor)

subject-token

return-token

TABLE A-95 readlink(2)

Event Name	Event ID	Event Class	Mask
AUE_READLINK	22	fr	0x00000001

Format:
header-token
path-token
[*attr-token*]
subject-token
return-token

TABLE A-96 rename(2)

Event Name	Event ID	Event Class	Mask
AUE_RENAME	42	fc,fd	0x00000030

Format:
header-token
path-token (from name)
[*attr-token*] (from name)
[*path-token*] (to name)
subject-token
return-token

TABLE A-97 rmdir(2)

Event Name	Event ID	Event Class	Mask
AUE_RMDIR	48	fd	0x00000020

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-98 semctl(2) - getall

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_GETALL	105	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is not valid.

TABLE A-99 semctl(2) - GETNCNT command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_GETNCNT	102	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is not valid.

TABLE A-100 semctl(2) - GETPID command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_GETPID	103	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is not valid.

TABLE A-101 semctl(2) - GETVAL command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_GETVAL	104	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the semaphore ID is not valid.

TABLE A-102 semctl(2) - GETZCNT command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_GETZCNT	106	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the semaphore ID is not valid.

TABLE A-103 semctl(2) - IPC_RMID command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_RMID	99	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is not valid.

TABLE A-104 semctl(2) - IPC_SET command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_SET	100	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is not valid.

TABLE A-105 semctl(2) - SETALL command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_SETALL	108	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the semaphore ID is not valid.

TABLE A-106 semctl(2) - SETVAL command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_SETVAL	107	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the semaphore ID is not valid.

TABLE A-107 semctl(2) - IPC_STAT command

Event Name	Event ID	Event Class	Mask
AUE_SEMCTL_STAT	101	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

TABLE A-108 semget(2)

Event Name	Event ID	Event Class	Mask
AUE_SEMGET	109	ip	0x00000200

Format:

header-token

[*ipc-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the system call failed.

TABLE A-109 semop(2)

Event Name	Event ID	Event Class	Mask
AUE_SEMOP	110	ip	0x00000200

Format:

header-token

argument-token (1, "sem ID", semaphore ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the semaphore ID is not valid.

TABLE A-110 setaudit(2)

Event Name	Event ID	Event Class	Mask
AUE_SETAUDIT	133	ad	0x00000800

Format (valid program stack address):

header-token

argument-token (1, "setaudit:audit", audit user ID)

argument-token (1, "setaudit:port", terminal ID)

argument-token (1, "setaudit:machine", terminal ID)

argument-token (1, "setaudit:as_success", preselection mask)

argument-token (1, "setaudit:as_failure", preselection mask)

argument-token (1, "setaudit:asid", audit session ID)

subject-token

return-token

Format (not valid program stack address):

header-token

subject-token

return-token

TABLE A-111 setauid(2)

Event Name	Event ID	Event Class	Mask
AUE_SETAUID	131	ad	0x00000800

Format:

header-token

argument-token (2, "setauid", audit user ID)

subject-token

return-token

TABLE A-112 setegid(2)

Event Name	Event ID	Event Class	Mask
AUE_SETEGID	214	pc	0x00000080

Format:

header-token

argument-token (1, "gid", group ID)

subject-token

return-token

TABLE A-113 seteuid(2)

Event Name	Event ID	Event Class	Mask
AUE_SETEUID	215	pc	0x00000080

Format:

header-token

argument-token (1, "gid", user ID)

subject-token

return-token

TABLE A-114 old setgid(2)

Event Name	Event ID	Event Class	Mask
AUE_SETGID	205	pc	0x00000080

Format:
header-token
argument-token (1, "gid", group ID)
subject-token
return-token

TABLE A-115 setgroups(2)

Event Name	Event ID	Event Class	Mask
AUE_SETGROUPS	26	pc	0x00000080

Format:
header-token
[*argument-token*] (1, "setgroups", group ID)
subject-token
return-token

One *argument-token* for each group set.

TABLE A-116 setpgrp(2)

Event Name	Event ID	Event Class	Mask
AUE_SETPGRP	27	pc	0x00000080

Format:

header-token

subject-token

return-token

TABLE A-117 setrlimit(2)

Event Name	Event ID	Event Class	Mask
AUE_SETRLIMIT	51	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-118 old setuid(2)

Event Name	Event ID	Event Class	Mask
AUE_OSETUID	200	pc	0x00000080

Format:

header-token

argument-token (1, "uid", user ID)

subject-token

return-token

Because of a current bug in the audit software, this token is reported as AUE_OSETUID.

TABLE A-119 shmat(2)

Event Name	Event ID	Event Class	Mask
AUE_SHMAT	96	ip	0x00000200

Format:

header-token

argument-token (1, "shmid", shared memory ID)

argument-token (2, "shmaddr", shared mem addr)

[*ipc-token*]

[*ipc_perm-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the shared memory segment ID is not valid.

TABLE A-120 shmctl(2) - IPC_RMID command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_RMID	92	ip	0x00000200

Format:

header-token

argument-token (1, "shmid", shared memory ID)

[ipc-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the shared memory segment ID is not valid.

TABLE A-121 shmctl(2) - IPC_SET command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_SET	93	ip	0x00000200

Format:

header-token

argument-token (1, "shmid", shared memory ID)

[ipc-token]

[ipc_perm-token]

subject-token

return-token

The `ipc` and `ipc_perm` tokens are not included if the shared memory segment ID is not valid.

TABLE A-122 shmctl(2) - IPC_STAT command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_STAT	94	ip	0x00000200

Format:

header-token

argument-token (1, "shmID", shared memory ID)

[ipc-token]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included if the shared memory segment ID is not valid.

TABLE A-123 shmdt(2)

Event Name	Event ID	Event Class	Mask
AUE_SHMDT	97	ip	0x00000200

Format:

header-token

argument-token (1, "shmaddr", shared mem addr)

subject-token

return-token

TABLE A-124 shmget(2)

Event Name	Event ID	EventClass	Mask
AUE_SHMGET	95	ip	0x00000200

Format:

header-token

argument-token (0, "shmid", shared memory ID)

[*ipc-token*]

[*ipc_perm-token*]

subject-token

return-token

The *ipc* and *ipc_perm* tokens are not included for failed events.

TABLE A-125 stat(2)

Event Name	Event ID	Event Class	Mask
AUE_STAT	16	fa	0x00000004

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-126 statfs(2)

Event Name	Event ID	EventClass	Mask
AUE_STATFS	54	fa	0x00000004

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-127 statvfs(2)

Event Name	Event ID	Event Class	Mask
AUE_STATVFS	234	fa	0x00000004

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-128 stime(2)

Event Name	Event ID	Event Class	Mask
AUE_STIME	201	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-129 symlink(2)

Event Name	Event ID	Event Class	Mask
AUE_SYMLINK	21	fc	0x00000010

Format:

header-token

text-token (symbolic link string)

path-token

[*attr-token*]

subject-token

return-token

TABLE A-130 sysinfo(2)

Event Name	Event ID	Event Class	Mask
AUE_SYSINFO	39	ad	0x00000800

Format:

header-token

argument-token (1, "cmd", command)

text-token (name)

subject-token

return-token

TABLE A-131 system booted

Event Name	Event ID	Event Class	Mask
AUE_SYSTEMBOOT	113	na	0x00000400

Format:

header-token

text-token ("booting kernel")

return-token

TABLE A-132 umount(2) - old version

Event Name	Event ID	Event Class	Mask
AUE_UMOUNT	12	ad	0x00000800

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-133 unlink(2)

Event Name	Event ID	Event Class	Mask
AUE_UNLINK	6	fd	0x00000020

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

TABLE A-134 old utime(2)

Event Name	Event ID	Event Class	Mask
AUE_UTIME	202	fm	0x00000008

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-135 utimes(2)

Event Name	Event ID	Event Class	Mask
AUE_UTIMES	49	fm	0x00000008

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-136 utssys(2) - fusers

Event Name	Event ID	Event Class	Mask
AUE_UTSSYS	233	ad	0x00000800

Format:

header-token

path-token

[attr-token]

subject-token

return-token

TABLE A-137 vfork(2)

Event Name	Event ID	Event Class	Mask
AUE_VFORK	25	pc	0x00000080

Format:

header-token

argument-token (0, "child PID", pid)

subject-token

return-token

The `fork` return values are undefined because the audit record is produced at the point that the child process is spawned.

TABLE A-138 vtrace(2)

Event Name	Event ID	Event Class	Mask
AUE_VTRACE	36	pc	0x00000080

Format:
header-token
subject-token
return-token

TABLE A-139 xmknod(2)

Event Name	Event ID	Event Class	Mask
AUE_XMKNOD	240	fc	0x00000010

Format:
header-token
path-token
[attr-token]
subject-token
return-token

TABLE A-140 xstat(2)

Event Name	Event ID	Event Class	Mask
AUE_XSTAT	235	fa	0x00000004

Format:

header-token

path-token

[*attr-token*]

subject-token

return-token

User-Level Generated Audit Records

These audit records are created by applications that operate outside the kernel. The records are sorted alphabetically by program. The description of each record includes:

- The name of the program
- A man page reference (if appropriate)
- The audit event number
- The audit event name
- The audit record structure

TABLE A-141 allocate-device success

Event Name	Program	Event ID	Event Class	Mask
AUE_allocate_succ	/usr/sbin/ allocate	6200	ad	0x00000800

Format:

header-token

subject-token

newgroups-token

exit-token

TABLE A-142 allocate-device failure

Event Name	Program	Event ID	Event Class	Mask
AUE_allocate_fail	/usr/sbin/ allocate	6201	ad	0x00000800

Format:

header-token

subject-token

newgroups-token

exit-token

TABLE A-143 deallocate-device success

Event Name	Program	Event ID	Event Class	Mask
AUE_allocate_succ	/usr/sbin/ allocate	6202	ad	0x00000800

Format:

header-token

subject-token

newgroups-token

exit-token

TABLE A-144 deallocate-device failure

Event Name	Program	Event ID	Event Class	Mask
AUE_allocate_fail	/usr/sbin/ allocate	6203	ad	0x00000800

Format:

header-token

subject-token

newgroups-token

exit-token

TABLE A-145 allocate-list devices success

Event Name	Program	Event ID	Event Class	Mask
AUE_listdevice_succ	/usr/sbin/ allocate	6205	ad	0x00000800

Format:

header-token

subject-token

[group-token]

exit-token

TABLE A-146 allocate-list devices failure

Event Name	Program	Event ID	Event Class	Mask
AUE_listdevice_fail	/usr/sbin/ allocate	6206	ad	0x00000800

Format:

header-token

subject-token

[group-token]

exit-token

TABLE A-147 at-create crontab

Event Name	Program	Event ID	Event Class	Mask
AUE_at_create	/usr/bin/at	6144	ad	0x00000800

Format:
header-token
subject-token
 [*group-token*]
exit-token

TABLE A-148 at-delete atjob (at or atrm)

Event Name	Program	Event ID	Event Class	Mask
AUE_at_delete	/usr/bin/at	6145	ad	0x00000800

Format:
header-token
subject-token
 [*group-token*]
exit-token

TABLE A-149 at-permission

Event Name	Program	Event ID	Event Class	Mask
AUE_at_perm	/usr/bin/at	6146	ad	0x00000800

Format:

header-token

subject-token

[group-token]

exit-token

TABLE A-150 crontab-crontab created

Event Name	Program	Event ID	Event Class	Mask
AUE_crontab_create	/usr/bin/ crontab	6148	ad	0x00000800

Format:

header-token

subject-token

[group-token]

exit-token

TABLE A-151 crontab-crontab deleted

Event Name	Program	Event ID	Event Class	Mask
AUE_crontab_delete	/usr/bin/ crontab	6149	ad	0x00000800

Format:

header-token

subject-token

[*group-token*]

exit-token

TABLE A-152 cron-invoke atjob or crontab

Event Name	Program	Event ID	Event Class	Mask
AUE_cron_invoke	/usr/bin/ crontab	6147	ad	0x00000800

Format:

header-token

subject-token

text-token (program)

text-token (shell)

text-token (cmd)

exit-token

TABLE A-153 crontab-permission

Event Name	Program	Event ID	Event Class	Mask
AUE_crontab_perm	/usr/bin/ crontab	6150	ad	0x00000800

Format:
header-token
subject-token
 [*group-token*]
exit-token

TABLE A-154 halt(1m)

Event Name	Program	Event ID	Event Class	Mask
AUE_halt_solaris	/usr/sbin/halt	6160	ad	0x00000800

Format:
header-token
subject-token
return-token

TABLE A-155 inetd

Event Name	Program	Event ID	Event Class	Mask
AUE_inetd_connect	/usr/sbin/inetd	6151	na	0x00000400

Format:

header-token

subject-token

text-token (service name)

return-token

TABLE A-156 init(1m)

Event Name	Program	Event ID	Event Class	Mask
AUE_init_solaris	/sbin/init; /usr/sbin/init; /usr/sbin/shutdown	6166	ad	0x00000800

Format:

header-token

subject-token

text-token (init level)

return-token

TABLE A-157 ftp access

Event Name	Program	Event ID	Event Class	Mask
AUE_ftp	/usr/sbin/ in.ftpd	6165	lo	0x00001000

Format:
header-token
subject-token
text-token (error message, failure only)
return-token

TABLE A-158 login - local

Event Name	Program	Event ID	Event Class	Mask
AUE_login	/usr/sbin/login	6152	lo	0x00001000

Format:
header-token
subject-token
text-token (error message)
return-token

TABLE A-159 login - rlogin

Event Name	Program	Event ID	Event Class	Mask
AUE_rlogin	/usr/sbin/login	6155	lo	0x00001000

Format:

header-token

subject-token

text-token (error message)

return-token

TABLE A-160 login - telnet

Event Name	Program	Event ID	Event Class	Mask
AUE_telnet	/usr/sbin/login	6154	lo	0x00001000

Format:

header-token

subject-token

text-token (error message)

return-token

TABLE A-161 logout

Event Name	Program	Event ID	Event Class	Mask
AUE_logout	/usr/sbin/login	6153	lo	0x00001000

Format:

header-token

subject-token

return-token

TABLE A-162 mount

Event Name	Program	Event ID	Event Class	Mask
AUE_mountd_mount	/usr/lib/nfs/ mountd	6156	na	0x00000400

Format:

header-token

subject-token

text-token (remote client hostname)

path-token (mount dir)

text-token (error message, failure only)

return-token

TABLE A-163 unmount

Event Name	Program	Event ID	Event Class	Mask
AUE_mountd_umount	/usr/lib/nfs/ mountd	6157	na	0x00000400

Format:

header-token

subject-token

text-token (remote client hostname)

path-token (mount dir)

text-token (error message, failure only)

return-token

TABLE A-164 passwd

Event Name	Program	Event ID	Event Class	Mask
AUE_passwd	/usr/bin/passwd	6163	lo	0x00001000

Format:

header-token

subject-token

text-token (error message)

return-token

TABLE A-165 poweroff(1m)

Event Name	Program	Event ID	Event Class	Mask
AUE_poweroff_solaris	/usr/sbin/ poweroff	6169	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-166 reboot(1m)

Event Name	Program	Event ID	Event Class	Mask
AUE_reboot_solaris	/usr/sbin/ reboot	6161	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-167 rexd

Event Name	Program	Event ID	Event Class	Mask
AUE_rexd	/usr/sbin/ rpc.rexd	6164	10	0x00001000

Format:

header-token

subject-token

text-token (error message, failure only)

text-token (hostname)

text-token (username)

text-token (command to be executed)

exit-token

TABLE A-168 rexecd

Event Name	Program	Event ID	Event Class	Mask
AUE_rexecd	/usr/sbin/ in.rexecd	6162	10	0x00001000

Format:

header-token

subject-token

text-token (error message, failure only)

text-token (hostname)

text-token (username)

text-token (command to be executed)

exit-token

TABLE A-169 rsh access

Event Name	Program	Event ID	Event Class	Mask
AUE_rshd	/usr/sbin/ in.rshd	6158	lo	0x00001000

Format:

header-token

subject-token

text-token (command string)

text-token (local user)

text-token (remote user)

return-token

TABLE A-170 shutdown(1b)

Event Name	Program	Event ID	Event Class	Mask
AUE_shutdown_solaris	/usr/ucb/ shutdown	6168	ad	0x00000800

Format:

header-token

subject-token

return-token

TABLE A-171 su

Event Name	Program	Event ID	Event Class	Mask
AUE_su	/usr/bin/su	6159	lo	0x00001000

Format:

header-token

text-token (error message)

subject-token

return-token

TABLE A-172 admin(1m)

Event Name	Program	Event ID	Event Class	Mask
AUE_uadmin_solaris	/sbin/uadmin; /usr/sbin/uadmin	6167	ad	0x00000800

Format:

header-token

subject-token

text-token (function)

text-token (argument)

return-token

Event-to-System Call Translation

Table A-173 associates an audit event name with the system call or kernel event that created it. Table A-174 associates an audit event with the application or command that generated it.

TABLE A-173 Event-to-System Call Translation

Audit Event	System Call
AUE_ACCESS	Table A-5
AUE_ACCT	Table A-6
AUE_ADJTIME	Table A-7
AUE_AUDIT	Table A-8
AUE_AUDITON_GETCAR	Table A-9
AUE_AUDITON_GETCLASS	Table A-10
AUE_AUDITON_GETCOND	Table A-11
AUE_AUDITON_GETCWD	Table A-12
AUE_AUDITON_GETKMASK	Table A-13
AUE_AUDITON_GETSTAT	Table A-14
AUE_AUDITON_GPOLICY	Table A-15
AUE_AUDITON_GQCTRL	Table A-16
AUE_AUDITON_SETCLASS	Table A-17
AUE_AUDITON_SETCOND	Table A-18
AUE_AUDITON_SETKMASK	Table A-19

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_AUDITON_SETSMASK	Table A-20
AUE_AUDITON_SETSTAT	Table A-21
AUE_AUDITON_SETUMASK	Table A-22
AUE_AUDITON_SPOLICY	Table A-23
AUE_AUDITON_SQCTRL	Table A-24
AUE_AUDITSVC	Table A-25
AUE_CHDIR	Table A-26
AUE_CHMOD	Table A-27
AUE_CHOWN	Table A-28
AUE_CHROOT	Table A-29
AUE_CLOSE	Table A-30
AUE_CORE	Table A-90
AUE_CREAT	Table A-31
AUE_ENTERPROM	Table A-32
AUE_EXEC	Table A-33

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_EXECVE	Table A-34
AUE_EXIT	Table A-36
AUE_EXITPROM	Table A-35
AUE_FCHDIR	Table A-37
AUE_FCHMOD	Table A-38
AUE_FCHOWN	Table A-39
AUE_FCHROOT	Table A-40
AUE_FCNTL	Table A-41
AUE_FORK	Table A-42
AUE_FORK1	Table A-43
AUE_FSTATFS	Table A-44
AUE_GETAUDIT	Table A-45
AUE_GETAUID	Table A-46
AUE_GETMSG	Table A-47
AUE_GETPMSG	Table A-50

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_GETPORTAUDIT	Table A-51
AUE_IOCTL	Table A-52
AUE_KILL	Table A-53
AUE_LCHOWN	Table A-54
AUE_LINK	Table A-55
AUE_LSTAT	Table A-56
AUE_LXSTAT	Table A-57
AUE_MEMCNTL	Table A-58
AUE_MKDIR	Table A-59
AUE_MKNOD	Table A-60
AUE_MMAP	Table A-61
AUE_MODADDMAJ	Table A-62
AUE_MODCONFIG	Table A-63
AUE_MODLOAD	Table A-64
AUE_MODUNLOAD	Table A-65

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_MOUNT	Table A-66
AUE_MSGCTL_RMID	Table A-67
AUE_MSGCTL_SET	Table A-68
AUE_MSGCTL_STAT	Table A-69
AUE_MSGGET	Table A-70
AUE_MSGRCV	Table A-71
AUE_MSGSND	Table A-72
AUE_MUNMAP	Table A-73
AUE_NICE	Table A-74
AUE_OPEN_R	Table A-75
AUE_OPEN_RC	Table A-76
AUE_OPEN_RT	Table A-78
AUE_OPEN_RTC	Table A-77
AUE_OPEN_RW	Table A-79
AUE_OPEN_RWC	Table A-80

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_OPEN_RWT	Table A-82
AUE_OPEN_RWTC	Table A-81
AUE_OPEN_W	Table A-83
AUE_OPEN_WC	Table A-84
AUE_OPEN_WT	Table A-86
AUE_OPEN_WTC	Table A-85
AUE_OSETUID	Table A-118
AUE_PATHCONF	Table A-87
AUE_PIPE	Table A-88
AUE_PRIOCNLSYS	Table A-89
AUE_PUTMSG	Table A-91
AUE_PUTPMSG	Table A-94
AUE_READLINK	Table A-95
AUE_RENAME	Table A-96
AUE_RMDIR	Table A-97

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_SEMCTL_GETALL	Table A-98
AUE_SEMCTL_GETNCNT	Table A-99
AUE_SEMCTL_GETPID	Table A-100
AUE_SEMCTL_GETVAL	Table A-101
AUE_SEMCTL_GETZCNT	Table A-102
AUE_SEMCTL_RMID	Table A-103
AUE_SEMCTL_SET	Table A-104
AUE_SEMCTL_SETALL	Table A-105
AUE_SEMCTL_SETVAL	Table A-106
AUE_SEMCTL_STAT	Table A-107
AUE_SEMGET	Table A-108
AUE_SEMOP	Table A-109
AUE_SETAUDIT	Table A-110
AUE_SETAUID	Table A-111
AUE_SETEGID	Table A-112

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_SETEUID	Table A-113
AUE_SETGID	Table A-114
AUE_SETGROUPS	Table A-115
AUE_SETPGRP	Table A-116
AUE_SETRLIMIT	Table A-117
AUE_SETUID	Reported as AUE_OSETUID, see Table A-118
AUE_SHMAT	Table A-119
AUE_SHMCTL_RMID	Table A-120
AUE_SHMCTL_SET	Table A-121
AUE_SHMCTL_STAT	Table A-122
AUE_SHMDT	Table A-123
AUE_SHMGET	Table A-124
AUE_SOCKETACCEPT	Table A-48
AUE_SOCKETCONNECT	Table A-92
AUE_SOCKETRECEIVE	Table A-49
AUE_SOCKETSEND	Table A-93

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_STAT	Table A-125
AUE_STATFS	Table A-126
AUE_STATVFS	Table A-127
AUE_STIME	Table A-128
AUE_SYMLINK	Table A-129
AUE_SYSINFO	Table A-130
AUE_SYSTEMBOOT	Table A-131
AUE_UMOUNT	Table A-132
AUE_UNLINK	Table A-133
AUE_UTIME	Table A-134
AUE_UTIMES	Table A-135
AUE_UTSSYS	Table A-136
AUE_VFORK	Table A-137
AUE_VTRACE	Table A-138

TABLE A-173 Event-to-System Call Translation *(continued)*

Audit Event	System Call
AUE_XMKNOD	Table A-139
AUE_XSTAT	Table A-140

TABLE A-174 Event-to-Command Translation

Audit Event	Command
AUE_allocate_succ	Table A-141
AUE_allocate_fail	Table A-142
AUE_deallocate_succ	Table A-143
AUE_deallocate_fail	Table A-144
AUE_listdevice_succ	Table A-145
AUE_listdevice_fail	Table A-146
AUE_at_create	Table A-147
AUE_at_delete	Table A-148
AUE_at_perm	Table A-149
AUE_crontab_create	Table A-150

TABLE A-174 Event-to-Command Translation *(continued)*

Audit Event	Command
AUE_crontab_delete	Table A-151
AUE_cron_invoke	Table A-152
AUE_crontab_perm	Table A-153
AUE_halt_solaris	Table A-154
AUE_inetd_connect	Table A-155
AUE_init_solaris	Table A-156
AUE_ftpd	Table A-157
AUE_login	Table A-158
AUE_rlogin	Table A-159
AUE_telnet	Table A-160
AUE_logout	Table A-161
AUE_mountd_mount	Table A-162
AUE_mountd_umount	Table A-163
AUE_passwd	Table A-164
AUE_poweroff_solaris	Table A-165

TABLE A-174 Event-to-Command Translation *(continued)*

Audit Event	Command
AUE_reboot_solaris	Table A-166
AUE_rexd	Table A-167
AUE_rexecd	Table A-168
AUE_rshd	Table A-169
AUE_shutdown_solaris	Table A-170
AUE_su	Table A-171
AUE_uadmin_solaris	Table A-172

BSM Reference

BSM brings a number of additional utilities to the Solaris operating environment. The utilities are listed here in four sections, each of which has a table below. Each table gives utility names and a short description of the task performed by each utility. The sections are identified by the man page suffix.

TABLE B-1 Section 1M-Maintenance Commands

Command	Task
<code>allocate(1M)</code>	Allocate a device
<code>audit(1M)</code>	Control the audit daemon
<code>audit_startup(1M)</code>	Initialize the audit subsystem
<code>audit_warn(1M)</code>	Run the audit daemon warning script
<code>auditconfig(1M)</code>	Configure auditing
<code>auditd(1M)</code>	Control audit trail files

TABLE B-1 Section 1M-Maintenance Commands *(continued)*

Command	Task
<code>auditreduce(1M)</code>	Merge and select audit records from audit trail files
<code>auditstat(1M)</code>	Display kernel audit statistics
<code>bsmconv(1M)</code>	Enable a Solaris system to use the Basic Security Module
<code>bsmunconv(1M)</code>	Disable the Basic Security Module and return to the Solaris operating environment (see the <code>bsmconv(1M)</code> man page)
<code>deallocate(1M)</code>	Deallocate a device
<code>dminfo(1M)</code>	Report information about a device entry in a device maps file
<code>list_devices(1M)</code>	List allocatable devices
<code>praudit(1M)</code>	Print contents of an audit trail file

TABLE B-2 Section 2-System Calls

System Call	Task
<code>audit(2)</code>	Write a record to the audit log
<code>auditon(2)</code>	Manipulate auditing

TABLE B-2 Section 2-System Calls *(continued)*

System Call	Task
<code>auditsvc(2)</code>	Write audit log to specified file descriptor
<code>getaudit(2)</code>	Get process audit information
<code>getauid(2)</code>	Get user audit identity
<code>setaudit(2)</code>	Get process audit information (see <code>getaudit(2)</code>)
<code>setauid(2)</code>	Get user audit identity (see <code>getauid(2)</code>)

TABLE B-3 Section 3-C Library Functions

Library Call	Task
<code>au_open(3)</code> , <code>au_close(3)</code> , <code>au_write(3)</code>	Construct and write audit records
<code>au_preselect(3)</code>	Preselect an audit event
<code>au_to_arg(3)</code> , <code>au_to_attr(3)</code> , <code>au_to_data(3)</code> , <code>au_to_groups(3)</code> , <code>au_to_in_addr(3)</code> , <code>au_to_ipc(3)</code> , <code>au_to_ipc_perm(3)</code> , <code>au_to_iport(3)</code> , <code>au_to_me(3)</code> , <code>au_to_opaque(3)</code> , <code>au_to_path(3)</code> , <code>au_to_process(3)</code> , <code>au_to_return(3)</code> , <code>au_to_socket(3)</code> , <code>au_to_text(3)</code>	Create audit record tokens (see <code>au_to(3)</code> for all of these functions)
<code>au_user_mask(3)</code>	Get user's binary preselection mask
<code>getacinfo(3)</code> , <code>getacdir(3)</code> , <code>getacflg(3)</code> , <code>getacmin(3)</code> , <code>getacna(3)</code> , <code>setac(3)</code> , <code>endac(3)</code>	Get audit control file information

TABLE B-3 Section 3-C Library Functions *(continued)*

Library Call	Task
<code>getaclassent(3)</code> , <code>getaclassnam(3)</code> , <code>setaclass(3)</code> , <code>endaclass(3)</code> , <code>getaclassnam_r(3)</code> , <code>getaclassent_r(3)</code>	Get <code>audit_class</code> entry
<code>getauditflags(3)</code> , <code>getauditflagsbin(3)</code> , <code>getauditflagschar(3)</code>	Convert audit flag specifications
<code>getauevent(3)</code> , <code>getauevnam(3)</code> , <code>getauevnum(3)</code> , <code>getauevnonam(3)</code> , <code>setauevent(3)</code> , <code>endauevent(3)</code> , <code>getauevent_r(3)</code> , <code>getauevnam_r(3)</code> , <code>getauevnum_r(3)</code>	Get <code>audit_user</code> entry
<code>getauusernam(3)</code> , <code>getauuserent(3)</code> , <code>setauuser(3)</code> , <code>endauuser(3)</code>	Get <code>audit_user</code> entry
<code>getfauditflags(3)</code>	Generate the process audit state

TABLE B-4 Section 4-Headers, Tables, and Macros

Files	Task
<code>audit.log(4)</code>	Gives format for an audit trail file
<code>audit_class(4)</code>	Gives audit class definitions
<code>audit_control(4)</code>	Controls information for system audit daemon
<code>audit_data(4)</code>	Holds current information on the audit daemon
<code>audit_event(4)</code>	Holds audit event definition and class mapping

TABLE B-4 Section 4-Headers, Tables, and Macros *(continued)*

Files	Task
<code>audit_user(4)</code>	Holds per-user auditing data file
<code>device_allocate(4)</code>	Contains physical device information
<code>device_maps(4)</code>	Contains physical device information

Index

Special Characters

- # for comments in files, 58, 60
- * in device_allocate file, 60, 61
- + audit flag prefix, 10, 11
- audit flag prefix, 10, 11
- \ ending file lines, 58, 60
- ^+ audit flag prefix, 11
- ^- audit flag prefix, 11

A

- a option of auditreduce command, 52
- access audit record, 84
- acct audit record, 84
- ad audit flag, 9
- adding devices, 66
- adjtime audit record, 85
- administering auditing
 - See also* audit records; audit tokens; audit trail,
- audit administration account, 32, 33
- audit classes
 - auditconfig command options, 35
 - changing definitions, 37
 - flags and definitions, 9, 10
 - mapping events, 7, 37
 - overview, 7, 8
 - selecting for auditing, 7

audit events

- audit tokens, 42
- auditconfig command options, 34, 35
- categories, 7
- event-to-system call translation
 - table, 172, 182
- including in audit trail, 7
- kernel events, 7, 34, 35, 42
- mapping to classes, 7, 37
- numbers, 7
- overview, 7, 8
- record formats and, 41
- user-level events, 7, 35, 42

- audit files, 24, 27
 - auditreduce command, 19, 21
 - combining, 19, 21, 24
 - copying login/logout messages to
 - single file, 51, 52
 - directory locations, 25, 28
 - displaying in entirety, 51
 - file token, 45, 75
 - managing size of, 17
 - minimum free space for file
 - systems, 12
 - names, 25, 27
 - nonactive files marked
 - not_terminated, 27, 52
 - order for opening, 12
 - overview, 24, 25
 - permissions, 28
 - printing, 51
 - reducing, 19, 21, 24
 - reducing storage-space
 - requirements, 21, 23
 - switching to new file, 17
 - time stamps, 26
- audit flags, 8, 11
 - auditconfig command options, 35
 - audit_control file line, 12
 - audit_user file, 13, 14
 - definitions, 9, 10
 - machine-wide, 8, 12
 - overview, 8
 - policy flags, 36
 - prefixes, 11
 - process preselection mask, 14
 - syntax, 10
- audit partitions, 27, 29
- audit records, 8
- audit trail creation, 15, 17
 - audit daemon's role, 16, 17
 - audit_data file, 16
 - directory suitability, 17
 - managing audit file size, 17
 - overview, 15
- audit trail overflow prevention, 33, 34
- auditreduce command, 19, 21, 50, 53
 - a option, 52
 - b option, 52
 - capabilities, 50
 - cleaning not_terminated files, 27, 52
 - d option, 51
 - described, 19, 40, 50
 - distributed systems, 50
 - examples, 51, 52
 - O option, 24, 27, 51, 52
 - options, 19, 52, 53
 - time stamp use, 26
 - without options, 19, 21
- audit_control file
 - audit_user file modification, 13
 - overview, 12
 - prefixes in flags line, 11
 - problem with contents, 18
- audit_user file audit fields, 13, 14
- audit_warn script, 16 to 18
- configuration
 - audit trail overflow prevention, 33, 34
 - auditconfig command, 34, 35
 - overview, 29, 30
 - planning, 30, 33
 - setting audit policies, 36
- cost control, 21, 23
 - analysis, 21
 - processing time, 21
 - storage, 21, 23
- efficiency, 23, 24
- normal users, 23
- overview, 6
- process audit characteristics, 14, 15
 - audit ID, 15
 - audit session ID, 15
 - process preselection mask, 14, 22, 23
 - terminal ID, 15
- startup, 6
- administrative audit class, 9
- all
 - audit class, 10
 - audit flag
 - caution for using, 10
 - described, 10
 - in user audit fields, 13
 - allhard string with audit_warn script, 18

- allocatable devices, *see* device allocation,
- allocate audit record
 - allocate-list device failure, 159
 - allocate-list device success, 158
 - deallocate device, 157
 - deallocate device failure, 158
 - device allocate failure, 157
 - device allocate success, 156
- allocate command
 - See also* device allocation,
 - how the allocate mechanism works, 64, 66
 - options, 57
 - using, 68
- allocate error state, 58
- allocating devices, *see* device allocation,
- allsoft string with audit_warn script, 18
- always-audit flags
 - described, 13, 14
 - process preselection mask, 14
- analysis, 39, 54
 - audit record format, 41, 50
 - auditing features, 39, 40
 - auditreduce command, 41, 50, 53
 - costs, 21
 - praudit command, 41, 53, 54
 - tools, 40, 41
- ap audit flag, 9
- application audit class, 9
- arbitrary token, 44, 72
- Archive tape drive clean script, 60
- arg token, 44, 73
- arge policy
 - exec_env token and, 74
 - flag, 36
- argv policy
 - exec_args token and, 74
 - flag, 36
- asterisk (*) in device_allocate file, 60, 61
- at audit record
 - at-create crontab, 159
 - at-delete atjob, 160
 - at-permission, 160
- attr token, 45, 73
- audio devices
 - device-clean scripts, 63
- audio devices, *see* device allocation,
 - device-clean scripts,
- AUDIOGETREG ioctl system call, 63
- AUDIOSETREG ioctl system call, 63
- audio_clean script, 63
- AUDIO_DRAIN ioctl system call, 63
- AUDIO_SETINFO ioctl system call, 63
- audit -n command, 17
- audit -s command
 - preselection mask for existing processes, 12
 - rereading audit files, 16
 - resetting directory pointer, 17
- audit -t command, 15
- audit administration account, 32, 33
- audit attributes, *see* audit tokens,
- audit audit record, 85
- audit classes
 - auditconfig command options, 35
 - changing definitions, 37
 - flags and definitions, 9, 10
 - mapping events, 7, 37
 - overview, 7, 8
 - selecting for auditing, 7
- audit daemon
 - audit trail creation, 15 to 17
 - audit_startup file, 6
 - audit_warn script
 - conditions invoking, 18
 - described, 16, 17
 - execution of, 16
 - directories suitable to, 17
 - enabling auditing, 6
 - functions, 16
 - order audit files are opened, 12
 - rereading the audit_control file, 12
 - terminating, 15
- audit events
 - See also* audit classes,
 - audit_event file
 - audit event type, 41
 - overview, 7, 8
 - categories, 7
 - event-to-system call translation table, 172, 182
 - including in audit trail, 7
 - kernel events
 - audit tokens, 42
 - auditconfig command options, 34, 35
 - described, 7

- mapping to classes, 7, 37
- numbers, 7
- overview, 7, 8
- record formats and, 41
- user-level events
 - audit tokens, 42
 - auditconfig command options, 35
 - described, 7
- audit files, 24, 27
 - See also* trail; directories,
 - auditreduce command, 19, 21
 - combining, 19, 21, 24
 - copying login/logout messages to single file, 51, 52
 - directory locations, 25, 28
 - displaying in entirety, 51
 - file token, 45, 75
 - managing size of, 17
 - minimum free space for file systems, 12
 - names, 25, 27
 - closed files, 26
 - form, 25
 - still-active files, 26
 - time stamps, 26
 - use, 26
 - nonactive files marked not_terminated, 27, 52
 - order for opening, 12
 - overview, 24, 25
 - permissions, 28
 - printing, 51
 - reducing, 19, 21, 24
 - reducing storage-space requirements, 21, 23
 - switching to new file, 17
 - time stamps, 26
- audit flags, 8, 11
 - auditconfig command options, 35
 - audit_control file line, 12
 - audit_user file, 13, 14
 - definitions, 9, 10
 - machine-wide, 8, 12
 - overview, 8
 - policy flags, 36
 - prefixes, 10, 11
 - process preselection mask, 14
 - syntax, 10
- audit ID, 6, 15, 40
- audit log files, *see* audit files,
- audit partitions, 27, 29
- audit policies
 - See also* audit flags,
 - auditconfig options, 35
 - setting, 36
- audit records, 69, 172
 - See also* audit tokens; specific audit records,
 - audit directories full, 16, 18, 83
 - converting to human-readable format, 8, 19, 41, 53, 54
 - displaying, 41
 - format or structure, 41, 50, 69, 84
 - kernel-level generated, 84, 156
 - overview, 8
 - policy flags, 36
 - reducing audit files, 24
 - selecting, 40
 - self-contained records, 40
 - tools, 40, 41
 - user-level generated, 156, 172
- audit server mount-point path names, 28
- audit session ID, 15, 40
- audit threshold, 12
- audit tokens
 - arbitrary token, 44, 72
 - arg token, 44, 73
 - attr token, 45, 73
 - audit record format, 41, 50, 69, 70
 - described, 8
 - exec_args token, 73
 - exec_env token, 74
 - exit token, 45, 74
 - file token, 45, 75
 - groups token, 46, 75
 - header token, 42, 43, 76
 - in_addr token, 46, 76
 - ip token, 46, 76
 - ipc token, 46, 77
 - ipc_perm token, 47, 78
 - ipport token, 47, 78
 - newgroups token, 78
 - opaque token, 47, 79
 - order in audit record, 42
 - path token, 47, 79
 - policy flags, 36

- process token, 48, 80
- return token, 48, 80
- seq token, 49, 81
- socket token, 49, 81
- socket-inet token, 81
- subject token, 49, 82
- table of, 70, 71
- text token, 50, 82
- trailer token, 42, 44, 83
- types, 41

audit trail

- See also* audit files, audit records; audit tokens,
- analysis, 39, 54
 - audit record format, 41, 50
 - auditing features, 39, 40
 - auditreduce command, 41, 50, 53
 - costs, 21
 - praudit command, 41, 53, 54
 - tools, 40, 41
- creating, 15, 17, 24
 - audit daemon's role, 15 to 17
 - audit_data file, 16
 - directory suitability, 17
 - managing audit file size, 17
 - overview, 15
- directory locations, 25, 28
- events included, 7
- merging all files, 19, 21
- monitoring in real time, 23
- overflow prevention, 33, 34

auditconfig command

- audit flags as arguments, 8
- options, 34, 35
- prefixes for flags, 11
- reducing storage-space requirements, 23

auditd daemon

- audit trail creation, 15 to 17
- audit_startup file, 6
- audit_warn script
 - conditions invoking, 18
 - described, 16, 17
 - execution of, 16
- directories suitable to, 17
- enabling auditing, 6
- functions, 16
- order audit files are opened, 12
- rereading the audit_control file, 12
- terminating, 15

auditing, *see* administering auditing; audit trail,

auditon audit record

- A_GETCAR command, 86
- A_GETCLASS command, 86
- A_GETCOND command, 87
- A_GETCWD command, 87
- A_GETKMASK command, 88
- A_GETSTAT command, 88
- A_GPOLICY command, 89
- A_GQCTRL command, 89
- A_SETCLASS command, 90
- A_SETCOND command, 90
- A_SETKMASK command, 91
- A_SETSMASK command, 91
- A_SETSTAT command, 92
- A_SETUMASK command, 92
- A_SPOLICY command, 93
- A_SQCTRL command, 93

auditreduce command, 19, 21

- a option, 52
- b option, 52
- capabilities, 50
- cleaning not_terminated files, 27, 52
- d option, 51
- described, 19, 40, 50
- distributed systems, 50
- examples, 51, 52
- m option, 53
- O option, 24, 27, 51, 52
- options, 19, 52, 53
- time stamp use, 26
- without options, 19, 21

auditsvc

- audit record, 94
- system call
 - fails, 18, 83

audit_control file

- audit daemon rereading after editing, 12
- audit_user file modification, 13
- dir: line
 - described, 12
 - examples, 13, 29
 - files subdirectory, 28
- examples, 13, 29

- flags: line
 - described, 12
 - prefixes in, 11
 - process preselection mask, 14
- minfree: line
 - audit_warn condition, 18
 - described, 12
- naflags: line, 12
- overview, 12
- prefixes in flags line, 11
- problem with contents, 18
- audit_data file, 16
- audit_event file
 - See also* audit events,
 - audit event type, 41
 - overview, 7, 8
- audit_startup file, 6
- audit_user file
 - prefixes for flags, 11
 - process preselection mask, 14
 - user audit fields, 13, 14
- audit_warn script, 17, 18
 - allhard string, 18
 - allsoft string, 18
 - audit daemon execution of, 16
 - auditsvc string, 18
 - conditions invoking, 18, 19
 - described, 16, 17
 - ebusy string, 18
 - hard string, 18
 - postsigterm string, 18
 - soft string, 18
 - tmpfile string, 18
- AUE_... names, 7
 - event-to-system call translation table, 172, 182
- automatically enabling auditing, 6

B

- b option of auditreduce command, 52
- backslash (\) ending file lines, 58, 60
- Basic Security Module (BSM)
 - client-server relationships, 2
 - disabling, 2
 - enabling, 1, 2
 - installing, 1, 3
 - packages, 1

- binary audit record format, 41
- BSM, *see* Basic Security Module (BSM),
- bsmconv script
 - devicemaps file creation, 58
 - enabling BSM, 1, 2
- bsmunconv script, 2

C

- C2 TCSEC features, 55
- carat (^) in audit flag prefixes, 11
- cartridge tape drives, *see* tape drives,
- CD-ROM drives
 - See also* device allocation,
 - device-clean scripts, 62
- change password audit record, 168
- chdir audit record, 94
- chkconf option of auditconfig command, 34
- chmod audit record, 95
- chown audit record, 95
- chroot audit record, 96
- cl audit flag, 9
- classes
 - auditconfig command options, 35
 - changing definitions, 37
 - flags and definitions, 9, 10
 - mapping events, 7, 37
 - overview, 7, 8
 - selecting for auditing, 7
- clean scripts, *see* device-clean scripts,
- cleaning not_terminated files, 27, 52
- clients, enabling BSM for, 3
- close audit record, 96
- cnt policy, 32, 33
 - flag, 36
- combining audit files, 24
 - auditreduce command, 19, 21
- commands
 - See also* specific commands,
 - device-allocation utilities, 57
- comments
 - device_allocate file, 60
 - device_maps file, 58
- conf option of auditconfig command, 34
- configuring
 - audit trail overflow prevention, 33, 34
 - auditconfig command, 34, 36

- overview, 29, 30
- planning, 30, 33
- setting audit policies, 36
- converting audit records to human-readable
 - format, 8, 19, 41, 53, 54
- copying login/logout messages to single
 - file, 51, 52
- cost control, 21, 23
 - analysis, 21
 - processing time, 21
 - storage, 21, 23
- creat audit record, 97
- creating the audit trail, 15, 17
 - audit daemon's role, 16, 17
 - audit_data file, 16
 - directory suitability, 17
 - managing audit file size, 17
 - overview, 15
- cron job, 17
- crontab audit record
 - cron-invoke atjob or crontab, 162
 - crontab-crntab created, 161
 - crontab-crntab deleted, 161
 - crontab-permission, 162

D

- d option
 - auditreduce command, 51, 53
- daemon, audit, *see* audit daemon,
- date-time auditreduce command options, 52
- deallocate command
 - allocate error state, 58
 - described, 57, 67
 - device-clean scripts and, 63
 - using, 68
- debugging sequence number, 49, 81
- defaults
 - audit policies, 36
 - audit_startup file, 6
 - machine-wide, 8
 - praudit output format, 53, 54
 - header token, 43
- device allocation, 55, 68
 - adding devices, 66
 - allocatable devices, 60, 66

- allocate command
 - how the allocate mechanism
 - works, 64, 66
 - options, 57
 - using, 68
- allocate error state, 58
- allocating a device, 68
- components of the allocation
 - mechanism, 56
- deallocate command
 - allocate error state, 58
 - described, 57, 67
 - device-clean scripts and, 63
 - using, 68
- device-clean scripts, 61, 63
 - adding devices, 66
 - audio devices, 63
 - CD-ROM drives, 62
 - described, 61
 - diskette drives, 62
 - options, 63
 - tape drives, 60, 61
 - writing new scripts, 63
- device_allocate file, 59, 61
- device_maps file, 58, 59
- list_devices command, 57, 67
- lock file setup, 63, 66
- managing devices, 66
- reallocating, 57
- risks associated with device use, 56
- using device allocations, 68
- utilities, 57
- device-clean scripts
 - adding devices, 66
 - audio devices, 63
 - CD-ROM drives, 62
 - described, 61
 - diskette drives, 62
 - options, 63
 - tape drives, 60, 61
 - writing new scripts, 63
- devices
 - See also* device allocation,
 - adding, 66
 - lock files, 63, 66
 - managing, 66
- device_allocate file

- format, 60, 61
 - overview, 59, 61
- device_maps file
 - format, 58, 59
 - overview, 58
- dir: line in audit_control file
 - described, 12
 - example, 13, 29
 - for files subdirectory, 28
- directories
 - audit daemon pointer, 17
 - audit directories full, 16, 18, 83
 - audit directory locations, 25, 28
 - audit partitions, 27, 29
 - audit_control file definitions, 12
 - diskfull machines, 25, 28
 - files subdirectory, 28
 - mounting audit directories, 25
 - permissions, 28
 - suitable to audit daemon, 17
- disabling BSM, 2
- disk-space requirements, 21, 23
- diskette drives
 - See also* device allocation,
 - device-clean scripts, 62
- diskfull machines' audit directory, 25, 28
- diskless clients, enabling BSM for, 3
- displaying
 - audit log in entirety, 51
 - audit records, 41
- distributed systems' auditreduce command
 - use, 50
- dminfo command, 58
- drives, *see* device allocation,

E

- ebusy string and audit_warn script, 18
- efficiency, 23, 24
- eject command, 62
- enabling
 - auditing, 6
 - BSM, 1, 2
- ending
 - disabling BSM, 2
 - signal received during auditing
 - shutdown, 18
 - terminating audit daemon, 15

- enter prom audit record, 97
- errors
 - allocate error state, 58
 - audit directories full, 16, 18, 83
 - internal errors, 18
 - /etc/security directory, 28
 - /etc/security/audit directory, 25, 28
 - /etc/security/audit/bsmconv script
 - enabling BSM, 1, 2
 - devicemaps file creation, 58
 - /etc/security/audit/bsmunconv script, 2
 - /etc/security/audit_control file, *see*
 - audit_control file,
 - /etc/security/audit_data file, 16
 - /etc/security/audit_event file
 - See also* audit events,
 - overview, 7, 8, 41
 - /etc/security/audit_startup file, 6
 - /etc/security/audit_warn script, 16 to 18
 - /etc/security/dev lock files, 63, 66
 - event modifier field flags (header token), 76
 - event numbers, 7
- events
 - See also* audit classes,
 - categories, 7
 - event-to-system call translation table, 172, 182
 - including in audit trail, 7
 - kernel events
 - audit tokens, 42
 - auditconfig command options, 34, 35
 - described, 7
 - mapping to classes, 7, 37
 - numbers, 7
 - overview, 7, 8
 - record formats and, 41
 - user-level events
 - audit tokens, 42
 - auditconfig command options, 35
 - described, 7
- ex audit flag, 9
- exec audit class, 9
- exec audit record, 98
- execve audit record, 98
- exec_args token, 73
- exec_env token, 74
- exit audit record, 99

- exit prom audit record, 99
- exit token, 45, 74
- export list, 25

F

- F option
 - allocate command, 57, 63
- fa audit flag, 9
- failure
 - audit flag prefix, 10
 - turning off audit flags for, 11
- fc audit flag, 9
- fchdir audit record, 100
- fchmod audit record, 100
- fchown audit record, 101
- fchroot audit record, 102
- fcntl audit record, 103
- fd audit flag, 9
- fd_clean script, 62
- file systems, *see* audit files; directories,
- file token, 45, 75
- file vnode token, 45, 73
- files subdirectory, 28
- files, audit, *see* audit files,
- files, lock, 63, 66
- file_attr_acc audit class, 9
- file_attr_mod audit class, 9
- file_close audit class, 9
- file_creation audit class, 9
- file_deletion audit class, 9
- file_read audit class, 9
- file_write audit class, 9
- flags, 8, 11
 - auditconfig command options, 35
 - audit_control file line, 12
 - audit_user file, 13, 14
 - definitions, 9, 10
 - machine-wide, 8, 12
 - overview, 8
 - policy flags, 36
 - prefixes, 10, 11
 - process preselection mask, 14
 - syntax, 10
- flags: line in audit_control file
 - described, 12
 - prefixes in, 11
 - process preselection mask, 14

- fm audit flag, 9
- forced cleanup, 63
- fork audit record, 103
- fork1 audit record, 104
- fr audit flag, 9
- fstatfs audit record, 104
- ftpd login audit record, 164
- fw audit flag, 9

G

- getaudit audit record, 105
- getaudit audit record, 105
- getclass option of auditconfig command, 35
- getcond option of auditconfig command, 34
- getmsg audit record, 106
 - socket accept, 106
 - socket receive, 107
- getpinfo option of auditconfig command, 35
- getpmsg audit record, 107
- getpolicy option of auditconfig command, 35
- getportaudit audit record, 108
- graphics tablets, *see* device allocation,
- group policy
 - flag, 36
 - groups token, 46, 75
 - newgroups token, 78
- groups token, 46, 75

H

- halt: machine halt audit record, 163
- hard string with audit_warn script, 18
- hard-disk-space requirements, 21, 23
- header token
 - described, 43, 76
 - event-modifier field flags, 76
 - fields, 43
 - format, 76
 - order in audit record, 42, 76
 - praudit display, 43
- human-readable audit record format
 - See also* audit tokens,
 - converting audit records to, 8, 19, 41, 53, 54
 - described, 41, 50

I

- I option
 - deallocate command, 57, 63
- IDs
 - audit, 6, 15, 40
 - audit session, 15, 40
 - audit user, 40
 - auditconfig command options, 35
 - terminal, 15
- in.ftpd audit record, 164
- in.rexecd audit record, 170
- in.rshd: rshd access denials/grants audit record, 170
- inetd: inetd service request audit record, 163
- init: init service request audit record, 164
- installing BSM, 1, 3
- Internet-related tokens
 - in_addr token, 46, 76
 - ip token, 46, 76
 - ipport token, 47, 78
 - socket token, 49, 81
 - socket-inet token, 81
- in_addr token, 46, 76
- io audit flag, 9
- ioctl audit class, 9
- ioctl system calls, 9, 63
- ioctl: ioctl to special devices audit record, 108
- ip audit flag, 9
- ip token, 46, 76
- ipc audit class, 9
- ipc token, 46, 77
- ipc type field values (ipc token), 77
- ipc_perm token, 47, 78
- ipport token, 47, 78
- item size field values (arbitrary token), 72

K

- kernel events
 - See also* audit events,
 - audit records, 84, 156
 - audit tokens, 42
 - auditconfig command options, 34, 35
 - described, 7
- kill audit record, 110

L

- l option
 - praudit command, 53
- lchown audit record, 111
- link audit record, 111
- list_devices command, 57, 67
- lo audit flag, 9
- lock files
 - how the allocate mechanism works, 64, 66
 - setting up, 63
- log files, *see* audit files,
- login audit record
 - logout, 166
 - rlogin, 165
 - telnet login, 166
 - terminal login, 165
- login/logout messages, copying to single file, 51, 52
- login_logout audit class, 9
- lsevent option of auditconfig command, 35
- lspolicy option of auditconfig command, 35, 36
- lstat audit record, 111
- lxstat audit record, 112

M

- m option of auditreduce command, 53
- machine halt audit record, 163
- machine reboot audit record, 169
- managing devices, 66
- mappings, class, 7, 37
- mask, process preselection
 - auditconfig command options, 35
 - described, 14
 - machine-wide, 12
 - reducing storage costs, 22, 23
- memcntl audit record, 112
- minfree: line in audit_control file
 - audit_warn condition, 18
 - described, 12
 - determining space needed, 31
- minus (-) audit flag prefix, 10, 11
- mkdir audit record, 113
- mknod audit record, 113
- mmap audit record, 114
- modctl audit record

- MODADDMAJBIND command, 115
- MODCONFIG command, 116
- MODLOAD command, 116
- MODUNLOAD command, 117
- modems, *see* device allocation,
- monitoring audit trail in real time, 23
- mount audit record, 117
- mountd audit record
 - NFS mount request, 167
 - NFS unmount request, 167
- mounting audit directories, 25
- msgctl audit record
 - IPC_RMID command, 118
 - IPC_SET command, 119
 - IPC_STAT command, 119
- msgget audit record, 120
- msgrev audit record, 120
- msgsnd audit record, 121
- mt command, device-cleanup option, 62
- munmap audit record, 121

N

- na audit flag, 9
- naflags: line in audit_control file, 12
- names
 - audit classes, 9, 10
 - audit files
 - closed files, 26
 - form, 25
 - still-active files, 26
 - time stamps, 26
 - use, 26
 - audit flags, 9, 10
 - device names
 - device_allocate file, 61
 - device_maps file, 59
 - IDs
 - audit, 6, 15
 - audit session, 15, 40
 - auditconfig command options, 35
 - terminal, 15
 - kernel events, 7
 - mount-point path names on audit
 - servers, 28
 - user-level events, 7
- network audit class, 9
- never-audit flags, 13, 14

- newgroups token, 78
- NFS mount request audit record, 167
- NFS unmount request audit record, 167
- nice audit record, 122
- no audit flag, 9
- nonattributable flags in audit_control file, 12
- non_attrib audit class, 9
- normal users, auditing, 23
- not_terminated files, cleaning, 27, 52
- no_class audit class, 9
- nt audit flag, 9
- null audit class, 9
- numbers, event, 7

O

- O option of auditreduce command, 24, 27, 51, 52
- object-reuse requirement, 55, 61, 63
 - device-clean scripts
 - adding devices, 66
 - audio devices, 63
 - CD-ROM drives, 62
 - described, 61
 - diskette drives, 62
 - tape drives, 60, 61
 - writing new scripts, 63
- opaque token, 47, 79
- open audit record
 - read, 122
 - read, create, 123
 - read, create, truncate, 123
 - read, truncate, 124
 - read, write, 124
 - read, write, create, 125
 - read, write, create, truncate, 125
 - read, write, truncate, 126
 - write, 126
 - write, create, 127
 - write, create, truncate, 127
 - write, truncate, 128
- ot audit flag, 10
- other audit class, 10
- overflow prevention for audit trail, 33, 34

P

- partitions, audit, 27, 29
- passwd audit record, 168
- path policy flag, 36
- path token, 47, 79
- pathconf audit record, 128
- pc audit flag, 9
- permissions for audit file systems, 29
- pipe audit record, 129
- plus (+) audit flag prefix, 10, 11
- policies
 - See also* audit flags,
 - auditconfig options, 35
 - setting, 36
- postsigterm string and audit_warn script, 18
- pound sign (#) for comments in files, 58, 60
- poweroff audit record, 168
- praudit command
 - See also* audit tokens,
 - converting audit records to
 - human-readable format, 8, 19
 - described, 41
 - human-readable format, 42, 50
 - output formats, 53, 54
 - piping auditreduce output to, 51
 - using, 53, 54
- prefixes in audit flags, 10, 11
- preselection mask
 - auditconfig command options, 35
 - described, 14
 - machine-wide, 12
 - reducing storage costs, 22, 23
- primary audit directory, 12, 27
- print format field values (arbitrary token), 72
- printing audit log, 51
- priocntlsys audit record, 129
- process audit characteristics, 14, 15
 - audit ID, 15
 - audit session ID, 15
 - process preselection mask, 14, 22, 23
 - terminal ID, 15
- process audit class, 9
- process dumped core audit record, 130
- process groups tokens
 - groups token, 46, 75
 - newgroups token, 78
- process preselection mask

- auditconfig command options, 35
- described, 14
- reducing storage costs, 22, 23
- process token, 48, 80
- processing time costs, 21
- putmsg audit record, 130
 - socket connect, 131
 - socket send, 131
- putpmsg audit record, 132

R

- r praudit output format, 53
 - header token, 43
- raw praudit output format, 53
 - header token, 43
- readlink audit record, 132
- reallocating devices, 57
- reboot: machine reboot audit record, 169
- records, *see* audit records,
- reducing audit files, 24
 - auditreduce command, 19, 21
 - storage-space requirements, 21, 23
- rename audit record, 133
- return token, 48, 80
- rewoffl option of mt command, 62
- risks associated with device use, 56
- rmdir audit record, 133
- rpc.rexd audit record, 169
- rshd access denials/grants audit record, 170

S

- S option of st_clean script, 63
- s praudit output format, 53
 - header token, 43
- /sbin/init audit record, 164
- SCSI devices
 - See also* device allocation,
 - st_clean script, 60
- secondary audit directory, 12, 27
- security risks associated with device use, 56
- selecting audit records, 40
- semctl audit record
 - GETALL command, 134
 - GETNCNT command, 134
 - GETPID command, 135

- GETVAL command, 135
- GETZCNT command, 136
- IPC_RMID command, 136
- IPC_SET command, 137
- IPC_STAT command, 138
- SETALL command, 137
- SETVAL command, 138
- semget audit record, 139
- semop audit record, 139
- seq policy flag, 37
- seq token, 49, 81
- servers, enabling BSM for clients, 3
- session ID, 15, 40
- setaudit audit record, 140
- setaudit audit record, 141
- setclass option of auditconfig command, 35
- setcond option of auditconfig command, 35
- setegid audit record, 141
- seteuid audit record, 142
- setgid audit record, 142
- setgroups audit record, 143
- setpgrp audit record, 143
- setpmask option of auditconfig command, 35
- setpolicy option of auditconfig command, 35, 36
- setrlimit audit record, 144
- setsmask option of auditconfig command, 35
- setuid audit record, 144
- setumask option of auditconfig command, 35
- SHIELD Basic Security Module, *see* Basic Security Module (BSM),
- shmat audit record, 145
- shmctl audit record
 - IPC_RMID command, 145
 - IPC_SET command, 146
 - IPC_STAT command, 146
- shmdt audit record, 147
- shmget audit record, 147
- short praudit output format, 53
 - header token, 43
- shutdown audit record, 171
- shutting down, *see* terminating,
- signal received during auditing shutdown, 18
- size
 - managing audit files, 17
 - reducing audit files, 24
 - auditreduce command, 19, 21
 - storage-space requirements, 21, 23
- socket accept audit record, 106
- socket connect audit record, 131
- socket receive audit record, 107
- socket send audit record, 131
- socket token, 49, 81
- socket-inet token, 81
- soft limit
 - audit_warn condition, 18
 - determining space needed, 31
 - minfree: line described, 12
- soft string with audit_warn script, 18
- Solaris SHIELD Basic Security Module, *see* Basic Security Module (BSM),
- sr_clean script, 62
- standard cleanup, 63
- starting, *see* enabling,
- stat audit record, 148
- statfs audit record, 148
- statvfs audit record, 149
- stime audit record, 149
- storage costs, 21, 23
- storage overflow prevention, 33, 34
- st_clean script for tape drives, 60, 62
- su audit record, 171
- subject token, 49, 82
- success
 - audit flag prefix, 10
 - turning off audit flags for, 11
- SUNWcar package, 1
- SUNWcsr package, 1
- SUNWcsu package, 1
- SUNWhea package, 1
- SUNWman package, 1
- symlink audit record, 150
- sysinfo audit record, 150
- system booted audit record, 151
- system calls
 - arg token, 44, 73
 - auditsvc fails, 18, 83
 - close, 9
 - event numbers, 7
 - event-to-system call translation table, 172, 182
 - exec_args token, 73
 - exec_env token, 74
 - ioctl, 9, 63

- return token, 48, 80
- System V IPC
 - ipc audit class, 9
 - ipc token, 46, 77
 - ipc_perm token, 47, 78

T

- tail command, 23
- tape drives
 - See also* device allocation,
 - device-clean scripts, 61
 - risks associated with use, 56
 - st_clean script, 60
- TCP address, 47, 78
- TCSEC (Trusted Computer System Evaluation Criteria) C2 features, 55
- temporary file cannot be used, 18
- terminal ID, 15
- terminals, *see* device allocation,
- terminating
 - audit daemon, 15
 - signal received during auditing
 - shutdown, 18
- text token, 50, 82
- time stamps in audit files, 26
- time-date auditreduce command options, 52
- tmpfile string and audit_warn script, 18
- tokens, *see* audit tokens,
- trail policy flag, 36
- trail, *see* audit trail,
- trailer token
 - described, 44, 83
 - fields, 44
 - format, 83
 - order in audit record, 42, 83
 - praudit display, 44
- Trusted Computer System Evaluation Criteria (TCSEC) C2 features, 55

U

- U option
 - allocate command, 57
- uadmin audit record, 172
- UDP address, 47, 78
- umount: old version audit record, 151
- unlink audit record, 152

- user audit fields, 13, 14
- user ID (audit ID), 6, 15, 40
- user-level events
 - See also* audit events,
 - audit records, 156, 172
 - audit tokens, 42
 - auditconfig command options, 35
 - described, 7
 - /usr/sbin/uadmin audit record, 172
 - /usr/bin/at audit record
 - at-create crontab, 159, 160
 - /usr/bin/crontab audit record
 - crontab-crontab created, 161, 162
 - /usr/bin/login audit record
 - rlogin, 165, 166
 - /usr/bin/passwd: change password audit record, 168
 - /usr/bin/su audit record, 171
 - /usr/lib/nfs/mountd audit record
 - NFS mount request, 167
 - /usr/sbin/allocate audit record
 - device allocate success, 156 to 159
 - /usr/sbin/auditd daemon, *see* audit daemon,
 - /usr/sbin/halt audit record, 163
 - /usr/sbin/in.ftpd audit record, 164
 - /usr/sbin/in.rexecd audit record, 170
 - /usr/sbin/in.rshd audit record, 170
 - /usr/sbin/inetd audit record, 163
 - /usr/sbin/init audit record, 164
 - /usr/sbin/poweroff audit record, 168
 - /usr/sbin/reboot audit record, 169
 - /usr/sbin/rpc.rexd audit record, 169
 - /usr/sbin/shutdown audit record, 164
 - /usr/ucb/shutdown audit record, 171
- utilities
 - device allocation, 57, 58
- utime audit record, 152
- utimes audit record, 153
- utssys - fusers audit record, 153

V

- vfork audit record, 154
- viewing, *see* displaying,
- vnode token, 45, 73
- vtrace audit record, 154

W

writing new device-clean scripts, 63

X

xmknod audit record, 155

xstat audit record, 155

Xylogics tape drive clean script, 60