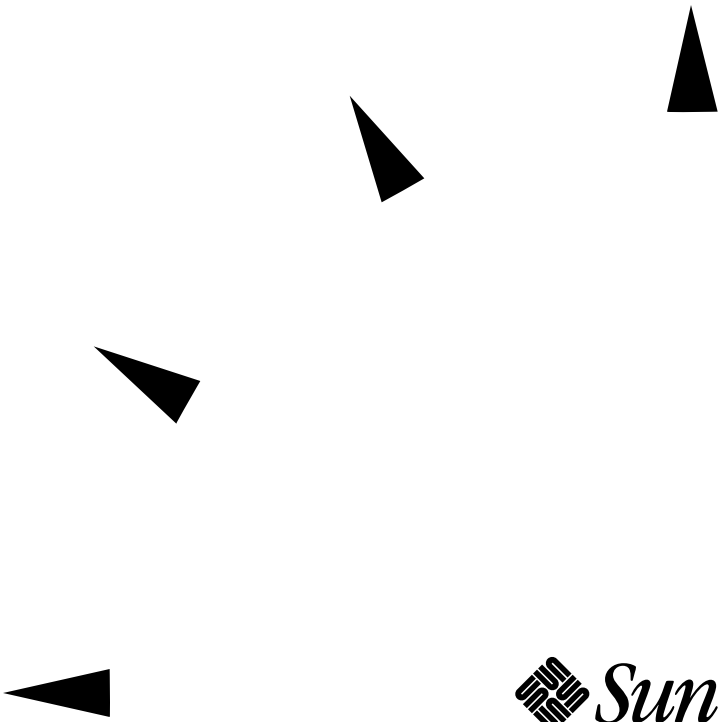
Trusted Solaris Audit Administration



Sun Microsystems Federal, Inc.
A Sun Microsystems, Inc. Business
901 San Antonio Road
Palo Alto, CA 94303
U.S.A.

Part No. 805-8010-10
Revision A, July 1997

THE NETWORK IS THE COMPUTER™

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

Auditing is a security feature required for a C2 rating in TCSEC. C2 discretionary-access control and identification and authentication features are provided by the standard Solaris system. The Trusted Solaris 2.5 operating system is in evaluation for a B1+ Trusted Computer System Evaluation Criteria (TCSEC) evaluation from the U.S. National Security Agency, and an ITSEC evaluation in the United Kingdom of assurance level E3 and functionality F-B1.

Who Should Use This Book

Trusted Solaris Audit Administration is intended for the system administrator whose duties include setting up and maintaining auditing file systems, and for the security administrator whose duties include determining what will be audited and analyzing the auditing trail. The system administrator should be familiar with file system administration, such as NFS-mounting, sharing directories, exporting directories, and creating disk partitions. The security administrator should be familiar with the site security policy, and with the help of the system administrator, be able to create and modify shell scripts.

How This Book Is Organized

Chapter 1, "Auditing Basics," 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 2, "Auditing Setup," covers setting up and maintaining auditing at your site. The latter part of the chapter contains procedures for setting up and maintaining auditing.

Chapter 3, "Audit Trail Management and Analysis," describes how the audit daemon creates the audit trail, and how to manage audit files and read the contents. The latter part of the chapter contains procedures for merging audit files, selecting records, and reading the audit trail.

Chapter 4, "Troubleshooting Auditing," contains procedures for troubleshooting the auditing subsystem.

Appendix A, "Event-to-Class Mappings," lists audit events by their default audit class and alphabetically. It also connects them to their system calls and user commands.

Appendix B, "Audit Record Descriptions," describes in detail the content of the audit records generated, including a description of every audit token.

Appendix C, "Audit Reference," lists and describes the man pages added for the auditing subsystem in Trusted Solaris and file protections on the auditing subsystem.

Appendix D, "Auditing Checklist and Worksheets," supplies an auditing worksheet to help you lay out the auditing network at your site.

Related Books

All sites should have the following books or information available when setting up auditing:

From Sun Microsystems

- *Trusted Solaris 2.5 Release Notes*
  Describes any late-breaking news about auditing including known problems.
- *Trusted Solaris Administrator's Procedures*
  Describes administration tasks, such as assuming a role, in detail.

From Elsewhere

- *Your site security policy*
  Describes the security policy and security procedures at your site.

Other books on auditing that might be of interest include:

- *A Guide to Understanding Audit in Trusted Systems*
- *Auditing in a UNIX System*
- *DoD Trusted Computer System Evaluation Criteria (the Orange Book)*
- *Compartmented Mode Workstation Evaluation Criteria*
- *Guideline for Trusted Facility Management and Audit, Virgil D. Gligor, 1985*

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase document 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 `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

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

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell root prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
secadmin prompt	`secadmin$`
admin prompt	`admin$`

Auditing Basics 1

This chapter explains how auditing works on one workstation and on a network of Trusted Solaris workstations.

Auditing makes it possible to:

- Monitor security-relevant events that take place on a workstation
- Record the events in a network-wide audit trail
- Detect misuse or unauthorized activity (by analyzing the audit trail)
- Review patterns of access, and see the access histories of individuals and objects
- Discover attempts to bypass the protection mechanisms
- Discover extended use of privilege that occurs when a user assumes an administrative role
- Supply additional assurance that attempts to bypass protection mechanisms are recorded and discovered

Auditing may serve as a deterrent: if users know that their actions are likely to be audited, they may be less likely to attempt malicious activities.

1

Auditing Overview

Auditing is included in the full release and is part of the release media of the Trusted Solaris operating system. By default, auditing is enabled. It can be disabled by running a simple script. All of the auditing software is included in the initial system installation in the following packages:

- `SUNWcar` – Core architecture
- `SUNWcsr` – Core SPARC
- `SUNWcsu` – Core SPARC
- `SUNWtsolr` – Trusted Solaris policy
- `SUNWtsolu` – Trusted Solaris policy
- `SUNWhea` – Header files
- `SUNWman` – On-line manual pages

Auditing in Trusted Solaris is enabled by default, configurable by the system and security administrators, and extensible. By default, audit records are stored in *`workstation_name`*`:/var/audit/`. Events in the audit classes `login_logout` and `non-attributable` are audited for the root user.

The system administrator can provide dedicated partitions for audit records. The audit analyst can collect all records from all workstations in a Trusted Solaris network into one audit trail. The auditing records from a network of workstations can be viewed as one large file. Record selection using a variety of criteria is possible.

After audit data is collected into one audit trail, selection (called *post-selection*) and interpretation tools enable the audit reviewer to examine specific parts of the audit trail. For example, records can be selected for individual users or groups, for a host name, for a certain type of event on a specific day, or for a time of day.

To simplify audit administration, Trusted Solaris auditing provides classes of auditable events. When the security administrator specifies a class of events to be audited, all events in that class are audited. User commands or kernel system calls are auditable events. Classes of events to be audited can be specified per workstation. Specific users (like `root`, for example) can be audited specially.

The security administrator can modify and extend the provided event - class mappings. For some events, event details that are not required by site security policy can be omitted from the audit record. Audit classes can be audited for

failure, for success, or for both per workstation. Selecting which activities to monitor is called *pre-selection*. When the auditing subsystem encounters error conditions, the security administrator can specify whom to email with the information.

In the Trusted Solaris auditing subsystem, audit records are protected from snooping by the sensitivity label `admin_high`. Audit configuration files are accessible by the appropriate administrative role only, and sending records to the audit queue requires privilege. A special privilege, `proc_audit_appl`, is provided for ISVs and integrators to add their applications' audit records to the audit queue. Audit event numbers from 32768 to 65535 are available for third-party trusted applications.

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 when users change identity, for example, by assuming an administrative role, all of their actions are tracked with the same audit ID.

The rest of this chapter describes the auditing subsystem. Chapter 2, "Auditing Setup" describes how to set up and administer auditing. The latter part of the chapter contains setup and maintenance procedures. Chapter 3, "Audit Trail Management and Analysis", describes the audit trail, how to manage its files, and how to read them. The latter part of the chapter contains typical procedures for managing and analyzing the audit trail.

The Audit Mechanism

Auditing is enabled by an audit daemon that uses six configurable audit files: `audit_event(4TSOL)`, `audit_class(4TSOL)`, `audit_control(4TSOL)`, `audit_user(4TSOL)`, `audit_startup(1MTSOL)`, and `audit_warn(1MTSOL)`. These files are in the `/etc/security` directory and determine what to audit, where to put the audit logs, and what to do when there is trouble. By default, events in the `lo` (login/logout) audit class are audited for the `root` role, the audit records are written to the `/var/audit` directory, and no one receives mail when there is trouble.

You can suspend and re-enable auditing without rebooting the workstation, and you can dynamically change what is being audited.

1

Audit Startup

Auditing is enabled when the audit daemon starts, usually when the workstation is booted (see the `auditd(1MTSOL)` man page). When troubleshooting, the daemon can be started manually by executing `/usr/sbin/auditd` in an `admin_high` shell in the `root` role.

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(1MTSOL)` man page). A default `audit_startup` script that automatically configures the event-to-class mappings and sets the audit policies is created during audit package installation.

The security administrator can edit the `audit_startup` script to alter the default audit policy. See “Setting Audit Policies” on page 25 for more information on audit policy.

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(4TSOL)` 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. Administrators name an audit class (called an *audit flag*) when specifying for the audit daemon what is to be audited. 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 are made in the `audit_event` file; new classes are added to the `audit_class` file.

Whether an auditable event is recorded in the audit trail depends on whether the administrator preselects an audit class that includes the specific event.

Audit Classes

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 global classes, `all` and `no`. `all` is a conjunction of all allowed classes; `no` is the invalid class. Events mapped to the `no` 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
0x00000080:pc:process
0x00000100:nt:network
0x00000200:ip:ipc
0x00000400:na:non-attribute
0x00000800:ad:administrative
0x00001000:lo:login or logout
0x00004000:ap:application
0x20000000:io:ioctl
0x40000000:ex:exec
0x80000000:ot:other
0xffffffff:all:all classes
```

If the `no` class is actually turned on for auditing, the audit trail fills up with records for the audit event `AUE_NULL`.

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.

Within kernel events there are two pseudo-events,

- AUE_MAC audits MAC decisions
- AUE_UPRIV audits use-of-privilege decisions

If either of these pseudo-events are preselected, audit information is collected internally even if the underlying kernel event is not selected. For example, if the kernel event AUE_OPEN_R is not selected for auditing but the pseudo-event AUE_MAC or AUE_UPRIV is enabled, the kernel event AUE_OPEN_R will be written to the audit trail if a MAC decision or use-of-privilege decision was part of the system call AUE_OPEN_R.

User-Level Events

Events generated by trusted application software outside the kernel range from 2048 to 65535. The event names begin with AUE_, followed by a lowercase mnemonic for the event. The file `/etc/security/audit_event` lists individual events in numerical order. See also “Audit Events Listed by Audit Class” on page 94 and “Audit Events in Alphabetical Order” on page 112. Table 1-1 shows general categories of user-related events.

Table 1-1 Audit Event Categories

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

Non-attributable Events

Events that are not attributable to a user, such as AUE_ENTERPROM.

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 and the audit policy. The audit record descriptions in Appendix B list all the audit tokens defined for each event and what each token means.

Audit tokens construct audit records in an audit file. An audit trail is one or more audit files in a distributed system. The construction of the audit trail is shown in Figure 1-1 on page 7. The audit trail may be converted to a human readable format by `praudit` (see the `praudit(1MTSOL)`man page). Specific audit records can be selectively chosen using the `auditreduce(1MTSOL)` command. See Chapter 3, “Audit Trail Management and Analysis”, for details.

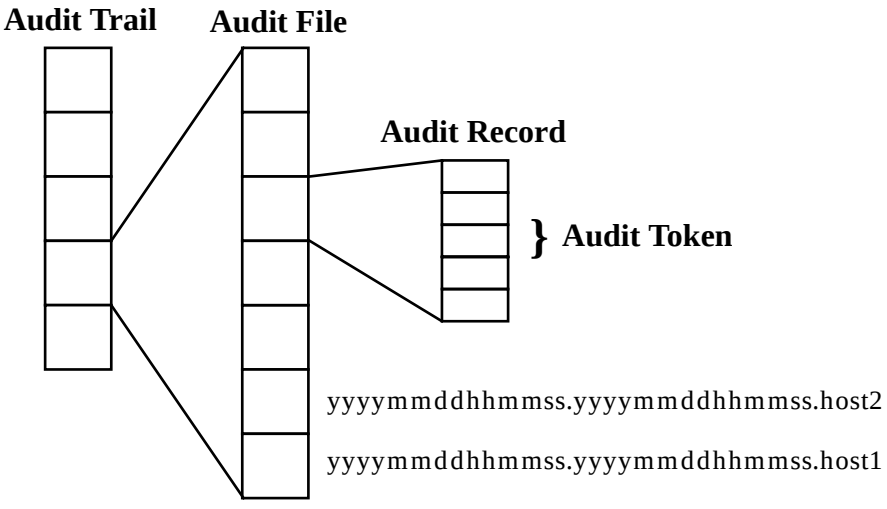


Figure 1-1 From the Audit Token to the Audit Trail

1

Audit Flags

Audit *flags* are the short names for the audit classes. Audit flags are used to indicate which classes to audit in the `audit_control(4TSOL)` file, the `audit_user(4TSOL)` file, and as arguments to the `auditconfig(1MTSOL)` command.

The `audit_control(4TSOL)`file is described in “Auditing a Workstation” on page 12. The `audit_control(4TSOL)` file is described in “The audit_user File” on page 14.

Definitions of Audit Flags

Each predefined audit class is shown in Table 1-2 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(4tsol).:

Table 1-2 Audit Classes

Short Name	Long Name	Description
`no`	`no_class`	Null value for turning off event preselection Events in this class are not audited.
`na`	`non_attrib`	Nonattributable events
`ad`	`administrative`	Administrative actions
`fr`	`file_read`	Read of data, open for reading, etc.
`fw`	`file_write`	Write of data, open for writing, etc.
`fa`	`file_attr_acc`	Access of object attributes: `stat`, `pathconf`, etc.
`fm`	`file_attr_mod`	Change of object attributes: `chown`, `flock`, etc.
`fc`	`file_creation`	Creation of object
`fd`	`file_deletion`	Deletion of object
`cl`	`file_close`	System call `close`(2tsol)

Table 1-2 Audit Classes *(Continued)*

Short Name	Long Name	Description
ex	exec	Program execution
io	ioctl	`ioctl` system call
lo	login_logout	Login and logout events, including telnet, rlogin
pc	process	Process operations: `fork`, `exit`, etc.
nt	network	Network events: `bind`, `connect`, `accept`, etc.
ip	ipc	System V IPC operations
ap	application	Application-defined event
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
-lo         # audit for failure
+lo         # audit for success
lo          # audit for success and failure
```

The audit flag `+lo` means “all successful attempts to log in and log out”. The audit flag `-lo` means “all failed attempts to log in”. (You cannot fail an attempt to logout.). The audit flag `lo` means “all successful attempts to log in and log out and all failed attempts to log in”.

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.

Table 1-3 shows prefixes that specify whether the audit class is audited for success or failure or both.

Table 1-3 Prefixes Used in Audit Flags

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

Prefixes to Modify Previously Set Audit Flags

Use the modify prefixes in any of three ways: in the `flags` line in the `audit_control(4TSOL)` file to modify already-specified flags, as flags in the user's entry in the `audit_user(4TSOL)`file, or as arguments to the `auditconfig(1MTSOL)` command.

The prefixes in Table 1-4 along with audit flags, turn on or turn off previously specified audit classes. These prefixes turn on or off previously specified flags only.

Table 1-4 Prefixes Used to Modify Already-Specified Audit Flags

Prefix	Definition
^-	Turn off for failed attempts
^+	Turn off for successful attempts
^	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.

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

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."

Audit Storage

On every workstation, the /etc/security/audit directory contains subdirectories with all the audit log files. The /etc/security directory contains files related to audit configuration. Because the /etc/security directory contains the per-workstation audit_data file, which is used by the audit daemon at boot time, the /etc/security directory must be part of the root file system.

The audit postselection 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 file server audubon are /etc/security/audit/audubon and /etc/security/audit/audubon.1.

Each audit file system has a subdirectory named files. This files subdirectory is where the audit files are located and where the auditreduce commands looks for them. For example, the audit file system on audit server audubon has a files subdirectory whose full path name is: /etc/security/audit/audubon/files.

The local audit_control file on each workstation directs the audit daemon to put the audit files in the files subdirectory. For example, the dir: line for the audit_control file on a workstation mounting the audit file system from eagle is:

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

The extra level of hierarchy prevents a workstation'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 the clients use the same naming convention for their local audit log files, `/etc/security/audit/`*client-name*, audit files cannot be created unintentionally in the local mount-point directory if the mount fails.

Permissions on Audit Directories

Audit directory permissions on the `/etc/security/audit/`*workstation-name* directory and the `files` directory directly beneath it are shown in Table 1-5.

Table 1-5 Audit Directory and File Permissions

Owner	Group	Permissions
audit	audit	750

Auditing a Workstation

Auditing is set per workstation by the security administrator in the file `audit_control`. This file on each workstation is read by the audit daemon (see the `audit_control(4TSOL)` man page). The `audit_control` file is located in the `/etc/security` directory.

A separate `audit_control` file is maintained on each workstation because the `dir:` lines, and perhaps the `minfree:` line are specific to the workstation. In a distributed system, the other lines should be identical.

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 workstation. 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 for Storing Audit Data” on page 18.

  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 workstation 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 workstation, the second is the secondary audit directory where the audit daemon puts audit trail files when the first one fills, and so forth.

The security administrator modifies the default `audit_control` file during the configuration process on each workstation.

After the `audit_control` file is configured, the security administrator on a distributed system distributes it to the other workstations. After any change in the file, the administrator runs `audit -s` on every workstation on the network to instruct the audit daemon to reread its `audit_control` file.

Note – The `audit -s` command does not change the preselection mask for existing processes. Use `auditconfig`, `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 workstation `willet`. `willet` uses two audit file systems on the audit server `egret`, and a third audit file system mounted from the audit administration server `audubon`, which is used to store audit records only when the audit file system on `egret` fills up or is unavailable. The `minfree` value of 20 percent specifies that the warning script (see the `audit_warn(1MTSOL)` man page) is run when the file systems are 80 percent filled and the audit data for the current workstation 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/egret/files
dir:/etc/security/audit/egret.1/files
#
# Audit filesystem used when egret fills up
#
dir:/etc/security/audit/audubon
```

Auditing User Exceptions

The security 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 add user entries to the `audit_user` file. 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.

Note – Alterations to a static auditing database (`audit_control`, `audit_user`, `audit_startup`, or `audit_warn`) on one workstation should be copied to all workstations on the network. See "To Distribute Audit Configuration Files to a Network of Workstations" on page 53.

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

The `audit_user` *File*

If it is desirable to audit some users differently from others, the administrator can 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.

So, what is audited for an individual user is the combination of the workstation audit flags and the user's always and never audit flags, as shown below.

```
What is audited = (Workstation Audit Flags + User's Always Audit) - User's Never Audit
```

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, 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 `katya` 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 `katya`. Here are two possible `audit_user` entries.

The correct entry:

```
katya:all,^+fr:
```

The incorrect entry:

```
katya: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 is 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.

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 workstation at a time, since the audit command only applies to the current workstation where you are logged in.

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.

Process Audit Characteristics

The following audit characteristics are set at initial login:

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

Process Preselection Mask

When a user logs in, `login` combines the workstation-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 that 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 will be 0.

The `audit_data` *File*

When `auditd` starts on each workstation, 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/egret.1/files/19910320100002.not_terminated.lazy
```

1

The Audit Daemon's Role

The following list summarizes what the audit daemon, `auditd(1MTSOL)`, 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. Your site should customize `audit_warn` to suit your needs. The `audit_warn` script is described in "The audit_warn Script" on page 19.
- With the system default configuration, when all audit directories are full, processes that generate audit records are suspended and `auditd` writes a message to the console and to the `audit_warn` alias. (The auditing policy can be reconfigured with `auditconfig`.) At this point only the system administrator could 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 workstation 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 and uses those 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.

What Makes a Directory Suitable for Storing Audit Data

A directory is *suitable* for storing audit records 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(1MTSOL)` 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 auditing.

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 that 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.

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

The cost in 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.

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.

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

- Number of users
- Number of workstations
- 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 quickly. 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, not auditing the `fr` class 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.

Auditing Efficiently

What to audit, when to audit it, and where to store the files are factors to consider when enforcing your site's security goals while auditing more efficiently. For example, you might try:

- 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 workstations when suspicious events occur.) See "To Read a Current Audit File" on page 77 for an example.
- Setting the public object flag on publicly accessible files or directories. This reduces the potential size of the audit trail while not compromising security, because the viewing of publicly accessible files and directories is not generally interesting for audit purposes. Files so marked do not generate audit records for the following audit events, even if the classes for those events are turned on for auditing:
  AUE_ACCESS, AUE_STAT, AUE_LSTAT, AUE_READLINK, AUE_STATFS, AUE_FSTATFS, AUE_PATHCONF, AUE_OPEN_R, AUE_FGETCMWLABEL, AUE_GETCMWFSRANGE, AUE_GETCMWLABEL, AUE_GETFILEPRIV, AUE_LGETCMWLABEL, AUE_GETMLDADORN, AUE_GETSLDNAME, AUE_MLDLSTAT, AUE_MLDSTAT, AUE_OSTAT, AUE_FUSERS, AUE_STATVFS, AUE_XSTAT, and AUE_LXSTAT.

  See "To Set Public Object Bit on Publicly Accessible Files" on page 59 for the procedure.

- Reducing the disk-storage requirements for audit files by combining, reducing, and compressing them (see “To Combine Selected Audit Files” on page 80), and developing procedures for storing them offline.

The `auditconfig` *Command*

The `auditconfig` command provides a command line interface to get and set audit configuration information and audit policy. It can be used in the `audit_startup(1MTSOL)` script to set audit policies when the audit daemon is started. See the `auditconfig(1MTSOL)` man page and “Dynamic Procedures” on page 60, for examples of the use of the `auditconfig` command.

`-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 workstation’s auditing condition. Table 1-6 shows the possible responses.

Table 1-6 Possible Auditing Conditions

Response	Meaning
`auditing`	Auditing is enabled and turned on.
`no audit`	Auditing is enabled but turned off.
`disabled`	The audit module is not enabled.

`-setcond` *condition*

Set the workstation’s auditing condition: `auditing` or `noaudit`. To disable auditing, run the `bsmunconv` script. See “To Disable Auditing” on page 39 for the procedure.

`-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.

`-setkmask` *+/-audit_flags*

Set the kernel preselection mask for non-attributable events to the specified audit flags.

`-setkmaskac`

Set the kernel preselection mask for non-attributable events to the classes specified in the *naflags* field of the `audit_control` file.

`-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` *auid 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” below.

Setting Audit Policies

You can use `auditconfig` with the `-setpolicy` flag to change the default Trusted Solaris audit policies. Setting audit policies means to add optional audit tokens to the audit record. The `auditconfig` command with the `-lspolicy` argument shows the audit policies that are optional. See “To Determine Current Audit Policy” on page 61 for the audit policies and their short descriptions. The following gives longer descriptions of the less easily understood policy flags.

`ahlt`

Halt the machine if an asynchronous audit event occurs which can not be delivered to the audit queue. The default is not to halt the workstation.

`cnt`

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

Note – To return to the default, remove the `cnt` policy. See “To Create an Admin_High Workspace” on page 61 for examples of replacing, adding, and removing audit policies.

`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.

`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.)

1

Auditing Setup 2

The focus of this chapter is on setting up auditing for a network of Trusted Solaris workstations. It also describes how to set up auditing for a non-networked Trusted Solaris workstation.

Planning Auditing at Your Site

When the system administrator and security administrator configure the first workstation for Trusted Solaris, auditing is enabled and a limited number of audit records are collected to a default audit location, *workstation_name*:/var/audit. The security administrator needs to plan what to audit and whether to customize site-specific event-to-class mappings. The system administrator plans disk space (local and remote) for the audit files, an audit administration server, and the order of installation.

Planning auditing for a non-networked workstation is a bit simpler. For a single workstation, customizing event-to-class mappings may not be worth the time. Your most important task is to ensure that auditing does not slow down your work. Planning the size and locations of auditing partitions can prevent work slowdown, and a regular maintenance schedule can automatically back up and free up the audit partition for more audit records.

Planning What to Audit

Trusted Solaris collects user actions and non-attributable events into audit classes. It is these audit classes, each of which holds a number of events, that are audited for success, for failure, or for both.

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.

See “Audit Events Listed by Audit Class” on page 94 for a list of provided audit classes. “Audit Events in Alphabetical Order” on page 112 lists all audit events defined in Trusted Solaris alphabetically. Both lists show each audit event’s corresponding system call or user command, and point you to its audit record format.

The security administrator plans what to audit based on the site security policy. You can configure a system-wide setup and user exceptions/additions.

1. **Decide if non-attributable events should be audited.**
   The audit flag `na` represents the non-attributable class of events. For example, accessing the PROM, booting, and remote mounting are non-attributable events. See "Events in Audit Class na" on page 107 for a list of the events in the default `non-attributable` class.

   When you audit a class, you audit all events in that class. If you want to customize the non-attributable class, see "Planning a Site-Specific Event-to-Class Mapping" on page 31.

   To audit non-attributable events, you will enter the `na` flag on the `naflags:` line of the `audit_control` file.

2. **Decide whether to audit them for success, for failure, or for both.**
   To audit non-attributable events for success, the `naflags:` line of the `audit_control` file would look like:

```
naflags:+na
```

   To audit non-attributable events for failure:

```
naflags:-na
```

   To audit non-attributable events for both:

```
naflags:na
```

3. **Decide if `all` events will be audited.**

Note – The class `all` includes all auditable events in the Trusted Solaris software system. While unusual circumstances may dictate use of this class, typically you would avoid auditing all events.

---

4. **If you are not going to audit all events, repeat Step 1 and Step 2 for the other audit classes for the class** `na`**.**
   The decisions you have made in Step 3 and Step 4 you will enter in the `audit_control` file when establishing auditing on the first workstation.

   To audit these events, you will enter their flag on the `flags:` line of the `audit_control` file, just as you entered the `na` flag in the `naflags:` line.

5. **Determine if there are particular users or roles that should be audited slightly differently than the system-wide setup.**
   You will enter user exceptions to the system setup in the `audit_user` file.

6. **Be consistent.**
   All workstations in a Trusted Solaris network should have identical `naflags:` entries in their `audit_control` files.

   All workstations in a Trusted Solaris network should have identical `flags:` entries in their `audit_control` files.

   All workstations in a Trusted Solaris network should have identical `audit_user` files.

Considerations When Planning What to Audit

What is audited at your site is based on your site policy and the costs of auditing (time, efficiency, disk space). For a discussion of the costs, see “Controlling Audit Costs” on page 21. The following are factors to consider when using auditing as it is implemented in Trusted Solaris.

- Every audit record in Trusted Solaris stands alone, so records can quickly fill up disk space.
  Therefore, you might want to start with a small amount of auditing and see how the audit partitions fill. You can then make more educated estimates of disk requirements and an audit archiving schedule. You can refine audit classes as you get an estimate of the size of the audit trail.

- The number of events in an audit class does not necessarily correlate to how many records are generated.
  For example, the `file read` class contains about the same number of events as the `login or logout` class. Enabling the `file read` class for success is likely to generate many more records than enabling the `login or logout` class for success.

- Auditing for failure locates abnormal events; auditing for success monitors system use.
  If site policy requires monitoring of system use, you will want to set aside more space for the audit trail than if you are auditing for abnormal events.
- Auditing for failure may generate many fewer records than auditing for success.
  For example, auditing for failure of `file read` events in a Trusted Solaris system of sophisticated users can generate many fewer records than turning on the `file read` class for success.
- Configuring the audit classes differently, or setting up new audit classes for audit events can more efficiently satisfy your site requirements. By excluding audit events that site policy does not require to be audited, the audit trail is smaller.
  For example, you may want to create a class `de` for devices. When configuring devices, audit the class for success to generate a record of what devices have been set up and tested. When all devices have been configured, you may want to audit the class for failure.
- Configuring some classes to be audited intermittently may satisfy your site requirements.
  For example, you may want to audit the audit class you created, `de`, intermittently. A cron job, or the command `auditconfig(1MTSOL)`, enable you to turn auditing on and off for particular classes and set other audit flags dynamically.

Planning a Site-Specific Event-to-Class Mapping

Optional: Skip this section if you are using the default event-to-class mappings provided by Trusted Solaris. Do not skip this section if you have decided to rearrange what events are assigned to what classes, or to create new classes or new events.

Trusted Solaris allows 32 audit classes. Nineteen (19) are defined, including the class `all`. Your site may therefore define up to 13 classes.

The security administrator plans site-specific mappings. To plan site-specific mappings:

1. **Decide what classes are needed.**

2. **Decide what events belong in what classes.**

   a. **Decide what events should be copied to another class or classes.**
   An audit event can belong to more than one class. For example, the audit event AUE_RENAME belongs to the classes `file create` and `file delete` in the default event-to-class mapping.

   b. **Decide what events should be moved to another class or classes.**

   c. **Decide what events should be added to a class or classes.**

3. **For each class, decide whether to audit it for success, for failure, or for both.**

When new software programs include audit events not provided by Trusted Solaris 2.5 software, add the events to existing classes or create a new classes for the new events.

Considerations When Changing Event-to-Class Mappings

The following are factors to consider when changing the contents of default audit classes and creating new ones in Trusted Solaris.

- This document, *Trusted Solaris Audit Administration*, reports the default auditing configuration.
  Document your site's modifications to the auditing defaults, and make the document available to the administrators handling audit administration.
- If you are networked, you must change the auditing configuration files on all the workstations when you change the files on one workstation.
  A network of Trusted Solaris workstations behaves like one workstation. When auditing is enabled, it is enabled on every workstation, and every workstation is audited for the same classes, has the same defaults, has the same user exceptions, and has the same event-to-class mappings as every other Trusted Solaris workstation in the system.

Planning Space for Audit Records

Storing audit records on a non-networked workstation involves setting up at least two local partitions dedicated to audit records, one primary and one backup, and planning a maintenance schedule.

Storing audit records for a network of workstations involves setting up a local (backup) partition dedicated to audit records, plus a network of audit servers with partitions for remote (primary) audit storage, and plus an audit administration server from which the entire *audit trail* can be monitored. The audit trail is every audit file (audit files hold audit records generated on a workstation) created by every workstation on the network.

Planning Space on a Non-Networked Workstation

On a non-networked workstation, plan the size of a disk partition to hold audit records. For efficiency, it is best to place the audit records on a separate disk. For safety, you may want to create two audit partitions on that disk, one as the primary storage area and the other as a backup if the first partition gets full. Set filesystem security attributes to set on the audit directory to prevent snooping on the audit trail.

1. **Estimate the volume of auditing between audit record backups.**
   Balance your security needs against the availability of disk space for audit trail storage.

   A rule of thumb is to assign 100 MB of space per workstation. However, the disk space requirements for the workstation are based on how much auditing you perform and may be far greater than this figure.

   “Controlling Audit Costs” on page 21 and “Auditing Efficiently” on page 22 provide guidance on how to reduce storage requirements.

2. **Decide at what point the audit filesystem sends a warning that it is filling up.**
   You will specify what is called the *minfree limit* for audit partitions in the `audit_control` file. This is the percentage of disk space remaining when the audit administrator is sent an email message (by the audit_warn alias you set up in the `audit_warn` file) that the disk is getting full. The default is to send the warning when there is 20% disk space remaining. This percentage is tunable.

Planning Space on a Network of Workstations

A networked system should include audit servers to store audit files for users’ workstations, an audit administration server for central audit analysis and backup, and a local audit partition on every workstation. You may want to set

filesystem security attributes on the directories and mount points to prevent snooping on the audit trail. Use the worksheet in Appendix D, "Auditing Checklist and Worksheets" to record your auditing plan, or use another mechanism that helps you track the auditing network that you set up.

1. **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.

   A rule of thumb is to assign 100 MB of space for each workstation that will be on the distributed system, but remember that the disk space requirements at your site is based on how much auditing you perform and may be far greater than this figure per workstation. If you are able to dedicate a local and a remote disk for auditing, one way to set up audit partitions is to divide each disk into two partitions.

   "Controlling Audit Costs" on page 21 and "Auditing Efficiently" on page 22 provide guidance on how to reduce storage requirements while still maintaining site security.

2. **Decide at what point each audit filesystem for the workstation sends a warning that it is filling up.**
   You will specify what is called the *minfree limit* for audit partitions in the `audit_control` file. This is the percentage of disk space remaining when the audit administrator is sent an email message (by the audit_warn alias you set up in the `audit_warn` file) that the disk is getting full. The default is to send the warning when there is 20% disk space remaining. This percentage is tunable.

3. **Determine which workstations will be audit servers.**
   The system administrator and you will install these workstations before installing the audit client workstations.

4. **Plan a local audit partition for each workstation.**
   The local partition provides a backup in cases where the audit server's partitions are full or when the network is unreachable.

5. **Determine which clients will use which audit filesystems on which audit server.**
   Using the worksheet "Audit Server-Client Network Worksheet" on page 272, lay out the network. Figure 2-1 shows an audit server, `egret`, with file systems `/etc/security/audit/egret`*[.n]*`/files` available to store remote hosts' audit records.

.

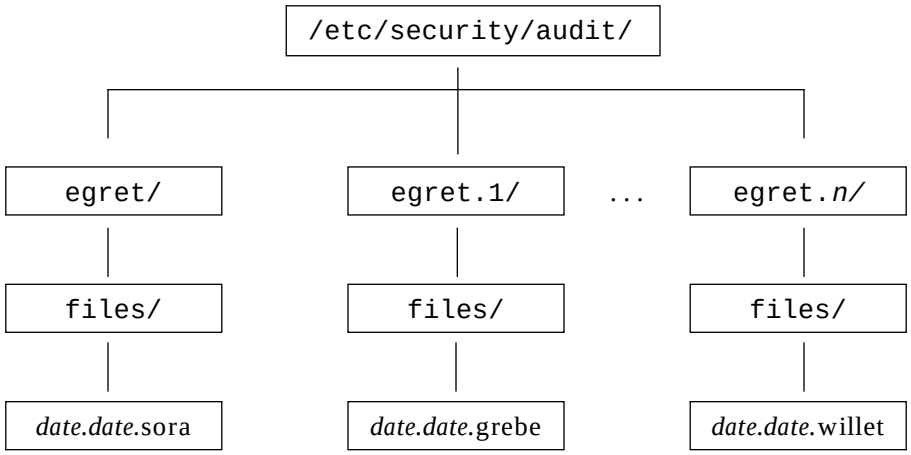


Figure 2-1 Audit Server `egret`'s Audit File Systems

6. **Follow the naming conventions for audit file systems.**
   As illustrated in Figure 2-1, the convention for naming the audit file systems on a workstation is:
   /etc/security/audit/*workstationname*/files
   /etc/security/audit/*workstationname*.1/files
   /etc/security/audit/*workstationname*.2/files
   /etc/security/audit/*workstationname*.3/files ...

   For an explanation of the naming scheme, see "Audit Storage" on page 11.

Planning the Rollout

Rolling out the auditing plan to the workstations is a job coordinated by the system administrator, who sets up the disks and the network of audit storage, and the security administrator, who decides what is to be audited and enters the information in the audit configuration files. Together, you want to set up an audited network of workstations where:

- From one workstation, the audit analyst is able to read every audit file on every workstation in the network, and the system operator is able to back up every audit file on every workstation on the network.

How: Create an administration server, and mount all audit directories on the server.

- The audit trail is not available for snooping.

  How: Protect audit directories with appropriate discretionary access controls and mandatory access controls. You may want to audit directory access.

- Each workstation in a Trusted Solaris system is writing records to the audit trail from the first time it is in multiuser mode, and thereafter.

  How: Create audit servers before you create user workstations. On all workstations, create a dedicated audit partition during installation.

- Every workstation is audited identically.

  How: Create a central location for all audit configuration files: `audit_event`, `audit_class`, `audit_control`, `audit_startup`, `audit_user`, and `audit_warn`. The examples use the directory `/export/home/tmp` on the NIS+ master. Copy these files to a tape or diskette that is copied to every workstation.

- When an end user's workstation is configured, it is able to immediately send its audit records to an audit server.

  How: Create the audit servers and configure them for receiving audit records before the end user workstations are set up. Create a procedure to copy the system-wide audit configuration files to each workstation and to modify the `audit_control` file for the audit storage locations for that workstation.

- End user's workstations are not slowed down by writing audit records.

  How: Regular archiving of the audit trail frees up audit server disk space. Placing the local audit storage on a separate or little-used disk will enable the end user to work quickly when audit records are stored locally.

Rolling Out Auditing at Your Site

To roll out auditing, the system administrator sets up the audit administration server, the audit file servers, the local audit partitions, and what usernames are warned of audit trouble. The security administrator edits the `audit_control` file(4tsol) on the NIS+ root master, and edits other audit configuration files before copying them to a central directory for distribution by tape or floppy. The audit configuration files are copied from the tape to each workstation as it

is configured by the install team. The security administrator edits the `dir:` lines in the `audit_control` file on each workstation before the system is rebooted.

System Administrator's Audit Setup Tasks

Table 2-1 Basic Auditing Setup by the System Administrator

Task	For the procedure, see...
Create audit partitions	"To Create Dedicated Audit Partitions" on page 41
Create audit administration server	*Trusted Solaris Installation and Configuration* or *Trusted Solaris Administrator's Procedures*
Install audit file servers	Plan to install them before audit clients
Create files directory	"To Create an Audit Directory" on page 46
Export audit partitions (networks only)	"To Share an Audit File System" on page 47
Edit Aliases database	"To Warn of Audit Trouble" on page 52
Mount audit partitions (networks only)	"To Mount an Audit File System" on page 47

Security Administrator's Audit Setup Tasks - Basic

Table 2-2 Basic Auditing Setup by the Security Administrator

Task	For the procedure, see...
On first workstation	
Edit audit_control file	"To Set Audit Flags" on page 50 "To Reserve Free Space on an Audit File System" on page 49 "To Specify the Audit File Storage Locations" on page 49
Set Solaris security attributes	"To Protect an Audit File System" on page 45
Edit audit_user file	"To Set User Exceptions to the Audit Flags" on page 51
Edit audit_startup file	"To Set Audit Policy Permanently" on page 52
Copy for distribution (networks only)	"To Distribute Audit Configuration Files to a Network of Workstations" on page 53
Set security attributes	"To Protect an Audit File System" on page 45

Security Administrator's Audit Setup Tasks - Advanced

Table 2-3 Advanced Auditing Setup by the Security Administrator

Task	For the procedure, see...
On first workstation	
Edit audit_event file	"To Add Audit Events" on page 57 "To Change Event-Class Mappings" on page 58
Edit audit_class file	"To Add Audit Classes" on page 56
Copy for distribution (networks only)	"To Distribute Audit Configuration Files to a Network of Workstations" on page 53

Audit Shutdown and Startup Procedures

The following procedures describe how to enable and disable auditing for one or more workstations. The procedures should be performed under the supervision of the security administrator. The commands should be run only on a server or standalone system and never on a diskless client.

▼ To Disable Auditing

1. **Shut down the workstation by choosing Shut Down from the Trusted Path menu off the Front Panel.**

Note – This should be done *only if* auditing is not a site security requirement, or in cases of audit file overflow. The security administrator is responsible.

---

2. **Boot into single-user mode after supplying the PROM password:**

```
Type b (boot), c (continue), or n (new command mode)
> n
PROM Password: <password> <Return>
Type help for more information
ok boot -s
```

3. **Log in as root.**

4. **Change directories to the** `/etc/security/audit` **directory, and execute the** `bsmunconv(1MTSOL)` **script located there. The script disables auditing.**

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

5. **After the script finishes, reboot using the** `exit` **command.**

```
# exit
```

▼ To Enable Auditing

By default, auditing is enabled. If you have disabled auditing, enable it using the following procedure.

1. **Shut down the workstation by choosing Shut Down from the Trusted Path menu off the Front Panel.**

2. **Boot into single-user mode after supplying the PROM password:**

```
Type b (boot), c (continue), or n (new command mode)
> n
PROM Password: <password> <Return>
Type help for more information
ok boot -s
```

3. **Log in as root.**

4. **Change directories to the** `/etc/security/audit` **directory, and execute the** `bsmconv(1MTSOL)` **script located there.** The script sets up a Trusted Solaris workstation to run auditing after a reboot**.**

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

5. **After the script finishes, exit the system using the** `exit` **command.** Auditing is enabled.

```
# exit
```

Basic Audit Setup Procedures

The following procedures describe how to set up auditing for one or more workstations.

Role - secadmin
Label - admin_low
Tool - profile shell

The callouts in the procedures, like the one on the left, indicate what role can do the task or step in the procedure, the label of the task's process, and the tool with which to perform the task. See "To Execute Commands that Require Privilege" on page 42 for how to assume a role and open a privileged shell.

▼ To Create Dedicated Audit Partitions

User - install team
Label - Not Applicable
Tool - installation program

♦ **During installation, create dedicated audit partition(s) when formatting the disks.**
Use the naming convention `/etc/security/audit/`*workstation_name*(.*n*)

A diskfull workstation 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.

See "Audit Storage" on page 11 for an explanation of the naming convention.

On an audit file server, most partitions hold audit files, as is shown in the following example of the `egret` audit file server:

Disk	Slice	Mount point	Size
c0t2d0	**s0**	/etc/security/audit/egret	1.0 GB
	s1	/etc/security/audit/egret.1	.98 GB
	s2	entire disk	1.98 GB
c0t2d1	**s0**	/etc/security/audit/egret.2	502 MB
	s1	/etc/security/audit/egret.3	500 MB
	s2	entire disk	1002 MB

Note – Another disk holds `egret`'s / (root) and `/swap` partitions.

On a diskfull workstation, at least one partition should be dedicated to audit files, as is shown in the following example of the workstation `willet`:

Disk	Slice	Mount point	Size (MB)
c0t3d0	**s0**	/	70
	s1	swap	180
	s2	entire disk	1002
	s3	/usr	350
	s4	/etc/security/audit/willet	202
	s7	/export/home	200

Hints

A rule of thumb is to assign 200 MB of space for each workstation. However, the disk space requirements at your site will be based on how much auditing you perform and may be far greater than this figure.

Fewer and large partitions are more efficient than more and smaller ones.

Note – To add a disk to hold audit partitions after installing the workstation, see the Solaris 2.5 *System Administration Guide, Volume II*.

▼ To Execute Commands that Require Privilege

Most commands for setting up auditing require the use of a profile shell, where commands can run with privilege. Auditing also requires the use of actions in the System_Admin and Solstice_Apps folders of Application Manager.

▼ Log on and assume an administrative role

1. **Log on to the workstation as yourself.**

   a. **Enter your user name and press the Return key.**
   If the workstation is protected against logins, the Enable Logins dialog is displayed.

   b. **If you are allowed to enable logins, click the Yes button after Login:.**
   If you are not allowed to enable logins, ask the administrator to enable logins.

   c. **Enter your password.**

   d. **Click OK to dismiss the dialog.**
   You are presented with the message of the day and a label builder screen. In a single-label system, the screen describes your session label. In a multilabel system, it presents you with a label builder to choose your session clearance.

   e. **Accept the default unless you have a reason not to.**
   Press the Return key or click the OK button and be logged in.

2. **Assume an administrative role that you have been assigned.**

   a. **Click the right mouse button in the middle of the Front Panel.**

a. **Choose Assume *administrative* Role from the menu.**

b. **At the password prompt, enter the password for that role.**

▼ Open a profile shell

Role - ***administrative***
Label - admin_low
Shell - profile shell

1. **Launch a terminal by clicking the left mouse button on the triangle above the pad icon on the Front Panel.**
   A subpanel is displayed that includes an icon of a terminal.
2. **Click the terminal icon once.**
3. **Enter the `clist` command.**

```
# clist
```

   If a list of commands prints, you are in a profile shell.
   If `clist` is not a recognized command,

4. **Open a profile shell by entering the `pfsh(1MTSOL)` command.**

```
# pfsh
```

   The shell is now a profile shell and recognizes the clist command.

Role - root
Label - admin_low
Shell - profile shell
Privileges - all

5. **If you are in the role `root`, you can open another shell using a shell command (`sh`, `ksh`, or `csh`).**

```
# sh
```

   The shell now recognizes all privileges. This is useful for formatting disks, and other tasks that require extensive privilege.

▼ To Remove Free Space (Optional)

Role - admin
Label - admin_low
Tool - profile shell

1. **Unmount the audit partitions from the workstation by running the** `umount(1MTSOL)` **command.**

   For example, on the audit file server `egret`:

```
$ umount /etc/security/audit/egret
$ umount /etc/security/audit/egret.1
$ umount /etc/security/audit/egret.2
$ umount /etc/security/audit/egret.3
```

2. **Reduce reserved filesystem space on each partition to 0% with the command** `tunefs -m 0`**.**
   The security administrator sets the reserved filesystem space (called the minfree limit) in the `audit_control(4TSOL)` file.

   For example, on the audit file server `egret`:

```
$ tunefs -m 0 /etc/security/audit/egret
$ tunefs -m 0 /etc/security/audit/egret.1
$ tunefs -m 0 /etc/security/audit/egret.2
$ tunefs -m 0 /etc/security/audit/egret.3
```

   Similarly, on the workstation `willet`:

```
$ umount /etc/security/audit/willet
$ tunefs -m 0 /etc/security/audit/willet
```

   See the `tunefs(1M)` man page for more information on the advantages and disadvantages of tuning a file system.

▼ To Protect an Audit File System

Role - ***administrative***
Label - admin_low
Tool - profile shell

1. **Set the appropriate file permissions on every audit file system while the file system is unmounted.**
   For example, on the audit file server `egret`:

```
egret$ chmod -R 750 /etc/security/audit/egret
egret$ chmod -R 750 /etc/security/audit/egret.1
egret$ chmod -R 750 /etc/security/audit/egret.2
egret$ chmod -R 750 /etc/security/audit/egret.3
```

   On the workstation `willet`:

```
willet$ chmod -R 750 /etc/security/audit/willet
```

Role - secadmin
Label - admin_high
Tool - profile shell

2. **Set any Trusted Solaris security attribute defaults required by your site security policy on every audit file system while the file system is unmounted.**
   To run the command at the label admin_high, you must create an admin_high workspace. Follow the procedure in "To Create an Admin_High Workspace" on page 61.

   For example, the following command on the audit file server `egret` should be repeated for all of its audit partitions:

```
egret$ setfsattr -l "admin_high;admin_high" -s "admin_high[admin_high]" \
/etc/security/audit/egret
```

   On the workstation `willet`:

```
willet$ setfsattr -l "admin_high;admin_high" -s "admin_high[admin_high]" \
/etc/security/audit/willet
```

The `-l` option ensures that all files created in the filesystem are labeled `admin_high`, and the `-s` option sets the partition's default sensitivity label for the audit files. See the `setfsattr(1MTSOL)` man page for more information.

Note – The local audit file systems must already be in the host's `/etc/vfstab` file.

▼ To Create an Audit Directory

Role - admin
Label - admin_high
Tool - profile shell

1. **In an `admin_high` profile shell, remount the local audit file systems.**
   Follow the procedure in "To Create an Admin_High Workspace" on page 61 to get an `admin_high` process.

   For example, on the audit file server `egret`:

```
egret$ mount /etc/security/audit/egret
egret$ mount /etc/security/audit/egret.1
egret$ mount /etc/security/audit/egret.2
egret$ mount /etc/security/audit/egret.3
```

   Similarly, on the workstation `willet`:

```
willet$ mount /etc/security/audit/willet
```

Role - admin
Label - admin_high
Tool - profile shell

2. **Create a directory named `files` at the top of each mounted audit partition.**
   For example, on the audit file server `egret`:

```
egret$ mkdir /etc/security/audit/egret/files
egret$ mkdir /etc/security/audit/egret.1/files
egret$ mkdir /etc/security/audit/egret.2/files
egret$ mkdir /etc/security/audit/egret.3/files
```

   On the workstation `willet`:

```
willet$ mkdir /etc/security/audit/willet/files
```

▼ To Share an Audit File System

♦ **Enter every local audit file system in the local host's `dfstab(4)` file.**
Share every audit file system.

Role - admin
Label - admin_low
Action - Share Filesystems

a. Click the Application Manager, double-click the System_Admin folder, and double-click the Share Filesystems action.

b. Enter and protect each local audit directory in the `dfstab(4)` file.
For example, the audit file server `egret` has the following entries:

```
share -F nfs -o ro -d "local audit files" /etc/security/audit/egret
share -F nfs -o rw=willet:audubon -d "audit willet" /etc/security/audit/egret.1
share -F nfs -o rw=grebe:audubon -d "audit grebe" /etc/security/audit/egret.2
share -F nfs -o rw=sora:audubon -d "audit sora" /etc/security/audit/egret.3
```

The workstation `willet` has the following entry:

```
share -F nfs -o ro -d "local audit files" /etc/security/audit/willet
```

Role - admin
Label - admin_low
Tool - profile shell

3. Reboot the workstation, or run the commands `unshareall` and `shareall`.

```
audubon# unshareall
audubon# shareall
audubon# showmount -e
```

▼ To Mount an Audit File System

Role - admin
Label - admin_low
Tool - profile shell

1. On `audubon`, the audit administration server, create a mount point for every audit directory in the Trusted Solaris network.
For example, on the audit administration server `audubon`:

```
audubon$ mkdir /etc/security/audit/willet
audubon$ mkdir /etc/security/audit/egret
audubon$ mkdir /etc/security/audit/egret.1
…
```

Role - admin
Label - admin_low
Action - Set Mount Points

2. **Enter every audit partition on the network in the audit administration server's `vfstab(4)` file.**
   Mount audit directories with the read-write (`rw`) option. Mount remote partitions using the `soft` option.

   a. **Click the Application Manager, double-click the System_Admin folder, and double-click the Set Mount Points action.**

   b. **Enter the mount points in the `vfstab(4)` file.**
   The following shows part of the `vfstab` file on audubon:

```
egret:/etc/security/audit/egret - /etc/security/audit/egret nfs - yes bg,soft,nopriv
egret:/etc/security/audit/egret.1 - /etc/security/audit/egret.1 nfs - yes bg,soft,nopriv
egret:/etc/security/audit/egret.2 - /etc/security/audit/egret.2 nfs - yes bg,soft,nopriv
egret:/etc/security/audit/egret.3 - /etc/security/audit/egret.3 nfs - yes bg,soft,nopriv
willet:/etc/security/audit/willet - /etc/security/audit/willet nfs - yes bg,soft,nopriv
…
```

Role - admin
Label - admin_low
Tool - profile shell
Action - Set Mount Points

3. **On each workstation, create the mount points for the remote audit file servers' partitions that are used by the workstation, and enter them in the `vfstab(4)` file.**
   For example, to create the mount points on the workstation `willet`:

```
willet$ mkdir /etc/security/audit/egret
willet$ mkdir /etc/security/audit/audubon.2
```

   a. **Click the Application Manager, double-click the System_Admin folder, and double-click the Set Mount Points action.**

   b. **Enter the mount points in the `vfstab(4)` file.**
   The following shows part of the `vfstab` file on `willet`:

```
egret:/etc/security/audit/egret - /etc/security/audit/egret nfs - yes bg,soft,nopriv
audubon:/etc/security/audit/audubon.2 - /etc/security/audit/audubon.2 nfs - yes nopriv
```

▼ To Reserve Free Space on an Audit File System

Role - secadmin
Label - admin_low
Action - Audit Control

1. **Enter reserve free space in the `audit_control` file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Control action.**
2. **Enter a value between 10 and 20 on the `minfree:` line.**

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

3. **Write the file and exit the editor.**

▼ To Specify the Audit File Storage Locations

Role - secadmin
Label - admin_low
Action - Audit Control

1. **Enter audit storage locations in the `audit_control` file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Control action.**
2. **On the first workstation installed, enter its local audit filesystem as the value of the `dir:` line.**
   The following shows the `audit_control`(4) file for `grebe`, the NIS+ root master.

```
dir:/etc/security/audit/grebe/files
flags:
minfree:20
naflags:
```

3. **When the audit file servers have been installed and configured, add their (mounted) filesystem names plus their top-level directory,** `files` **to the** `dir:` **entry.**
   The mounted file systems are listed before the workstation's local file system, as in:

```
dir:/etc/security/audit/egret/files
dir:/etc/security/audit/egret.1/files
dir:/etc/security/audit/grebe/files
flags:
minfree:20
naflags:
```

4. **Write the file and exit the editor.**

Role - secadmin
Label - admin_high
Shell - profile shell

5. **In an** `admin_high` **profile shell, execute the** `audit -s` **command to have the audit daemon re-read the** `audit_control` **file and write audit records to the designated directory.**:

```
$ audit -s
```

   By default, the audit records have been stored in `/var/audit`. The audit records will now be stored in the first directory in the `audit_control` file.

▼ To Set Audit Flags

Role - secadmin
Label - admin_low
Action - Audit Control

1. **Enter system-wide audit flags in the** `audit_control` **file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Control action.**

2. **Enter the** `na` **class in the** `naflags:` **line if your site is auditing non-attributable events.**

```
dir:/etc/security/audit/egret/files
dir:/etc/security/audit/egret.1/files
dir:/etc/security/audit/grebe/files
flags:
minfree:20
naflags:na
```

3. **Enter other classes in the** `flags:` **line if your workstation is auditing user-level events.**

```
dir:/etc/security/audit/egret/files
dir:/etc/security/audit/egret.1/files
dir:/etc/security/audit/grebe/files
flags:lo,ad,-all,^-fc
minfree:20
naflags:na
```

See "Sample audit_control File" on page 13 for an explanation of the syntax of the audit flags' fields.

4. **Write the file and exit the editor.**

Note – On a distributed system, the audit flags in the `audit_control` file must be identical on every workstation on the network. See "To Distribute Audit Configuration Files to a Network of Workstations" on page 53 for a process to distribute master copies of files to all workstations on the network.

▼ To Set User Exceptions to the Audit Flags

Role - secadmin
Label - admin_low
Action - Audit Users

♦ **Enter user exceptions to system-wide audit flags in the** `audit_user` **file.**

a. **Open the System_Admin folder from the Application Manager.**

b. **Double-click the Audit Users action.**

c. **Enter the user exceptions, write the file, and exit the editor.**

For example, the following entry audits the role `root` for logins and logouts, and never audits the `fc` class, even if it is being audited for the workstation. The `jane` entry audits her for all flags specified in the `audit_control` file except for successful `file_read` events. Null events, `no`, are never audited.

```
# User Level Audit User File
#
# File Format
#
#       username:always:never
#
root:lo:no,fc
jane:all,^+fr:no
```

▼ To Warn of Audit Trouble

Role - ***administrative***
Label - admin_low
Tool - Database Manager

1. **Enter mail alias to warn of audit trouble in the Aliases database.**
   a. **Open the Solstice_Apps folder from the Application Manager.**
   b. **Double-click the Database Manager.**
   c. **Choose NIS+, and Aliases, and press Return.**
2. **Add an alias called `audit_warn` for notifying its members of audit trouble.**
   For example, this `audit_warn` alias emails the security administrator and the system administrator when the auditing subsystem needs attention.

```
Alias: audit_warn
Expansion: secadmin@grebe,admin@grebe
```

▼ To Set Audit Policy Permanently

Role - secadmin
Label - admin_low
Action - Audit Startup

1. **Enter permanent audit policy in the `audit_startup` file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Startup action.**

2. **Create a script that calls the** `auditconfig(1MTSOL)` **command with policy options.**
   The sample `audit_startup(1MTSOL)` script below adds ACLs to audit records, counts audit records when the workstation's audit filesystems are full, and at startup, prints the current audit policy to standard i/o.

```
#!/bin/sh
auditconfig -setpolicy acl
auditconfig -setpolicy +cnt
auditconfig -getpolicy
```

3. **Write the file and exit the editor**

▼ To Distribute Audit Configuration Files to a Network of Workstations

Role - root
Label - admin_low
Tool - regular shell

1. **Create a directory on the first workstation to hold copies of the audit configuration files customized for your site.**
   The directory would include your customized versions of `audit_control`, `audit_user`, `audit_startup`, and `audit_warn`. If you have modified event-to-class mappings, it would include `audit_event` and `audit_class`. It would not include `audit_data`.

   For example, on `grebe`, the first workstation in a network:

```
# mkdir /export/home/tmp
```

2. **Copy the modified files from the** `/etc/security` **directory to the** `/export/home/tmp` **directory.**

```
# cp /etc/security/audit_control /export/home/tmp/audit_control
# cp /etc/security/audit_control /export/home/tmp/audit_user
# cp /etc/security/audit_startup /export/home/tmp/audit_startup
# cp /etc/security/audit_event /export/home/tmp/audit_event
```

3. **Allocate the tape or diskette device.**
   Follow the procedure in "Allocate a device" on page 55.

4. **Run the** `tar(1TSOL)` **command to copy the contents of the** `/export/home/tmp` **directory to tape or to diskette.**

   a. **To copy to tape**

```
# cd /export/home/tmp
# tar cv audit_control audit_user audit_startup audit_event
```

   b. **To copy to diskette**

```
# cd /export/home/tmp
# tar cvf /dev/diskette audit_control audit_user audit_startup audit_event
```

   c. **To eject a diskette**

```
# eject
```

5. **Remove the tape or diskette.**

6. **Deallocate the tape or diskette device.**
   Follow the procedure in "Deallocate a device" on page 56.

Role - secadmin
Label - admin_low
Tool - profile shell

7. **As each new workstation is configured, copy the files from the tape or diskette to the correct directory on the new workstation.**

   a. **Prepare the directory for the new files.**

```
# cd /etc/security
# mv audit_control audit_control.orig
# mv audit_startup audit_startup.orig
# mv audit_warn audit_user.orig
# mv audit_event audit_event.orig
```

   b. **Allocate the appropriate device at the label admin_low.**
   Follow the procedure in "Allocate a device" on page 55.

i. To copy from tape

```
# tar xv audit_control audit_user audit_startup audit_event
```

ii. To copy from diskette

Use the command `eject`(1) to eject the diskette.

```
# tar xvf /dev/diskette audit_control audit_user audit_startup audit_event
```

c. Deallocate the device.

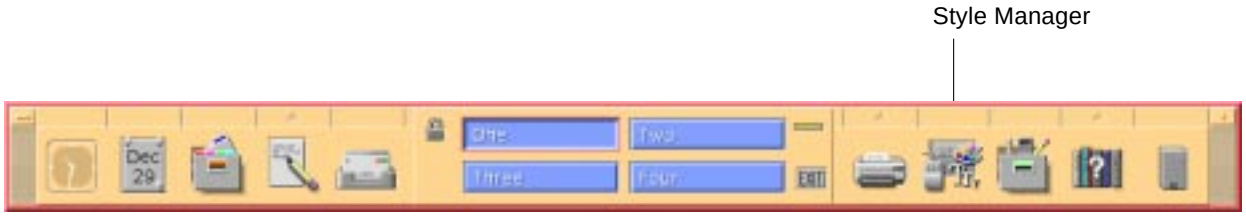
Follow the procedure in "Deallocate a device" on page 56.

Role - secadmin
Label - admin_low
Action - Audit Control

8. Modify the `audit_control` file on each new workstation with that workstation's remote and local audit filesystems.

▼ To Allocate and Deallocate Devices

The Device Allocation Manager on the Style Manager's Trusted Desktop subpanel is used to allocate and deallocate devices.



▼ Allocate a device

Role - ***administrative***
Label - admin_low
Tool - Device Manager

1. Allocate the tape or diskette drive using the Device Allocation Manager.

a. Click the left mouse button on the triangle above the Style Manager icon on the Front Panel.

A subpanel is displayed that includes the Device Manager icon.

b. Click the Device Manager icon once.

c. **Double-click the device to be allocated.**
mag_tape_0 allocates a tape device.
floppy_0 allocates a diskette.

d. **Click OK in the label builder; the file you load will be labeled admin_low.**

2. **Place the tape or diskette in the appropriate drive.**

▼ Deallocate a device

Role - ***administrative***
Label - admin_low
Tool - Device Manager

1. **Deallocate the tape or diskette drive using the Device Allocation Manager.**

a. **Double-click the device to be deallocated.**
A window appears listing devices being deallocated.

b. **When prompted, press the Return key to dismiss the window.**

2. **Remove the tape or diskette from the drive.**

Advanced Audit Setup Procedures

The following procedures describe how to modify the default audit classes and audit events, and to set a public object bit on files and folders to reduce unnecessary auditing.

▼ To Add Audit Classes

Role - secadmin
Label - admin_low
Action - Audit Classes

1. **Add audit classes in the `audit_classes` file.**

a. **Open the System_Admin folder from the Application Manager.**

b. **Double-click the Audit Classes action.**

2. **Add the classes you planned in "Planning a Site-Specific Event-to-Class Mapping" on page 31, write the file, and exit the editor.**

Caution – Do not reassign the hexadecimal numbers already in use.

Role - secadmin
Label - admin_low
Action - Audit Events

3. **Open the Audit Events action to add the new class to each event in the new class.**
   For events in more than one class, use a comma (no space) to delimit the classes.
4. **Write the file and exit the editor.**
5. **Make any changes to** `audit_control(4TSOL)` **and** `audit_user(4TSOL)` **to audit the events in the new classes.**
   See "To Set Audit Flags" on page 50 and "To Set User Exceptions to the Audit Flags" on page 51 for details of the procedures.

Note – On a distributed system, the `audit_class`, `audit_event`, `audit_startup`, and `audit_user` files must be identical on every workstation on the network. See "To Distribute Audit Configuration Files to a Network of Workstations" on page 53 for a process to distribute master copies of files to all workstations on the network.

Role - secadmin
Label - admin_low
Shell - profile shell

6. **Reboot, or as** `secadmin` **in an** `admin_low` **profile shell, run the** `auditconfig(1MTSOL)` **command with appropriate options.**
   In the following example, the audit session ID is 159, and the new classes are gr (for graphic applications) and db (for databases applications).

```
$ auditconfig -setsmask 159 gr,db
```

▼ To Add Audit Events

Role - secadmin
Label - admin_low
Action - Audit Events

1. **Add audit events in the** `audit_event(4TSOL)` **file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Events action.**
2. **Add the events you planned in "Planning a Site-Specific Event-to-Class Mapping" on page 31, write the file, and exit the editor.**
   For events in more than one class, use a comma (no space) to delimit the classes.

Note – Third-party applications can use the event numbers 32768 through 65536 only. See "Audit Event Categories" on page 6 for more information.

---

3. **Make any changes to** `audit_control(4TSOL)` **and** `audit_user(4TSOL)` **to audit the events in the new classes.**
   See "To Set Audit Flags" on page 50 and "To Set User Exceptions to the Audit Flags" on page 51 for details of the procedures.

Note – On a distributed system, the `audit_class`, `audit_event`, `audit_startup`, and `audit_user` files must be identical on every workstation on the network. See "To Distribute Audit Configuration Files to a Network of Workstations" on page 53 for a process to distribute master copies of files to all workstations on the network.

Role - secadmin
Label - admin_low
Shell - profile shell

4. **Reboot, or as** `secadmin` **in an** `admin_low` **profile shell, run the** `auditconfig(1MTSOL)` **command with appropriate options.**
   In the following example, the audit session ID is 159, and the new events are in the classes gr (for graphic applications) and db (for databases applications).

```
# auditconfig -setsmask 159 gr,db
```

▼ To Change Event-Class Mappings

Role - secadmin
Label - admin_low
Action - Audit Events

1. **Change event-class mappings inthe** `audit_control(4TSOL)` **file.**
   a. **Open the System_Admin folder from the Application Manager.**
   b. **Double-click the Audit Events action.**
2. **Edit the file to change the class mapping for each event to be changed, write the file, and exit the editor.**
   If you are changing events above number 2048, this is all you need to do.

Note – On a distributed system, the `audit_class`, `audit_event`, `audit_startup`, and `audit_user` files must be identical on every workstation on the network. See "To Distribute Audit Configuration Files to a Network of Workstations" on page 53 for a process to distribute master copies of files to all workstations on the network.

If you modify a kernel event mapping (numbers 1 to 2047):

   a. **Reboot the system, or**

Role - secadmin
Label - admin_low
Shell - profile shell

b. In an `admin_low` profile shell, change the runtime event-to-class mappings:

```
$ auditconfig -conf
```

▼ To Set Public Object Bit on Publicly Accessible Files

Setting the public object bit can reduce the size of the audit trail when you are auditing the predefined audit classes `file attribute access` (audit flag *fa*) and `file read` (audit flag *fr*).

Role - secadmin
Label - admin_low
Shell - profile shell

♦ **Set the public object bit on a local directory of publicly accessible files using the `setfattrflag(1TSOL)` command with the `-p 1` option.**
The following command sets the public object bit on the /etc directory. A search of the `/etc` directory, or a read of files in the `/etc` directory will not result in an audit record.

```
$ setfattrflag -p 1 /etc
```

- **Set the public object bit on a mounted file system of publicly accessible files using the `attr_flag` security attribute.**
  For example, the `vfstab_adjunct(4TSOL)` file for the mounted file system `/spublic` sets the public object flag for all files in the file system.

```
#       Modified template.
#
/spublic; \
acc_acl=; \
def_acl=; \
mode=; \
attr_flag=public; \
gid=; \
uid=; \
ilabel=; \
slabel=; \
forced=; \
allowed=; \
low_range=; \
hi_range=; \
mld_prefix=mldroot; \
mnt_flg=; \
audit_psa=;
#
```

See the man page for `mount_ufs`(1mtsol), and "Managing Files and File Systems" in the *Trusted Solaris Administrator's Procedures* for more details.

Dynamic Procedures

Dynamic controls apply to one workstation at a time, since the audit command only applies to the current workstation where you are logged in. Use dynamic controls to test auditing on a workstation (estimate volume of records, for example), or to add an auditing flag without having to reboot the workstation. However, if you make dynamic changes on one workstation for other than testing purposes, you should make the changes on all workstations.

▼ To Determine Current Audit Policy

Role - secadmin
Label - admin_low
Shell - profile shell

The `auditconfig(1MTSOL)` command enables an appropriately configured role to determine audit policy and to see what policies can be set. If your role is not configured to determine the policy, or if auditing is turned off, the command `auditconfig -getpolicy` returns an error.

```
$ auditconfig -getpolicy
    audit policies = none
$ auditconfig -lspolicy
    policy string    description:
    arge        include exec environment args in audit recs
    argv        include exec args in audit recs
    cnt         when no more space, drop recs and keep a count
    group       include supplementary groups in audit recs
    seq         include a sequence number in audit recs
    trail       include trailer tokens in audit recs
    path        allow multiple paths per event
    acl         include ACL information in audit recs
    ahlt        halt machine if we can't record an async event
    ilabel      include information labels in audit recs
    passwd      include cleartext passwords in audit recs
    slabel      include sensitivity labels in audit recs
    windata_down   include downgraded information in audit recs
    windata_up     include upgraded information in audit recs
    all         all policies
    none        no policies
```

▼ To Create an Admin_High Workspace

To change audit policy, to label files admin_high or to move them to an admin_high directory, to reset the audit daemon, and to make other changes in auditing requires an admin_high process. Admin_high processes can only be started interactively from an admin_high workspace.

1. **Click the right button on the Front Panel and choose Assume secadmin Role from the menu.**
   A secadmin role workspace becomes the current workspace.

2. **In the current workspace, click the right button on the workspace name (secadmin) button and choose Change Workspace SL from the menu.**

3. **In the label builder, type in** `admin_high` **in the Update With field.**
4. **Click the Update button to the right of the field.**
5. **Click OK at the bottom of the label builder.**
   The color of the workspace button turns to black, indicating an admin_high workspace. An admin_high workspace is available only to an administrative role.

▼ To Set Audit Policy Temporarily

The `auditconfig` command enables you to change audit policy, such as the action to take when its audit storage filesystems are full. Since the policy variable is a dynamic kernel variable, the policy that you set is in effect until the workstation next boots. See the `auditconfig(1MTSOL)` man page for a list of policy parameters.

Role - secadmin
Label - admin_low
Shell - profile shell

♦ **To set policies in one invocation of the command, or to override all current policies, separate the policies with commas (no spaces):**

```
$ auditconfig -setpolicy trail,seq
$ auditconfig -getpolicy
    audit policies = trail,seq
$ auditconfig -setpolicy cnt,acl
$ auditconfig -getpolicy
    audit policies = cnt,acl
```

♦ **To add policies to the current policies, preface each added policy with a plus (+):**

```
$ auditconfig -setpolicy trail,seq
$ auditconfig -getpolicy
    audit policies = trail,seq
$ auditconfig -setpolicy +cnt
$ auditconfig -setpolicy +acl
$ auditconfig -getpolicy
    audit policies = seq,trail,cnt,acl
```

♦ **To remove policies from the current policies, preface each policy to be removed with a minus (-):**

```
$ auditconfig -setpolicy trail,seq
$ auditconfig -getpolicy
    audit policies = trail,seq
$ auditconfig -setpolicy -seq
$ auditconfig -getpolicy
    audit policies = trail
```

In the examples above, the `trail` and `seq` tokens are added to debug audit trail discrepancies. To set policies permanently, enter the `auditconfig` command in the `audit_startup(1MTSOL)` script. See "To Set Audit Policy Permanently" on page 52 for how to edit the script.

▼ To Change Audit Flags Dynamically

The `auditconfig(1MTSOL)` command enables you to change audit flags dynamically, such as adding extra flags to a user, a session, or a process while the user, session, or process is active. Since the flags are added dynamically, they are in effect until the user logs out, the session ends, or the process ends.

Role - secadmin
Label - admin_low
Shell - profile shell

♦ **To set a particular user to be additionally audited for successful file reads:**

```
$ auditconfig -setumask <audit user id> +fr
```

♦ **To set a particular session to be additionally audited for failed file attribute access:**

```
$ auditconfig -setsmask <audit session id> -fa
```

♦ **To set a particular process to be additionally audited for successful and unsuccessful file attribute modifications:**

```
$ ps -ef | grep <application to be monitored>
$ auditconfig -setpmask <process id> fm
```

▼ To Stop the Audit Daemon

Only one audit daemon may run at a time. An attempt to start a second one will result in an error message, and the new one will exit. If there is a problem with the audit daemon, terminate the audit daemon gracefully, then restart it manually.

Role - secadmin
Label - admin_high
Shell - profile shell

♦ **To stop the audit daemon in event of trouble:**

```
$ audit -t
```

This is not recommended. Audit records may be lost.

▼ To Start the Audit Daemon

The audit daemon starts when the workstation is brought up to multiuser mode, and restarts when the audit daemon is instructed by the `audit -s` command to reread an audit configuration file.

Role - secadmin
Label - admin_high
Shell - profile shell

♦ **To restart the audit daemon in event of trouble or a change to an audit configuration file:**

```
$ audit -s
```

The pointer may be reset to the beginning of the list of audit directories when the administrator enters the `audit -s` command.

▼ To Send Audit Records to a New Audit File

Role - secadmin
Label - admin_high
Shell - profile shell

♦ **To change the current audit file for audit records being generated on the workstation:**
The new file is created in the same directory as the current file. The directory must be able to contain files labeled `admin_high`. .

```
$ audit -n <filename>
```

Audit Trail Management and Analysis 3

The tools described in this chapter manage the audit files generated on a workstation or on a distributed system. Managing the audit trail involves file tasks and interpretive tasks. File tasks handle disk space issues, such as combining multiple audit files into one and renaming files. Interpretive tasks cover audit analysis, such as selecting audit records based on audit event, user, host machine, and time of day. Sophisticated postprocessing using shell scripts can create auditing reports.

The Audit Trail

The collection of all audit files in a distributed system is called the *audit trail*. 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.

Audit files by default are stored in the *audit root directory*, defined as `/etc/security/audit/*/files`. Once each workstation has created an audit root directory, and the directories have been mounted (with mount points that follow the naming convention) on the audit administration server, the management tools, `auditreduce` and `praudit`, can examine the entire audit trail. See “Basic Audit Setup Procedures” on page 40 for how to set up an audit trail.

Even though it is possible to locate audit directories within other file systems that are not dedicated to auditing, this is not recommended. If other factors dictate placing audit files on a partition not dedicated to auditing, only do so for directories of last resort. Directories of last resort would be directories where audit files would be 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 would be in an 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 is unworkable in a security-conscious production environment.

How the Audit Trail Is Created

The *audit trail* is created by the audit daemon, `auditd(1MTSOL)`. The audit daemon starts on each workstation when the workstation is booted. After `auditd` starts, 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(4TSOL)` man page for a description of the file format.

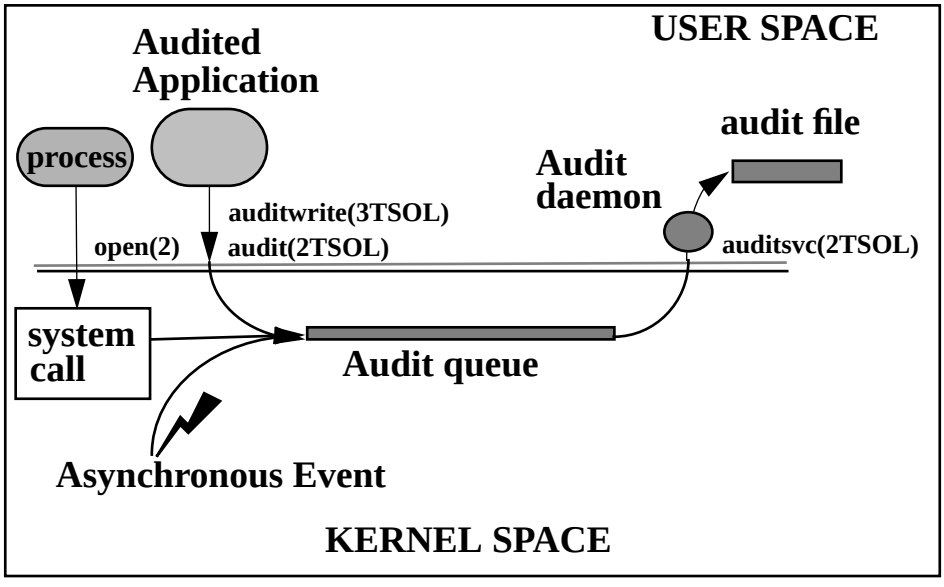


Figure 3-1 How Auditing Works

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 workstation's activities are suspended because the kernel's audit buffers are full. The audit operations can continue because `auditd` is not audited.

Audit Record Format

Audit files consist of self-contained audit records of user-level and kernel-level events that have been preselected for auditing by the security administrator. An *audit record c*onsists of a sequence of *audit tokens*, each of which describes an attribute of the event being audited. 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.

Each audit 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).

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

"Audit Token Structure" on page 126, gives a detailed description of each data element in each token and shows sample output. "Audit Records" on page 157 lists all the audit records generated by Trusted Solaris auditing. The records are listed alphabetically by kernel event and by user event. A cross-reference table to the audit records is found in "Audit Event to Audit Class Mappings" on page 112.

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' attributes, such as the owner or mode.

For example, the AUE_LSTAT kernel event, whose audit record is described in Table B-64 on page 185, has the following tokens:

- header
- path
- attribute (optional)
- subject
- return

If the `trail` policy has been turned on using the `auditconfig` command, the `trailer` token appears in the audit record after the `return` token.

Human-Readable Audit Record Format

This section provides examples of audit records in text format. Audit records are stored in binary format. Running the binary records through the `praudit(1MTSOL)` command produces text output, which can be sent to standard output, a printer, or a scripting program to produce reports. For a complete description of `praudit`, see the `praudit(1MTSOL)` man page. For an example of a scripting program, see "To Perform Selections Using a praudit Script" on page 82.

Reading an Audit Token

The following examples of a `header` token show the form that `praudit` produces by default. Examples are also provided of raw (`-r`) and short (`-s`) options.

Every audit record begins with a `header` token. The `header` token gives information common to all audit records. When displayed by `praudit` in default format, a `header` token looks like the following example from `ioctl`:

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

The fields are:

- A token ID, here in text form, **`header`**
- The record length in bytes, including the `header` and `trailer` tokens, here **`240`**
- An audit record structure version number, here, version **`1`**
- An event ID identifying the type of audit event, here in text form, **`ioctl(2)`**
- An event ID modifier with descriptive information about the event type, here the descriptive field is empty
- The time and date the record was created, here **`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,,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 20 is the header token ID and `158` is the event number for this event.

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

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

Reading an Audit Record

Every audit record contains at least the `header` token and one other token. For example, the audit record for the audit event AUE_login contains five tokens. See “login(1TSOL) — local” on page 247 for a full description of its audit record format.

When displayed by `praudit` in default format, the audit record for AUE_login looks like this, one token per line:

```
header,90,3,login - local,,Tue Jul 8 15:12:01 1997, +520002000
msec,
text,emily
text,successful login
subject,emily,emily,staff,emily,staff,14094,14094,0 0 willet,
return,success,0
sequence,17
trailer,90
```

The tokens are:

- A header token
- A text token (login name)
- A text token (for success or failure)
- A subject token
- A return token

When this audit file collected records, the audit policy tokens `sequence` and `trailer` were turned on, so all audit records including this one contain the following tokens:

- A sequence token
- A trailer token

Note the following features in the audit record:

- Each user's processes is assigned a unique audit ID that stays the same even when the user ID changes (14094)
- Each session has an audit session ID (14094)
- Audit records are self-contained

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

The Trusted Solaris 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.

Note – You should archive system administration files with audit file archives. Information that is referred to in the audit trail but changes as site personnel and equipment change, such as users and their UIDs, affects your ability to interpret records.

Using `praudit -l`, the audit record displays on one line, like this:

```
header,90,3,login - local,,Tue Jul 8 15:12:01 1997, +520002000
msec,text,emily,text,successful login,subject,emily,emily,
staff,emily,staff,14094,14094,0 0 willet,return,success,0,
sequence,17,trailer,90
```

Using `praudit -r` the audit record displays like this:

```
20,90,3,6152,0x0000,872028721,520002000
40,emily
40,successful login
36,6001,6001,10,6001,10,14094,14094,0 0 129.150.110.2
39,0,0
47,17
19,90
```

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 workstation that generated them. The contents of the audit files are binary, protected at the sensitivity label admin_high, and accessible in a profile shell only by an administrative role with the Audit Review profile.

Audit File Naming

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

```
start-time.finish-time.workstation
```

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 *workstation* is the name of the workstation that generated the file. Some examples of these names can be found in "Example of a Closed Audit File Name" on page 73.

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

```
start-time.not_terminated.workstation
```

How Audit File Names Are Used

The file name time stamps are used by the `auditreduce` command 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 online, and searching them all for records generated in the last 24 hours would be 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 saving 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.patchwork
```

The audit log files are named by the beginning date, so the example above was started in 1997, 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 `patchwork` 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:

```
19970320005243.19970327225351.patchwork
```

The example above was started in 1997, 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 `patchwork` at the end is the host name of the workstation whose audit data is being collected.

3

Audit Files Management

Two commands, `praudit(1MTSOL)` and `auditreduce(1MTSOL)`, enable the audit reviewer to process the Trusted Solaris audit records. The `praudit` command makes the records readable, and the `auditreduce` command enables selecting particular audit records and merging the records into one audit trail.

Note – The `auditreduce` command can only find records that have been preselected by the security administrator. Events that are not recorded in the audit trail are unavailable to postselection tools.

Merging the Audit Trail

The `auditreduce` command merges audit records from one or more input audit files to create a single, chronologically ordered output file. On a distributed system, the input audit files originate from different workstations. Therefore, when issued from the audit administration server, the `auditreduce` command treats the distributed system as if it were one workstation. This treatment simplifies audit administration. Coupled with backup audit partitions, the distributed system is robust in the face of workstation failures.

The `auditreduce` command also includes options for selecting sets of records to examine. For instance, records from the past 24 hours can be selected to generate a daily report; all records generated by a specific user can be selected to examine that user's activities; or all records caused by a specific event type can be selected to see how often that type occurs.

Selecting Records from the Audit Trail

Options to the `auditreduce(1MTSOL)` command allow you to select audit records based on file characteristics and record characteristics, as shown in Table 3-1.

Table 3-1 Some Options to the `auditreduce` Command

Characteristic	**Option(s)**
Time, date (start, finish)	-d, -a, -f
Host (workstation) ID	-M, -h, -S
Audit class	-c
Audit event	-m
Audit session ID	
Audit User ID – AUID	-u
Effective and Real User ID – EUID, RUID	-e, -r
Effective and Real Group ID – EGID, RGID	-f, -g
Process ID – PID	-j
Sensitivity label	-s
Information label	-i
Filename	*filename*

Uppercase options select operations or parameters for *files*, and lowercase options select parameters for *records*. When piped through `praudit`, audit files processed by the `auditreduce` command are readable. Otherwise, they remain in binary format.

The merging and selecting functions of `auditreduce` are logically independent. The `auditreduce` command selects messages from the input files as the records are read, before the files are merged and written to disk.

3

Using the auditreduce *and* praudit ***Commands***

This section describes a few common uses of `auditreduce` and `praudit` to select and manage data. See the `auditreduce(1MTSOL)` man page for more examples.

Role - secadmin
Label - admin_high
Shell - profile shell

Prerequisites for running the `auditreduce(1MTSOL)` and `praudit(1MTSOL)` commands:

- You are in an administrative role that includes the Audit Review profile.
- You are in an admin_high workspace of that role.
  To create an admin_high workspace, see "To Create an Admin_High Workspace" on page 61 in Chapter 2, "Auditing Setup".
- You have launched a terminal window and opened a profile shell [`pfsh(1MTSOL)`].
- You are running with all privileges [`sh(1)`].

To access the audit trail for a distributed system:

- You issue the `auditreduce` command from the audit administration server.

▼ To Read a Closed Audit File

The `praudit` command enables you to display audit records interactively and create very basic reports. For multiple files, the input is piped from `auditreduce`.

♦ **Specify the audit file as the file argument to the `praudit` command.**

```
$ praudit 19970401000000.19970601000000.grebe
```

This displays audit token per line to standard output.

♦ **Specify the audit file as the file argument to the `praudit -l` command.**

```
$ praudit -l 19970401000000.19970601000000.grebe
```

This displays one audit record per line to standard output.

▼ To Read a Current Audit File

♦ **Use the** `tail(1)` **command to see what is currently being written to an active audit file.**

```
$ praudit | tail -40 19970401000000.not_terminated.grebe
```

This displays the latest 40 tokens that were recorded to standard output.

▼ To Display Several Audit Files as One Audit File

♦ **To display several audit files in chronological order in the terminal window, pipe the output of** `auditreduce` **into** `praudit`**.**

```
$ auditreduce 19970413000000.19970413235959.willet \
19970413000000.19970413235959.grebe | praudit
```

♦ **To display the whole audit trail in the terminal window, pipe the output of** `auditreduce` **into** `praudit`**.**

```
$ auditreduce | praudit
```

Note – The `auditreduce` command without options does not disturb open audit files.

▼ To Print an Audit Log

♦ **Use** `praudit` **with a pipe to** `lp`**, to send the output of one file to the printer.**

```
$ praudit 19970413000000.19970413235959.audubon | lp
```

♦ **Use** `auditreduce` **piped through** `praudit` **with a pipe to** `lp`**, to send the output of all closed audit files to the printer.**

```
$ auditreduce | praudit | lp
```

Note – In the Trusted Solaris environment, the printer must be able to accept admin_high print jobs.

▼ To Display User Activity on a Selected Date

♦ **Use the** `-d` **option to the** `auditreduce` **command to see audit information collected during a specified 24-hour period.**
In the following example, the security administrator checks to see when a user named `doris` logged in and logged out on April 13, 1997, by requesting the `lo` message class. The short-form date is in the form *yymmdd*. (The long form is described in the `auditreduce`(1MTSOL) man page.)

```
$ auditreduce -d 970413 -u doris -c lo | praudit
```

▼ To Print User Activity on a Selected Date

♦ **Use the** `auditreduce` **command with a pipe through** `praudit` **to** `lp`**, to send selected output to a printer.**

Note – In the Trusted Solaris environment, the printer must be able to accept admin_high print jobs.

```
$ auditreduce -d 970413 -u doris -c lo | praudit | lp
```

▼ 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 970413 -O /usr/audit_summary/logins
```

The `-O` option creates an audit file in the `/usr/audit_summary` directory. The file name has 14-character timestamps for both start-time and end-time, and the suffix `logins`:

```
/usr/audit_summary/19970413000000.19970413235959.logins
```

▼ To Display Audit Records Created Before or After a Designated Date

The *date-time* options `-b` and `-a` allow specifying records before or after a particular day and time. A day begins at *yyyymmdd*00:00:00 and ends at *yyyymmdd*23:59:59. 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.

The `auditreduce -a` command with the date shown in the following screen example sends all audit records created after midnight on July 15, 1997 through `praudit` to standard output.

```
$ auditreduce -a 97071500:00:00 | praudit
```

If `-a` is not specified, `auditreduce` defaults to 00:00:00, January 1, 1970.

The `auditreduce -b` command with the same date shown above sends all audit records created before midnight on July 15, 1997 through `praudit` to standard output.

```
$ auditreduce -b 97071500:00:00 | praudit
```

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 "To Copy Login/Logout Messages to a Single File" on page 79.

▼ To Find an Audit Event

♦ **Use the message type selection for** auditreduce **(**-m **option) to find a particular audit event.**
The -m option accepts either numeric message identifiers or AUE_*xxxxx* event names. The screen example below finds all kernel-level login events in the audit trail and displays them to standard output.

```
$ auditreduce -m AUE_LOGIN | praudit
```

The auditreduce command rejects an incorrect format, but does not describe the correct format.

▼ To Combine Selected Audit Files

Although auditreduce can do this type of combination and deletion automatically (see the -C and -D options in the auditreduce(1MTSOL) man page), it is often easier to select the files manually (perhaps with find) and use the auditreduce command to combine just the named set of files.

1. **List the audit files as arguments to the** auditreduce **command.**
In the following example, a recurring job that starts a bit before midnight merges the audit files from two days before. The final time on the file is the time the job ended, here just before midnight, Greenwich Mean Time (GMT).

```
$ auditreduce 19970413000000.19970413235959.grebe \
19970413000000.19970413235959.willet \
19970413000000.19970413235959.sora
$ ls *audubon
19970413000000.19970414235959.server_name
```

2. **Delete the input files and move the output file to the audit root directory on the administration server.**
   In this example, the auditreduce command was run on the audit administration server, `audubon`, and then placed in its audit root directory so that future calls to `auditreduce` locate the file.

```
$ rm /etc/security/tsol/grebe/files/19970413000000.19970413235959.grebe
$ rm /etc/security/tsol/willet/files/19970413000000.19970413235959.willet
$ rm /etc/security/tsol/sora/files/19970413000000.19970413235959.sora
$ mv 19970413000000.19970414235959.audubon /etc/security/audit/audubon/files/
```

▼ To Reduce Audit Files

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. The following example selects just the audit records from April 1997.

```
$ auditreduce -m AUE_LOGIN -a 19970401000000 -b 19970501000000 \
-O /usr/audit_summary/logins_april97
```

The output is a smaller file containing just the April 1997 login/logout records. Note that the end-time stamp is the date (in GMT) that the command was run (June 1, 1997), not the last date of the merged records. You specified the file suffix, `logins_april97`, on the command line with the directory name.

```
/usr/audit_summary/19970401000000.19970601000000.logins_april97
```

▼ To Change the `praudit` Field Separator to a Tab

When the `praudit` command displays an audit token, it separates the data fields with commas. However, if a field (such as a time stamp) contains a comma, this cannot be distinguished from a field-separating comma.

♦ **Press the <Tab> key as the value of the `-d` option to `praudit`.**

```
$ praudit -d"<press tab key>" 19970413120429.19970413180433.grebe
```

There is no space between the -d option and the delimiter.
Surround the delimiter with double quotes.
The delimiter can be up to four characters long.

▼ To Change the `praudit` Token Separator to a Tab

Audit tokens are separated by newlines by default. When audit records are printed one per line using the `-l` option, the audit token separator is the same as the audit field separator. In the following screen example, the audit tokens are separated by tabs, as are the audit fields.

```
$ praudit -l -d"<press tab key>" 19970413120429.19970413180433.grebe
```

▼ To Perform Selections Using a `praudit` Script

To accomplish more sophisticated display and reports, process the output from `praudit` with `sed` or `awk`, or write programs to interpret and process the binary audit records.

It is sometimes useful to manipulate `praudit` 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 newlines.

3

Troubleshooting Auditing 4

Another auditing task is to handle audit anomalies as they occur. Typical tasks that audit analysts and system administrators face are discussed below.

Preventing Audit Trail Overflow

When all audit file systems for a workstation 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.

Site security policy may require a different solution. There are other candidates: preventing overflow, keeping a count of dropped audit records, and halting the workstation.

If your security policy requires that overflow be prevented so that no audit data is ever lost, see “To Prevent Audit Trail Overflow by Planning Ahead” on page 86.

If your security policy permits the loss of some audit data rather than suspending system activities due to audit trail overflow. In that case, you can set the `auditconfig` policy to drop or count records. See "To Handle an Audit Filesystem Overflow" on page 87 for how to drop or count records.

If your security policy requires you to handle filesystem overflow by halting the affected workstation, you must enter the workstation in single-user mode. This is not a secure practice. See "To Handle an Audit Filesystem Overflow" on page 87 for the procedure.

▼ To Prevent Audit Trail Overflow by Planning Ahead

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 all audit file systems.**
   The schedule must allow files to be deleted from the system before the hard limit of the system is reached. Scripts, including modified `audit_warn` scripts, can automatically move audit files to a separate disk before archiving.
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.**
   For example, the current list of users and passwords, the directory listings on the workstations, and other volatile information should be saved.
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 would be a summary file containing only the audit records for all logins and logouts. See Chapter 3, "Audit Trail Management and Analysis."

▼ To Handle an Audit Filesystem Overflow

Role - secadmin
Label - admin_high
Tool - profile shell

♦ **To set the audit policy that a count of audit records is kept when the audit file systems are full:**

```
$ auditconfig -setpolicy +cnt
```

♦ **To set the audit policy that the workstation is shut down when its audit file systems are full:**

```
$ auditconfig -setpolicy +ahlt
```

To set one of the above policies permanently, enter the command in the `audit_startup(1MTSOL)` script. See "To Set Audit Policy Permanently" on page 52 for how to edit the script.

Note – On a distributed system, the same audit policy should be applied to all workstations.

Cleaning up an Audit File Marked `not_terminated`

Occasionally, if an audit daemon dies while its audit file is still open, or a server becomes inaccessible and forces the workstation 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.

The `auditreduce(1MTSOL)` 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 incomplete 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.

▼ To Clean Up a `not_terminated` Audit File

Role - secadmin
Label - admin_high
Tool - profile shell

1. **Check the `/etc/security/audit_data` file to determine the current process number of the audit daemon.**
   If that process is still running, and if the file name in `audit_data(4TSOL)` is the same as the file in question, do not clean the file.
2. **Issue the command `auditreduce` with the `-O` (capital o) option.**
3. **Provide the workstation name as the argument to `-O`, and the incomplete file name. To delete the original record, use the `-D` option.**

```
$ auditreduce -O workstation 19970413120429.not_terminated.workstation
```

This creates a new audit file with the correct name, cleans up pointers to other files, and copies all the records to the new file. The end-time is the time when the command was executed; the correct suffix is *workstation*, explicitly specified.

4. **If you did not use the `-D` option, verify that the new file contains the original file's records, then delete the original file.**

```
$ ls -l 19970413120429*.workstation


$ rm 19970413120429.not_terminated*
```

Using the sequence *Token for Debugging*

When an audit trail created from merging records from several workstations appears to have the records listed out of order, you can debug the audit trail discrepancies using the sequence token. Since the sequence token is not recorded by default, the security administrator adds it to the audit policy. The audit policy must be set identically on all workstations contributing to the audit trail.

When the audit trail has been debugged, the security administrator removes the token.

▼ To Add the sequence Token to the Audit Record

Role - secadmin
Label - admin_high
Tool - profile shell

1. To add the `seq` audit policy dynamically, on the command line:

```
$ auditconfig -setpolicy +seq
$ auditconfig -getpolicy
       cnt,seq
```

Role - secadmin
Label - admin_high
Action - Audit Startup

2. To add the `seq` audit policy permanently, in the `audit_startup` file:

```
#!/bin/sh
auditconfig -setpolicy cnt,seq
```

▼ To Prevent the sequence Token from Being Part of Audit Records

Role - secadmin
Label - admin_high
Tool - profile shell

1. To remove the `seq` audit policy dynamically, on the command line:

```
$ auditconfig -setpolicy -seq
$ auditconfig -getpolicy
cnt
```

Role - secadmin
Label - admin_high
Action - Audit Startup

2. To remove the `seq` audit policy from the `audit_startup` file:

```
#!/bin/sh
auditconfig -setpolicy cnt
```

Starting the Audit Daemon Manually

Role - secadmin
Label - admin_high
Tool - profile shell

On a distributed system, if many workstations havse lost their audit daemon, bring up the audit daemons in order. First execute the command on the audit administration server, then on the audit servers, and finally on the audit clients.

♦ **Execute the command `/usr/sbin/auditd`.**

```
$ /usr/sbin/auditd
```

4

Workstations are Being Audited Differently

If you change audit configuration files on one workstation and fail to copy it to the other workstations on the network, the workstations will be audited differently. Therefore,

Role - secadmin
Label - admin_low
Tool - profile shell

1. **Copy the audit configuration files from a central location to every workstation.**
   Follow the procedure in "To Distribute Audit Configuration Files to a Network of Workstations" on page 53.
2. **Check that the audit class mappings for attributable and nonattributable events match the kernel cache.**

Role - secadmin
Label - admin_high
Tool - profile shell

▼ To set audit class mappings for attributable events

1. **First, check to see if the kernel preselection mask matches the class mappings in the `flags:` field of the `audit_control(4TSOL)` file by issuing the command:**

```
$ auditconfig -chkconf
```

2. **If the runtime class mappings differ from the kernel cache, issue the command:**

```
$ auditconfig -conf
```

Role - secadmin
Label - admin_high
Tool - profile shell

▼ To set audit class mappings for nonattributable audit events

1. **First, check to see if the kernel preselection mask matches the nonattributable events in the `naflags:` field of the `audit_control(4TSOL)` file by issuing the command:**

```
$ auditconfig -getkmask
```

♦ **If they differ, issue the command:**

```
$ auditconfig -setkmaskac
```

Finding Failed Login Attempts

Role - admin
Label - admin_high
Tool - profile shell

♦ **Enter** `-lo` **as the value of the** `-c` **option to** `auditreduce(1MTSOL)`**.**

```
$ auditreduce -c -lo -O /usr/audit_summary/logins_failed
```

The value "`-lo`" is the audit flag for failed (`-`) login (audit class `lo`) attempts. The command produces a binary file in the `/usr/audit_summary` directory with all failed login attempts on the distributed system. The `/usr/audit_summary` directory is labeled `admin_high`.

```
/usr/audit_summary/19970313120429.19970613120415.logins_failed
```

Note – This command works only if the security administrator has pre-selected failed logins for the workstation, distributed system, or users.

4

Event-to-Class Mappings A

This appendix has two parts. The first part lists audit events by audit class. The second part lists the default audit events in alphabetical order. See the file `/etc/security/audit_event` for a list of events by audit event number.

A

Audit Events Listed by Audit Class

Trusted Solaris provides the audit classes listed in Table A-1. They are in the file `/etc/security/audit_class`.

Table A-1 Audit Classes (Default)

Long Name	Short Name	Audit Mask	List of Events per Class
administrative	ad	0x00000800	Table A-2 on page 95
all classes	`all`	0xffffffff	Table A-18 on page 112
application	ap	0x00004000	No predefined audit events.
file close	cl	0x00000040	Table A-3 on page 100
exec	ex	0x40000000	Table A-4 on page 100
file attribute access	fa	0x00000004	Table A-5 on page 100
file create	fc	0x00000010	Table A-6 on page 101
file delete	fd	0x00000020	Table A-7 on page 102
file attribute modify	fm	0x00000008	Table A-8 on page 103
file read	fr	0x00000001	Table A-9 on page 103
file write	fw	0x00000002	Table A-10 on page 104
ioctl	io	0x20000000	No predefined audit events.
ipc	ip	0x00000200	Table A-11 on page 105
login or logout	lo	0x00001000	Table A-12 on page 107
non-attribute	na	0x00000400	Table A-13 on page 107
invalid class	no	0x00000000	Table A-14 on page 108
network	nt	0x00000100	Table A-15 on page 109
other	ot	0x80000000	Table A-16 on page 110
process	pc	0x00000080	Table A-17 on page 110

Events in Audit Class ad

Table A-2 lists in alphabetical order the administrative class of audit events provided in the Trusted Solaris 2.5 release.

Table A-2 Administrative Audit Events (Default)

Audit Event Number and Event		Where Described
18	AUE_ACCT	"acct(2)" on page 158
50	AUE_ADJTIME	"adjtime(2TSOL)" on page 158
57	AUE_ASYNC_DAEMON	No longer supported
114	AUE_ASYNC_DAEMON_EXIT	No longer supported
224	AUE_AUDITON_GETCAR	"auditon(2TSOL) — get current active root" on page 159
231	AUE_AUDITON_GETCLASS	"auditon(2TSOL) — get event class" on page 159
229	AUE_AUDITON_GETCOND	"auditon(2TSOL) — get audit state" on page 160
223	AUE_AUDITON_GETCWD	"auditon(2TSOL) — get current working directory" on page 160
221	AUE_AUDITON_GETKMASK	"auditon(2TSOL) — get kernel mask" on page 160
225	AUE_AUDITON_GETSTAT	"auditon(2TSOL) — get audit statistics" on page 161
141	AUE_AUDITON_GPOLICY	"auditon(2TSOL) — GETPOLICY command" on page 161
145	AUE_AUDITON_GQCTRL	"auditon(2TSOL) — get audit queue control parameters" on page 161
139	AUE_AUDITON_GTERMID	No longer supported
144	AUE_AUDITON_SESTATE	No longer supported
232	AUE_AUDITON_SETCLASS	"auditon(2TSOL) — set event class" on page 162
230	AUE_AUDITON_SETCOND	"auditon(2TSOL) — set audit state" on page 162
222	AUE_AUDITON_SETKMASK	"auditon(2TSOL) — set kernel mask" on page 163
228	AUE_AUDITON_SETSMASK	"auditon(2TSOL) — set mask per session ID" on page 163
226	AUE_AUDITON_SETSTAT	"auditon(2TSOL) — reset audit statistics" on page 163

Table A-2 Administrative Audit Events (Default)

Audit Event Number and Event		Where Described
227	AUE_AUDITON_SETUMASK	"auditon(2TSOL) — set mask per uid" on page 164
142	AUE_AUDITON_SPOLICY	"auditon(2TSOL) — SETPOLICY command" on page 164
146	AUE_AUDITON_SQCTRL	"auditon(2TSOL) — set audit queue control parameters" on page 165
140	AUE_AUDITON_STERMID	No longer supported
150	AUE_AUDITSTAT	"auditstat(2)" on page 165
136	AUE_AUDITSVC	"auditsvc(2TSOL)" on page 166
10	AUE_CHMOD	"chmod(2TSOL)" on page 167
513	AUE_CLOCK_SETTIME	"clock_settime(3RTSOL)" on page 168
61	AUE_EXPORTFS	System 4 (BSD) system call
132	AUE_GETAUDIT	"getaudit(2TSOL)" on page 177
130	AUE_GETAUID	"getauid(2TSOL)" on page 177
147	AUE_GETKERNSTATE	No longer supported
149	AUE_GETPORTAUDIT	"getportaudit(2)" on page 180
134	AUE_GETUSERAUDIT	No longer supported
9	AUE_MKNOD	"mknod(2TSOL)" on page 187
246	AUE_MODADDMAJ	"modctl(2) — bind module" on page 188
245	AUE_MODCONFIG	"modctl(2) — configure module" on page 189
242	AUE_MODCTL	Placeholder
243	AUE_MODLOAD	"modctl(2) — load module" on page 189
244	AUE_MODUNLOAD	"modctl(2) — unload module" on page 190
62	AUE_MOUNT	"mount(2TSOL)" on page 190
115	AUE_NFSSVC_EXIT	No longer supported
58	AUE_NFS_GETFH	No longer supported
53	AUE_NFS_SVC	No longer supported
60	AUE_QUOTACTL	No longer supported

Table A-2 Administrative Audit Events (Default)

Audit Event Number and Event		Where Described
20	AUE_REBOOT	No longer supported
133	AUE_SETAUDIT	“setaudit(2TSOL)” on page 217
131	AUE_SETAUID	“setauid(2TSOL)” on page 217
59	AUE_SETDOMAINNAME	No longer supported
29	AUE_SETHOSTNAME	No longer supported
148	AUE_SETKERNSTATE	No longer supported
51	AUE_SETRLIMIT	“setrlimit(2TSOL)” on page 222
37	AUE_SETTIMEOFDAY	No longer supported
135	AUE_SETUSERAUDIT	No longer supported
201	AUE_STIME	“stime(2TSOL)” on page 229
28	AUE_SWAPON	No longer supported
239	AUE_SYSINFO	“sysinfo(2TSOL)” on page 230
12	AUE_UMOUNT	“umount(2TSOL) — old version” on page 231
56	AUE_UNMOUNT	No longer supported
233	AUE_UTSSYS	“utssys(2) — fusers” on page 233
9018	AUE_add_drv	“add_drv(1MTSOL)” on page 236
6200	AUE_allocate_succ	“allocate(1MTSOL) device success” on page 237
6201	AUE_allocate_fail	“allocate(1MTSOL) device failure” on page 237
6144	AUE_at_create	“at(1TSOL) - create crontab” on page 239
6145	AUE_at_delete	“at(1TSOL) - delete atjob file (at or atrm)” on page 239
6146	AUE_at_perm	“at(1TSOL) - permission” on page 240
9016	AUE_audit	“audit(1MTSOL)” on page 240
9015	AUE_auditwrite	“auditwrite(3TSOL)” on page 241
9029	AUE_chroot_cmd	“chroot(1MTSOL)” on page 241
6147	AUE_cron_invoke	“crontab(1TSOL) - invoke atjob or crontab” on page 243
6148	AUE_crontab_create	“crontab(1TSOL) - crontab created” on page 242

Table A-2 Administrative Audit Events (Default)

Audit Event Number and Event		Where Described
6149	AUE_crontab_delete	"crontab(1TSOL) - crontab deleted" on page 242
6150	AUE_crontab_perm	"crontab(1TSOL) - permission" on page 243
6202	AUE_deallocate_succ	"allocate(1MTSOL) device success" on page 237
6203	AUE_deallocate_fail	"deallocate(1MTSOL) device failure" on page 238
9025	AUE_dispadmin	"dispadmin(1MTSOL)" on page 244
9032	AUE_eeprom	"eeprom(1MTSOL)" on page 244
9031	AUE_fuser	"fuser(1MTSOL)" on page 245
6160	AUE_halt_solaris	"halt(1MTSOL)" on page 245
6205	AUE_listdevice_succ	"allocate(1MTSOL) - list devices success" on page 238
6206	AUE_listdevice_fail	"allocate(1MTSOL) - list devices failure" on page 238
9020	AUE_modload	"modload(1MTSOL)" on page 248
9021	AUE_modunload	"modload(1MTSOL)" on page 249
9026	AUE_pbind	"pbind(1M)" on page 251
9009	AUE_pfsh_priv	"pfsh(1MTSOL)" on page 251
9010	AUE_pfsh_nopriv	"pfsh(1MTSOL)" on page 251
9008	AUE_pfsh_trusted_nopriv	"pfsh(1MTSOL)" on page 251
9007	AUE_pfsh_trusted_priv	"pfsh(1MTSOL)" on page 251
9027	AUE_psradm	"psradm(1MTSOL)" on page 252
6161	AUE_reboot_solaris	"reboot(1MTSOL)" on page 252
9019	AUE_rem_drv	"rem_drv(1MTSOL)" on page 254
9028	AUE_run_level_change	"init(1M) - run level change" on page 255
9014	AUE_sendmail_defer	"sendmail(1MTSOL)" on page 255
9013	AUE_sendmail_deliver	"sendmail(1MTSOL)" on page 255
9012	AUE_sendmail_upgrade	"sendmail(1MTSOL) - upgrade" on page 256
9017	AUE_uauth	"uauth" on page 259
9022	AUE_setuname	"setuname(1MTSOL)" on page 256

Table A-2 Administrative Audit Events (Default)

Audit Event Number and Event	Where Described
9030 AUE_swap	“swap(1MTSOL)” on page 258
9023 AUE_uadmin_cmd	“uadmin(1MTSOL)” on page 258
9024 AUE_uname_set	“uname(1TSOL)” on page 259

A

Events in Audit Class ap

The `application` audit class contains no audit events in the default configuration.

Events in Audit Class cl

Table A-3 lists in alphabetical order the `file close` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-3 File Close Audit Events (Default)

Audit Event Number and Event		Where Described
112	AUE_CLOSE	"close(2)" on page 169
213	AUE_MUNMAP	"munmap(2)" on page 196

Events in Audit Class ex

Table A-4 lists in alphabetical order the `exec` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-4 Exec Audit Events (Default)

Audit Event Number and Event		Where Described
7	AUE_EXEC	"exec(2TSOL)" on page 171
23	AUE_EXECVE	"execve(2TSOL)" on page 171

Events in Audit Class fa

Table A-5 lists in alphabetical order the `file attribute` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-5 File Attribute Access Audit Events (Default)

Audit Event Number and Event		Where Described
14	AUE_ACCESS	"access(2TSOL)" on page 157
220	AUE_AUDITSYS	Placeholder
66	AUE_BSMSYS	Placeholder

Table A-5 File Attribute Access Audit Events (Default)

Audit Event Number and Event		Where Described
55	AUE_FSTATFS	“fstatfs(2)” on page 176
17	AUE_LSTAT	“lstat(2TSOL)” on page 185
236	AUE_LXSTAT	“lxstat(2)” on page 186
64	AUE_MSGSYS	Placeholder
3	AUE_OPEN	Placeholder
199	AUE_OSTAT	No longer supported
71	AUE_PATHCONF	“pathconf(2)” on page 203
67	AUE_RFSSYS	Placeholder
63	AUE_SEMSYS	Placeholder
65	AUE_SHMSYS	Placeholder
16	AUE_STAT	“stat(2TSOL)” on page 227
54	AUE_STATFS	“statfs(2)” on page 228
234	AUE_STATVFS	“statvfs(2TSOL)” on page 228
25	AUE_VPIXSYS	Placeholder
235	AUE_XSTAT	“xstat(2)” on page 235

Events in Audit Class fc

Table A-6 lists in alphabetical order the `file create` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-6 File Create Audit Events (Default)

Audit Event Number and Event		Where Described
8	AUE_CHDIR	“chdir(2TSOL)” on page 166
111	AUE_CORE	“process dumped core” on page 205
4	AUE_CREAT	“creat(2TSOL)” on page 169
5	AUE_LINK	“link(2TSOL)” on page 185
47	AUE_MKDIR	“mkdir(2TSOL)” on page 187
73	AUE_OPEN_RC	“open(2TSOL) — read,creat” on page 198

Table A-6 File Create Audit Events (Default)

Audit Event Number and Event		Where Described
75	AUE_OPEN_RTC	"open(2TSOL) — read,trunc,creat" on page 198
81	AUE_OPEN_RWC	"open(2TSOL) — read,write,creat" on page 200
83	AUE_OPEN_RWTC	"open(2TSOL) — read,write,trunc,creat" on page 200
77	AUE_OPEN_WC	"open(2TSOL) — write,creat" on page 202
79	AUE_OPEN_WTC	"open(2TSOL) — write,trunc,creat" on page 202
42	AUE_RENAME	"rename(2TSOL)" on page 208
48	AUE_RMDIR	"rmdir(2TSOL)" on page 208
21	AUE_SYMLINK	"symlink(2TSOL)" on page 229
240	AUE_XMKNOD	"xmknod(2)" on page 234

Events in Audit Class fd

Table A-7 lists in alphabetical order the `file delete` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-7 File Delete Audit Events (Default)

Audit Event Number and Event		Where Described
44	AUE_FTRUNCATE	No longer supported
74	AUE_OPEN_RT	"open(2TSOL) — read,trunc" on page 199
75	AUE_OPEN_RTC	"open(2TSOL) — read,trunc,creat" on page 198
82	AUE_OPEN_RWT	"open(2TSOL) — read,write,trunc" on page 201
83	AUE_OPEN_RWTC	"open(2TSOL) — read,write,trunc,creat" on page 200
78	AUE_OPEN_WT	"open(2TSOL) — write,trunc" on page 203
79	AUE_OPEN_WTC	"open(2TSOL) — write,trunc,creat" on page 202
42	AUE_RENAME	"rename(2TSOL)" on page 208
6	AUE_UNLINK	"unlink(2TSOL)" on page 231

Events in Audit Class fm

Table A-8 lists in alphabetical order the `file attribute modify` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-8 File Attribute Modify Audit Events (Default)

Audit Event Number and Event		Where Described
11	AUE_CHOWN	"chown(2TSOL)" on page 167
202	AUE_UTIME	"old utime(2)" on page 232
39	AUE_FCHMOD	"fchmod(2TSOL)" on page 173
38	AUE_FCHOWN	"fchown(2TSOL)" on page 174
30	AUE_FCNTL	"fcntl(2TSOL)" on page 175
45	AUE_FLOCK	Placeholder
158	AUE_IOCTL	"ioctl(2)" on page 183
237	AUE_LCHOWN	"lchown(2TSOL)" on page 184
49	AUE_UTIMES	"utimes(2TSOL)" on page 232
19	AUE_MCTL	No longer supported

Events in Audit Class fr

Table A-9 lists lists in alphabetical order the `file read` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-9 File Read Audit Events (Default)

Audit Event Number and Event		Where Described
72	AUE_OPEN_R	"open(2TSOL) — read" on page 197
73	AUE_OPEN_RC	"open(2TSOL) — read,creat" on page 198
74	AUE_OPEN_RT	"open(2TSOL) — read,trunc" on page 199
75	AUE_OPEN_RTC	"open(2TSOL) — read,trunc,creat" on page 198
80	AUE_OPEN_RW	"open(2TSOL) — read,write" on page 199
81	AUE_OPEN_RWC	"open(2TSOL) — read,write,creat" on page 200

Table A-9 File Read Audit Events (Default)

Audit Event Number and Event	Where Described
82 AUE_OPEN_RWT	"open(2TSOL) — read,write,trunc" on page 201
83 AUE_OPEN_RWTC	"open(2TSOL) — read,write,trunc,creat" on page 200
22 AUE_READLINK	"readlink(2TSOL)" on page 207

Events in Audit Class fw

Table A-10 lists in alphabetical order the `file write` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-10 File Write Audit Events (Default)

Audit Event Number and Event	Where Described
80 AUE_OPEN_RW	"open(2TSOL) — read,write" on page 199
81 AUE_OPEN_RWC	"open(2TSOL) — read,write,creat" on page 200
82 AUE_OPEN_RWT	"open(2TSOL) — read,write,trunc" on page 201
83 AUE_OPEN_RWTC	"open(2TSOL) — read,write,trunc,creat" on page 200
76 AUE_OPEN_W	"open(2TSOL) — write" on page 201
77 AUE_OPEN_WC	"open(2TSOL) — write,creat" on page 202
78 AUE_OPEN_WT	"open(2TSOL) — write,trunc" on page 203
79 AUE_OPEN_WTC	"open(2TSOL) — write,trunc,creat" on page 202

Events in Audit Class io

The `ioctl` audit class contains no audit events in the default configuration.

Events in Audit Class ip

Table A-11 lists in alphabetical order the `ipc` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-11 IPC Audit Events (Default)

Audit Event Number and Event		Where Described
514	AUE_GETMSGQCMWLABEL	“getmsgqcmwlabel(2TSOL)” on page 179
515	AUE_GETSEMCMWLABEL	“getsemcmwlabel(2TSOL)” on page 180
516	AUE_GETSHMCMWLABEL	“getshmcmwlabel(2TSOL)” on page 181
84	AUE_MSGCTL	Illegal command
85	AUE_MSGCTL_RMID	“msgctl(2TSOL) — IPC_RMID command” on page 191
86	AUE_MSGCTL_SET	“msgctl(2TSOL) — IPC_SET command” on page 192
87	AUE_MSGCTL_STAT	“msgctl(2TSOL) — IPC_STAT command” on page 193
88	AUE_MSGGET	“msgget(2TSOL)” on page 193
174	AUE_MSGGETL	“msggetl(2TSOL)” on page 194
89	AUE_MSGRCV	“msgrcv(2) / msgrcvl(2TSOL)” on page 195
175	AUE_MSGRCVL	“msgrcv(2) / msgrcvl(2TSOL)” on page 195
90	AUE_MSGSND	“msgsnd(2)” on page 195
176	AUE_MSGSNDL	“msgsndl(2TSOL)” on page 196
98	AUE_SEMCTL	Illegal command
105	AUE_SEMCTL_GETALL	“semctl(2TSOL) — getall” on page 209
102	AUE_SEMCTL_GETNCNT	“semctl(2TSOL) — GETNCNT command” on page 209
103	AUE_SEMCTL_GETPID	“semctl(2TSOL) — GETPID command” on page 210
104	AUE_SEMCTL_GETVAL	“semctl(2TSOL) — GETVAL command” on page 210
106	AUE_SEMCTL_GETZCNT	“semctl(2TSOL) — GETZCNT command” on page 211

Table A-11 IPC Audit Events (Default)

Audit Event Number and Event		Where Described
99	AUE_SEMCTL_RMID	"semctl(2TSOL) — IPC_RMID command" on page 211
100	AUE_SEMCTL_SET	"semctl(2TSOL) — IPC_SET command" on page 212
108	AUE_SEMCTL_SETALL	"semctl(2TSOL) — SETALL command" on page 212
107	AUE_SEMCTL_SETVAL	"semctl(2TSOL) — SETVAL command" on page 213
101	AUE_SEMCTL_STAT	"semctl(2TSOL) — IPC_STAT command" on page 213
109	AUE_SEMGET	"semget(2TSOL)" on page 214
177	AUE_SEMGETL	"semgetl(2TSOL)" on page 215
110	AUE_SEMOP	"semop(2TSOL)" on page 215
517	AUE_SEMOPL	"semopl(2TSOL)" on page 216
96	AUE_SHMAT	"shmat(2)" on page 223
91	AUE_SHMCTL	Placeholder
92	AUE_SHMCTL_RMID	"shmctl(2) — IPC_RMID command" on page 224
93	AUE_SHMCTL_SET	"shmctl(2TSOL) — IPC_SET command" on page 224
94	AUE_SHMCTL_STAT	"shmctl(2TSOL) — IPC_STAT command" on page 225
97	AUE_SHMDT	"shmdt(2)" on page 225
95	AUE_SHMGET	"shmget(2TSOL)" on page 226
178	AUE_SHMGETL	"shmgetl(2TSOL)" on page 227

Events in Audit Class io

Table A-12 lists in alphabetical order the `login or logout` class of audit

events provided in the Trusted Solaris 2.5 release.

Table A-12 Login or Logout Audit Events (Default)

Audit Event Number and Event		Where Described
6165	AUE_ftpd	“in.ftpd(1MTSOL) ftp access” on page 246
6152	AUE_login	“login(1TSOL) — local” on page 247
6153	AUE_logout	“logout(1)” on page 248
6163	AUE_passwd	“passwd(1)” on page 250
6164	AUE_rexd	“rpc.rexd(1M)” on page 253
6162	AUE_rexecd	“in.rexecd(1MTSOL)” on page 253
6155	AUE_rlogin	“login(1TSOL) — rlogin” on page 247
6158	AUE_rshd	“in.rshd(1MTSOL) rsh access” on page 254
6159	AUE_su	“su(1M)” on page 257
6154	AUE_telnet	“login(1TSOL) — telnet” on page 247

Events in Audit Class na

Table A-13 lists in alphabetical order the `non-attribute` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-13 Non-attribute Audit Events (Default)

Audit Event Number and Event		Where Described
153	AUE_ENTERPROM	“enter prom” on page 170
154	AUE_EXITPROM	“exit prom” on page 172
113	AUE_SYSTEMBOOT	“system booted” on page 230
6151	AUE_inetd_connect	“inetd(1MTSOL)” on page 246
6156	AUE_mountd_mount	“mountd(1M) – NFS mount” on page 249
6157	AUE_mountd_umount	“mountd(1MTSOL) – NFS unmount” on page 250

A

Events in Audit Class no

Table A-14 lists in alphabetical order the `invalid class` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-14 Invalid Class Audit Events (Default)

Audit Event Number and Event		Where Described
211	AUE_AUDIT	“audit(2TSOL)” on page 159
209	AUE_DUP2	No longer supported
208	AUE_FSTAT	“fstatfs(2)” on page 176
193	AUE_GETDENTS	No longer supported
13	AUE_JUNK	
194	AUE_LSEEK	Placeholder
518	AUE_MAC	“MAC check” on page 235
210	AUE_MMAP	“mmap(2)” on page 188
242	AUE_MODCTL	Placeholder
197	AUE_NFS	Placeholder
0	AUE_NULL	Indirect system call
185	AUE_PIPE	“pipe(2)” on page 204
192	AUE_READ	Placeholder
206	AUE_READL	Placeholder
198	AUE_READV	Placeholder
207	AUE_READVL	Placeholder
189	AUE_RECV	Placeholder
187	AUE_SEND	Placeholder
519	AUE_OFLOAT	“information label float - object” on page 182
520	AUE_SFLOAT	“information label float - subject” on page 181
186	AUE_SOCKETPAIR	Placeholder
521	AUE_UPRIV	“Use of privilege” on page 235
195	AUE_WRITE	Placeholder
196	AUE_WRITEV	Placeholder

Events in Audit Class nt

Table A-15 lists in alphabetical order the `network` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-15 Network Audit Events (Default)

Audit Event Number and Event		Where Described
33	AUE_ACCEPT	No longer supported
34	AUE_BIND	No longer supported
32	AUE_CONNECT	No longer supported
217	AUE_GETMSG	“getmsg(2)” on page 177
219	AUE_GETPMSG	“getpmsg(2)” on page 179
173	AUE_ONESIDE	No longer supported
216	AUE_PUTMSG	“putmsg(2)” on page 205
218	AUE_PUTPMSG	“putpmsg(2)” on page 207
191	AUE_RECVFROM	No longer supported
190	AUE_RECVMSG	No longer supported
188	AUE_SENDMSG	No longer supported
184	AUE_SENDTO	No longer supported
35	AUE_SETSOCKOPT	No longer supported
45	AUE_SHUTDOWN	No longer supported
247	AUE_SOCKACCEPT	“getmsg(2) — accept” on page 178
248	AUE_SOCKCONNECT	“putmsg(2)-connect” on page 206
183	AUE_SOCKET	No longer supported
250	AUE_SOCKRECEIVE	“getmsg(2) — receive” on page 178
249	AUE_SOCKSEND	“putmsg(2)-send” on page 206

A

Events in Audit Class ot

Table A-16 lists in alphabetical order the `other` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-16 Other Audit Events (Default)

Audit Event Number and Event		Where Described
238	AUE_MEMCNTL	“memcntl(2)” on page 186

Events in Audit Class pc

Table A-17 lists in alphabetical order the `process` class of audit events provided in the Trusted Solaris 2.5 release.

Table A-17 Process Audit Events (Default)

Audit Event Number and Event		Where Described
24	AUE_CHROOT	“chroot(2TSOL)” on page 168
1	AUE_EXIT	“exit(2)” on page 172
68	AUE_FCHDIR	“fchdir(2TSOL)” on page 173
69	AUE_FCHROOT	“fchroot(2TSOL)” on page 174
2	AUE_FORK	“fork(2TSOL)” on page 175
241	AUE_FORK1	“fork1(2TSOL)” on page 176
15	AUE_KILL	“kill(2TSOL)” on page 184
52	AUE_KILLPG	No longer supported
203	AUE_NICE	“old nice(2TSOL)” on page 197
204	AUE_OSETPGRP	
200	AUE_OSETUID	No longer supported
212	AUE_PRIOCNTLSYS	“priocntl(2TSOL)” on page 204
214	AUE_SETEGID	“setegid(2TSOL)” on page 220
215	AUE_SETEUID	“seteuid(2TSOL)” on page 220
205	AUE_SETGID	“old setgid(2TSOL)” on page 221
26	AUE_SETGROUPS	“setgroups(2TSOL)” on page 221

Table A-17 Process Audit Events (Default)

Audit Event Number and Event		Where Described
27	AUE_SETPGRP	“setpgrp(2)” on page 222
31	AUE_SETPRIORITY	No longer supported
41	AUE_SETREGID	Placeholder
40	AUE_SETREUID	Placeholder
	AUE_SETUID (event name is AUE_OSETUID	“old setuid(2TSOL)” on page 223
25	AUE_VFORK	“vfork(2TSOL)” on page 233
36	AUE_VTRACE	“vtrace(2)” on page 234

A

Audit Events in Alphabetical Order

Table A-18 lists in alphabetical order the `all` class of audit events provided in the Trusted Solaris 2.5 release. The list is alphabetical by type of event, kernel events before user events.

Table A-18 Audit Event to Audit Class Mappings *(1 of 12)*

Audit Event Number and Event	Where Described	Audit Class
33 AUE_ACCEPT		network
14 AUE_ACCESS	"access(2TSOL)" on page 157	file attribute access
18 AUE_ACCT	"acct(2)" on page 158	administrative
50 AUE_ADJTIME	"adjtime(2TSOL)" on page 158	administrative
57 AUE_ASYNC_DAEMON		administrative
114 AUE_ASYNC_DAEMON_EXIT		administrative
211 AUE_AUDIT	"audit(2TSOL)" on page 159	invalid class
224 AUE_AUDITON_GETCAR	"auditon(2TSOL) — get current active root" on page 159	administrative
231 AUE_AUDITON_GETCLASS	"auditon(2TSOL) — get event class" on page 159	administrative
229 AUE_AUDITON_GETCOND	"auditon(2TSOL) — get audit state" on page 160	administrative
223 AUE_AUDITON_GETCWD	"auditon(2TSOL) — get current working directory" on page 160	administrative
221 AUE_AUDITON_GETKMASK	"auditon(2TSOL) — get kernel mask" on page 160	administrative
225 AUE_AUDITON_GETSTAT	"auditon(2TSOL) — get audit statistics" on page 161	administrative
141 AUE_AUDITON_GPOLICY	"auditon(2TSOL) — GETPOLICY command" on page 161	administrative
145 AUE_AUDITON_GQCTRL	"auditon(2TSOL) — get audit queue control parameters" on page 161	administrative
139 AUE_AUDITON_GTERMID		administrative
144 AUE_AUDITON_SESTATE		administrative

Table A-18 Audit Event to Audit Class Mappings *(2 of 12)*

Audit Event Number and Event		Where Described	Audit Class
232	AUE_AUDITON_SETCLASS	"auditon(2TSOL) — set event class" on page 162	administrative
230	AUE_AUDITON_SETCOND	"auditon(2TSOL) — set audit state" on page 162	administrative
222	AUE_AUDITON_SETKMASK	"auditon(2TSOL) — set kernel mask" on page 163	administrative
228	AUE_AUDITON_SETSMASK	"auditon(2TSOL) — set mask per session ID" on page 163	administrative
226	AUE_AUDITON_SETSTAT	"auditon(2TSOL) — reset audit statistics" on page 163	administrative
227	AUE_AUDITON_SETUMASK	"auditon(2TSOL) — set mask per uid" on page 164	administrative
142	AUE_AUDITON_SPOLICY	"auditon(2TSOL) — SETPOLICY command" on page 164	administrative
146	AUE_AUDITON_SQCTRL	"auditon(2TSOL) — set audit queue control parameters" on page 165	administrative
140	AUE_AUDITON_STERMID		administrative
150	AUE_AUDITSTAT	"auditstat(2)" on page 165	administrative
136	AUE_AUDITSVC	"auditsvc(2TSOL)" on page 166	administrative
220	AUE_AUDITSYS		file attr access
34	AUE_BIND		network
66	AUE_BSMSYS		file attr access
8	AUE_CHDIR	"chdir(2TSOL)" on page 166	file create
10	AUE_CHMOD	"chmod(2TSOL)" on page 167	administrative
11	AUE_CHOWN	"chown(2TSOL)" on page 167	file attr modify
24	AUE_CHROOT	"chroot(2TSOL)" on page 168	process
513	AUE_CLOCK_SETTIME	"clock_settime(3RTSOL)" on page 168	administrative
112	AUE_CLOSE	"close(2)" on page 169	file close
32	AUE_CONNECT		network
111	AUE_CORE	"process dumped core" on page 205	file create
4	AUE_CREAT	"creat(2TSOL)" on page 169	file create

Table A-18 Audit Event to Audit Class Mappings *(3 of 12)*

Audit Event Number and Event		Where Described	Audit Class
209	AUE_DUP2		invalid class
153	AUE_ENTERPROM	“enter prom” on page 170	non-attribute
7	AUE_EXEC	“exec(2TSOL)” on page 171	process, exec
23	AUE_EXECVE	“execve(2TSOL)” on page 171	process, exec
1	AUE_EXIT	“exit(2)” on page 172	process
154	AUE_EXITPROM	“exit prom” on page 172	non-attribute
61	AUE_EXPORTFS		administrative
68	AUE_FCHDIR	“fchdir(2TSOL)” on page 173	process
39	AUE_FCHMOD	“fchmod(2TSOL)” on page 173	file attr modify
38	AUE_FCHOWN	“fchown(2TSOL)” on page 174	file attr modify
69	AUE_FCHROOT	“fchroot(2TSOL)” on page 174	process
30	AUE_FCNTL	“fcntl(2TSOL)” on page 175	file attr modify
45	AUE_FLOCK		file attr modify
2	AUE_FORK	“fork(2TSOL)” on page 175	process
241	AUE_FORK1	“fork1(2TSOL)” on page 176	process
208	AUE_FSTAT		invalid class
55	AUE_FSTATFS	“fstatfs(2)” on page 176	file attr access
44	AUE_FTRUNCATE		file delete
132	AUE_GETAUDIT	“getaudit(2TSOL)” on page 177	administrative
130	AUE_GETAUID	“getauid(2TSOL)” on page 177	administrative
193	AUE_GETDENTS		invalid class
147	AUE_GETKERNSTATE		administrative
217	AUE_GETMSG	“getmsg(2)” on page 177	network
514	AUE_GETMSGQCMWLABEL	“getmsgqcmwlabel(2TSOL)” on page 179	ipc
219	AUE_GETPMSG	“getpmsg(2)” on page 179	network
149	AUE_GETPORTAUDIT	“getportaudit(2)” on page 180	administrative
515	AUE_GETSEMCMWLABEL	“getsemcmwlabel(2TSOL)” on page 180	ipc

Table A-18 Audit Event to Audit Class Mappings *(4 of 12)*

Audit Event Number and Event		Where Described	Audit Class
516	AUE_GETSHMCMWLABEL	“getshmcmwlabel(2TSOL)” on page 181	ipc
134	AUE_GETUSERAUDIT		administrative
519	AUE_OFLOAT	“information label float - object” on page 182	invalid class
158	AUE_IOCTL	“ioctl(2)” on page 183	file attr modify
13	AUE_JUNK		invalid class
15	AUE_KILL	“kill(2TSOL)” on page 184	process
52	AUE_KILLPG		process
237	AUE_LCHOWN	“lchown(2TSOL)” on page 184	file attr modify
5	AUE_LINK	“link(2TSOL)” on page 185	file create
194	AUE_LSEEK		invalid class
17	AUE_LSTAT	“lstat(2TSOL)” on page 185	file attr access
236	AUE_LXSTAT	“lxstat(2)” on page 186	file attr access
518	AUE_MAC	“MAC check” on page 235	invalid class
19	AUE_MCTL		file attr modify
238	AUE_MEMCNTL	“memcntl(2)” on page 186	other
47	AUE_MKDIR	“mkdir(2TSOL)” on page 187	file create
9	AUE_MKNOD	“mknod(2TSOL)” on page 187	administrative
210	AUE_MMAP	“mmap(2)” on page 188	invalid class
246	AUE_MODADDMAJ	“modctl(2) — bind module” on page 188	administrative
245	AUE_MODCONFIG	“modctl(2) — configure module” on page 189	administrative
242	AUE_MODCTL		invalid class
243	AUE_MODLOAD	“modctl(2) — load module” on page 189	administrative
244	AUE_MODUNLOAD	“modctl(2) — unload module” on page 190	administrative
62	AUE_MOUNT	“mount(2TSOL)” on page 190	administrative
84	AUE_MSGCTL		ipc
85	AUE_MSGCTL_RMID	“msgctl(2TSOL) — IPC_RMID command” on page 191	ipc

Table A-18 Audit Event to Audit Class Mappings *(5 of 12)*

Audit Event Number and Event		Where Described	Audit Class
86	AUE_MSGCTL_SET	"msgctl(2TSOL) — IPC_SET command" on page 192	ipc
87	AUE_MSGCTL_STAT	"msgctl(2TSOL) — IPC_STAT command" on page 193	ipc
88	AUE_MSGGET	"msgget(2TSOL)" on page 193	ipc
174	AUE_MSGGETL		ipc
89	AUE_MSGRCV	"msgrcv(2) / msgrcvl(2TSOL)" on page 195	ipc
175	AUE_MSGRCVL		ipc
90	AUE_MSGSND	"msgsnd(2)" on page 195	ipc
176	AUE_MSGSNDL	"msgsndl(2TSOL)" on page 196	ipc
64	AUE_MSGSYS		file attr access
213	AUE_MUNMAP	"munmap(2)" on page 196	file close
197	AUE_NFS		invalid class
115	AUE_NFSSVC_EXIT		administrative
58	AUE_NFS_GETFH		administrative
53	AUE_NFS_SVC		administrative
203	AUE_NICE	"old nice(2TSOL)" on page 197	process
0	AUE_NULL		invalid class
519	AUE_OFLOAT	"information label float - object" on page 182	invalid class
173	AUE_ONESIDE		network
3	AUE_OPEN		file attr access
72	AUE_OPEN_R	"open(2TSOL) — read" on page 197	file read
73	AUE_OPEN_RC	"open(2TSOL) — read,creat" on page 198	file read, create
74	AUE_OPEN_RT	"open(2TSOL) — read,trunc" on page 199	file read, delete
75	AUE_OPEN_RTC	"open(2TSOL) — read,trunc,creat" on page 198	file create, read, delete
80	AUE_OPEN_RW	"open(2TSOL) — read,write" on page 199	file read, write

Table A-18 Audit Event to Audit Class Mappings *(6 of 12)*

Audit Event Number and Event	Where Described	Audit Class
81 AUE_OPEN_RWC	“old nice(2TSOL)” on page 197	file read, write, create
82 AUE_OPEN_RWT	“open(2TSOL) — read,write,trunc” on page 201	file read, write, delete
83 AUE_OPEN_RWTC	“open(2TSOL) — read,write,trunc,creat” on page 200	file read, write, delete, create
76 AUE_OPEN_W	“open(2TSOL) — write” on page 201	file write
77 AUE_OPEN_WC	“open(2TSOL) — write,creat” on page 202	file write, create
78 AUE_OPEN_WT	“open(2TSOL) — write,trunc” on page 203	file write, delete
79 AUE_OPEN_WTC	“open(2TSOL) — write,trunc,creat” on page 202	file create, write, delete
204 AUE_OSETPGRP		process
200 AUE_OSETUID	“old setuid(2TSOL)” on page 223	process
199 AUE_OSTAT		file attr access
71 AUE_PATHCONF	“pathconf(2)” on page 203	file attr access
185 AUE_PIPE	“pipe(2)” on page 204	invalid class
212 AUE_PRIOCNTLSYS	“priocntl(2TSOL)” on page 204	process
216 AUE_PUTMSG	“putmsg(2)” on page 205	network
218 AUE_PUTPMSG	“putpmsg(2)” on page 207	network
60 AUE_QUOTACTL		administrative
192 AUE_READ		invalid class
206 AUE_READL		invalid class
22 AUE_READLINK	“readlink(2TSOL)” on page 207	file read
198 AUE_READV		invalid class
207 AUE_READVL		invalid class
20 AUE_REBOOT		administrative
189 AUE_RECV		invalid class

Table A-18 Audit Event to Audit Class Mappings *(7 of 12)*

Audit Event Number and Event		Where Described	Audit Class
191	AUE_RECVFROM		network
190	AUE_RECVMSG		network
42	AUE_RENAME	"rename(2TSOL)" on page 208	file create, delete
67	AUE_RFSSYS		file attr access
48	AUE_RMDIR	"rmdir(2TSOL)" on page 208	file create
98	AUE_SEMCTL		ipc
105	AUE_SEMCTL_GETALL	"semctl(2TSOL) — getall" on page 209	ipc
102	AUE_SEMCTL_GETNCNT	"semctl(2TSOL) — GETNCNT command" on page 209	ipc
103	AUE_SEMCTL_GETPID	"semctl(2TSOL) — GETPID command" on page 210	ipc
104	AUE_SEMCTL_GETVAL	"semctl(2TSOL) — GETVAL command" on page 210	ipc
106	AUE_SEMCTL_GETZCNT	"semctl(2TSOL) — GETZCNT command" on page 211	ipc
99	AUE_SEMCTL_RMID	"semctl(2TSOL) — IPC_RMID command" on page 211	ipc
100	AUE_SEMCTL_SET	"semctl(2TSOL) — IPC_SET command" on page 212	ipc
108	AUE_SEMCTL_SETALL	"semctl(2TSOL) — SETALL command" on page 212	ipc
107	AUE_SEMCTL_SETVAL	"semctl(2TSOL) — SETVAL command" on page 213	ipc
101	AUE_SEMCTL_STAT	"semctl(2TSOL) — IPC_STAT command" on page 213	ipc
109	AUE_SEMGET	"semget(2TSOL)" on page 214	ipc
177	AUE_SEMGETL	"semgetl(2TSOL)" on page 215	ipc
110	AUE_SEMOP	"semop(2TSOL)" on page 215	ipc
63	AUE_SEMSYS		file attr access
187	AUE_SEND		invalid class

Table A-18 Audit Event to Audit Class Mappings *(8 of 12)*

Audit Event Number and Event		Where Described	Audit Class
188	AUE_SENDMSG		network
184	AUE_SENDTO		network
133	AUE_SETAUDIT	“setaudit(2TSOL)” on page 217	administrative
131	AUE_SETAUID	“setauid(2TSOL)” on page 217	administrative
117	AUE_SETCLEARANCE	“setclearance(2TSOL)” on page 218	sensitivity label
116	AUE_SETCMWPLABEL	“setcmwplabel(2TSOL)” on page 219	sensitivity label
59	AUE_SETDOMAINNAME		administrative
214	AUE_SETEGID	“setegid(2TSOL)” on page 220	process
215	AUE_SETEUID	“seteuid(2TSOL)” on page 220	process
205	AUE_SETGID	“old setgid(2TSOL)” on page 221	process
26	AUE_SETGROUPS	“setgroups(2TSOL)” on page 221	process
29	AUE_SETHOSTNAME		administrative
148	AUE_SETKERNSTATE		administrative
27	AUE_SETPGRP	“setpgrp(2)” on page 222	process
31	AUE_SETPRIORITY		process
41	AUE_SETREGID		process
40	AUE_SETREUID		process
51	AUE_SETRLIMIT	“setrlimit(2TSOL)” on page 222	administrative
35	AUE_SETSOCKOPT		network
37	AUE_SETTIMEOFDAY		administrative
135	AUE_SETUSERAUDIT		administrative
520	AUE_SFLOAT	“information label float - subject” on page 181	invalid class
	AUE_SETUID	See “old setuid(2TSOL)” on page 223	
96	AUE_SHMAT	“shmat(2)” on page 223	ipc
91	AUE_SHMCTL		ipc

Table A-18 Audit Event to Audit Class Mappings *(9 of 12)*

Audit Event Number and Event	Where Described	Audit Class
92 AUE_SHMCTL_RMID	"shmctl(2) — IPC_RMID command" on page 224	ipc
93 AUE_SHMCTL_SET	"shmctl(2TSOL) — IPC_SET command" on page 224	ipc
94 AUE_SHMCTL_STAT	"shmctl(2TSOL) — IPC_STAT command" on page 225	ipc
97 AUE_SHMDT	"shmdt(2)" on page 225	ipc
95 AUE_SHMGET	"shmget(2TSOL)" on page 226	ipc
178 AUE_SHMGETL	"shmgetl(2TSOL)" on page 227	ipc
65 AUE_SHMSYS		file attr access
45 AUE_SHUTDOWN		network
247 AUE_SOCKACCEPT	"getmsg(2) — accept" on page 178	network
248 AUE_SOCKCONNECT	"putmsg(2)-connect" on page 206	network
183 AUE_SOCKET		network
186 AUE_SOCKETPAIR		invalid class
250 AUE_SOCKRECEIVE	"getmsg(2) — receive" on page 178	network
249 AUE_SOCKSEND	"putmsg(2)-send" on page 206	network
16 AUE_STAT	"stat(2TSOL)" on page 227	file attr access
54 AUE_STATFS	"statfs(2)" on page 228	file attr access
234 AUE_STATVFS	"statvfs(2TSOL)" on page 228	file attr access
201 AUE_STIME	"stime(2TSOL)" on page 229	administrative
28 AUE_SWAPON		administrative
21 AUE_SYMLINK	"symlink(2TSOL)" on page 229	file create
239 AUE_SYSINFO	"sysinfo(2TSOL)" on page 230	administrative
113 AUE_SYSTEMBOOT	"system booted" on page 230	non-attribute
43 AUE_TRUNCATE		file delete
12 AUE_UMOUNT	"umount(2TSOL) — old version" on page 231	administrative
6 AUE_UNLINK	"unlink(2TSOL)" on page 231	file delete

Table A-18 Audit Event to Audit Class Mappings *(10 of 12)*

Audit Event Number and Event		Where Described	Audit Class
56	AUE_UNMOUNT		administrative
521	AUE_UPRIV	“Use of privilege” on page 235	invalid class
202	AUE_UTIME	“old utime(2)” on page 232	file attr modify
49	AUE_UTIMES	“utimes(2TSOL)” on page 232	file attr modify
233	AUE_UTSSYS	“utssys(2) — fusers” on page 233	administrative
25	AUE_VFORK	“vfork(2TSOL)” on page 233	process
25	AUE_VPIXSYS		file attr access
36	AUE_VTRACE	“vtrace(2)” on page 234	process
195	AUE_WRITE		invalid class
128	AUE_WRITEL		sl
196	AUE_WRITEV		invalid class
129	AUE_WRITEVL		sl
240	AUE_XMKNOD	“xmknod(2)” on page 234	file create
235	AUE_XSTAT	“xstat(2)” on page 235	file attr access
9018	AUE_add_drv	“add_drv(1MTSOL)” on page 236	administrative
6200	AUE_allocate_succ	“allocate(1MTSOL) device success” on page 237	administrative
6201	AUE_allocate_fail	“allocate(1MTSOL) device failure” on page 237	administrative
6144	AUE_at_create	“at(1TSOL) - create crontab” on page 239	administrative
6145	AUE_at_delete	“at(1TSOL) - delete atjob file (at or atrm)” on page 239	administrative
6146	AUE_at_perm	“at(1TSOL) - permission” on page 240	administrative
9016	AUE_audit	“audit(1MTSOL)” on page 240	administrative
9015	AUE_auditwrite	“auditwrite(3TSOL)” on page 241	administrative
9029	AUE_chroot_cmd	“chroot(1MTSOL)” on page 241	administrative
6147	AUE_cron_invoke	“crontab(1TSOL) - invoke atjob or crontab” on page 243	administrative
6148	AUE_crontab_create	“crontab(1TSOL) - crontab created” on page 242	administrative

Table A-18 Audit Event to Audit Class Mappings *(11 of 12)*

Audit Event Number and Event		Where Described	Audit Class
6149	AUE_crontab_delete	"crontab(1TSOL) - crontab deleted" on page 242	administrative
6150	AUE_crontab_perm	"crontab(1TSOL) - permission" on page 243	administrative
6202	AUE_deallocate_succ	"allocate(1MTSOL) device success" on page 237	administrative
6203	AUE_deallocate_fail	"deallocate(1MTSOL) device failure" on page 238	administrative
9025	AUE_dispadmin	"dispadmin(1MTSOL)" on page 244	administrative
9032	AUE_eeprom	"eeprom(1MTSOL)" on page 244	administrative
	AUE_exportfs	"share(1M)" on page 257	non-attribute
	AUE_exportfs	"unshare(1M)" on page 260	non-attribute
6165	AUE_ftpd	"in.ftpd(1MTSOL) ftp access" on page 246	login
9031	AUE_fuser	"fuser(1MTSOL)" on page 245	administrative
6160	AUE_halt_solaris	"halt(1MTSOL)" on page 245	administrative
6151	AUE_inetd_connect	"inetd(1MTSOL)" on page 246	non-attribute
6205	AUE_listdevice_succ	"allocate(1MTSOL) - list devices success" on page 238	administrative
6206	AUE_listdevice_fail	"allocate(1MTSOL) - list devices failure" on page 238	administrative
6152	AUE_login	"login(1TSOL) — local" on page 247	login
6153	AUE_logout	"logout(1)" on page 248	login
9020	AUE_modload	"modload(1MTSOL)" on page 248	administrative
9021	AUE_modunload	"modload(1MTSOL)" on page 249	administrative
6156	AUE_mountd_mount	"mountd(1M) – NFS mount" on page 249	non-attribute
6157	AUE_mountd_umount	"mountd(1MTSOL) – NFS unmount" on page 250	non-attribute
6163	AUE_passwd	"passwd(1)" on page 250	login
9026	AUE_pbind	"pbind(1M)" on page 251	administrative
9010	AUE_pfsh_nopriv	"pfsh(1MTSOL)" on page 251	application
9009	AUE_pfsh_priv	"pfsh(1MTSOL)" on page 251	administrative
9008	AUE_pfsh_trusted_nopriv	"pfsh(1MTSOL)" on page 251	administrative

Table A-18 Audit Event to Audit Class Mappings *(12 of 12)*

Audit Event Number and Event		Where Described	Audit Class
9007	AUE_pfsh_trusted_priv	"pfsh(1MTSOL)" on page 251	administrative
9027	AUE_psradm	"psradm(1MTSOL)" on page 252	administrative
6161	AUE_reboot_solaris	"reboot(1MTSOL)" on page 252	administrative
6164	AUE_rexd	"rpc.rexd(1M)" on page 253	login
6162	AUE_rexecd	"in.rexecd(1MTSOL)" on page 253	login
9019	AUE_rem_drv	"rem_drv(1MTSOL)" on page 254	administrative
6155	AUE_rlogin	"login(1TSOL) — rlogin" on page 247	login
6158	AUE_rshd	"in.rshd(1MTSOL) rsh access" on page 254	login
9028	AUE_run_level_change	"init(1M) - run level change" on page 255	administrative
9014	AUE_sendmail_defer	"sendmail(1MTSOL)" on page 255	administrative
9013	AUE_sendmail_deliver	"sendmail(1MTSOL)" on page 255	administrative
9012	AUE_sendmail_upgrade	"sendmail(1MTSOL) - upgrade" on page 256	administrative
9017	AUE_uauth	"uauth" on page 259	administrative
9022	AUE_setuname	"setuname(1MTSOL)" on page 256	administrative
6159	AUE_su	"su(1M)" on page 257	login
9030	AUE_swap	"swap(1MTSOL)" on page 258	administrative
6154	AUE_telnet	"login(1TSOL) — telnet" on page 247	login
9023	AUE_uadmin_cmd	"uadmin(1MTSOL)" on page 258	administrative
9024	AUE_uname_set	"uname(1TSOL)" on page 259	administrative

A

Audit Record Descriptions

B

This appendix has two parts. The first part describes each part of an audit record structure and each audit token structure. The second part defines all of the audit records generated in Trusted Solaris software by event description.

B

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 B-1 shows a typical audit record.

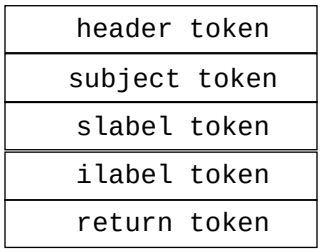


Figure B-1 Typical Audit Record

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 42 current tokens are shown in Table B-1. Those marked Yes in "TS2.5" are in Trusted Solaris 2.5 only. Those not marked Yes are modified versions of audit tokens from the Solaris Basic Security Module. The token scheme can be extended.

Table B-1 Trusted Solaris Audit Tokens

Token Name	Description	TS2.5
`acl`	access control list information	Yes
`arbitrary`	Data with format and type information	
`arg`	System call argument value	
`attr`	Vnode tokens	
`clearance`	clearance information	Yes
`exec_args`	Exec system call arguments	
`exec_env`	Exec system call environment variables	
`exit`	Program exit information	
`file`	Audit file information	

Table B-1 Trusted Solaris Audit Tokens

Token Name	Description	TS2.5
`groups`	Process groups information (obsolete)	
`header`	Indicates start of record	
`host`	indicates the host where the audit record was collected	Yes
`ilabel`	information label information	Yes
`in_addr`	Internet address	
`ip`	IP header information	
`ipc`	System V IPC information	
`ipc_perm`	System V IPC object tokens	
`iport`	Internet port address	
`liaison`	liaison information for Trusted Networking	Yes
`newgroups`	Process groups information	
`opaque`	Unstructured data (unspecified format)	
`path`	Path information (path)	
`priv`	use of privilege information	Yes
`privilege`	privilege set information	Yes
`process`	Process token information	
`return`	Status of system call	
`seq`	Sequence number token	
`slabel`	sensitivity label information	Yes
`socket`	Socket type and addresses	
`socket-inet`	Socket port and address	
`subject`	Subject token information (same structure as `process` token)	
`text`	ASCII string	
`trailer`	Indicates end of record	
`xatom`	X window atom identification	Yes
`xcolormap`	X window color information	Yes
`xcursor`	X window cursor information	Yes

Table B-1 Trusted Solaris Audit Tokens

Token Name	Description	TS2.5
`xfont`	X window font information	Yes
`xgc`	X window graphical context information	Yes
`xpixmap`	Xwindow pixel mapping information	Yes
`xproperty`	X window property information	Yes
`xselect`	X window data information	Yes
`xwindow`	X window window information	Yes

An audit record always contains a `header` token and may contain a `trailer` token. The `header` token indicates where the audit record begins in the audit trail. The optional `trailer` token allows backward seeks of 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. For an example of how to read an audit record, go to “Reading an Audit Record” on page 69.

`acl` *Token*

The `acl` token provides information about any access control lists in place on an object. If there is no current `acl`, this token is not written to the audit record. Also, unless required by an audit record format, this token is normally recorded only when the appropriate auditing policy is set. The fields are:

- A token ID
- A count of the number of element arrays in the token
- The object type of an array element
- The user / group id of an array element
- The permissions given to the subject

Figure B-2 shows the token format.

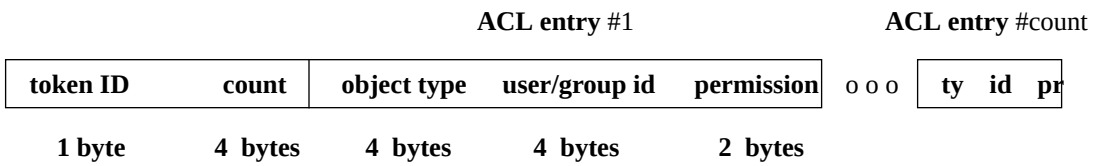


Figure B-2 `acl` Token Format

An `acl` token is displayed by `praudit(1MTSOL)` as follows:

```
acl,
```

`arbitrary` *Token*

The `arbitrary` token encapsulates data for the audit trail. It consists of four fixed fields and an array of data. 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

Figure B-3 shows the token format.

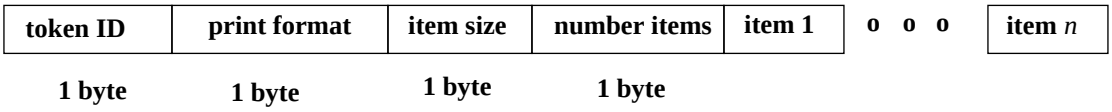


Figure B-3 `arbitrary` Token Format

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

Table B-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 B-3.

Table B-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)
`AUR_LONGLONG`	Data is in units of longlongs (8 bytes)

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

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

`arg` *Token*

The `arg` 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

Figure B-4 shows the token format.

token ID	argument #	argument value	text length	text
1 byte	1 byte	4 bytes	2 bytes	*n* bytes

Figure B-4 `arg` Token Format

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

```
argument,2,0xff87a2fc,strioctl:vnode
```

`attr` *Token*

The `attr` token contains information from the file vnode. 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. 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

See the `statvfs(2)` man page for further information about the file system ID and the device ID. Figure B-5 shows the 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 B-5 `attr` Token Format

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

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

`clearance` *Token*

The clearance token contains Trusted Solaris clearance information. The fields are:

- A token ID
- The CMW clearance, containing
  - A pad ID identifying the label type
  - The clearance's classifications
  - The clearance's compartments

Figure B-6 shows the token format.

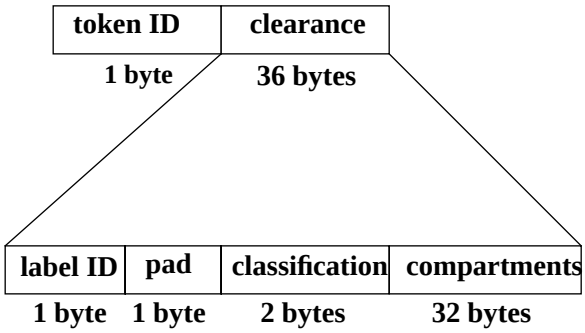


Figure B-6 `clearance` Token Format

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

```
clearance,TOP SECRET
```

`exec_args` *Token*

The `exec_args` token records the arguments to an exec system call. The fields are:

- A token ID

- A count that represents the number of arguments passed to the `exec` call
- Zero or more null-terminated strings, the arguments of the `exec` call

Figure B-7 shows an `exec_args` token.

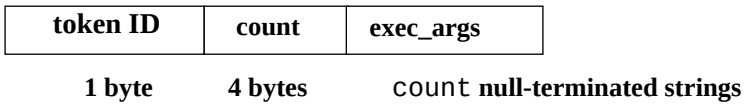


Figure B-7 `exec_args` Token Format

Note – The `exec_args` token is output only when the audit policy `argv` is active. See “Setting Audit Policies” on page 25 for more information.

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

```
exec_args,
```

`exec_env` *Token*

The `exec_env` token records the current environment variables to an `exec` system call. The fields are:

- A token ID
- A count of the current environment variables in the `exec` call
- Zero or more null-terminated strings, the variables of the `exec` call

Figure B-8 shows an `exec_env` token.

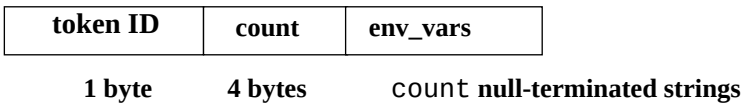


Figure B-8 `exec_env` Token Format

Note – The `exec_env` token is output only when the audit policy `arge` is active. See “Setting Audit Policies” on page 25 for more information.

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

```
exec_env,
```

`exit` *Token*

The `exit` token records the exit status of a program and a return value. 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

Figure B-9 shows an `exit` token.

token ID	status	return value
1 byte	4 bytes	4 bytes

Figure B-9 `exit` Token Format

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

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

`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 fields are:

- A token ID
- 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 (does not show)
- The file null-terminated name

Figure B-10 shows the token format.

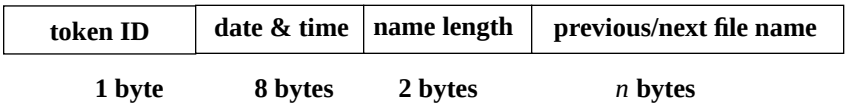


Figure B-10 `file` Token Format

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

```
file,Fri Jan 23 13:32:42 1997, + 79249 msec,
/etc/security/audit/patchwork/files/19920901202558.19920901203241.patchwork
```

`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. Note 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 fields are:

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

Figure B-11 shows a `groups` token.

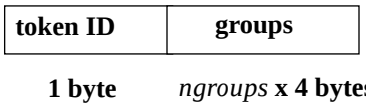


Figure B-11 `groups` Token Format

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,-1
```

Note – The `groups` token is output only when the audit policy `group` is active. See "The auditconfig Command" on page 23 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 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

Figure B-12 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 B-12 `header` Token Format

The event modifier field has the following flags defined:

Value	Constant Name	Description
0x0001	PAD_MACUSE	MAC decision was successful
0x0002	PAD_MACREAD	MAC read failure
0x0004	PAD_MACWRITE	MAC write failure
0x0008	PAD_MACSEARCH	MAC search failure
0x0010	PAD_MACKILL	MAC signal failure
0x0020	PAD_MACTRACE	MAC trace failure
0x0040	PAD_MACIOCTL	MAC ioctl failure
0x0080	PAD_SPRIVUSE	Successful use of privilege

Value	Constant Name	Description
0x0100	PAD_FPRIVUSE	Failed use of privilege
0x4000	PAD_NONATTR	Nonattributable event
0x8000	PAD_FAILURE	Failed audit event

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

```
header,408,3,ioctl(2),,Fri Jan 23 13:32:42 1997, + 79249 msec
```

`host` *Token*

The `host` token contains the machine ID for the workstation which generated this audit record. The fields are:

- A token ID
- The workstation ID of the host that generated the audit record

Figure B-13 shows the token format.

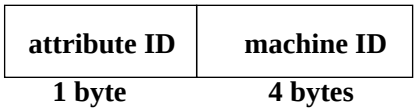


Figure B-13 `host` Token Format

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

```
host,patchwork
```

`ilabel` *Token*

The `ilabel` token contains an information label. The fields are:

- A token ID
- An information label

Figure B-14 shows the token format.

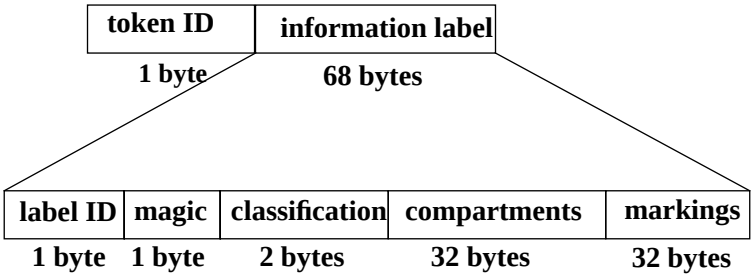


Figure B-14 `ilabel` Token Format

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

```
ilabel,ADMIN_LOW
```

`in_addr` *Token*

The `in_addr` token contains an Internet address. This 4-byte value is an Internet Protocol address. The fields are:

- A token ID
- An Internet address

Figure B-15 shows the token format.

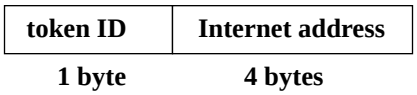


Figure B-15 `in_addr` Token Format

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

```
ip addr,129.150.110.3
```

`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 IP header structure is defined in `/usr/include/netinet/ip.h`. The fields are:

- A token ID
- A 20-byte copy of an IP header (all 20 bytes)

Figure B-16 shows the token format.

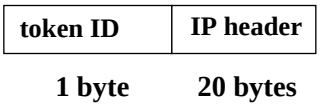


Figure B-16 `ip` Token Format

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

```
ip,0.0.0.0
```

`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. The fields are:

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

Figure B-17 shows the token format.

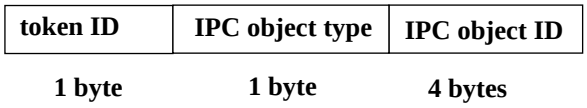


Figure B-17 `ipc` Token Format

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

```
IPC,msg,3
```

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 B-4. The values are defined in `/usr/include/bsm/audit.h`.

Table B-4 IPC Object Type Field

Name	Value	Description
`AU_IPC_MSG`	1	IPC message object
`AU_IPC_SEM`	2	IPC semaphore object
`AU_IPC_SHM`	3	IPC shared memory object

`ipc_perm` *Token*

The `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

The values are taken from the `ipc_perm` structure associated with the IPC object. Figure B-18 shows the 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 B-18 `ipc_perm` Token Format

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

```
IPC perm,root,wheel,root,wheel,0,0,0x00000000
```

`iport` *Token*

The `iport` token contains the TCP (or UDP) port address. The fields are:

- A token ID
- A TCP/UDP address

Figure B-19 shows the token format.

token ID	port ID
1 byte	2 bytes

Figure B-19 `iport` Token Format

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

```
iport,0xf6d6
```

`liaison` *Token*

The `liaison` token contains a liaison ID used by the Trusted Networking software. The fields are:

- A token ID
- The liaison ID

Figure B-20 shows the token format.

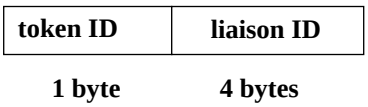


Figure B-20 `liaison` Token Format

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

```
liaison,17
```

`newgroups` *Token*

This token is the replacement for the `groups` token. Note 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 fields are:

- A token ID field
- A count of the number of groups contained in this audit record.
- Zero or more group entries.

Figure A-16 shows the token format.

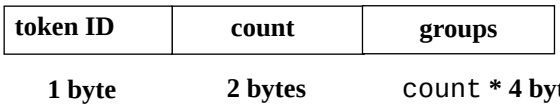


Figure B-21 `newgroups` Token Format

Note – The `newgroups` token is output only when the audit policy `group` is active. See "The auditconfig Command" on page 23 for more information.

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

```
newgroups,1,analysts
```

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

Figure B-22 shows the token format.

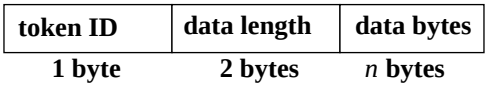


Figure B-22 opaque Token Format

An opaque token is displayed by praudit as follows:

```
opaque,12,0x4f504151554520444154 4100
```

path *Token*

The 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 to the object based on the real root of the system

Figure B-23 shows the token format.

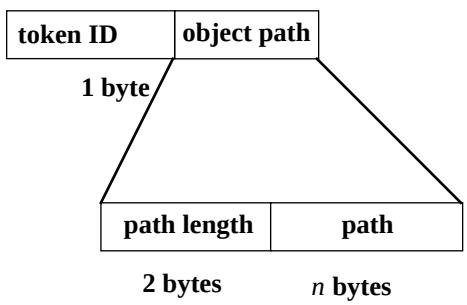


Figure B-23 path Token Format

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

```
path,/etc/security/audit/patchwork
```

`priv` *Token*

The `priv` token contains use of privilege information. The fields are:

- A token ID
- A success/failure field indicating whether the use of privilege was successful (1 success, 0 failure)
- The privilege being tested

Figure B-24 shows a `priv` token.

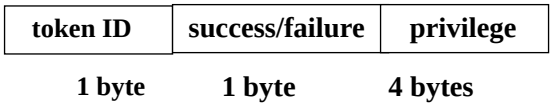


Figure B-24 `priv` Token Format

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

```
useofpriv,successful use of priv,net_reply_equal
```

`privilege` *Token*

The `privilege` token contains privilege information for an object or a subject. The fields are:

- A token ID
- The type of privilege
- The privilege set

where type is one of the following:

Value	Type
0	Unknown or Undefined
1	Forced
2	Allowed
3	Effective
4	Inheritable
5	Permitted
6	Saved

Figure B-25 shows the token format.

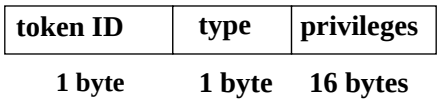


Figure B-25 `privilege` Token Format

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

```
privilege,1,proc_tcb_audit
```

`process` *Token*

The `process` token contains information describing a process as an object such as the recipient of a signal. 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 workstation ID

Figure B-26 shows the token format.

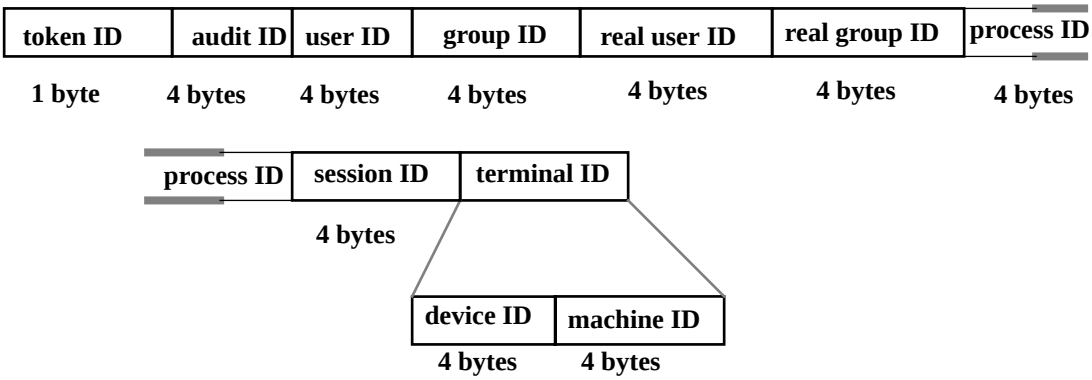


Figure B-26 Format for `process` and `subject` Tokens

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.

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

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

`return` *Token*

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). The token indicates exit status and other return values in application auditing. 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

Figure B-27 shows the token format.

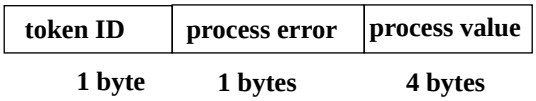


Figure B-27 `return` Token Format

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

```
return,failure: No such file or directory,-1
```

`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 fields are:

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

The sequence number is incremented every time an audit record is generated and put onto the audit trail. Figure B-28 shows the token format.

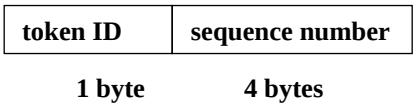


Figure B-28 `seq` Token Format

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

```
sequence,1292
```

`slabel` *Token*

The `slabel` token contains a sensitivity label. The fields are:

- A token ID
- A sensitivity label

Figure B-29 shows the token format.

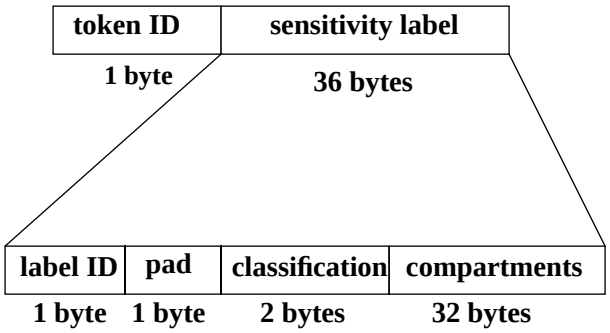


Figure B-29 `slabel` Token Format

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

```
slabel,ADMIN_LOW
```

`socket` *Token*

The `socket` token contains information describing 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

The socket type is taken from the designated socket and the port and Internet addresses are taken from the socket's *inpcb* control structure. Figure B-30 shows the token format.

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 B-30 `socket` Token Format

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

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

socket,0x0002,0x8008,patchwork
```

`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 fields are:

- A token ID
- A socket family field that indicates the Internet family (`AF_INET`, `AF_OSI`, and so on)
- The local port address
- The socket address

Figure B-31 shows the token format.

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

Figure B-31 `socket-inet` Token Format

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

```
socket,0x0002,0x8008,patchwork
```

`subject` *Token*

The `subject` token describes a subject (process). The structure is the same as the `process` token:

- 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 workstation ID

This token is always returned as part of kernel-generated audit records for system calls. The audit ID, user ID, group ID, process ID, and session ID are long instead of short. Figure B-26 on page 146 shows the token format.

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.

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

```
subject,install,root,root,root,root,276,264,0 0 patchwork
```

`text` *Token*

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

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

- A text string

Figure B-32 shows the token format.

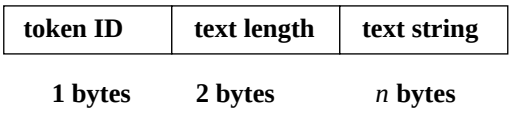


Figure B-32 `text` Token Format

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

```
text,emily
```

`trailer` *Token*

A `trailer` token it marks the end of an audit record to support backward seeks of the audit trail. 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 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

Figure B-33 shows the token format.

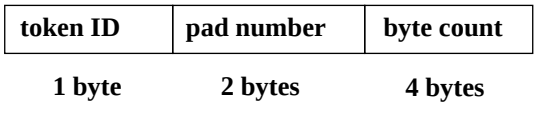


Figure B-33 `trailer` Token Format

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

```
trailer,136
```

`xatom` *Token*

The `xatom` token contains information concerning an X atom. The fields are:

- A token ID
- The string length
- A text string identifying the atom

Figure B-34 shows the token format.

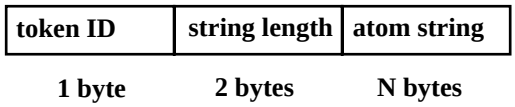


Figure B-34 `xatom` Token Format

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

```
xatom,
```

`xcolormap` *Token*

The `xcolormap` token contains information about the colormaps. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 shows the token format.

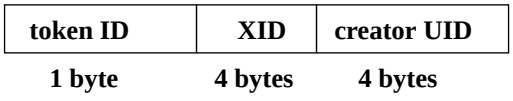


Figure B-35 Format for `xcolormap`, `xcursor`, `xfont`, `xgc`, `xpixmap`, and `xwindow` Tokens

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

```
xcolormap,
```

`xcursor` *Token*

The `xcursor` token contains information about the cursors. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 on page 152 shows shows the token format.

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

```
xcursor,
```

`xfont` *Token*

The `xfont` token contains information about the fonts. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 on page 152 shows the token format.

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

```
xfont,
```

`xgc` *Token*

The `xgc` token contains information about the xgc. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 on page 152 shows the token format.

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

```
xgc,
```

`xpixmap` *Token*

The `xpixmap` token contains information about the pixel mappings. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 on page 152 shows the token format.

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

```
xpixmap,
```

`xproperty` *Token*

The `xproperty` token contains information about various properties of a window. The fields are:

- A token ID
- The X server identifier
- The creator's user ID
- A string length
- A string (atom name)

Figure B-36 shows an `xproperty` token.

token ID	XID	creator UID	strlen	string (atom name)
1 byte	4 bytes	4 bytes	2 bytes	N bytes

Figure B-36 `xproperty` Token Format

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

```
xproperty,
```

`xselect` *Token*

The `xselect` token contains the data moved between windows. This data is a byte stream with no assumed internal structure, and a property string. This data is a byte stream with no assumed internal structure, and a property string. The fields are:

- A token ID
- The length of the property string
- The property string
- A length for the property type
- The property type string
- A length field that gives the number of bytes of data
- A byte string containing the data

Figure B-37 shows the token format.

token ID	property length	property string	prop type len	prop type	data length	window data
1 byte	2 bytes	N bytes	2 bytes	N bytes	2 bytes	N bytes

Figure B-37 `xselect` Token Format

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

```
xselect,
```

`xwindow` *Token*

The `xwindow` token contains information about a window. The fields are:

- A token ID
- The X server identifier
- The creator's user ID

Figure B-35 on page 152 shows the token format.

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

```
xwindow,
```

Audit Records

General Audit Record Structure

The audit records produced by Trusted Solaris 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 which 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 B-5 `access(2TSOL)`

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

`Format:`

header-token
path-token
[*attr-token*]
[slabel-token] (`object`)
subject-token
slabel-token (subject)
return-token

B

Table B-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 B-7 `adjtime(2TSOL)`

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

`Format:`
header-token
subject-token
return-token

Table B-8 audit(2TSOL)

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

Format:
header-token
subject-token
return-token

Table B-9 auditon(2TSOL) — get current active root

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

Format:
header-token
subject-token
return-token

Table B-10 auditon(2TSOL) — get event class

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

Format:
header-token
subject-token
return-token

Table B-11 auditon(2TSOL) — get audit state

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

Format:
header-token
subject-token
return-token

Table B-12 auditon(2TSOL) — get current working directory

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

Format:
header-token
subject-token
return-token

Table B-13 auditon(2TSOL) — get kernel mask

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

Format:
header-token
subject-token
return-token

Table B-14 auditon(2TSOL) — get audit statistics

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

Format:
header-token
subject-token
return-token

Table B-15 auditon(2TSOL) — GETPOLICY command

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

Format:
header-token
subject-token
return-token

Table B-16 auditon(2TSOL) — get audit queue control parameters

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

Format:
header-token
subject-token
return-token

Table B-17 auditon(2TSOL) — 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 B-18 auditon(2TSOL) — 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 B-19 auditon(2TSOL) — 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)
subject-token
return-token

Table B-20 auditon(2TSOL) — 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 B-21 auditon(2TSOL) — reset audit statistics

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

Format:

header-token
subject-token
return-token

Table B-22 auditon(2TSOL) — 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 B-23 auditon(2TSOL) — SETPOLICY 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 B-24 auditon(2TSOL) — set audit queue control parameters

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 B-25 auditstat(2)

Event Name	Event ID	Event Class	Mask
AUE_AUDITSTAT	150	ad	0x00000800

Format:

header-token
[*argument-token*]
subject-token
return-token

Table B-26 `auditsvc(2TSOL)`

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 (invalid file descriptor):

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

Table B-27 `chdir(2TSOL)`

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

Format:

header-token	
path-token	
[*attr-token*]	
[slabel-token]	(object)
subject-token	
slabel-token	(subject)
return-token	

Table B-28 `chmod(2TSOL)`

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*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-29 `chown(2TSOL)`

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*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-30 `chroot(2TSOL)`

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

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-31 `clock_settime(3RTSOL)`

Event Name	Event ID	Event Class	Mask
AUE_CLOCK_SETTIME	513	ad	0x00000800

Format:

header-token
slabel-token
return-token

Table B-32 `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*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Also for files closed on process termination.
The *argument-token* is only present with the close() system call. It may be removed in future releases
The *path-token* is present only with valid file descriptors.

Table B-33 `creat(2TSOL)`

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

Format
header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-34 `devpolicy(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_DEVPOLICY	270	ad	0x40000800

Format:

header-token	
argument-token	(cmd)
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-35 `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 B-36 exec(2TSOL)

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

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-37 execve(2TSOL)

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

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-38 `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 B-39 `exit(2)`

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

Format:
header-token
subject-token
return-token

Table B-40 `fchdir(2TSOL)`

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

Format:
- *header-token*
- [*path-token*]
- [*attr-token*]
- *[slabel-token]* (object)
- *subject-token*
- *slabel-token* (subject)
- *return-token*

Table B-41 `fchmod(2TSOL)`

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*]
- *[slabel-token]* (object)
- *subject-token*
- *slabel-token* (subject)
- *return-token*

Format (invalid file descriptor):
- *header-token*
- *argument-token* (2, "new file mode", mode)
- *argument-token* (1, "no path: fd", fd)
- *subject-token*
- *slabel-token* (subject)
- *return-token*

Table B-42 `fchown(2TSOL)`

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]
    [slabel-token]                   (object)
    subject-token
    slabel-token                     (subject)
    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
    slabel-token                (subject)
    return-token
```

Table B-43 `fchroot(2TSOL)`

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

```
Format:
    header-token
    [path-token]
    [attr-token]
    [slabel-token]                   (object)
    subject-token
    slabel-token                     (subject)
    return-token
```

Table B-44 `fcntl(2TSOL)`

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 B-45 `fork(2TSOL)`

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

`Format:`

header-token	
[*argument-token*]	`(0, "child PID", pid)`
subject-token	
slabel-token	`(subject)`
return-token	

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

Table B-46 `fork1(2TSOL)`

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

Format:

header-token	
[*argument-token*]	(0, "child PID", pid)
subject-token	
slabel-token	(subject)
return-token	

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

Table B-47 `fstatfs(2)`

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

Format (file descriptor):

header-token	
[*path-token*]	
[*attr-token*]	
[slabel-token]	(object)
subject-token	
slabel-token	(subject)
return-token	

Format (non-file descriptor):

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

Table B-48 `getaudit(2TSOL)`

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

Format:
- *header-token*
- *subject-token*
- *return-token*

Table B-49 `getauid(2TSOL)`

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

Format:
- *header-token*
- *subject-token*
- *return-token*

Table B-50 `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 B-51 `getmsg(2) — 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 B-52 `getmsg(2) — 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 B-53 `getmsgqcmwlabel(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_GETMSGQCMWLABEL	514	ip	0x00000200

Format:

header-token
argument-token (1, "msg ID", message ID)
[*ipc-token*]
[*ipc-perm-token*] (of the IPC)
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-54 `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 B-55 `getportaudit(2)`

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

Format:
header-token
subject-token
return-token

Table B-56 `getsemcmwlabel(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_GETSEMCMWLABEL	515	ip	0x00000200

Format:

header-token	
argument-token	(1, "sem ID", semaphore ID)
[*ipc-token*]	
[*ipc-perm-token*]	(of the IPC)
[*slabel-token*]	
subject-token	
slabel-token	
return-token	

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the sem ID is invalid.

Table B-57 `getshmcmwlabel(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_GETSHMCMWLABEL	516	ip	0x00000200

```
Format:
     header-token
     argument-token              (1, "shm ID", semaphore ID)
     [ipc-token]
     [ipc-perm-token]            (of the IPC)
     [slabel-token]
     subject-token
     slabel-token
     return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not
included if the shm ID is invalid.
```

Table B-58 `information label float - subject`

Event Name	Event ID	Event Class	Mask
AUE_SFLOAT	520	no	0x00000000

```
System call: varies

Format:
     header-token
     ilabel-token                (old ilabel)
     subject-token
     slabel-token
     return-token
```

Table B-59 information label float - object

Event Name	Event ID	Event Class	Mask
AUE_OFLOAT	519	no	0x00000000

System call: varies

Format (object is a process):

header-token
ilabel-token (old ilabel)
process-token
slabel-token
subject-token
slabel-token
return-token

Format (object is a file):

header-token
ilabel-token (old ilabel)
path-token
attr-token
subject-token
slabel-token
return-token

Format (object is unknown):

header-token
ilabel-token (old ilabel)
text-token (“No path”, Path not available)
attr-token
subject-token
slabel-token
return-token

Table B-60 `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 B-61 `kill(2TSOL)`

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

```
Format (valid process):
    header-token
    argument-token          (2, "signal", signo)
    [process-token]
    [slabel-token]          (process)
    subject-token
    slabel-token            (subject)
    return-token

Format (zero or negative process):
    header-token
    argument-token          (2, "signal", signo)
    argument-token          (1, "process", pid))
    subject-token
    slabel-token            (subject)
    return-token
```

Table B-62 `lchown(2TSOL)`

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 B-63 `link(2TSOL)`

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

Format:

header-token	
path-token	(from path)
[*attr-token*]	(from path)
[*slabel-token*]	(from path)
path-token	(to path)
[*attr-token*]	(to path)
[*slabel-token*]	(to path)
subject-token	
slabel-token	(subject)
return-token	

Table B-64 `lstat(2TSOL)`

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

Format:

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

Table B-65 `lxstat(2)`

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

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-66 `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 B-67 `mkdir(2TSOL)`

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

`Format:`

header-token
argument-token `(2, "mode", mode)`
path-token
`[`*attr-token*`]`
[slabel-token] *(*`object`*)*
subject-token
slabel-token *(*subject*)*
return-token

Table B-68 `mknod(2TSOL)`

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*`]`
[slabel-token] *(*`object`*)*
subject-token
slabel-token *(*subject*)*
return-token

Table B-69 `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 (invalid 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 B-70 `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 B-71 `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 B-72 `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 B-73 `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 B-74 `mount(2TSOL)`

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]
     [slabel-token]          (object)
     subject-token
     slabel-token            (subject)
     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 B-75 `msgctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-76 msgctl(2TSOL) — 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*]	
[*ipc-perm-token*]	(of the IPC's old values)
[*slabel-token*]	
[*ipc-perm-token*]	(of the IPC's new values)
[*slabel-token*]	
subject-token	
slabel-token	
subject-token	
return-token	

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-77 msgctl(2TSOL) — 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*]
[*ipc-perm-token*] (of the IPC)
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-78 msgget(2TSOL)

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

Format:

header-token
argument-token (1, "msg key", message key)
argument-token (2, "msg flag", message flags)
[*ipc-perm-token*] (of the IPC object)
[*slabel-token*]
[*ipc-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-79 `msggetl(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_MSGGETL	174	ip	0x00000200

Format:

header-token	
argument-token	(1, "msg key", message key)
argument-token	(2, "msg flag", message flags)
slabel-token	(desired SL)
[*ipc-perm-token*]	(of the IPC object)
[*slabel-token*]	
[*ipc-token*]	
subject-token	
slabel-token	
return-token	

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-80 msgrcv(2)/msgrcvl(2TSOL)

Event Name	Event ID	Event Class	Mask
AUE_MSGRCV	89	ip	0x00000200
AUE_MSGRCVL	175	ip	0x00000200

Format:

header-token
argument-token (1, "msg ID", message ID)
[*ipc-token*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-81 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*]
[*ipc-perm-token*] (of the IPC's new values)
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-82 `msgsndl(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_MSGSND	176	ip	0x00000200

Format:

header-token	
argument-token	(1, "msg ID", message ID)
[*ilabel-token*]	(argument passed)
[*ipc-token*]	
[*ipc-perm-token*]	
[*slabel-token*]	
subject-token	
slabel-token	
return-token	

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the msg ID is invalid.

Table B-83 `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 B-84 `old nice(2TSOL)`

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

Format:

header-token
subject-token
slabel-token (subject)
return-token

Table B-85 `open(2TSOL) – read`

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

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-86 `open(2TSOL) — read,creat`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RC	73	fc,fr	0x00000011

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-87 `open(2TSOL) — read,trunc,creat`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RTC	75	fc,fd,fr	0x00000031

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-88 `open(2TSOL) – read,trunc`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RT	74	fd,fr	0x00000021

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-89 `open(2TSOL) – read,write`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RW	80	fr,fw	0x00000003

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-90 `open(2TSOL) — 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*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-91 `open(2TSOL) — read,write,trunc,creat`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_RWTC	83	fr,fw,fc,fd	0x00000033

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-92 `open(2TSOL) – 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*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-93 `open(2TSOL) – write`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_W	76	fw	0x00000002

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-94 `open(2TSOL) — write,creat`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WC	77	fw,fc	0x00000012

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-95 `open(2TSOL) — write,trunc,creat`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WTC	79	fw,fc,fd	0x00000032

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-96 `open(2TSOL) – write,trunc`

Event Name	Event ID	Event Class	Mask
AUE_OPEN_WT	78	fw,fd	0x00000022

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-97 `pathconf(2)`

Event Name	Event ID	Event Class	Mask
AUE_PATHCONF	71	fa	0x00000004

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-98 `pipe(2)`

Event Name	Event ID	Event Class	Mask
AUE_PIPE	185	no	0x00000000

Format:

header-token
subject-token
slabel-token (subject)
return-token

Table B-99 `priocntl(2TSOL)`

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
slabel-token (subject)
return-token

Table B-100 `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 B-101 `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 B-102 `putmsg(2)-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 B-103 `putmsg(2)-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 B-104 `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 B-105 `readlink(2TSOL)`

Event Name	Event ID	Event Class	Mask
`AUE_READLINK`	`22`	`fr`	0x00000001

`Format:`

header-token
path-token
[*attr-token*]
[slabel-token] (`object`)
subject-token
slabel-token (subject)
return-token

Table B-106 `rename(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_RENAME	42	fc,fd	0x00000030

Format:

header-token	
path-token	(from name)
[*attr-token*]	(from name)
[*slabel-token*]	(from name)
[*path-token*]	(to name)
subject-token	
slabel-token	(subject)
return-token	

Table B-107 `rmdir(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_RMDIR	48	fd	0x00000020

Format:

header-token	
path-token	
[*attr-token*]	
[slabel-token]	*(*object*)*
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-108 semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-109 semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-110 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-111 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.

Table B-112 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-113 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.

Table B-114 `semctl(2TSOL) — 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]
     [ipc-perm-token]          (of the IPC's old values)
     [slabel-token]
     [ipc-perm-token]          (of the IPC's new values)
     [slabel-token]
     subject-token
     slabel-token
     return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not
included if the semaphore ID is invalid.
```

Table B-115 `semctl(2TSOL) — 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]
     [ipc-perm-token]
     [slabel-token]
     subject-token
     slabel-token
     return-token

The ipc and ipc_perm tokens are not included if the semaphore ID
is invalid.
```

Table B-116 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-117 `semctl(2TSOL) — 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*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

Table B-118 `semget(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SEMGET	109	ip	0x00000200

Format:

header-token
argument-token (1, "sem key", semaphore key)
argument-token (3, "sem flags", semaphore flags)
[*ipc-perm-token*]
[*slabel-token*]
[*ipc-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-119 `semgetl(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SEMGETL	177	ip	0x00000200

```
Format:
     header-token
     argument-token          (1, "sem key", semaphore key)
     argument-token          (3, "sem flags", semaphore flags)
     slabel-token
     [ipc-perm-token]
     [slabel-token]
     [ipc-token]
     subject-token
     slabel-token
     return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not
included if the system call failed.
```

Table B-120 `semop(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SEMOP	110	ip	0x00000200

```
Format:
     header-token
     argument-token          (1, "sem ID", semaphore ID)
     [ipc-token]
     [ipc-perm-token]
     [slabel-token]
     subject-token
     slabel-token
     return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not
included if the semaphore ID is invalid.
```

Table B-121 `semopl(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SEMOPL	517	ip	0x00000200

Format:

header-token
argument-token (1, "sem ID", semaphore ID)
ilabel-token
[*ipc-token*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and the slabel of the ipc tokens are not included if the semaphore ID is invalid.

Table B-122 `setaudit(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETAUDIT	133	ad	0x00000800

```
Format (valid program stack address):
    header-token
    argument-token       (1, "setaudit:auid", 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 (invalid program stack address):
    header-token
    subject-token
    return-token
```

Table B-123 `setauid(2TSOL)`

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 B-124 `setclearance(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETCLEARANCE	117	sl	0x00008000

Format:

header-token	
clearance-token	(argument value)
clearance-token	(subject before)
clearance-token	(subject later)
subject-token	
slabel-token	(subject)
return-token	

Table B-125 `setcmwplabel(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETCMWPLABEL	116	sl	0x00008000

```
Format (setting flag == SETCL_ALL):
    header-token
    slabel-token              (SL from input argument)
    ilabel-token              (IL from input argument)
    slabel-token              (SL of subject before)
    ilabel-token              (IL of subject before)
    argument-token            (2, “flag”, value)
    slabel-token              (SL of subject after)
    ilabel-token              (IL of subject after)
    subject-token
    slabel-token              (subject)
    return-token

Format (setting flag == SETCL_SL):
    header-token
    slabel-token              (SL from input argument)
    slabel-token              (SL of subject before)
    argument-token            (2, “flag”, value)
    slabel-token              (SL of subject after)
    subject-token
    slabel-token              (subject)
    return-token

Format (setting flag == SETCL_IL):
    header-token
    ilabel-token              (IL from input argument)
    ilabel-token              (IL of subject before)
    argument-token            (2, “flag”, value)
    ilabel-token              (IL of subject after)
    subject-token
    slabel-token              (subject)
    return-token
```

Table B-126 `setegid(2TSOL)`

Event Name	Event ID	Event Class	Mask
`AUE_SETEGID`	`214`	`pc`	0x00000080

`Format:`

header-token	
argument-token	`(1, "gid", group ID)`
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-127 `seteuid(2TSOL)`

Event Name	Event ID	Event Class	Mask
`AUE_SETEUID`	`215`	`pc`	0x00000080

`Format:`

header-token	
argument-token	`(1, "gid", user ID)`
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-128 `old setgid(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETGID	205	pc	0x00000080

Format:

header-token	
argument-token	(1, "gid", group ID)
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-129 `setgroups(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETGROUPS	26	pc	0x00000080

Format:

header-token	
[*argument-token*]	(1, "setgroups", group ID)
subject-token	
slabel-token	*(*subject*)*
return-token	

One *argument-token* for each group set.

Table B-130 `setpgrp(2)`

Event Name	Event ID	Event Class	Mask
AUE_SETPGRP	27	pc	0x00000080

Format:
- *header-token*
- *subject-token*
- *slabel-token*
- *return-token*

Table B-131 `setrlimit(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_SETRLIMIT	51	ad	0x00000800

Format:
- *header-token*
- *subject-token*
- *slabel-token* (subject)
- *return-token*

Table B-132 `old setuid(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_OSETUID	200	pc	0x00000080

Format:

header-token	
argument-token	(1, "uid", user ID)
subject-token	
slabel-token	*(*subject*)*
return-token	

Due to a current bug in the audit software, this token is reported as AUE_OSETUID.

Table B-133 `shmat(2)`

Event Name	Event ID	Event Class	Mask
AUE_SHMAT	96	ip	0x00000200

Format:

header-token	
argument-token	(1, "shm ID", shared memory ID)
argument-token	(2, "shm adr", shared mem addr)
[*ipc-token*]	
[*ipc_perm-token*]	
[*slabel-token*]	
subject-token	
slabel-token	
return-token	

The ipc, ipc_perm, and slabel tokens are not included if the shared memory segment ID is invalid.

Table B-134 shmctl(2TSOL) — IPC_RMID command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_RMID	92	ip	0x00000200

Format:

header-token	
argument-token	(1, "shm ID", shared memory ID)
[*ipc-token*]	
[*ipc-perm-token*]	
[*slabel-token*]	
subject-token	
slabel-token	
return-lbtoken	

The ipc, ipc_perm, and slabel tokens are not included if the shared memory segment ID is invalid.

Table B-135 shmctl(2TSOL) — IPC_SET command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_SET	93	ip	0x00000200

Format:

header-token	
argument-token	(1, "shm ID", shared memory ID)
[*ipc-token*]	
[*ipc_perm-token*]	(of the IPC's old values)
[*slabel-token*]	
[*ipc_perm-token*]	(of the IPC's new values)
[*slabel-token*]	
subject-token	
slabel-token	
return-token	

The ipc, ipc_perm, and slabel tokens are not included if the shared memory segment ID is invalid.

Table B-136 shmctl(2TSOL) — IPC_STAT command

Event Name	Event ID	Event Class	Mask
AUE_SHMCTL_STAT	94	ip	0x00000200

Format:

header-token
argument-token (1, "shm ID", shared memory ID)
[*ipc-token*]
[*ipc-perm-token*]
[*slabel-token*]
subject-token
slabel-token
return-token

The ipc, ipc_perm, and slabel tokens are not included if the shared memory segment ID is invalid.

Table B-137 shmdt(2)

Event Name	Event ID	Event Class	Mask
AUE_SHMDT	97	ip	0x00000200

Format:

header-token
argument-token (1, "shm adr", shared mem addr)
subject-token
slabel-token
return-token

Table B-138 `shmget(2TSOL)`

Event Name	Event ID	EventClass	Mask
AUE_SHMGET	95	ip	0x00000200

```
Format:
     header-token
     argument-token          (1, "shm ID", shared memory ID)
     argument-token          (3, "shm flag", shared memory flags)
     [ipc-perm-token]        (of the IPC's old values)
     [slabel-token]
     [ipc-perm-token]        (of the IPC's new values)
     [slabel-token]
     subject-token
     slabel-token
     subject-token

The ipc, ipc_perm, and slabel tokens are not included for failed
events.
```

Table B-139 shmgetl(2TSOL)

Event Name	Event ID	EventClass	Mask
AUE_SHMGETL	178	ip	0x00000200

Format:

header-token
argument-token (1, "shm ID", shared memory ID)
argument-token (3, "shm flag", shared memory flags)
slabel-token
[*ipc-perm-token*] (of the IPC's old values)
[*slabel-token*]
[*ipc-perm-token*] (of the IPC's new values)
[*slabel-token*]
subject-token
slabel-token
subject-token

The ipc, ipc_perm, and slabel tokens are not included for failed events.

Table B-140 stat(2TSOL)

Event Name	Event ID	Event Class	Mask
AUE_STAT	16	fa	0x00000004

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-141 `statfs(2)`

Event Name	Event ID	EventClass	Mask
AUE_STATFS	54	fa	0x00000004

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-142 `statvfs(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_STATVFS	234	fa	0x00000004

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-143 `stime(2TSOL)`

Event Name	Event ID	Event Class	Mask
`AUE_STIME`	`201`	`ad`	0x00000800

`Format:`
header-token
subject-token
return-token

Table B-144 `symlink(2TSOL)`

Event Name	Event ID	Event Class	Mask
`AUE_SYMLINK`	`21`	`fc`	0x00000010

`Format:`

header-token	
text-token	`(symbolic link string)`
path-token	
[*attr-token*]	
[slabel-token]	*(*`object`*)*
subject-token	
slabel-token	*(*subject*)*
return-token	

Table B-145 `sysinfo(2TSOL)`

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 B-146 `system booted`

Event Name	Event ID	Event Class	Mask
AUE_SYSTEMBOOT	113	na	0x00000400
Format:			
header-token			
text-token	("booting kernel")		
return-token			

Table B-147 `umount(2TSOL) — old version`

Event Name	Event ID	Event Class	Mask
AUE_UMOUNT	12	ad	0x00000800

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-148 `unlink(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_UNLINK	6	fd	0x00000020

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-149 `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 B-150 `utimes(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_UTIMES	49	fm	0x00000008

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

Table B-151 `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 B-152 `vfork(2TSOL)`

Event Name	Event ID	Event Class	Mask
AUE_VFORK	25	pc	0x00000080

Format:

header-token	
argument-token	(0, "child PID", pid)
subject-token	
slabel-token	*(*subject*)*
return-token	

The fork return values are undefined since the audit record is produced at the point that the child process is spawned.

Table B-153 `vtrace(2)`

Event Name	Event ID	Event Class	Mask
AUE_VTRACE	36	pc	0x00000080

Format:
header-token
subject-token
return-token

Table B-154 `xmknod(2)`

Event Name	Event ID	Event Class	Mask
AUE_XMKNOD	240	fc	0x00000010

Format:
header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Table B-155 `xstat(2)`

Event Name	Event ID	Event Class	Mask
AUE_XSTAT	235	fa	0x00000004

Format:

header-token
path-token
[*attr-token*]
[slabel-token] (object)
subject-token
slabel-token (subject)
return-token

Kernel-Level Pseudo-Events

Pseudo-events do not create audit records separate from the event being audited for MAC (Mandatory Access Control) or for use of privilege.

Table B-156 `MAC check`

Event Name	Event ID	Event Class	Mask
AUE_MAC	518	no	0x00000000

Table B-157 `Use of privilege`

Event Name	Event ID	Event Class	Mask
AUE_UPRIV	521	no	0x00000000

B

User-Level Generated Audit Records

These audit records are created by programs 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 B-158 `add_drv(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_add_drv`	/usr/sbin/add_drv	`9018`	`ad`	0x00000800

`Format:`

header-token	
subject-token	
groups-token	
slabel-token	
ilabel-token	
return-token	
exec_args-token	(command-line arguments)
text-token	(driver name)
text-token	(base directory)
text-token	(class name)
text-token	(aliases)

Table B-159 `allocate(1MTSOL) device success`

Event Name	Program	Event ID	Event Class	Mask
`AUE_allocate_succ`	/usr/sbin/allocate	`6200`	`ad`	0x00000800

`Format:`

header-token
text-token
path-token
subject-token
exit-token

Table B-160 `allocate(1MTSOL) device failure`

Event Name	Program	Event ID	Event Class	Mask
`AUE_allocate_fail`	/usr/sbin/allocate	`6201`	`ad`	0x00000800

`Format:`

header-token
text-token
subject-token
exit-token

Table B-161 `allocate(1MTSOL) device success`

Event Name	Program	Event ID	Event Class	Mask
`AUE_deallocate_succ`	/usr/sbin/deallocate	`6202`	`ad`	0x00000800

`Format:`

header-token
subject-token
newgroups-token
exit-token

Table B-162 `deallocate(1MTSOL) device failure`

Event Name	Program	Event ID	Event Class	Mask
`AUE_deallocate_fail`	/usr/sbin/deallocate	`6203`	`ad`	0x00000800

`Format:`
header-token
subject-token
newgroups-token
exit-token

Table B-163 `allocate(1MTSOL) - 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 B-164 `allocate(1MTSOL) - 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 B-165 `at(1TSOL) - create crontab`

Event Name	Program	Event ID	Event Class	Mask
`AUE_at_create`	/usr/bin/at	`6144`	`ad`	0x00000800

`Format:`

header-token
subject-token
return-token
exec_args-token
text-token (user name)
text-token (job queue)

Table B-166 `at(1TSOL) - delete atjob file (at or atrm)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_at_delete`	/usr/bin/at /usr/bin/atrm	`6145`	`ad`	0x00000800

`Format:`

header-token
subject-token
return-token
exec_args-token
text-token (user name)
text-token (job queue)

Table B-167 `at(1TSOL) - 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 B-168 `audit(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_audit`	/usr/sbin/audit	`9016`	`ad`	0x00000800

`Format:`
header-token
text-token (“new audit file” | “reread audit_control” | “terminate auditd” | “unknown option”)
return-token
subject-token
slabel-token

Table B-169 `auditwrite(3TSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_auditwrite`	auditwrite()	`9015`	`ad`	0x00000800

`Format:`

header-token	
text-token	*(*error description*)*
subject-token	
return-token	

Table 4-1 `chroot(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_chroot`	/usr/sbin/chroot	`9029`	`ad`	0x00000800

`Format:`

header-token	
subject-token	
groups-token	
slabel-token	
ilabel-token	
return-token	
exec_args-token	*(*command-line arguments*)*
path-token	*(*new root directory*)*
path-token	*(*command to execute*)*

Table B-170 crontab(1TSOL) - crontab created

Event Name	Program	Event ID	Event Class	Mask
AUE_crontab_create	/usr/bin/crontab	6148	ad	0x00000800

Format:
header-token
subject-token
return-token
exec_args-token
text-token (user name)

Table B-171 crontab(1TSOL) - crontab deleted

Event Name	Program	Event ID	Event Class	Mask
AUE_crontab_delete	/usr/bin/crontab	6149	ad	0x00000800

Format:
header-token
subject-token
return-token
exec_args-token
text-token (user name)

Table B-172 crontab(1TSOL) - 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
return-token
exec_args-token
text-token (user name)
text-token (job type: cron or at)
text-token (cron command or at job name)

Table B-173 crontab(1TSOL) - 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 B-174 `dispadmin(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_dispadmin`	/usr/sbin/dispadmin	`9025`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
text-token (scheduler class)
path-token (input file)

Table B-175 `eeprom(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_eeprom`	/usr/sbin/eeprom	`9032`	`ad`	0x00000800

`Format:`

header-token
return-token
path-token (prom device)
text-token (variable=value)

Table B-176 `fuser(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_fuser`	/usr/sbin/fuser	`9031`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
path-token (file name)
arg-token (1, “PID”, process-id)

Table B-177 `halt(1MTSOL)`

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 B-178 `inetd(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_inetd_connect`	`/usr/sbin/inetd`	`6151`	`na`	0x00000400
`Format:` *header-token* *subject-token* *text-token* *ip-address-token* *ip-port-token* *return-token*	`(service name)`			

Table B-179 `in.ftpd(1MTSOL) ftp access`

Event Name	Program	Event ID	Event Class	Mask
`AUE_ftpd`	`/usr/sbin/in.ftpd`	`6165`	`lo`	0x00001000
`Format:` *header-token* *subject-token* *text-token* *return-token*	`(error message, failure only)`			

Table B-180 `login(1TSOL) — local`

Event Name	Program	Event ID	Event Class	Mask
`AUE_login`	/usr/sbin/login	`6152`	`lo`	0x00001000

`Format:`
header-token
text-token
text-token `(message - success or failure)`
subject-token
return-token

Table B-181 `login(1TSOL) — 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 B-182 `login(1TSOL) — 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 B-183 `logout(1)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_logout`	/usr/sbin/login	`6153`	`lo`	0x00001000

`Format:`

header-token
subject-token
text-token
return-token

Table B-184 `modload(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_modload`	/usr/sbin/modload	`9020`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
text-token (module pathname)

Table B-185 `modload(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_modunload`	/usr/sbin/modunload	`9021`	`ad`	0x00000800

`Format:`

- *header-token*
- *subject-token*
- *groups-token*
- *slabel-token*
- *ilabel-token*
- *return-token*
- *exec_args-token* (command-line arguments)
- *text-token* (module name)

Table B-186 `mountd(1M) – NFS mount`

Event Name	Program	Event ID	Event Class	Mask
`AUE_mountd_mount`	/usr/lib/nfs/mountd	`6156`	`na`	0x00000400

`Format:`

- *header-token*
- *argument-token*
- *slabel-token* `(subject slabel)`
- *text-token* `(remote client hostname)`
- *path-token* `(mount dir)`
- *slabel-token* `(slabel of the directory)`
- *text-token* `(error message, failure only)`
- *attribute-token*
- *subject-token*
- *return-token*

Table B-187 mountd(1MTSOL) – NFS unmount

Event Name	Program	Event ID	Event Class	Mask
AUE_mountd_umount	/usr/lib/nfs/mountd	6157	na	0x00000400

Format:

header-token	
slabel-token	(subject slabel)
text-token	(remote client hostname)
path-token	(mount dir)
slabel-token	(slabel of the directory)
text-token	(error message, failure only)
attribute-token	
subject-token	
return-token	

Table B-188 passwd(1)

Event Name	Program	Event ID	Event Class	Mask
AUE_passpœœd	/usr/bin/passwd	6163	lo	0x00001000

Format:

header-token	
subject-token	
text-token	(error message)
return-token	

Table B-189 `pbind(1M)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_pbind`	/usr/sbin/pbind	9026	ad	0x00000800

`Format:`

header-token	
subject-token	
groups-token	
slabel-token	
ilabel-token	
return-token	
exec_args-token	(command-line arguments)
text-token	(action: “BIND” | “UNBIND”)
arg-token	(1, “CPU”, processor id)
arg-token	(2, ”PID”, process-id)

Table B-190 `pfsh(1MTSOL)`

Event Names	Program	Event IDs	Event Class	Mask
`AUE_pfsh_trusted_priv`	/usr/bin/pfsh	9007	ad	0x00000800
`AUE_pfsh_trusted_nopriv`		9008		
`AUE_pfsh_priv`		9009		
`AUE_pfsh_nopriv`		9010		

`Format:`

header-token	
path-token	`(of the executable)`
exec_args-token	
path-token	`(of current directory)`
privilege-token	
return-token	
exec_env-token	`(if AUDIT_ARGE is on)`
subject-token	
slabel-token	

Table B-191 `psradm(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_psradm`	/usr/sbin/psradm	`9027`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
text-token (action: “ON” | “OFF”)
arg-token (1, ”PID”, processor id)

Table B-192 `reboot(1MTSOL)`

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 B-193 `rpc.rexd(1M)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_rexd`	/usr/sbin/rpc.rexd	`6164`	`lo`	0x00001000

```
Format:
     header-token
     subject-token
     text-token          (error message, failure only)
     text-token          (hostname)
     text-token          (username)
     text-token          adap(command to be executed)
     exit-token
```

Table B-194 `in.rexecd(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_rexecd`	/usr/sbin/in.rexecd	`6162`	`lo`	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 B-195 `in.rshd(1MTSOL) 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 B-196 `rem_drv(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_rem_drv`	/usr/sbin/rem_drv	`9019`	`ad`	0x00000800

`Format:`

header-token	
subject-token	
groups-token	
slabel-token	
ilabel-token	
return-token	
exec_args-token	*(*command-line arguments*)*
text-token	*(*driver name*)*
[text-token]	*(*base directory*)*

Table B-197 `init(1M) - run level change`

Event Name	Program	Event ID	Event Class	Mask
`AUE_run_level_change`	/usr/sbin/init	`9024`	`ad`	0x00000800

`Format:`
header-token

Table B-198 `sendmail(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_sendmail_deliver`	/usr/lib/sendmail	`9013`	`ad`	0x00000800
`AUE_sendmail_defer`		`9014`		

`Format:`

header-token	
text-token	(message about status)
text-token	(to)
text-token	(message ID)
text-token	(from)
text-token	(from host)
text-token	(to user)
text-token	(to host)
return-token	
slabel-token	
ilabel-token	

Table B-199 `sendmail(1MTSOL) - upgrade`

Event Name	Program	Event ID	Event Class	Mask
`AUE_sendmail_upgrade`	/usr/lib/sendmail	`9012`	`ad`	0x00000800

`Format:`

header-token	
text-token	(message ID)
slabel-token	(old label)
slabel-token	(new label)
subject-token	
slabel-token	

Table B-200 `setuname(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_setuname`	/usr/bin/setuname	`9022`	`ad`	0x00000800

`Format:`

header-token	
subject-token	
groups-token	
slabel-token	
ilabel-token	
return-token	
exec_args-token	(command-line arguments)
text-token	(action: “ADD” | “DELETE”)
path-token	(swapname)

Table B-201 `share(1M)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_exportfs`	/usr/lib/fs.d/nfs/share		`na`	0x00000400

`Format:`

header-token	
subject-token	
slabel-token	*(*subject slabel*)*
path-token	*(*export directory*)*
slabel-token	*(*slabel of the directory*)*
text-token	*(*export options*)*
return-token	

Table B-202 `su(1M)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_su`	/usr/bin/su	`6159`	`lo`	0x00001000

`Format:`

header-token	
subject-token	
text-token	`(error message)`
return-token	

Table B-203 `swap(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_swap`	/usr/sbin/swap	`9030`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token
text-token (new node name | "*none*")
text-token (new systemname | "*none*")

Table B-204 `uadmin(1MTSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_uadmin_cmd`	/usr/sbin/uadmin	`9023`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
argument-token (1, "cmd", command code)
argument-token (2, "fcn", function code)

Table B-205 `uauth`

Event Name	Program	Event ID	Event Class	Mask
`AUE_uauth`	authorization used	`9017`	`ad`	0x00000800

`Format:`

header-token
text-token (user name)
text-token (authorization)
subject-token
return-token

Table B-206 `uname(1TSOL)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_uname_set`	/usr/bin/uname	`9024`	`ad`	0x00000800

`Format:`

header-token
subject-token
groups-token
slabel-token
ilabel-token
return-token
exec_args-token (command-line arguments)
text-token (new node name)

Table B-207 `unshare(1M)`

Event Name	Program	Event ID	Event Class	Mask
`AUE_exportfs`	/usr/lib/fs.d/nfs/share		`na`	0x00000400

`Format:`

header-token
subject-token
slabel-token (subject slabel)
path-token (export directory)
return-token

Audit Reference

Auditing brings a number of additional utilities to the Trusted 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. The fifth table gives the filesystem security attributes of files in the auditing subsystem.

Table C-1 Section 1MTSOL—Maintenance Commands

Command	**Task**
`audit(1MTSOL)`	Control the audit daemon
`audit_startup(1MTSOL)`	Initialize the audit subsystem
`audit_warn(1MTSOL)`	Run the audit daemon warning script
`auditconfig(1MTSOL)`	Configure auditing
`auditd(1MTSOL)`	Control audit trail files
`auditreduce(1MTSOL)`	Merge and select audit records from audit trail files
`auditstat(1MTSOL)`	Display kernel audit statistics
`bsmconv(1MTSOL)`	Enable Trusted Solaris auditing (default)
`bsmunconv(1MTSOL)`	Disable Trusted Solaris auditing
`praudit(1MTSOL)`	Print contents of an audit trail file

Table C-2 Section 2TSOL—System Calls

System Call	System Parameter	Task
`audit(2TSOL)`		Write a record to the audit log
`auditon(2TSOL)`		Manipulate auditing:
	A_GETPOLICY	Get audit policy flags
	A_SETPOLICY	Set audit policy flags
	A_GETKMASK	Get asynchronous audit event preselection mask
	A_SETKMASK	Set asynchronous audit event preselection mask
	A_GETQCTRL	Get the kernel audit queue control parameters
	A_SETQCTRL	Set the kernel audit queue control parameters
	A_GETSTAT	Get the audit system statistics
	A_SETSTAT	Reset the audit system statistics
	A_GETCOND	Determine if auditing is on/off/disabled
	A_SETCOND	Set auditing to on/off
	A_GETFSIZE	Get the size limit for an audit trail file
	A_GETCLASS	Return the event to class mapping for the designated event
	A_SETCLASS	Set the event to class mapping for the designated audit event
	A_GETPINFO	Get the audit information for the specified process
	A_SETPMASK	Set the preselection mask for a specified process
	A_SETUMASK	Set the process mask for all processes of a specified audit ID
	A_SETSMASK	Set the process mask for all processes of a specified session ID
	A_GETCWD	Get the current working directory for this process
	A_GETCAR	Get the current active root for this process
`auditsvc(2TSOL)`		Write audit log to specified file descriptor
`getaudit(2TSOL)`		Get process audit information

Table C-2 Section 2TSOL—System Calls

System Call	System Parameter	Task
setaudit(2TSOL)		Set process audit information
getaudit_x(2TSOL)		Get process audit information, extended mask
setaudit_x(2TSOL)		Set process audit information, extended mask
getauid(2TSOL)		Get user audit identity
setauid(2TSOL)		Set user audit identity

Table C-3 Section 3TSOL—C Library Functions

Library Call	Task
au_preselect(3TSOL)	Preselect an audit event
au_user_mask(3TSOL)	Get user's binary preselection mask
getacdir(3TSOL), getacmin(3TSOL), getacflg(3TSOL), getacna(3TSOL), setac(3TSOL), endac(3TSOL)	Get audit control file information
getauclassnam(3TSOL), getauclassnam_r(3TSOL), getauclassent(3TSOL), getauclassent_r(3TSOL), setauclass(3TSOL), endauclass(3TSOL)	Get audit_class(4TSOL) entries
getauditflagsbin(3TSOL), getauditflagschar(3TSOL)	Convert audit flag specifications
getauevent(3TSOL), getauevent_r(3TSOL), getauevnam(3TSOL), getauevnam_r(3TSOL), getauevnum(3TSOL), getauevnum_r(3TSOL), getauevnonam(3TSOL), setauevent(3TSOL), endauevent(3TSOL)	Get audit_event(4TSOL) entries
getauusernam(3TSOL), getauusernam_r(3TSOL), getauuserent(3TSOL), getauuserent_r(3TSOL), setauuser(3TSOL), endauuser(3TSOL)	Get audit_user(4TSOL) entries
getfauditflags(3TSOL), getfauditflags_x(3TSOL)	Generate the process audit state

Table C-4 Section 4TSOL—Headers, Tables, and Macros

Files	Task
`audit.log(4TSOL)`	Gives format for an audit trail file
`audit_class(4TSOL)`	Gives audit class definitions
`audit_control(4TSOL)`	Controls information for system audit daemon
`audit_data(4TSOL)`	Holds current information on the audit daemon
`audit_event(4TSOL)`	Holds audit event definition and class mapping
`audit_user(4TSOL)`	Holds per-user auditing data file

Table C-5 Filesystem Security Attributes for the Audit Subsystem

Name	IL[SL]	DAC	Owner	Group
`audit(1MTSOL)`	ADMIN_LOW[ADMIN_LOW]	555	bin	bin
`auditd(1MTSOL)`				
`auditconfig(1MTSOL)`				
`auditstat(1MTSOL)`				
`auditreduce(1MTSOL)`				
`praudit(1MTSOL)`				
`audit_warn(1MTSOL)`	ADMIN_LOW[ADMIN_LOW]	740	root	sys
`audit_startup(1MTSOL)`				
`bsmconv(1MTSOL)`	ADMIN_LOW[ADMIN_LOW]	744	root	sys
`bsmunconv(1MTSOL)`				
`audit.log(4TSOL)`	ADMIN_HIGH[ADMIN_HIGH]	400	root	root
`audit_class(4TSOL)`	ADMIN_LOW[ADMIN_LOW]	644	root	sys
`audit_control(4TSOL)`	ADMIN_LOW[ADMIN_LOW]	600	root	sys
`audit_data(4TSOL)`	ADMIN_LOW[ADMIN_LOW]	660	root	root
`audit_event(4TSOL)`	ADMIN_LOW[ADMIN_LOW]	644	root	sys
`audit_user(4TSOL)`	ADMIN_LOW[ADMIN_LOW]	600	root	sys

Auditing Checklist and Worksheets D

Purpose

The worksheet templates help you plan the details of auditing at your site.

How to Use the Worksheets

♦ **Fill out the information to plan auditing at your site.**

The worksheets provide the details of how you configured auditing at your site. They should be protected with care, locked away from unauthorized users, and used to help debug system problems when they occur.

D

Auditing Checklist

	Decisions have been made about ...	Role responsible for decision...
❑	Classes of events to audit for success	Security administrator
❑	Classes of events to audit for failure	Security administrator
❑	Classes of events to audit for both	Security administrator
❑	Audit policies	Security administrator
❑	Users/roles with additional auditing & for what	Security administrator
❑	Audit servers, partitions, size of partitions	System administrator
❑	Audit administration server	System administrator
❑	Primary and secondary audit partitions for each workstation	System administrator
❑	Size of local audit partitions for each workstation	System administrator
❑	Protections on audit file systems	Security administrator
❑	Who has access to the audit administration server	Security administrator
❑	Who has access to the audit servers	Security administrator
❑	Who receives audit trouble mail	Security administrator
❑	Who is responsible for audit archiving and when	System administrator

This function	is done by ...	user(s)	with profile	in role (optional)
audit records backup				
audit analysis & reports				

Security Administrator Worksheets

The security administrator is responsible for deciding what to audit. Optionally, the security administrator can refine existing event-to-class mappings, create new classes, and add new events.

What to Audit Worksheet

	Audited for Success	for Failure	for Both
Audit Classes (Enter in `audit_control`)			
flags:	+*flag*,	-*flag*,	*flag*
naflags:	+*flag*,	-*flag*,	*flag*
User Names for User Exceptions (Enter in `audit_user`)			
	Always	:	**Never**
Username:	+*flag*,-*flag*,^-*flag*,^+*flag*	:	+*flag*,-*flag*,^-*flag*,^+*flag*
***username1*:**			
***username2*:**			
Mail Addresses in `audit_warn` entry in Aliases database			
audit_warn:			
Printers (Can print at `admin_high` label; printer and output protected from snooping)			

D

Audit Class Modifications Worksheet (Advanced)

Audit Class Name and Number	Modified or New	Audit Event Members

Audit Event Additions Worksheet (Advanced)

Caution – Do not modify or add events with an event number less than 32768. These are reserved by the system.

- **Use numbers from 32768 - 65535 for third party trusted applications.**

Number	Audit Event	Maps to User Command	Belongs to Class(es)

D

System Administrator Worksheets

The system administrator is responsible for setting up and maintaining the network of audit servers and clients.

Audit Clients Worksheet

Audit clients share their local audit file system and mount the audit server partitions where they store their audit files. The security administrator puts the information in the `audit_control` file for each client.

- **Fill out each client's list of audit file systems in the order they are used, from primary to secondary to backup (local).**

Name	Audit File Systems	Size	Mount Point
tern (Client)	hurricane:/etc/security/audit/hurricane.4/files	800	/etc/security/audit/hurricane.4
(sample)	hurricane:/etc/security/audit/hurricane.3/files	800	/etc/security/audit/hurricane.3
	tornado:/etc/security/audit/tornado/files	1000	/etc/security/audit/tornado
	/etc/security/audit/tern/files	200	local (/etc/security/audit/tern)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)

Name	Audit File Systems	Size	Mount Point
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)
	/etc/security/audit/		local (/etc/security/audit/)

D

Audit Server-Client Network Worksheet

Name	Shared File Systems	Size	Audit Client: Primary or Secondary
Audit Administration Server (Shares no file systems; mounts all audit file systems)			
	None		
Audit Servers (Share all audit partitions for mounting by audit administration server and client workstations)			
hurricane (sample)	/etc/security/audit/hurricane	200	local
	/etc/security/audit/hurricane.1	1000	grebe (P),
	/etc/security/audit/hurricane.3	800	tern (P), willet (S)
	/etc/security/audit/hurricane.4	800	tern (P), willet (S)
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
	/etc/security/audit/		
Mail Addresses in `audit_warn` **Alias**			
Printer for Audit Administration Server			

Auditing for a Non-Networked Workstation Worksheet

Security Information	**Audited for Success**	**Audited for Failure**	**Audited for Both**
Audit Classes (Enter in `audit_control`**)**			
User Name for User Exceptions (Enter in `audit_user`**)**			
Mail Addresses in `audit_warn` **Alias**			
Printers (Can print at `admin_high` label; printer and output protected from snooping)			

Audit Partitions	**File System Name**	**Size**
Local	/etc/security/audit/	
Local Backup (optional)	/etc/security/audit/	

D

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

R

S

T

U

V

W

X