SunSHIELD Basic Security Module Guide
## Contents

1. **Installation** ......................................................... 1  
   - Enabling BSM .......................................................... 2  
   - Disabling BSM ......................................................... 2  
   - BSM and Client-Server Relationships ............................ 3  

2. **Administering Auditing** ............................................. 5  
   - More on Auditing ..................................................... 6  
   - Audit Startup .......................................................... 6  
   - Audit Classes and Events ......................................... 7  
     - Kernel Events ....................................................... 7  
     - User-Level Events ................................................ 8  
     - Audit Records ...................................................... 8  
   - Audit Flags ............................................................. 8  
     - Definitions of Audit Flags .................................... 9  
     - Audit Flag Syntax ................................................. 10  
     - Prefixes to Modify Previously Set Audit Flags ........... 11
Audit File Naming .................................................. 27
How Audit File Names Are Used ............................... 28
Time-Stamp Format and Interpretation....................... 28
Example of a File Name for a Still-Active File ............ 28
Example of a Closed Audit File Name ................. 29
Handling Nonactive Files Marked not_terminated .... 29
▼ To Create Audit Partitions and Export Them ........ 30
Example audit_control File Entries .................... 32
▼ To Configure Auditing ............................... 32
▼ To Plan Audit Configuration .............................. 33
Preventing Audit Trail Overflow .............................. 36
▼ To Prevent Audit Trail Overflow .................... 36
The auditconfig Command ................................. 37
Setting Audit Policies ........................................ 39
▼ To Change Which Events Are in Which Audit Classes 40
Changing Class Definitions .................................. 40

3. Audit Trail Analysis ............................................ 43
Auditing Features ............................................. 43
Audit User ID ............................................... 44
Audit Session ID ........................................... 44
Self-Contained Audit Records ................................ 44
Tools for Merging, Selecting, Viewing, and Interpreting Audit Records ........................................ 44
Audit Record Format ......................................... 45
Order of Audit Tokens .................................................. 46
Human-Readable Audit Record Format .......................... 46
header Token ............................................................ 47
trailer Token ............................................................ 48
arbitrary Token .......................................................... 48
arg Token ................................................................. 49
attr Token ................................................................. 49
exit Token ................................................................. 50
file Token ................................................................. 50
groups Token ............................................................. 50
in_addr Token ............................................................. 51
ip Token ................................................................. 51
ipc Token ................................................................. 51
ipc_perm Token ............................................................ 52
iport Token ............................................................... 52
opaque Token ............................................................ 52
path Token ............................................................... 53
process Token ............................................................. 53
return Token .............................................................. 54
seq Token ................................................................. 54
socket Token .............................................................. 54
subject Token ............................................................. 55
text Token ................................................................. 55
Using the auditreduce Command ................................. 56
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How auditreduce Helps in a Distributed System</td>
<td>56</td>
</tr>
<tr>
<td>Using auditreduce</td>
<td>57</td>
</tr>
<tr>
<td>▼ To Display the Whole Audit Log</td>
<td>57</td>
</tr>
<tr>
<td>▼ To Print the Whole Audit Log</td>
<td>57</td>
</tr>
<tr>
<td>▼ To Display User Activity on a Selected Data</td>
<td>57</td>
</tr>
<tr>
<td>▼ To Copy Login/Logout Messages to a Single File</td>
<td>57</td>
</tr>
<tr>
<td>▼ To Clean Up a not_terminated Audit File</td>
<td>58</td>
</tr>
<tr>
<td>Other Useful auditreduce Options</td>
<td>58</td>
</tr>
<tr>
<td>Using praudit</td>
<td>59</td>
</tr>
<tr>
<td>4. Device Allocation</td>
<td>61</td>
</tr>
<tr>
<td>Risks Associated With Device Use</td>
<td>62</td>
</tr>
<tr>
<td>Components of the Device-Allocation Mechanism</td>
<td>62</td>
</tr>
<tr>
<td>Using the Device-Allocation Utilities</td>
<td>63</td>
</tr>
<tr>
<td>The Allocate Error State</td>
<td>64</td>
</tr>
<tr>
<td>The device_maps File</td>
<td>64</td>
</tr>
<tr>
<td>The device_allocate File</td>
<td>66</td>
</tr>
<tr>
<td>Device-Clean Scripts</td>
<td>68</td>
</tr>
<tr>
<td>Object Reuse</td>
<td>68</td>
</tr>
<tr>
<td>Device-Clean Script for Tapes</td>
<td>69</td>
</tr>
<tr>
<td>Device-Clean Scripts for Diskettes and CD-ROM</td>
<td>69</td>
</tr>
<tr>
<td>Device-Clean Script for Audio</td>
<td>70</td>
</tr>
<tr>
<td>Writing New Device-Clean Scripts</td>
<td>70</td>
</tr>
<tr>
<td>Setting Up Lock Files</td>
<td>70</td>
</tr>
<tr>
<td>▼ To Set Up Lock Files for a Device to Be Made Allocatable</td>
<td>71</td>
</tr>
</tbody>
</table>
How the Allocate Mechanism Works .................... 71
Managing and Adding Devices ............................ 74
  ▼ To Manage Devices ................................ 74
  ▼ To Add a New Allocatable Device .................. 74
Using Device Allocations ................................. 75
  ▼ To Allocate a Device ................................. 75
  ▼ To Deallocate a Device .............................. 76
A. Audit Record Descriptions ......................... 77
Audit Record Structure ................................ 78
Audit Token Structure ................................ 78
  arbitrary Token .................................... 79
  arg Token .......................................... 81
  attr Token ......................................... 81
  exec_args Token ................................... 82
  exec_env Token .................................... 82
  exit Token ......................................... 83
  file Token ......................................... 83
  groups Token (Obsolete) ............................. 84
  header Token ...................................... 84
  in_addr Token ...................................... 85
  ip Token ........................................... 85
  ipc Token .......................................... 86
  ipc_perm Token .................................... 87
  iport Token ........................................ 88
<table>
<thead>
<tr>
<th>Token</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>newgroups Token</td>
<td>88</td>
</tr>
<tr>
<td>opaque Token</td>
<td>89</td>
</tr>
<tr>
<td>path Token</td>
<td>89</td>
</tr>
<tr>
<td>process Token</td>
<td>90</td>
</tr>
<tr>
<td>return Token</td>
<td>90</td>
</tr>
<tr>
<td>seq Token</td>
<td>91</td>
</tr>
<tr>
<td>socket Token</td>
<td>91</td>
</tr>
<tr>
<td>socket-inet Token</td>
<td>92</td>
</tr>
<tr>
<td>subject Token</td>
<td>92</td>
</tr>
<tr>
<td>text Token</td>
<td>93</td>
</tr>
<tr>
<td>trailer Token</td>
<td>94</td>
</tr>
<tr>
<td>Audit Records</td>
<td>95</td>
</tr>
<tr>
<td>General Audit Record Structure</td>
<td>95</td>
</tr>
<tr>
<td>Kernel-Level Generated Audit Records</td>
<td>95</td>
</tr>
<tr>
<td>User-Level Generated Audit Records</td>
<td>139</td>
</tr>
<tr>
<td>Event-to–System Call Translation</td>
<td>147</td>
</tr>
</tbody>
</table>

B. BSM Reference .............................. 155

Index ............................................ 159
Figures

Figure 2-1    Audit Trail Separated by Host ........................................... 21
Figure 2-2    Audit Trail Separated by Server ........................................... 22
Figure 2-3    audit_control File Entries ............................................... 34
Figure 4-1    Sample device_allocate File ............................................. 66
Figure A-1    Typical Audit Record ....................................................... 78
Figure A-2    arbitrary Token Format ..................................................... 80
Figure A-3    arg Token Format ............................................................. 81
Figure A-4    attr Token Format ............................................................. 81
Figure A-5    exec_args Token Format ...................................................... 82
Figure A-6    exec_env Token Format ....................................................... 82
Figure A-7    exit Token Format ............................................................. 83
Figure A-8    file Token Format ............................................................. 83
Figure A-9    groups Token Format ......................................................... 84
Figure A-10   header Token Format ......................................................... 85
Figure A-11   in_addr Token Format ....................................................... 85
Figure A-12   ip Token Format ............................................................... 86
| Figure A-13 | ipc Token Format | 86 |
| Figure A-14 | ipc_perm Token Format | 87 |
| Figure A-15 | iport Token Format | 88 |
| Figure A-16 | newgroups Token Format | 88 |
| Figure A-17 | opaque Token Format | 89 |
| Figure A-18 | path Token Format | 89 |
| Figure A-19 | process Token Format | 90 |
| Figure A-20 | return Token Format | 91 |
| Figure A-21 | seq Token Format | 91 |
| Figure A-22 | socket Token Format | 92 |
| Figure A-23 | socket-inet Token Format | 92 |
| Figure A-24 | subject Token Format | 93 |
| Figure A-25 | text Token Format | 93 |
| Figure A-26 | trailer Token Format | 94 |
**Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Audit Event Categories</td>
<td>8</td>
</tr>
<tr>
<td>2-2</td>
<td>Audit Classes</td>
<td>9</td>
</tr>
<tr>
<td>2-3</td>
<td>Prefixes Used in Audit Flags</td>
<td>10</td>
</tr>
<tr>
<td>2-4</td>
<td>Prefixes Used to Modify Already-Specified Audit Flags</td>
<td>11</td>
</tr>
<tr>
<td>2-5</td>
<td>Audit File Permissions</td>
<td>32</td>
</tr>
<tr>
<td>2-6</td>
<td>Possible Auditing Conditions</td>
<td>38</td>
</tr>
<tr>
<td>4-1</td>
<td>Device-Clean Script for the Three Supported Tape Devices</td>
<td>69</td>
</tr>
<tr>
<td>4-2</td>
<td>Device-Clean Scripts for the Diskette and CD-ROM Device</td>
<td>69</td>
</tr>
<tr>
<td>4-3</td>
<td>Device-Specification Options for <code>allocate</code></td>
<td>75</td>
</tr>
<tr>
<td>4-4</td>
<td>Options for the <code>list_devices</code> Command</td>
<td>75</td>
</tr>
<tr>
<td>A-1</td>
<td>Basic Security Module Audit Tokens</td>
<td>78</td>
</tr>
<tr>
<td>A-2</td>
<td><code>arbitrary</code> Token Print Format Field Values</td>
<td>80</td>
</tr>
<tr>
<td>A-3</td>
<td><code>arbitrary</code> Token Item Size Field Values</td>
<td>80</td>
</tr>
<tr>
<td>A-4</td>
<td>IPC Object Type Field</td>
<td>87</td>
</tr>
<tr>
<td>A-5</td>
<td>Event-to-System Call Translation</td>
<td>147</td>
</tr>
<tr>
<td>A-6</td>
<td>Event-to-Command Translation</td>
<td>152</td>
</tr>
<tr>
<td>Table B-1</td>
<td>Section 1M—Maintenance Commands</td>
<td>155</td>
</tr>
<tr>
<td>Table B-2</td>
<td>Section 2—System Calls</td>
<td>156</td>
</tr>
<tr>
<td>Table B-3</td>
<td>Section 3—C Library Functions</td>
<td>156</td>
</tr>
<tr>
<td>Table B-4</td>
<td>Section 4—Headers, Tables, and Macros</td>
<td>157</td>
</tr>
</tbody>
</table>
Preface

The Solaris™ SHIELD™ Basic Security Module (BSM) provides the security features defined as C2 in the Trusted Computer System Evaluation Criteria (TCSEC). The features provided by the BSM are the security auditing subsystem and a device-allocation mechanism that provides the required object-reuse characteristics for removable or assignable devices. C2 discretionary-access control and identification and authentication features are provided by the standard Solaris system.

Who Should Use This Book

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

How This Book Is Organized

Chapter 1, “Installation,” describes enabling and disabling the BSM. Topics covered include how to enable the Solaris system to use these additional security features, and how clients and servers interact in an enabled environment.
Chapter 2, “Administering Auditing,” explains the system management and configuration of the auditing subsystem. Topics discussed include managing audit trail storage, determining global and per-user preselection, and setting site-specific configuration options.

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

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

Appendix A, “Audit Record Descriptions,” describes in detail the content of the audit records generated.


What Typographic Changes Mean

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

<table>
<thead>
<tr>
<th>Typeface or Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AaBbCc123</td>
<td>The names of commands, files, and directories; on-screen computer output</td>
<td>Edit your .login file. Use ls -a to list all files. machine_name% You have mail.</td>
</tr>
</tbody>
</table>
Shell Prompts in Command Examples

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

Table P-2    Shell Prompts

<table>
<thead>
<tr>
<th>Shell</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C shell prompt</td>
<td><code>machine_name%</code></td>
</tr>
<tr>
<td>C shell superuser prompt</td>
<td><code>machine_name#</code></td>
</tr>
<tr>
<td>Bourne shell and Korn shell</td>
<td>$</td>
</tr>
<tr>
<td>prompt</td>
<td>#</td>
</tr>
</tbody>
</table>

**Table P-1    Typographic Conventions**

<table>
<thead>
<tr>
<th>Typeface or Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AaBbCc123</td>
<td>What you type, contrasted with on-screen computer output</td>
<td><code>machine_name% su</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>Password:</code></td>
</tr>
<tr>
<td>AaBbCc123</td>
<td>Command-line placeholder: replace with a real name or value</td>
<td>To delete a file, type <code>rm filename</code>.</td>
</tr>
<tr>
<td>AaBbCc123</td>
<td>Book titles, new words or terms, or words to be emphasized</td>
<td>Read Chapter 6 in User’s Guide.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These are called <em>class</em> options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You <em>must</em> be root to do this.</td>
</tr>
</tbody>
</table>
Starting with the Solaris 2.3 release, BSM has been included in the full release and is part of the release media. You do not need to install BSM separately because BSM is now enabled or disabled by running one of two simple scripts. All of the BSM software is included in the initial system installation, provided you install the following packages:

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

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

<table>
<thead>
<tr>
<th>Enabling BSM</th>
<th>page 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabling BSM</td>
<td>page 2</td>
</tr>
<tr>
<td>BSM and Client-Server ...</td>
<td>page 3</td>
</tr>
</tbody>
</table>
### Enabling BSM

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

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

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

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

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

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

### Disabling BSM

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

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

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

```bash
# /etc/telinit 6
```
BSM and Client-Server Relationships

The Solaris 2.1 release required two additional procedures for adding and deleting diskless clients from a BSM-enabled system. With the inclusion of BSM in the Solaris 2.3 and later releases, those procedures are no longer necessary. Enabling BSM on a server now automatically enables the BSM features on all of that server’s clients.
1
This chapter describes how to set up and administer auditing. Auditing allows system administrators to monitor the actions of the users. The auditing mechanism allows an administrator to detect potential security breaches. Auditing can reveal suspicious or abnormal patterns of system usage and provide the means to trace suspect actions back to a specific user. Auditing 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.

<table>
<thead>
<tr>
<th>More on Auditing</th>
<th>page 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Startup</td>
<td>page 6</td>
</tr>
<tr>
<td>Audit Classes and Events</td>
<td>page 7</td>
</tr>
<tr>
<td>Audit Flags</td>
<td>page 8</td>
</tr>
<tr>
<td>Process Audit Characteristics</td>
<td>page 14</td>
</tr>
<tr>
<td>How the Audit Trail Is Created</td>
<td>page 16</td>
</tr>
<tr>
<td>Controlling Audit Costs</td>
<td>page 22</td>
</tr>
<tr>
<td>Auditing Normal Users</td>
<td>page 25</td>
</tr>
<tr>
<td>Auditing Efficiently</td>
<td>page 25</td>
</tr>
<tr>
<td>Learning About the Audit Trail</td>
<td>page 26</td>
</tr>
<tr>
<td>Preventing Audit Trail Overflow</td>
<td>page 36</td>
</tr>
<tr>
<td>Setting Audit Policies</td>
<td>page 39</td>
</tr>
<tr>
<td>Changing Class Definitions</td>
<td>page 40</td>
</tr>
</tbody>
</table>
More on Auditing

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

Auditing makes it possible to:

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

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

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

The rest of this chapter describes how to set up and administer auditing. How to interpret the audit data is described in Chapter 4, “Audit Trail Analysis.”

Audit Startup

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

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

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

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

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

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

**Kernel Events**

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

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

<table>
<thead>
<tr>
<th>Number Range</th>
<th>Type of Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048–65535</td>
<td>User-level audit events</td>
</tr>
<tr>
<td>2048–32767</td>
<td>Reserved for SunOS user-level programs</td>
</tr>
<tr>
<td>32768–65536</td>
<td>Available for third-party applications</td>
</tr>
</tbody>
</table>

Audit Records

Each \textit{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 \textit{audit tokens}. Each time an audit record is created for an event, the record contains some or all of the tokens defined for it, depending on the nature of the event. The audit record descriptions in Appendix A list all the audit tokens defined for each event and what each token means.

Audit records are collected in a trail (see the \texttt{audit.log(4) man page}) and may be converted to a human readable format by \texttt{praudit} (see the \texttt{praudit(1M) man page}). See Chapter 3, “Audit Trail Analysis,” for details.

Audit Flags

Audit \textit{flags} indicate classes of events to audit. Machine-wide defaults for auditing are specified for all users on each machine by flags in the \texttt{audit_control} file, which is described in “The audit_control File” on page 11.
The system administrator can modify what gets audited for individual users by putting audit flags in a user’s entry in the `audit_user` file. The audit flags are also used as arguments to `auditconfig` (see the `auditconfig(1M)` man page).

**Definitions of Audit Flags**

Each predefined audit class is shown in Table 2-2 with the audit flag (which is the short name that stands for the class), the long name, a short description, and a longer definition. The system administrator uses the audit flags in the auditing configuration files to specify which classes of events to audit. Additional classes can be defined and existing classes can be renamed by modifying the `audit_class` file (see the `audit_class(4)` man page).

**Table 2-2  Audit Classes**

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

The following table shows prefixes that specify whether the audit class is audited for success or failure or both.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Audit for both success and failure</td>
</tr>
<tr>
<td>+</td>
<td>Audit for success only</td>
</tr>
<tr>
<td>-</td>
<td>Audit for failure only</td>
</tr>
</tbody>
</table>

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

Caution – The `all` flag can generate large amounts of data and fill up audit file systems quickly, so use it only if you have extraordinary reasons to audit everything.
Prefixes to Modify Previously Set Audit Flags

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

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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Turn off for failed attempts</td>
</tr>
<tr>
<td>+</td>
<td>Turn off for successful attempts</td>
</tr>
<tr>
<td>^</td>
<td>Turn off for both failed and successful attempts</td>
</tr>
</tbody>
</table>

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

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

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

The audit_control File

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

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

- The audit flags line (flags:) contains the audit flags that define what classes
  of events are audited for all users on the machine. The audit flags specified
  here are referred to as the machine-wide audit flags or the machine-wide audit
  preselection mask. Audit flags are separated by commas, with no spaces.

- The nonattributable flags line (naflags:) contains the audit flags that define
  what classes of events are audited when an action cannot be attributed to a
  specific user. The flags are separated by commas, with no spaces.

- The audit threshold line (minfree:) defines the minimum free-space level
  for all audit file systems. See “What Makes a Directory Suitable” on page 17.
  The minfree percentage must be greater than or equal to 0. The default is
  20 percent.

- The directory definition lines (dir:) define which audit file systems and
  directories the machine will use to store its audit trail files.

  There may be one or more directory definition lines. The order of the dir:
  lines is significant, because auditd opens audit files in the directories in the
  order specified (see the audit(1M) man page). The first audit directory
  specified is the primary audit directory for the machine, the second is the
  secondary audit directory where the audit daemon puts audit trail files
  when the first one fills, and so forth.

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

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

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

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

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

User Audit Fields in the audit_user File

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

In the audit_user file entry for each user, there are three fields. The first field is the username, the second field is the always-audit field, the third is the never-audit field.

The two auditing fields are processed in sequence, so auditing is enabled by the first field and turned off by the second.
Note – Avoid the common mistake of leaving the all set in the never-audit field. This causes all auditing to be turned off for that user, overriding the flags set in the always-audit field.

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

The correct entry:

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

The incorrect entry:

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

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

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

Process Audit Characteristics

The following audit characteristics are set at initial login:

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

**Process Preselection Mask**

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

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

\[
\text{user’s process preselection mask} = (\text{flags: line + always audit flags}) - \text{never audit flags}
\]

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

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

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

The audit_warn script is run by auditd whenever the daemon switches audit directories or encounters difficulty (such as a lack of storage). As distributed, the audit_warn script sends mail to an audit_warn alias and sends a message to the console. Your site should customize audit_warn to suit your needs. Customizing the audit_warn script is described in “The audit_warn Script” on page 18.

The audit_data File

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

```
# cat /etc/security/audit_data
116:/etc/security/audit/blinken.1/files/19910320100002.not_terminated.lazy
```
The Audit Daemon’s Role

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

• auditd opens and closes audit log files in the directories specified in the audit_control file in the order in which they are specified.

• auditd reads audit data from the kernel and writes it to an audit file.

• auditd executes the audit_warn script when the audit directories fill past limits specified in the audit_control file. The script, by default, sends warnings to the audit_warn alias and to the console.

• With the system default configuration, when all audit directories are full, processes that generate audit records are suspended and auditd writes a message to the console and to the audit_warn alias. (The auditing policy can be reconfigured with autoconfig.) At this point only the system administrator 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 machine is brought up to multiuser mode, or when the audit daemon is instructed by the audit -s command to reread the file after the file has been edited, auditd determines the amount of free space necessary and reads the list of directories from the audit_control file 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. The pointer may be reset to the beginning of the list if the administrator enters the audit -s command. When you use the audit -n command to instruct the daemon to switch to a new audit file, the new file is created in the same directory as the current file.

What Makes a Directory Suitable

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

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

Keeping Audit Files Manageable

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

The audit_warn Script

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

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

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

The audit_warn script is invoked with the string soft and the name of the directory whose space available has gone below the minimum. The audit daemon switches automatically to the next suitable directory, and writes the audit files there until this new directory reaches its minfree
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.

Using the auditreduce Command

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

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

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

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

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

Following are some of the actions performed by some of the options to the auditreduce command.
• You can request that the output contain audit records generated by only certain audit flags.

• You can request audit records generated by one particular user.

• You can request audit records generated on specific dates.

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

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

Figure 2-1  Audit Trail Separated by Host

The audit data may not be in the default directory. Perhaps because the partition for /etc/security/audit is very small or because you want to store audit data on another partition without symbolically linking that partition to /etc/security/audit. You can give auditreduce another directory (-R) to substitute for /etc/security/audit, or you can specify one particular subdirectory (-S):
You can direct `auditreduce` to treat only certain files by specifying them as command arguments:

```bash
```

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

### Controlling Audit Costs

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

- Costs in increased processing time
- Costs of analysis of audit data
- Costs of storage of audit data
Cost of Increased Processing Time

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

Cost of Analysis

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

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

Cost of Storage

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

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

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

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

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

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

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

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

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

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

To change the audit flags for a specific user to a supplied value, use the `auditconfig` command with the `-setpmask`, `-setsmask`, or `-setumask` options. The command changes the process audit flags for one process, one audit session ID, or one audit user ID respectively. See the `auditconfig(1M)` man page and “The auditconfig Command” on page 37.
Auditing Normal Users

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

Auditing Efficiently

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

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

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

To monitor the audit trail in real time and watch for unusual events, write a script that monitors creation of audit files on all the audit file servers and processes them with the tail command (see the tail(1) man page). The output of tail -0f, piped through praudit, yields a stream of audit records immediately as they are generated. This stream can be analyzed for unusual message types or other indicators and delivered to the auditor or used to trigger automatic responses. The script should be written to constantly watch the audit directories for the appearance of new not_terminated audit files, and also termination of outstanding tail processes when their files are no longer being written to (that is, have been replaced by new ones).
To Combine and Reduce audit Files

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

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

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

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

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

Learning About the Audit Trail

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

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

Most often the audit directories will be separate audit file system partitions. Even though they can be included in other file systems, this is not recommended.
As a rule, locate primary audit directories in dedicated audit file systems mounted on separate partitions. Normally, all audit file system are subdirectories of /etc/security/audit. These should be dedicated audit file systems to ensure that normal use of the partition is not interrupted, if the audit directories become filled with audit files.

Even though you can physically locate audit directories within other file systems that are not dedicated to auditing, do not do this except for directories of last resort. Directories of last resort 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 a software development environment if auditing is optional, and where it is more important to make full use of disk space than to keep an audit trail. Putting audit directories within other file systems would never be acceptable in a security-conscious production environment.

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

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

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

More About the Audit Files

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

Audit File Naming

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

```
start-time.finish-time.machine
```
where \textit{start-time} is the time of the first audit record in the audit file, \textit{finish-time} is the time of the last record, and \textit{machine} is the name of the machine that generated the file. Some examples of these names can be found in “Example of a Closed Audit File Name” on page 29.

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

\begin{verbatim}
start-time.not_terminated.machine
\end{verbatim}

\textbf{How Audit File Names Are Used}

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

\textbf{Time-Stamp Format and Interpretation}

The \textit{start-time} and \textit{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.

\begin{verbatim}
YYYYMMDDHHMMSS
\end{verbatim}

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 \texttt{auditreduce}.

\textbf{Example of a File Name for a Still-Active File}

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

\begin{verbatim}
YYYYMMDDHHMMSS.not_terminated.hostname
\end{verbatim}
Here is an example:

```
19900327225243.not_terminated.lazy
```

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

**Example of a Closed Audit File Name**

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

```
YYYYMMDDHHMMSS.YYYYYMDDHHMMSS.hostname
```

Here is an example:

```
19900320005243.19900327225351.lazy
```

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

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

**Handling Nonactive Files Marked `not_terminated`**

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

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

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

▼ To Create Audit Partitions and Export Them

1. Assign at least one primary audit directory to each machine. The primary audit directory is the directory where a machine places its audit files under normal conditions.

2. Assign at least one secondary audit directory to each machine that is located on a different audit file server than the primary directory. The secondary audit directory is where a machine places audit files if the primary directory is full or inaccessible, because of network failure, NFS server crash, or some other reason.

3. On every diskfull machine create a local audit directory of last resort (preferably a dedicated audit file system) that is used when the network is inaccessible or the primary and secondary directories are unusable.

4. Spread the directories used as primary and secondary destinations evenly over the set of audit servers in the system.
5. Create audit file systems according to the requirements discussed in this section.

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

The audit 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 server winken are /etc/security/audit/winken and /etc/security/audit/winken.1.

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

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

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

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

Make sure that each audit directory contains nothing except audit files.
6. Assign the required permissions to the audit file systems.
   The permissions that must appear on the /etc/security/audit/server-name
directory and the files directory that must be created beneath it on
the audit server are shown in Table 2-5.

<table>
<thead>
<tr>
<th>Owner</th>
<th>Group</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>staff</td>
<td>2750</td>
</tr>
</tbody>
</table>

Table 2-5  Audit File Permissions

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

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

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

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

2. Assign the audit file systems to the dedicated partitions.
   Each diskfull machine should have a backup audit directory on the local
machine in case it’s NFS-mounted audit file systems are not available.
3. While each machine is in single-user mode, run `tunefs -m 0` on each dedicated audit partition to reduce reserved file system space to 0 percent. A reserved space percentage (called the `minfree` limit) is specified for audit partitions in the `audit_control` file. The default is 20 percent, and this percentage is tunable. Because this value is set by each site in the `audit_control` file, you should remove the automatically reserved file system space that is set aside by default for all file systems.

4. Set the required permissions on each of the audit directories on the audit servers, and make a subdirectory in each audit directory called `files`. Use `chown` and `chmod` to assign each audit directory and each `files` subdirectory the required permissions.

5. If using audit servers, export the audit directories with the `/etc/dfs/dfstab` file.

6. Create the `audit_control` file entries for all the audit directories in the `audit_control` file on each machine, specifying the `files` subdirectory.

7. On each audit client, create the entries for the audit file systems in the `/etc/vfstab` file.

8. On each audit client, create the mount point directories and use `chmod` and `chown` to set the correct permissions.

▼ To Plan Audit Configuration

First, plan for audit trail storage.

1. In the `/etc/security/audit_class` file, define the classes needed at your site.
   If the default classes are suitable, you do not need to define new ones. See the `audit_class(4)` man page.

2. Set up event-to-class mapping in `/etc/security/audit_event`. This step is not needed if the default mapping suits your site’s needs. See the `audit_event(4)` man page.

3. Determine how much auditing your site needs to do.
   Balance your site’s security needs against the availability of disk space for audit trail storage.
See “Controlling Audit Costs” on page 22, “Auditing Efficiently” on page 25, and “Learning About the Audit Trail” on page 26 for guidance on how to reduce storage requirements while still maintaining site security and on how to design audit storage.

4. Determine which machines will be audit servers and which will be clients of the audit servers.

5. Determine the names and locations of audit file systems.

6. Plan which machines will use which audit file systems on the audit servers.

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

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

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

See “Process Audit Characteristics” on page 14.

3. Determine the minimum free space (minfree), also called the soft limit, that should be on an audit file system before a warning is sent. When the amount of space available goes below the minfree percentage, the audit daemon switches to the next suitable audit file system and send a notice that the soft limit has been exceeded. (What makes an audit file system suitable is defined in “What Makes a Directory Suitable” on page 17.)

A certain amount of auditing is configured by default on each machine. The default audit_control file contains the lines shown in Figure 2-3, which set the audit directory as /var/audit, one system-wide audit flag (lo), a minfree threshold of 20 percent, and one nonattributable flag.

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

*Figure 2-3 audit_control File Entries*

4. Edit the `/etc/security/audit_control` file.
a. Specify which audit file systems to use for audit trail storage on this machine.
   Make a `dir:` entry for each audit directory available to the current machine. See “Learning About the Audit Trail” on page 26 for how to set up the audit directory scheme for the distributed system.

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

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

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

5. Use `auditconfig` to modify the audit policy if modification is desired.
   See the `auditconfig(1M)` man page or “The auditconfig Command” on page 37. The policy variable is a dynamic kernel variable, so its value is not saved when the system is brought down. Therefore, you should set the desired policy using the appropriate startup script.

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

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

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

   **Warning** – This entry must be placed below the entry for the root user for processes owned by root to function properly.
b. To add a corresponding entry into the `/etc/shadow` file, type the following.

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

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

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

```
audit:no:all
```

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

```
# passwd audit
```

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

**Preventing Audit Trail Overflow**

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

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

▼ **To Prevent Audit Trail Overflow**

If your security policy requires that all audit data be saved, do the following:
1. Set up a schedule to regularly archive audit files and to delete the archived audit files from the audit file system.

2. Manually archive audit files by backing them up on tape or moving them to an archive file system.

3. Store context-sensitive information that will be needed to interpret audit records along with the audit trail.

4. Keep records of what audit files are moved off line.

5. Store the archived tapes appropriately.

6. Reduce the volume of audit data you store by creating summary files. You can extract summary files from the audit trail using options to `auditreduce`, so that the summary files contain only records for certain specified types of audit events. An example of this would be a summary file containing only the audit records for all logins and logouts. See Chapter 3, “Audit Trail Analysis.”

**The `auditconfig` Command**

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

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

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

Get the machine-auditing condition. Table 2-6 shows the possible responses.

<table>
<thead>
<tr>
<th>Response</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditing</td>
<td>Auditing is enabled and turned on.</td>
</tr>
<tr>
<td>no audit</td>
<td>Auditing is enabled but turned off.</td>
</tr>
<tr>
<td>disabled</td>
<td>The audit module is not enabled.</td>
</tr>
</tbody>
</table>

-setcond condition

Set the machine-auditing condition: auditing or noaudit.

-getclass event_number

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

-setclass event_number audit_flags

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

-lsevent

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

-getpinfo pid

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

-setpmask pid flags

Set the preselection mask of the specified process.

-setsmask asid flags

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

-setumask 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 Solaris-BSM audit policies. The auditconfig command with the -lspolicy argument shows the audit policies that you can change. The policy flags are described below.

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

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

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

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

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

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

seq

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

▼ To Change Which Events Are in Which Audit Classes

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

1. Edit the /etc/security/audit_event file to change the class mapping for each event to be changed.

2. Reboot the system or run auditconfig -conf to change the runtime kernel event-to-class mappings.

Changing Class Definitions

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

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

<table>
<thead>
<tr>
<th>Mask</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
<td>invalid class</td>
</tr>
<tr>
<td>0x1</td>
<td>fr</td>
<td>file read</td>
</tr>
<tr>
<td>0x2</td>
<td>fw</td>
<td>file write</td>
</tr>
<tr>
<td>0x4</td>
<td>fa</td>
<td>file attribute access</td>
</tr>
<tr>
<td>0x8</td>
<td>fm</td>
<td>file attribute modify</td>
</tr>
<tr>
<td>0x10</td>
<td>fc</td>
<td>file create</td>
</tr>
<tr>
<td>0x20</td>
<td>fd</td>
<td>file delete</td>
</tr>
<tr>
<td>0x40</td>
<td>cl</td>
<td>file close</td>
</tr>
<tr>
<td>0xffffffff</td>
<td>all</td>
<td>all classes</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Auditing Features</th>
<th>page 43</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools for Merging, Selecting, Viewing, and Interpreting Audit Records</td>
<td>page 44</td>
</tr>
<tr>
<td>Audit Record Format</td>
<td>page 45</td>
</tr>
<tr>
<td>Using the auditreduce Command</td>
<td>page 56</td>
</tr>
<tr>
<td>Using praudit</td>
<td>page 59</td>
</tr>
</tbody>
</table>

Auditing Features

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

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

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

Audit User ID

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

Audit Session ID

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

Self-Contained Audit Records

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

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

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

• The auditreduce command allows you to choose sets of records to examine. For instance, you can select all records from the past 24 hours to generate a daily report; you can select all records generated by a specific user to examine that user’s activities; or you can select all records caused by a specific event type to see how often that type occurs.
• The `praudit` command allows you to display audit records interactively and create very basic reports. `praudit` displays records in one of several human-readable but otherwise non-interpreted forms. You may accomplish more sophisticated display and reporting by postprocessing the output from `praudit` (with `sed` or `awk`, for instance) or by writing programs that interpret and process the binary audit records.

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

## Audit Record Format

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

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

## Binary Format

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

## Audit Event Type

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

## Audit Token Types

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

**Order of Audit Tokens**

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

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

- header
- subject
- return
- trailer

The `trailer` token is optional, depending on the `trail` policy that can be set with the `auditconfig` command.

**Human-Readable Audit Record Format**

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

The following token examples show the form that `praudit` produces by default. Examples are also provided of raw (`-r`) and short (`-s`) options. When `praudit` displays an audit token, it begins with the token type, followed by the data from the token. Each data field from the token is separated from other fields by a comma. However, if a field (such as a path name) contains a comma, this cannot be distinguished from a field-separating comma. Use a different field separator or the output will contain commas. The token type is displayed by default as a name, like `header`, or in `-r` format as a decimal number (18).
The individual tokens are described in the following order:

- “header Token” on page 47
- “trailer Token” on page 48
- “arbitrary Token” on page 48
- “arg Token” on page 49
- “attr Token” on page 49
- “exit Token” on page 50
- “file Token” on page 50
- “groups Token” on page 50
- “in_addr Token” on page 51
- “ip Token” on page 51
- “ipc Token” on page 51
- “ipc_perm Token” on page 52
- “iport Token” on page 52
- “opaque Token” on page 52
- “path Token” on page 53
- “process Token” on page 53
- “return Token” on page 54
- “seq Token” on page 54
- “socket Token” on page 54
- “subject Token” on page 55
- “text Token” on page 55

**header Token**

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

- A token ID
- The record length in bytes, including the header and trailer tokens
- An audit record structure version number
- An event ID identifying the type of audit event
- An event ID modifier with descriptive information about the event type
- The time and date the record was created
When displayed by `praudit` in default format, a header token looks like the following example from `ioctl`:

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

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

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

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

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

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

**trailer Token**

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

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

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

```
trailer,136
```

**arbitrary Token**

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

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

An arbitrary token is displayed by praudit as follows:

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

**arg Token**

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

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

An arg token is displayed by praudit as follows:

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

**attr Token**

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

- A token ID
- The file access mode and type
- The owner user ID
- The owner group ID
- The file system ID
- The inode ID
- The device ID that the file might represent
An attr token is displayed by praudit as follows:

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

**exit Token**

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

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

An exit token is displayed by praudit as follows:

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

**file Token**

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

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

A file token is displayed by praudit as follows:

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

**groups Token**

A groups token records the groups entries from a process’s credential. The fields are:
• A token ID
• An array of groups entries of size NGROUPS_MAX (16)

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

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

**in_addr Token**

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

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

```
ip addr, 129.150.113.7
```

**ip Token**

The `ip` token contains a copy of an Internet Protocol header. The fields are:
• A token ID
• A 20-byte copy of an IP header

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

```
ip address, 0.0.0.0
```

**ipc Token**

This token contains the System V IPC message/semaphore/shared-memory handle used by a caller to identify a particular IPC object. The fields are:
• A token ID
• An IPC object type identifier
• The IPC object handle
An ipc token is displayed by praudit as follows:

```
IPC, msg, 3
```

**ipc_perm Token**

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

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

An ipc_perm token is displayed by praudit as follows:

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

**iport Token**

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

- A token ID
- A TCP/UDP address

An iport token is displayed by praudit as follows:

```
iport, 0xf6d6
```

**opaque Token**

The opaque token contains unformatted data as a sequence of bytes. The fields are:
• A token ID
• A byte count of the data array
• An array of byte data

An opaque token is displayed by praudit as follows:

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

**path Token**

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

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

A path token is displayed by praudit as follows:

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

**process Token**

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

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

A process token is displayed by praudit as follows:

```
process,root,root,wheel,root,wheel,0,0,0.0.0.0
```
return Token

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

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

A return token is displayed by praudit as follows:

```
return,success,0
```

seq Token

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

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

A seq token is displayed by praudit as follows:

```
sequence,1292
```

socket Token

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

- A token ID
- A socket type field (TCP/UDP/UNIX)
- The local port address
- The local Internet address
- The remote port address
- The remote Internet address
A socket token is displayed by `praudit` as follows:

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

**subject Token**

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

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

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

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

**text Token**

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

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

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

```
text,aw_test_token
```
Using the auditreduce Command

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

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

The praudit command, described in “Using praudit” on page 59,” makes the records human-readable.

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

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

How auditreduce Helps in a Distributed System

When multiple machines running Solaris BSM are administered as part of a distributed system, each machine performs auditable events, and each machine writes audit records to its own machine-specific audit file. This simplifies software and is robust in the face of machine failures. However, unless a tool existed to make it easier, you would have to look at every one of the files to determine a particular user’s actions.

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

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

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

▼ To Display the Whole Audit Log

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

```
# auditreduce | praudit
```

▼ To Print the Whole Audit Log

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

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

▼ To Display User Activity on a Selected Data

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

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

▼ To Copy Login/Logout Messages to a Single File

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

```
# auditreduce -c lo -d 870413 -O /usr/audit_summary/logins
```
The -O option creates an audit file with 14-character timestamps for both start-time and end-time, and the suffix logins:

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

**To Clean Up a not_terminated Audit File**

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

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

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

**Other Useful auditreduce Options**

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

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

If `–a` is not specified, `auditreduce` defaults to 00:00:00, January 1, 1970. If `–b` is not specified, `auditreduce` defaults to the current time of day (GMT). The `–d` option selects a particular 24-hour period, as shown in “To Copy Login/Logout Messages to a Single File” on page 57.
The `auditreduce -a` command with the date shown in the following screen example sends all audit records created after midnight on July 15, 1991, to `praudit`.

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

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

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

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

### Using `praudit`

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

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

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

```
header,240,1,ioc1(2),es,Tue Sept  1 16:11:44 1992, + 270000 msec
```
And here is the output from `praudit -r` for the same header token:

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

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

```bash
#!/bin/sh
praudit | sed -e '1,2d' -e 's/\^header/\^header/' \| 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.

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

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

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

| Risks Associated With Device Use                  | page 62 |
| Components of the Device-Allocation Mechanism     | page 62 |
| Using the Device-Allocation Utilities             | page 63 |
| The Allocate Error State                          | page 64 |
| The device_maps File                              | page 64 |
| The device_allocate File                          | page 66 |
| Device-Clean Scripts                              | page 68 |
| Setting Up Lock Files                             | page 70 |
| Managing and Adding Devices                       | page 74 |
| Using Device Allocations                          | page 75 |
For one example of the security risks associated with the use of various I/O devices, consider how cartridge devices are typically used. Often several users share a single tape drive, which may be located in an office or lab away from where an individual user’s own machine is located. This means that, after the user loads a tape into the tape drive, some length of time may elapse before the user can return to the machine to invoke the command that reads or writes data to or from the tape. Then another time lapse occurs before the user is able to return and take the tape out of the drive. Because tape devices are typically accessible to all users, during the time when the tape is unattended a unauthorized user could access or overwrite data on the tape.

The device-allocation mechanism makes it possible to assign certain devices to one user at a time, so that the device can only be accessed by that user while it is assigned to that user’s name.

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

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

Components of the Device-Allocation Mechanism

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

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

How any user invokes the allocate, dealloc, dminfo, and list_devices commands is described in “Using the Device-Allocation Utilities” on page 63. All of the options and other descriptions are defined in the man pages.

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

Using the Device-Allocation Utilities

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

allocate

-F device_special_filename

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

-U username

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

deleolate

-F device_special_filename

Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. When a user forgets to deallocate a tape drive, you can force deallocation using the -F option while you are root.
-I
Forces deallocation of all allocatable devices. This option should only be used at system initialization.

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

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

The Allocate Error State

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

The device_maps File

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

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

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

device-name:device-type:device-list

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

Leading and trailing blanks are allowed in any of the fields.

device-name

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

device-type

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

device-list

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

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

```
fd0:
    fd:
        /dev/fd0 /dev/fd0a /dev/fd0b /dev/fd0c /dev/rfd0 /dev/rfd0a /dev/rfd0b /dev/rfd0c:
        .
        .
    st0:
        st:
            /dev/rst0 /dev/rst8 /dev/rst16 /dev/rst24 /dev/nrst0 /dev/nrst8 /dev/nrst16 /dev/nrst24:
```

The **device_allocate File**

Modify the device_allocate file to change devices from allocatable to nonallocatable or to add new devices. Figure 4-1 shows a sample device_allocate file.

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

*Figure 4-1  Sample device_allocate File*

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

The entries for devices in the device_allocate file may be modified by the administrator after installation. Any device that needs to be allocated before use must be defined in the device_allocate file on each machine. Currently, cartridge tape drives, diskette drives, CD-ROM devices, and audio chips are considered allocatable and have device-clean scripts.
Note – If you add an Xylogics tape drive or an Archive tape drive, they can also use the `st_clean` script supplied for SCSI devices. Other devices that you could make allocatable are modems, terminals, graphics tablets, and the like, but you need to create your own device-clean scripts for such devices, and the script must fulfill object-reuse requirements for that type of device.

An entry in the `device_allocate` file does not mean the device is allocatable, unless the entry specifically states the device is allocatable. Notice in Figure 4-1 on page 66 an asterisk (*) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable; that is, the system administrator does not require a user to allocate the device before it is used nor to deallocate it afterward. Any other string placed in this field indicates that the device is allocatable.

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

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

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

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

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

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

**device-name**

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

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

reserved

These fields are reserved for future use.

alloc

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

device-clean

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

Device-Clean Scripts

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

Object Reuse

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

The three supported tape devices and the device-clean script for each are shown in Table 4-1.

Table 4-1 Device-Clean Script for the Three Supported Tape Devices

<table>
<thead>
<tr>
<th>Tape Device Type</th>
<th>Device-Clean Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI 1/4-inch tape</td>
<td>st_clean</td>
</tr>
<tr>
<td>Archive 1/4-inch tape</td>
<td>st_clean</td>
</tr>
<tr>
<td>Open-reel 1/2-inch tape</td>
<td>st_clean</td>
</tr>
</tbody>
</table>

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

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

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

Device-Clean Scripts for Diskettes and CD-ROM

The device-clean scripts for the diskettes and CD-ROM devices are shown in Table 4-2.

Table 4-2 Device-Clean Scripts for the Diskette and CD-ROM Device

<table>
<thead>
<tr>
<th>Disk Device Type</th>
<th>Device-Clean Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>diskette</td>
<td>fd_clean</td>
</tr>
<tr>
<td>CD-ROM</td>
<td>sr_clean</td>
</tr>
</tbody>
</table>

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

The audio device is cleaned up with an audio-clean script. The script performs an AUDIO_DRAIN ioctl system call to flush the device, and then an AUDIO_SETINFO ioctl system call to reset the device configuration to default. In addition, the script retrieves the audio chip registers using the AUDIOGETREG ioctl system call. Any registers deviating from default are reset using the AUDIOSETREG ioctl system call.

Writing New Device-Clean Scripts

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

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

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

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

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

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

Setting Up Lock Files

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

If no lock file exists for an allocatable device, the device cannot be allocated, and no one can access the device.
To Set Up Lock Files for a Device to Be Made Allocatable

1. Use the `dminfo` command to get the device name for the device from its entry in the `device_maps` file.
   See “The device_maps File” on page 64 and the `dminfo(1M)` and `device_maps(4)` man pages. For example, the device name for device type st is `st0`. Use the device name as the name of the lock file.

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

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

How the Allocate Mechanism Works

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

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

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

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

In this screen, user vanessa allocates device st0.

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

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

If it finds the lock file for the device with the correct ownership and permissions, the allocate command then checks to make sure the device has an entry in the device_allocate file and that the entry specifies that the device is allocatable.
In this example, the default `device_allocate` entry for the `st0` device specifies that the device is allocatable. Because the `allocate` command finds that all the above conditions are met, the device is allocated to `vanessa`.

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

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

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

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

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

▼ To Manage Devices

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

▼ To Add a New Allocatable Device

1. Create an entry for any new allocatable device on the machine in the device_allocate file.
   How to do this is described in “The device_allocate File” on page 66.

2. Create an empty lock file for each allocatable device in the /etc/security/dev directory.
   How to do this is described in “Setting Up Lock Files” on page 70.

3. Create a device-clean script if needed, for each new device.
   If you add a Xylogics or an Archive tape drive, you can use the st_clean script; otherwise, create your own. How to create a device-handling script is described in “Device-Clean Scripts” on page 68.

4. Make all device-special files for the device to be owned by user bin, group bin and mode 000.
   You can run the dminfo command to get a listing from the device_maps file of all the device-special files that are associated with the device you are making allocatable.
Using Device Allocations

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

- **allocate** assigns a device to a user.
  You can specify the device in either of the two ways shown in Table 4-3.

- **deallocate** releases a previously allocated device.

- **list_devices** allows you to see a list of all allocatable devices, devices currently allocated, and allocatable devices not currently allocated.

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>device-name</td>
<td>Allocate the device that matches the device name.</td>
</tr>
<tr>
<td>-g device-type</td>
<td>Allocate the device that matches the device group type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l</td>
<td>List all allocatable devices or information about the device.</td>
</tr>
<tr>
<td>-n</td>
<td>List devices not currently allocated or information about the device.</td>
</tr>
<tr>
<td>-u</td>
<td>List devices currently allocated or information about the device.</td>
</tr>
</tbody>
</table>

**To Allocate a Device**

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

```
sar1% allocate st0
```
If the command cannot allocate the device, an error message displays in the console window. A list of all error messages appears in the `allocate(1M)` man page.

▼ To Deallocate a Device

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

```
sarl% deallocate st0
```

Deallocation allows other users to allocate the device when you are finished.
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 by the Basic Security Module by event description.
Audit Record Structure

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

<table>
<thead>
<tr>
<th>Audit Record Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>header token</td>
</tr>
<tr>
<td>arg token</td>
</tr>
<tr>
<td>data token</td>
</tr>
<tr>
<td>subject token</td>
</tr>
<tr>
<td>return token</td>
</tr>
</tbody>
</table>

Figure A-1 Typical Audit Record

Audit Token Structure

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

Table A-1 Basic Security Module Audit Tokens

<table>
<thead>
<tr>
<th>Token Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary</td>
<td>Data with format and type information</td>
</tr>
<tr>
<td>arg</td>
<td>System call argument value</td>
</tr>
<tr>
<td>attr</td>
<td>Vnode tokens</td>
</tr>
<tr>
<td>exec_args</td>
<td>Exec system call arguments</td>
</tr>
<tr>
<td>exec_env</td>
<td>Exec system call environment variables</td>
</tr>
<tr>
<td>exit</td>
<td>Program exit information</td>
</tr>
<tr>
<td>file</td>
<td>Audit file information</td>
</tr>
<tr>
<td>groups</td>
<td>Process groups information (obsolete)</td>
</tr>
<tr>
<td>header</td>
<td>Indicates start of record</td>
</tr>
<tr>
<td>in_addr</td>
<td>Internet address</td>
</tr>
</tbody>
</table>
An audit record always contains a header token and a trailer token. The header token indicates where the audit record begins in the audit trail. Every audit record contains a subject token, except for audit records from some nonattributable events. In the case of attributable events, these two tokens refer to the values of the process that caused the event. In the case of asynchronous events, the process tokens refer to the system.

**arbitrary** Token

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

### Table A-1  Basic Security Module Audit Tokens  (Continued)

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

---

Audit Record Descriptions

79
example, short), and a count field that gives the number of following items. The remainder of the token is composed of one or more items of the specified type. The arbitrary token appears as follows:

<table>
<thead>
<tr>
<th>token ID</th>
<th>print format</th>
<th>item size</th>
<th>number items</th>
<th>item 1</th>
<th>item n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Figure A-2* arbitrary Token Format

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

*Table A-2* arbitrary Token Print Format Field Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUP_BINARY</td>
<td>Print date in binary</td>
</tr>
<tr>
<td>AUP_OCTAL</td>
<td>Print date in octal</td>
</tr>
<tr>
<td>AUP_DECIMAL</td>
<td>Print date in decimal</td>
</tr>
<tr>
<td>AUP_HEX</td>
<td>Print date in hex</td>
</tr>
<tr>
<td>AUP_STRING</td>
<td>Print date as a string</td>
</tr>
</tbody>
</table>

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

*Table A-3* arbitrary Token Item Size Field Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUR_BYTE</td>
<td>Data is in units of bytes (1 byte)</td>
</tr>
<tr>
<td>AUR_SHORT</td>
<td>Data is in units of shorts (2 bytes)</td>
</tr>
<tr>
<td>AUR_LONG</td>
<td>Data is in units of longs (4 bytes)</td>
</tr>
</tbody>
</table>
arg Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>argument #</th>
<th>argument value</th>
<th>text length</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>4 bytes</td>
<td>2 bytes</td>
<td>n bytes</td>
</tr>
</tbody>
</table>

Figure A-3 arg Token Format

attr Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>file mode</th>
<th>owner UID</th>
<th>owner GID</th>
<th>file system ID</th>
<th>file inode ID</th>
<th>device ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Figure A-4 attr Token Format
exec_args Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>count</th>
<th>env_args</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>count null-terminated strings</td>
</tr>
</tbody>
</table>

Figure A-5  exec_args Token Format

Note – The exec_args token is output only when the audit policy argv is active. See “Setting Audit Policies” on page 39 for more information.

exec_env Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>count</th>
<th>env_args</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>count null-terminated strings</td>
</tr>
</tbody>
</table>

Figure A-6  exec_env Token Format

Note – The exec_env token is output only when the audit policy argv is active. See “Setting Audit Policies” on page 39 for more information.
exit Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>status</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Figure A-7  exit Token Format

file Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>date &amp; time</th>
<th>name length</th>
<th>previous/next file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>8 bytes</td>
<td>2 bytes</td>
<td>n bytes</td>
</tr>
</tbody>
</table>

Figure A-8  file Token Format
groups Token (Obsolete)

This token has been replaced by the newgroups token, which provides the same type of information but requires less space. A description of the groups token is provided here for completeness, but the application designer should use the newgroups token. 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 groups token has two fixed fields: a token ID field that identifies this as a groups token and a count that represents the number of groups contained in this audit record. The remainder of the token is composed of zero or more group entries. Figure A-9 shows a groups token.

```
<table>
<thead>
<tr>
<th>token ID</th>
<th>groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>ngroups x 4 bytes</td>
</tr>
</tbody>
</table>
```

Figure A-9  groups Token Format

Note – The groups token is output only when the audit policy group is active. See “The auditconfig Command” on page 37 for more information.

header Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>byte count</th>
<th>version #</th>
<th>event ID</th>
<th>ID modifier</th>
<th>date and time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>1 byte</td>
<td>2 bytes</td>
<td>2 bytes</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

*Figure A-10 header Token Format*

The event modifier field has the following flags defined:

- 0x4000 PAD_NOTATTR nonattributable event
- 0x8000 PAD_FAILURE fail audit event

**in_addr Token**

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>Internet address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-11 in_addr Token Format*

**ip Token**

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>IP header</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>20 bytes</td>
</tr>
</tbody>
</table>

Figure A-12 ip Token Format

**ipc Token**

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>IPC object type</th>
<th>IPC object ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Figure A-13 ipc Token Format

**Note** – The IPC object identifiers violate the context-free nature of the Solaris
CMW audit tokens. No global “name” uniquely identifies IPC objects; instead,
they are identified by their handles, which are valid only during the time the
IPC objects are active. The identification should not be a problem since the
System V IPC mechanisms are seldom used and they all share the same audit
class.
The IPC object type field may have the values shown in Table A-4. The values are defined in /usr/include/bsm/audit.h.

Table A-4  IPC Object Type Field

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU_IPC_MSG</td>
<td>1</td>
<td>IPC message object</td>
</tr>
<tr>
<td>AU_IPC_SEM</td>
<td>2</td>
<td>IPC semaphore object</td>
</tr>
<tr>
<td>AU_IPC_SHM</td>
<td>3</td>
<td>IPC shared memory object</td>
</tr>
</tbody>
</table>

**ipc_perm Token**

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>owner uid</th>
<th>owner gid</th>
<th>creator uid</th>
<th>creator gid</th>
<th>ipc mode</th>
<th>sequence ID</th>
<th>IPC key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 byte</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-14*  `ipc_perm` Token Format
iport *Token*

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

\[
\begin{array}{|c|c|}
\hline
\text{token ID} & \text{port ID} \\
1 \text{ byte} & 2 \text{ bytes} \\
\hline
\end{array}
\]

*Figure A-15 iport Token Format*

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

\[
\begin{array}{|c|c|c|}
\hline
\text{token ID} & \text{count} & \text{groups} \\
1 \text{ byte} & 2 \text{ bytes} & \text{count} \times 4 \text{ bytes} \\
\hline
\end{array}
\]

*Figure A-16 newgroups Token Format*

**Note** – The newgroups token is output only when the audit policy group is active. See “The auditconfig Command” on page 37 for more information.
opaque Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>data length</th>
<th>data bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>2 bytes</td>
<td>n bytes</td>
</tr>
</tbody>
</table>

Figure A-17  opaque Token Format

path Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>object path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>path length</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>n bytes</td>
</tr>
</tbody>
</table>

Figure A-18  path Token Format
process Token

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>token ID</td>
<td>1 byte</td>
</tr>
<tr>
<td>audit ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>user ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>group ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>real user ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>real group ID</td>
<td>4 bytes</td>
</tr>
<tr>
<td>process ID</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

**Figure A-19** process Token Format

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

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

return Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>process error</th>
<th>process value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-20  return Token Format*

seq Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-21  seq Token Format*

socket Token

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

<table>
<thead>
<tr>
<th>Token ID</th>
<th>socket type</th>
<th>local port</th>
<th>local Internet address</th>
<th>remote port</th>
<th>remote Internet address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-22 socket Token Format*

socket-inet *Token*

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

<table>
<thead>
<tr>
<th>Token ID</th>
<th>socket family</th>
<th>local port</th>
<th>socket address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>2 bytes</td>
<td>2 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

*Figure A-23 socket-inet Token Format*

subject *Token*

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>audit ID</th>
<th>user ID</th>
<th>group ID</th>
<th>real user ID</th>
<th>real group ID</th>
<th>process ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Figure A-24 subject Token Format

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

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

text Token

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>text length</th>
<th>text string</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bytes</td>
<td>2 bytes</td>
<td>n bytes</td>
</tr>
</tbody>
</table>

Figure A-25 text Token Format
trailer Token

The two tokens, header and trailer, are special in that they distinguish the endpoints of an audit record and bracket all the other tokens. A header token begins an audit record. A trailer token ends an audit record. It is an optional token that is added as the last token of each record only when the AUDIT_TRAIL audit policy has been set.

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

<table>
<thead>
<tr>
<th>token ID</th>
<th>pad number</th>
<th>byte count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>2 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

Figure A-26 trailer Token Format

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

General Audit Record Structure

The audit records produced by Basic Security Module have a sequence of tokens. Certain tokens are optional within an audit record, according to the current audit policy. The `group`, `sequence`, and `trailer` tokens fall into this category. The administrator can determine if these are included in an audit record with the `auditconfig` command `-getpolicy` option.

“Event–to–System Call Translation” on page 147 includes two tables which include all possible audit events and indentifies which kernel or user event created the audit event. Table A-5 on page 147 maps audit events to system calls. Table A-6 on page 152 maps audit events to an application or command.

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

access

<table>
<thead>
<tr>
<th>system call</th>
<th>access</th>
<th>see access(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>14</td>
<td>AUE_ACCESS</td>
</tr>
<tr>
<td>event class</td>
<td>fa</td>
<td>0x00000004</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[attr-token]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
acct

system call acct see acct(2)
event ID 18 AUE_ACCT
event class ad 0x00000800
audit record
<path non-zero>
header-token
path-token
[attr-token]
subject-token
return-token
<path zero>
header-token
argument-token
(1,"accounting off", 0)
subject-token
return-token

adjtime

system call adjtime see adjtime(2)
event ID 50 AUE_ADJTIME
event class ad (0x00000800)
audit record
header-token
subject-token
return-token

audit

system call audit see audit(2)
event ID 211 AUE_AUDIT
event class no (0x00000000)
audit record
header-token
subject-token
return-token
auditon: A_GETCAR
system call auditon see auditon(2)
event ID 224 AUE_AUDITON_GETCAR (0x00000800)
event class ad
audit record
  header-token
  subject-token
  return-token

auditon: A_GETCLASS
system call auditon see auditon(2)
event ID 231 AUE_AUDITON_GETCLASS (0x00000800)
event class ad
audit record
  header-token
  subject-token
  return-token

auditon: A_GETCOND
system call auditon see auditon(2)
event ID 229 AUE_AUDITON_GETCOND (0x00000800)
event class ad
audit record
  header-token
  subject-token
  return-token

auditon: A_GETCWD
system call auditon see auditon(2)
event ID 223 AUE_AUDITON_GETCWD (0x00000800)
event class ad
audit record
  header-token
  subject-token
  return-token
auditon: A_GETKMASK

  system call    auditon    see auditon(2)
  event ID      221        AUE_AUDITON_GETKMASK
  event class   ad         (0x00000800)
  audit record  
     header-token
     subject-token
     return-token

auditon: A_GETSTAT

  system call    auditon    see auditon(2)
  event ID      225        AUE_AUDITON_GETSTAT
  event class   ad         (0x00000800)
  audit record  
     header-token
     subject-token
     return-token

auditon: A_GETPOLICY

  system call    auditon    see auditon(2)
  event ID      114        AUE_AUDITON_GPOLICY
  event class   ad         (0x00000800)
  audit record  
     header-token
     subject-token
     return-token

auditon: A_GETQCTRL

  system call    auditon    see auditon(2)
  event ID      145        AUE_AUDITON_GQCTRL
  event class   ad         (0x00000800)
  audit record  
     header-token
     subject-token
     return-token
auditon: A_SETCLASS
system call auditon see auditon(2)
event ID 232 AUE_AUDITON_SETCLASS (0x00000800)
event class ad
audit record
  header-token
  [argument-token] (2,"setclass:ec_event",event number)
  [argument-token] (3,"setclass:ec_class",class mask)
subject-token
return-token

auditon: A_SETCOND
system call auditon see auditon(2)
event ID 230 AUE_AUDITON_SETCOND (0x00000800)
event class ad
audit record
  header-token
  [argument-token] (3,"setcond",audit state)
subject-token
return-token

auditon: A_SETKMASK
system call auditon see auditon(2)
event ID 222 AUE_AUDITON_SETKMASK (0x00000800)
event class ad
audit-record
  header-token
  [argument-token] (2,"setkmask:as_failure",kernel mask)
subject-token
return-token

auditon: A_SETSMASK
system call auditon see auditon(2)
event ID 228 AUE_AUDITON_SETSMASK (0x00000800)
event class ad
audit record
  header-token
### auditon: A_SETSTAT

- **System Call:** `auditon`
- **Event ID:** 226
- **Event Class:** `ad`
- **Audit Record:**
  - **Header Token:**
  - **Subject Token:**
  - **Return Token:**

### auditon: A_SETUMASK

- **System Call:** `auditon`
- **Event ID:** 227
- **Event Class:** `ad`
- **Audit Record:**
  - **Header Token:**
  - **Argument Tokens:**
    - (3,"setumask:as_success",audit ID mask)
    - (3,"setumask:as_failure",audit ID mask)
  - **Subject Token:**
  - **Return Token:**

### auditon: A_SETPOLICY

- **System Call:** `auditon`
- **Event ID:** 142
- **Event Class:** `ad`
- **Audit Record:**
  - **Header Token:**
  - **Argument Tokens:**
    - (1,"policy",audit policy flags)
  - **Subject Token:**
  - **Return Token:**
auditon: A_SETQCTRL

- **System Call:** `auditon`
- **Event ID:** 146 (AUE_AUDITON_SQCTRL)
- **Event Class:** ad (0x00000800)

**Audit Record**

- **Header-Token**
- **Argument-Tokens**:
  - `(3,"setqctrl:aq_hiwater",queue control param.)`
  - `(3,"setqctrl:aq_lowater",queue control param.)`
  - `(3,"setqctrl:aq_bufsz",queue control param.)`
  - `(3,"setqctrl:aq_delay",queue control param.)`

- **Subject-Token**
- **Return-Token**

auditsvc

- **System Call:** `auditsvc`
- **Event ID:** 136 (AUE_AUDITSVC)
- **Event Class:** ad (0x00000800)

**Audit Record**

- **Header-Token**
- **Path-Token**
- **Attr-Token**
- **Subject-Token**
- **Return-Token**

- **<Valid File Descriptor>**
- **<Invalid File Descriptor>**
  - **Header-Token**
  - **Argument-Token** `(1, "no path: fd",fd)`
  - **Subject-Token**
  - **Return-Token**

chdir

- **System Call:** `chdir`
- **Event ID:** 8 (AUE_CHDIR)
- **Event Class:** pc (0x00000800)

**Audit Record**

- **Header-Token**
- **Path-Token**
chmod

<table>
<thead>
<tr>
<th>system call</th>
<th>chmod</th>
<th>see chmod(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>10</td>
<td>AUE_CHMOD</td>
</tr>
<tr>
<td>event class</td>
<td>fm</td>
<td>0x00000008</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[attr-token]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

chown

<table>
<thead>
<tr>
<th>system call</th>
<th>chown</th>
<th>see chown(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>11</td>
<td>AUE_CHOWN</td>
</tr>
<tr>
<td>event class</td>
<td>fm</td>
<td>0x00000008</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[attr-token]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

chroot

<table>
<thead>
<tr>
<th>system call</th>
<th>chroot</th>
<th>see chroot(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>24</td>
<td>AUE_CHROOT</td>
</tr>
<tr>
<td>event class</td>
<td>pc</td>
<td>0x00000080</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Audit Record Descriptions

AUE_CLOSE

```
[attr-token]
subject-token
return-token
```

close

- System call: `close`
- Event ID: 112
- Event class: `cl`
- Audit record:
  - `<file system object>`
  - `header-token`
  - `argument-token`²
  - `[path-token]³
  - `[attr-token]`
  - `subject-token`
  - `return-token`

AUE_CREAT

```
creat
```

- System call: `creat`
- Event ID: 4
- Event class: `fc`
- Audit record:
  - `header-token`
  - `path-token`
  - `[attr-token]`
  - `subject-token`
  - `return-token`

AUE_ENTERPROM

```
enter prom
```

- System call: `---`
- Event ID: 153
- Event class: `na`

---

1. Also for files closed on process termination.
2. Only present with `close()` system call.
3. This token may be removed in future releases.
4. Only with valid file descriptors.
audit record
  header-token
  text-token
  subject-token
  return-token

exec
  system call exec see exec(2)
  event ID 7 AUE_EXEC
  event class pc,ex 0x40000080
  audit record
    header-token
    path-token
    [attr-token]
    subject-token
    return-token

execve
  system call execve see exec(2)
  event ID 23 AUE_EXECVE
  event class pc,ex 0x40000080
  audit record
    header-token
    path-token
    [attr-token]
    subject-token
    return-token

exit prom
  system call ---
  event ID 154 AUE_EXITPROM
  event class na (0x00000400)
  audit record
    header-token
    text-token
    subject-token
    return-token

  (addr,"monitor PROM" | "kadb")
exit

system call       exit
                 see exit(2)
event ID1         1
                 AUE_EXIT
event class1      pc
                 0x00000080
audit record
                 header-token
                 subject-token
                 return-token

fchdir

system call      fchdir
                 see chdir(2)
event ID          68
                 AUE_FCHDIR
event class       pc
                 0x00000080
audit record
                 header-token
                 [path-token]
                 [attr-token]
                 subject-token
                 return-token

fchmod

system call      fchmod
                 see chmod(2)
event ID          39
                 AUE_FCHMOD
event class       fm
                 0x00000008
audit record
                 <valid file descriptor>
                 header-token
                 argument-token
                 (2,"new file mode", mode)
                 [path-token]
                 [attr-token]
                 subject-token
                 return-token

                 <invalid file descriptor>
                 header-token
                 argument-token
                 (2,"new file mode", mode)
                 argument-token
                 (1,"no path: fd",fd)
                 subject-token
                 return-token
fchown

system call: fchown
see chown(2)

event ID: 38
AUE_FCHOWN

event class: fm
0x00000008

audit record

<valid file descriptor>

header-token

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

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

(path-token)

(attr-token)

subject-token

return-token

<non-file descriptor>

header-token

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

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

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

(subject-token)

return-token

fchroot

system call: fchroot
see chroot(2)

event ID: 69
AUE_FCHROOT

event class: pc
0x00000080

audit record

(header-token)

(path-token)

(attr-token)

subject-token

return-token

fcntl

system call: fcntl
see fcntl(2)

event ID: 30
AUE_FCNTL (cmd=F_GETLK, F_SETLK, F_SETLK)

event class: fm
0x00000008

audit record

<bad file descriptor>

(header-token)
1. Note that the `fork()` return values are undefined since the audit record is produced at the point that the child process is spawned.

2. Note that the `fork1()` return values are undefined since the audit record is produced at the point that the child process is spawned.
fstatfs

system call  fstatfs  4.x call, see statvfs(2)
event ID  55  AUE_FSTATFS
event class  fa  (0x00000004)
audit record
  <file descriptor>
    header-token
    [path-token]
    [attr-token]
    subject-token
    return-token
  <non-file descriptor>
    header-token
    argument-token (1, "no path: fd", fd)
    subject-token
    return-token

getaudit

system call  getaudit  see getaudit(2)
event ID  132  AUE_GETAUID
  event class  ad  (0x00000800)
audit record
    header-token
    subject-token
    return-token

getauid

system call  getauid  see getauid(2)
event ID  130  AUE_GETAUID
  event class  ad  (0x00000800)
audit record
    header-token
    subject-token
    return-token
Audit Record Descriptions

getmsg

system call getmsg see getmsg(2)
event ID 217 AUE_GETMSG
event class nt (0x00000100)
audit record
    header-token
    argument-token (1,"fd",file descriptor)
    argument-token (4,"pri",priority)
    subject-token
    return-token

getmsg: socket accept

system call getmsg see getmsg(2)
event ID 247 AUE_SOCKACCEPT
event class nt (0x00000100)
audit record
    header-token
    socket-inet-token
    argument-token (1,"fd",file descriptor)
    argument-token (4,"pri",priority)
    subject-token
    return-token

getmsg: socket receive

system call getmsg see getmsg(2)
event ID 250 AUE_SOCKRECEIVE
event class nt (0x00000100)
audit record
    header-token
    socket-inet-token
    argument-token (1,"fd",file descriptor)
    argument-token (4,"pri",priority)
    subject-token
    return-token
getpmsg
system call     getpmsg     see getpidmsg(2)
  event ID      219        AUE_GETPMSG
  event class   nt         (0x00000100)
  audit record  
    header-token
    argument-token
    subject-token
    return-token

getportaudit
system call     getportaudit
  event ID      149        AUE_GETPORTAUDIT
  event class   ad         (0x00000800)
  audit record  
    header-token
    argument-token
    subject-token
    return-token

ioctl
system call     ioctl
  event ID      158        AUE_IOCTL
  event class   io         (0x20000000)
  audit record  
    <good file descriptor>
      header-token
      path-token
      [attr-token]
      argument-token
      argument-token
      subject-token
      return-token
    <socket>
      header-token
      [socket-token]
      argument-token
      argument-token
      subject-token
      return-token
<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
<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

kill
system call kill
see kill(2)
event ID 15 AUE_KILL
event class pc (0x00000080)
audit record
<valid process>
header-token
argument-token (2,"signal",signo)
[process-token]
subject-token
return-token
<zero or negative process>
header-token
argument-token (2,"signal",signo)
argument-token (1,"process",pid))
subject-token
return-token

lchown
system call lchown
see chown (2)
event ID 237 AUE_LCHOWN
event class fm 0x00000008
audit record
header-token
argument-token (2, "new file uid", uid)
argument-token (3, "new file gid", gid)
path-token
[attr-token]
subject-token
return-token

link
system call link see link(2)
event ID 5 AUE_LINK (0x00000010)
event class fc
audit record
header-token
path-token (from path)
[attr-token] (from path)
path-token (to path)
subject-token
return-token

lstat
system call lstat see stat(2)
event ID 17 AUE_LSTAT (0x00000004)
event class fa
audit record
header-token
path-token
[attr-token]
subject-token
return-token

lxstat
system call lxstat AUE_LXSTAT (0x00000004)
event ID 236
event class fa
audit record
header-token
path-token
memcntl

system call memcntl
event ID 238
event class ot
audit record
  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)

mkdir

system call mkdir
event ID 47
event class fc
audit record
  header-token
  argument-token (2,"mode",mode)

mknod

system call mknod
event ID 9
event class fc
audit record
  header-token
  argument-token (2,"mode",mode)
argument-token (3,"dev",dev)
path-token
[attr-token]
subject-token
return-token

mmap
system call mmap
event ID 210
event class no
audit record
<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
<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

modctl: MODADDMAJBIND
system call modctl
event ID 246
event class ad
audit record
  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)
Audit Record Descriptions

modctl: MODCONFIG
system call modctl
event ID 245
event class ad
audit record
header-token

modctl: MODLOAD
system call modctl
event ID 243
event class ad
audit record
header-token

modctl: MODUNLOAD
system call modctl
event ID 244
event class ad
audit record
header-token
mount

system call     mount     see mount (2)
event ID        62        AUE_MOUNT
event class     ad        (0x00000800)

audit record
<unix filesystem>
  header-token
  argument-token (3,"flags",flags)
  text-token     (filesystem type)
  path-token
  [attr-token]
  subject-token
  return-token

<nfs filesystem>
  header-token
  argument-token (3,"flags",flags)
  text-token     (filesystem type)
  text-token     (host name)

msgctl: IPC_RMID

system call     msgctl     see msgctl (2)
event ID        85        AUE_MSGCTL_RMID
event class     ip        (0x00000200)

audit record
  header-token
  argument-token (1,"msg ID",message ID)
  [ipc-token]\1
  subject-token
  return-token

msgctl: IPC_SET

system call     msgctl     see msgctl (2)
event ID        86        AUE_MSGCTL_SET
event class     ip        (0x00000200)

audit record

1. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
Audit Record Descriptions

header-token
argument-token (1,"msg ID",message ID)
[ipc-token]
subject-token
return-token

msgctl: IPC_STAT

system call msgctl see msgctl(2)
event ID 87 AUE_MSGCTL_STAT (0x00000200)
event class ip
audit record

header-token
argument-token (1,"msg ID",message ID)
[ipc-token]
subject-token
return-token

msgget

system call msgget see msgget(2)
event ID 88 AUE_MSGGET (0x00000200)
event class ip
audit record

header-token
[ipc-token]
subject-token
return-token

msgrcv

system call msgrcv see msgop(2)
event ID 89 AUE_MSGRCV (0x00000200)
event class ip
audit record

header-token

1. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
2. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
3. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
argument-token
{ipc-token}¹
subject-token
return-token

msgsnd

system call    msgsnd
event ID    90
event class    ip
audit record
header-token
argument-token
{ipc-token}²
subject-token
return-token

munmap

system call    munmap
event    214
class    cl
audit record
header-token
argument-token
argument-token
subject-token
return-token

nice

system call    nice
event ID    203
event class    pc
audit record

1. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
2. The ipc and ipc_perm tokens are not included if the msg ID is invalid.
open: read

system call open
event ID 72
event class fr
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

open: read, create

system call open
event ID 73
event class fc,fr
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

open: read, create, truncate

system call open
event ID 75
event class fc,fd,fr
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token
open: read, truncate

system call open see open(2)
event ID 74 AUE_OPEN_RT
event class fd, fr (0x00000021)
audit record
  header-token
  path-token [attr-token]
  subject-token
  return-token

open: read, write

system call open see open(2)
event ID 80 AUE_OPEN_RW (0x00000003)
event class fr, fw

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

open: read, write, create

system call open see open(2)
event ID 81 AUE_OPEN_RWC (0x00000013)
event class fr, fw, fc

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

open: read, write, create, truncate

system call open see open(2)
event ID 83 AUE_OPEN_RWTC (0x00000033)
event class fr, fw, fc, fd
### open: read, write, truncate

<table>
<thead>
<tr>
<th>System Call</th>
<th>open</th>
<th>See: open(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
<td>82</td>
<td>AUE_OPEN_RWT</td>
</tr>
<tr>
<td>Event Class</td>
<td>fr, fw, fd (0x00000023)</td>
<td></td>
</tr>
</tbody>
</table>

#### Audit Record
- header-token
- path-token
- [attr-token]
- subject-token
- return-token

### open: write

<table>
<thead>
<tr>
<th>System Call</th>
<th>open</th>
<th>See: open(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
<td>76</td>
<td>AUE_OPEN_W</td>
</tr>
<tr>
<td>Event Class</td>
<td>fw</td>
<td>(0x00000002)</td>
</tr>
</tbody>
</table>

#### Audit Record
- header-token
- path-token
- [attr-token]
- subject-token
- return-token

### open: write, create

<table>
<thead>
<tr>
<th>System Call</th>
<th>open</th>
<th>See: open(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
<td>77</td>
<td>AUE_OPEN_WC</td>
</tr>
<tr>
<td>Event Class</td>
<td>fw, fc</td>
<td>(0x00000012)</td>
</tr>
</tbody>
</table>

#### Audit Record
- header-token
- path-token
- [attr-token]
- subject-token
- return-token
open: write, create, truncate

system call  open  see open(2)
event ID     79     AUE_OPEN_WTC
event class  fw, fc, fd (0x00000032)
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

open: write, truncate

system call  open  see open(2)
event ID     78     AUE_OPEN_WT
event class  fw, fd (0x00000022)
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

pathconf

system call  pathconf  see fpathconf(2)
event ID     71     AUE_PATHCONF
event class  fa (0x00000004)
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

pipe

system call  pipe  see pipe(2)
event ID     185     AUE_PIPE
event class  no (0x00000000)
audit record
header-token
subject-token
return-token

priocntl

system call      priocntl      see priocntl(2)
event ID         212          AUEPRIOCNTLSYS
event class      pc           (0x00000080)
audit record
     header-token
     argument-token   (1,"pc_version",priocntl version num.)
     argument-token   (3,"cmd",command)
     subject-token
     return-token

process dumped core

system call      ---
event ID         111          AUECORE
event class      fc           0x00000010
audit record
     header-token
     path-token
     [attr-token]
     argument-token   (1,"signal",signal)
     subject-token
     return-token

putmsg

system call      putmsg      see putmsg(2)
event ID         216          AUE_PUTMSG
event class      nt           (0x000000100)
audit record
     header-token
     argument-token   (1,"fd",file descriptor)
     argument-token   (4,"pri",priority)
     subject-token
     return-token
putmsg: socket connect

system call putmsg see putmsg(2)
event ID 248 AUE_SOCKCONNECT
event class nt (0x00000100)
audit record
  header-token
  socket-inet-token
  argument-token (1,"fd", file descriptor)
  argument-token (4,"pri", priority)
  subject-token
  return-token

putmsg: socket send

system call putmsg see putmsg(2)
event ID 249 AUE_SOCKSEND
event class nt (0x00000100)
audit record
  header-token
  socket-inet-token
  argument-token (1,"fd", file descriptor)
  argument-token (4,"pri", priority)
  subject-token
  return-token

putpmsg

system call putpmsg see putmsg(2)
event ID 218 AUE_PUTPMSG
event class nt (0x00000100)
audit record
  header-token
  argument-token (1,"fd", file descriptor)
  subject-token
  return-token

readlink

system call readlink see readlink(2)
event ID 22 AUE_READLINK
event class fr (0x00000001)
rename

system call  rename  see rename(2)
event ID     42      AUE_RENAME
event class  fc,fd  (0x00000030)
audit record
  header-token
  path-token  (from name)
  [attr-token]  (from name)
  [path-token]  (to name)
  subject-token
  return-token

rmdir

system call  rmdir  see rmdir(2)
event ID     48      AUE_RMDIR
event class  fd      (0x00000020)
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

semctl: GETALL

system call  semctl  see semctl(2)
event ID     105     AUE_SEMCTL_GETALL
event class  ip      (0x00000020)
audit record
  header-token
  argument-token  (1,"sem ID", semaphore ID)
semctl: GETNCNT

system call  semctl  see semctl(2)
event ID     102 AUE_SEMCTL_GETNCNT
event class  ip  (0x00000200)
audit record
header-token
argument-token
[ipc-token]
subject-token
return-token

semctl: GETPID

system call  semctl  see semctl(2)
event ID     103 AUE_SEMCTL_GETPID
event class  ip  (0x00000200)
audit record
argument-token
[ipc-token]
subject-token
return-token

semctl: GETVAL

system call  semctl  see semctl(2)
event ID     104 AUE_SEMCTL_GETVAL
event class  ip  (0x00000200)
audit record
header-token
argument-token

1. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
2. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
3. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
### semctl: GETZCNT

<table>
<thead>
<tr>
<th>System Call</th>
<th>semctl</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>106</td>
<td>ip</td>
<td>(0x00000200)</td>
</tr>
</tbody>
</table>

**Header Token**: 1

**Argument Token**: (1,”sem ID”, semaphore ID)

**Return Token**: 

---

### semctl: IPC_RMID

<table>
<thead>
<tr>
<th>System Call</th>
<th>semctl</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>99</td>
<td>ip</td>
<td>(x0000200)</td>
</tr>
</tbody>
</table>

**Header Token**: 1

**Argument Token**: (1,”sem ID”, semaphore ID)

**Return Token**: 

---

### semctl: IPC_SET

<table>
<thead>
<tr>
<th>System Call</th>
<th>semctl</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>ip</td>
<td>(0x00000200)</td>
</tr>
</tbody>
</table>

**Header Token**: 1

**Argument Token**: (1,”sem ID”, semaphore ID)

---

1. The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is invalid.
2. The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is invalid.
3. The *ipc* and *ipc_perm* tokens are not included if the semaphore ID is invalid.
semctl: SETALL

system call  semctl  see semctl(2)
  event ID  108  AUE_SEMCTL_SETALL
  event class ip  (0x00000200)
  audit record
    header-token
    argument-token
    [ipc-token]
    subject-token
    return-token

semctl: SETVAL

system call  semctl  see semctl(2)
  event ID  107  AUE_SEMCTL_SETVAL
  event class ip  (0x00000200)
  audit record
    header-token
    argument-token
    [ipc-token]
    subject-token
    return-token

semctl: IPC_STAT

system call  semctl  see semctl(2)
  event ID  101  AUE_SEMCTL_STAT
  event class ip  (0x00000200)
  audit record
    header-token
    argument-token
    (1,"sem ID", semaphore ID)

1. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
2. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
3. The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
[ipc-token]
subject-token
return-token

semget
system call  semget
event ID     109
event class  ip
audit record
  header-token
  [ipc-token]¹
  subject-token
  return-token

semop
system call  semop
event ID     110
event class  ip
audit record
  header-token
  argument-token
  [ipc-token]²
  subject-token
  return-token

setaudit
system call  setaudit
event ID     133
event class  ad
audit record
  <valid program stack address>
    header-token
    argument-token  (1,"setaudit:auid",audit user ID)
    argument-token  (1,"setaudit:port",terminal ID)

¹. The ipc and ipc_perm tokens are not included if the system call failed.
². The ipc and ipc_perm tokens are not included if the semaphore ID is invalid.
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
<invalid program stack address>

header-token
subject-token
return-token

setauid

system call setauid see getauid(2)
event ID 131 AUE_SETAUID
event class ad (0x00000800)
audit record

header-token
argument-token
subject-token
return-token

setegid

system call setegid see setuid(2)
event ID 214 AUE_SETEGID
event class pc (0x00000080)
audit record

header-token
argument-token
subject-token
return-token

seteuid

system call seteuid see setuid(2)
event ID 215 AUE_SETEUID
event class pc (0x00000080)
audit record

header-token
argument-token (1,"gid",user ID)
subject-token
return-token

setgid
system call setgid see setuid(2)
event ID 205 AUE_SETGID
event class pc (0x00000080)
audit record
  header-token
  argument-token (1,"gid",group ID)
  subject-token
  return-token

setgroups
system call setgroups see getgroups(2)
event ID 26 AUE_SETGROUPS
event class pc (0x00000080)
audit record
  header-token
  [argument-token] (1,"setgroups",group ID)\textsuperscript{1}
  subject-token
  return-token

setpgrp
system call setpgrp see setpgrp(2)
event ID 27 AUE_SETPGRP
event class pc (0x00000080)
audit record
  header-token
  subject-token
  return-token

\textsuperscript{1} One token for each group set.
setrlimit
system call: setrlimit
  see getrlimit(2)
  event ID 51
  event class ad
  audit record
    header-token
    subject-token
    return-token

setuid
system call: setuid
  see setuid(2)
  event ID 200
  event class pc
  audit record
    header-token
    argument-token
    subject-token
    return-token

shmat
system call: shmat
  see shmop(2)
  event ID 96
  event class ip
  audit record
    header-token
    argument-token
    argument-token
    [ipc-token]^2
    [ipc_perm-token]
    subject-token
    return-token

1. Due to a current bug in the audit software, this token is reported as AUE_OSETUID
2. The ipc and ipc_perm tokens are not included if the shared memory segment ID is invalid.
shmctl: IPC_RMID

<table>
<thead>
<tr>
<th>system call</th>
<th>shmctl</th>
<th>see shmctl(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>92</td>
<td>AUE_SHMCTL_RMID</td>
</tr>
<tr>
<td>event class</td>
<td>ip</td>
<td>(0x00000200)</td>
</tr>
<tr>
<td>audit record</td>
<td>header-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>argument-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ipc-token]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>subject-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return-lbtoken</td>
<td></td>
</tr>
</tbody>
</table>

shmctl: IPC_SET

<table>
<thead>
<tr>
<th>system call</th>
<th>shmctl</th>
<th>see shmctl(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>93</td>
<td>AUE_SHMCTL_SET</td>
</tr>
<tr>
<td>event class</td>
<td>ip</td>
<td>(0x00000200)</td>
</tr>
<tr>
<td>audit record</td>
<td>header-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>argument-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ipc-token]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ipc_perm-token]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>subject-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return-token</td>
<td></td>
</tr>
</tbody>
</table>

shmctl: IPC_STAT

<table>
<thead>
<tr>
<th>system call</th>
<th>shmctl</th>
<th>see shmctl(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>94</td>
<td>AUE_SHMCTL_STAT</td>
</tr>
<tr>
<td>event class</td>
<td>ip</td>
<td>(0x00000200)</td>
</tr>
<tr>
<td>audit record</td>
<td>header-token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>argument-token</td>
<td></td>
</tr>
</tbody>
</table>

1. The ipc and ipc_perm tokens are not included if the shared memory segment ID is invalid.
2. The ipc and ipc_perm tokens are not included if the shared memory segment ID is invalid.
shmdt

system call shmdt  see shmop(2)
event ID  97             AUE_SHMDT
event class ip  (0x00000200)
audit record
  header-token
  argument-token
  subject-token
  return-token

shmget

system call shmget  see shmget(2)
event ID  95             AUE_SHMGET
event class ip  (0x00000200)
audit record
  header-token
  argument-token
  [ipc-token]1
  [ipc_perm-token]2
  subject-token
  return-token

stat

system call stat  see stat(2)
event ID  16             AUE_STAT
event class fa  (0x00000004)
audit record
  header-token
  path-token

1. The ipc and ipc_perm tokens are not included if the shared memory segment ID is invalid.
2. The ipc and ipc_perm tokens are not included for failed events.
statfs

system call statfs
event ID 54
event class fa
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

4.x call, see statvfs(2)
AUE_STATFS
(0x00000004)

statvfs

system call statvfs
event ID 234
event class fa
audit record
  header-token
  path-token
  [attr-token]
  subject-token
  return-token

see statvfs(2)
AUE_STATVFS
(0x00000004)

stime

system call stime
event ID 201
event class ad
audit record
  header-token
  subject-token
  return-token

see stime(2)
AUE_STIME
(0x00000800)
### symlink

<table>
<thead>
<tr>
<th>system call</th>
<th>syscall</th>
<th>see symlink(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>21</td>
<td>AUE_SYMLINK</td>
</tr>
<tr>
<td>event class</td>
<td>fc</td>
<td>(0x00000010)</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td>(symbolic link string)</td>
</tr>
<tr>
<td>text-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[attr-token]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### sysinfo

<table>
<thead>
<tr>
<th>system call</th>
<th>syscall</th>
<th>see sysinfo(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>239</td>
<td>AUE_SYSINFO</td>
</tr>
<tr>
<td>event class</td>
<td>ad</td>
<td>(0x00000800)</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argument-token</td>
<td></td>
<td>(1,&quot;cmd&quot;,command)</td>
</tr>
<tr>
<td>text-token</td>
<td></td>
<td>(name)</td>
</tr>
<tr>
<td>subject-token</td>
<td></td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### system booted

<table>
<thead>
<tr>
<th>system call</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>113</td>
</tr>
<tr>
<td>event class</td>
<td>na</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
</tr>
<tr>
<td>text-token</td>
<td></td>
</tr>
<tr>
<td>return-token</td>
<td></td>
</tr>
</tbody>
</table>

### umount: old version

<table>
<thead>
<tr>
<th>system call</th>
<th>umount</th>
<th>see umount(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>12</td>
<td>AUE_UMOUNT</td>
</tr>
<tr>
<td>event class</td>
<td>ad</td>
<td>(0x00000800)</td>
</tr>
<tr>
<td>audit record</td>
<td></td>
<td></td>
</tr>
<tr>
<td>header-token</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
path-token
[attr-token]
subject-token
return-token

unlink

- system call: **unlink**
- event ID: 6
- event class: **fd**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token

- see unlink(2)
- event ID: 6
- event class: **fd**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token

utime

- system call: **utime**
- event ID: 202
- event class: **fm**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token

- see utime(2)
- event ID: 202
- event class: **fm**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token

utimes

- system call: **utimes**
- event ID: 49
- event class: **fm**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token

- see utimes(2)
- event ID: 49
- event class: **fm**
- audit record
  - header-token
  - path-token
  [attr-token]
  - subject-token
  - return-token
### utssys - fusers

<table>
<thead>
<tr>
<th>System Call</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>utssys</td>
<td>233</td>
<td>ad</td>
<td>AUE_UTSSYS</td>
<td>233</td>
<td>ad</td>
<td>0x00000800</td>
</tr>
</tbody>
</table>

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

### vfork

<table>
<thead>
<tr>
<th>System Call</th>
<th>Event ID</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>vfork</td>
<td>25</td>
<td>pc</td>
<td>AUE_VFORK</td>
<td>25</td>
<td>pc</td>
<td>0x00000800</td>
<td></td>
</tr>
</tbody>
</table>

#### header-token
#### argument-token (0,”child PID”,pid)
#### subject-token
#### return-token

### vtrace

<table>
<thead>
<tr>
<th>System Call</th>
<th>Event ID</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>vtrace</td>
<td>36</td>
<td>pc</td>
<td>AUE_VTRACE</td>
<td>36</td>
<td>pc</td>
<td>0x00000800</td>
<td></td>
</tr>
</tbody>
</table>

#### header-token
#### subject-token
#### return-token

### xmknod

<table>
<thead>
<tr>
<th>System Call</th>
<th>Event ID</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmknod</td>
<td>240</td>
<td>fc</td>
<td>AUE_XMKNOD</td>
<td>240</td>
<td>fc</td>
<td>0x0000010</td>
<td></td>
</tr>
</tbody>
</table>

---

1. Note that the fork return values are undefined since the audit record is produced at the point that the child process is spawned.
User-Level Generated Audit Records

These audit records are created by applications that operate outside the kernel. The records are sorted alphabetically by program. The description of each record includes

- The name of the program
- A man page reference (if appropriate)
- The audit event number
- The audit event name
- The audit record structure

allocate: device allocate

<table>
<thead>
<tr>
<th>program</th>
<th>/usr/sbin/allocate</th>
<th>see allocate(1M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>6200</td>
<td>AUE_allocate_succ</td>
</tr>
<tr>
<td>event class</td>
<td>ad</td>
<td>(0x00000800)</td>
</tr>
</tbody>
</table>

audit record

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

xstat

system call xstat
event ID 235 AUE_XSTAT
event class fa (0x00000004)

audit record

header-token
path-token
[attr-token]
subject-token
return-token
allocate: device allocate failure

program    /usr/sbin/allocate  see allocate(1M)
event ID    6201               AUE_allocate_fail
event class ad                   (0x00000800)
audit record
  header-token
  subject-token
  newgroups-token
  exit-token

allocate: deallocate device

program    /usr/sbin/allocate  see allocate(1M)
event ID    6202               AUE_deallocate_succ
event class ad                   (0x00000800)
audit record
  header-token
  subject-token
  newgroups-token
  exit-token

allocate: deallocate device failure

program    /usr/sbin/allocate  see allocate(1M)
event ID    6203               AUE_deallocate_fail
event class ad                   (0x00000800)
audit record
  header-token
  subject-token
  newgroups-token
  exit-token

allocate: list device

program    /usr/sbin/allocate  see allocate(1M)
event ID    6205               AUE_listdevice_succ
event class ad                   (0x00000800)
audit record
  header-token
<table>
<thead>
<tr>
<th>Function</th>
<th>Program</th>
<th>See</th>
<th>Event ID</th>
<th>Event Class</th>
<th>Audit Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocate: list device failure</td>
<td>/usr/sbin/allocate</td>
<td>allocate(1M)</td>
<td>6206</td>
<td>ad</td>
<td>AUE_listdevice_fail (0x00000800)</td>
</tr>
<tr>
<td>at: create crontab</td>
<td>/usr/bin/at</td>
<td>at(1)</td>
<td>6144</td>
<td>ad</td>
<td>AUE_at_create (0x00000800)</td>
</tr>
<tr>
<td>at: delete atjob</td>
<td>/usr/bin/at</td>
<td>at(1)</td>
<td>6145</td>
<td>ad</td>
<td>AUE_at_delete (0x00000800)</td>
</tr>
<tr>
<td>at: at-permission</td>
<td>/usr/bin/at</td>
<td>at(1)</td>
<td>6146</td>
<td>ad</td>
<td>AUE_at_perm (0x00000800)</td>
</tr>
</tbody>
</table>
audit record
  header-token
  subject-token
  [group-token]
  exit-token

crontab: crontab created
program /usr/bin/crontab see crontab(1)
  event ID 6148 AUE_crontab_create
  event class ad (0x00000800)
  audit record
  header-token
  subject-token
  [group-token]
  exit-token

crontab: crontab deleted
program /usr/bin/crontab see crontab(1)
  event ID 6149 AUE_crontab_delete
  event class ad (0x00000800)
  audit record
  header-token
  subject-token
  [group-token]
  exit-token

crontab: cron-invoke atjob or crontab
program /usr/bin/crontab see crontab(1)
  event ID 6147 AUE_cron_invoke
  event class ad (0x00000800)
  audit record
  header-token
  subject-token
  text-token (program)
  text-token (shell)
  text-token (cmd)
  exit-token
crontab: crontab-permission

program /usr/bin/crontab see crontab(1)
event ID 6150 AUE_crontab_perm
event class ad (0x00000800)
audit record
  header-token
  subject-token
  [group-token]
  exit-token

halt

program /usr/sbin/halt see halt(1M)
event ID 6160 AUE_halt_solaris
event class ad (0x00000800)
audit record
  header-token
  subject-token
  return-token

inetd

program /usr/sbin/inetd see inetd(1M)
event ID 6151 AUE_inetd_connect
event class na (0x00000400)
audit record
  header-token
  subject-token
  text-token
  return-token

in.ftpd

program /usr/sbin/in.ftpd see in.ftpd(1M)
event ID 6165 AUE_ftpd
event class 10 (0x00001000)
audit record
  header-token
  subject-token
  text-token
  return-token

(error message, failure only)
login: terminal login
program /usr/bin/login see login(1)
event ID 6152 AUE_login
event class lo (0x00001000)
audit record
  header-token
  subject-token
  text-token
  return-token
(error message)

login: rlogin
program /usr/bin/login see login(1)
event ID 6155 AUE_rlogin
event class lo (0x00001000)
audit record
  header-token
  subject-token
  text-token
  return-token
(error message)

login: telnet login
program /usr/bin/login see login(1)
event ID 6154 AUE_telnet
event class lo (0x00001000)
audit record
  header-token
  subject-token
  text-token
  return-token
(error message)

login: logout
program /usr/bin/login see login(1)
event ID 6153 AUE_logout
event class lo (0x00001000)
audit record
  header-token
  subject-token
  return-token
mountd: NFS mount

<table>
<thead>
<tr>
<th>program</th>
<th>/usr/lib/nfs/mountd</th>
<th>see mountd(1M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>6156</td>
<td>AUE_mountd_mount</td>
</tr>
<tr>
<td>event class</td>
<td>na</td>
<td>(0x00000400)</td>
</tr>
</tbody>
</table>

audit record

header-token
subject-token
text-token
path-token
text-token
return-token

mountd: NFS unmount request

<table>
<thead>
<tr>
<th>program</th>
<th>/usr/lib/nfs/mountd</th>
<th>see mountd(1M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>6157</td>
<td>AUE_mountd_umount</td>
</tr>
<tr>
<td>event class</td>
<td>na</td>
<td>(0x00000400)</td>
</tr>
</tbody>
</table>

audit record

header-token
subject-token
text-token
path-token
text-token
return-token

passwd

<table>
<thead>
<tr>
<th>program</th>
<th>/usr/bin/passwd</th>
<th>see passwd(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event ID</td>
<td>6163</td>
<td>AUE_passwd</td>
</tr>
<tr>
<td>event class</td>
<td>lo</td>
<td>(0x00001000)</td>
</tr>
</tbody>
</table>

audit record

header-token
subject-token
text-token
return-token

(repeat with the same format for reboot)

(repeat with the same format for reboot)

Audit Record Descriptions
audit record
  header-token
  subject-token
  return-token

rpc.rexd

  program /usr/sbin/rpc.rexd  see rpc.rexd(1M)
  event ID  6164  AUE_rexd
  event class lo  (0x00001000)

  header-token
  subject-token
  text-token
  text-token
  text-token
  text token
  exit-token

  (error message, failure only)
  (hostname)
  (username)
  (command to be executed)

in.rexed

  program /usr/sbin/in.rexed  see in.rexed(1M)
  event ID  6162  AUE_rexed
  event class lo  (0x00001000)

  header-token
  subject-token
  text-token
  text-token
  text-token
  text token
  exit-token

  (error message, failure only)
  (hostname)
  (username)
  (command to be executed)

in.rshd

  program /usr/sbin/in.rshd  see in.rshd(1M)
  event ID  6158  AUE_rshd
  event class lo  (0x00001000)

  header-token
  subject-token
Audit Record Descriptions

Event–to–System Call Translation

Table A-5 on page 147 associates an audit event name with the system call or kernel event that created it. Table A-6 on page 152 associates an audit event with the application or command that generated it.

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>System Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_ACCESS</td>
<td>“access” on page 95</td>
</tr>
<tr>
<td>AUE_ACCT</td>
<td>“acct” on page 96</td>
</tr>
<tr>
<td>AUE_ADJTIME</td>
<td>“adjtime” on page 96</td>
</tr>
<tr>
<td>AUE_AUDIT</td>
<td>“audit” on page 96</td>
</tr>
<tr>
<td>AUE_AUDITON_GETCAR</td>
<td>“auditon: A_GETCAR” on page 97</td>
</tr>
<tr>
<td>AUE_AUDITON_GETCLASS</td>
<td>“auditon: A_GETCLASS” on page 97</td>
</tr>
<tr>
<td>AUE_AUDITON_GETCOND</td>
<td>“auditon: A_GETCOND” on page 97</td>
</tr>
<tr>
<td>AUE_AUDITON_GETCWD</td>
<td>“auditon: A_GETCWD” on page 97</td>
</tr>
<tr>
<td>AUE_AUDITON_GETKMASK</td>
<td>“auditon: A_GETKMASK” on page 98</td>
</tr>
<tr>
<td>AUE_AUDITON_GETSTAT</td>
<td>“auditon: A_GETSTAT” on page 98</td>
</tr>
<tr>
<td>AUE_AUDITON_GPOLICY</td>
<td>“auditon: A_GETPOLICY” on page 98</td>
</tr>
</tbody>
</table>
Table A-5  Event-to-System Call Translation  (2 of 6)

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>System Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_AUDITON_GQCTRL</td>
<td>“auditon: A_GETQCTRL” on page 98</td>
</tr>
<tr>
<td>AUE_AUDITON_SETCLASS</td>
<td>“auditon: A_SETCLASS” on page 99</td>
</tr>
<tr>
<td>AUE_AUDITON_SETCOND</td>
<td>“auditon: A_SETCOND” on page 99</td>
</tr>
<tr>
<td>AUE_AUDITON_SETKMASK</td>
<td>“auditon: A_SETKMASK” on page 99</td>
</tr>
<tr>
<td>AUE_AUDITON_SETSMASK</td>
<td>“auditon: A_SETSMASK” on page 99</td>
</tr>
<tr>
<td>AUE_AUDITON_SETSTAT</td>
<td>“auditon: A_GETSTAT” on page 98</td>
</tr>
<tr>
<td>AUE_AUDITON_SETUMASK</td>
<td>“auditon: A_SETUMASK” on page 100</td>
</tr>
<tr>
<td>AUE_AUDITON_SPOLICY</td>
<td>“auditon: A_SETPOLICY” on page 100</td>
</tr>
<tr>
<td>AUE_AUDITON_SQCTRL</td>
<td>“auditon: A_SETQCTRL” on page 101</td>
</tr>
<tr>
<td>AUE_AUDITSVC</td>
<td>“auditsvc” on page 101</td>
</tr>
<tr>
<td>AUE_CHDIR</td>
<td>“chdir” on page 101</td>
</tr>
<tr>
<td>AUE_CHMOD</td>
<td>“chmod” on page 102</td>
</tr>
<tr>
<td>AUE_CHOWN</td>
<td>“chown” on page 102</td>
</tr>
<tr>
<td>AUE_CHROOT</td>
<td>“chroot” on page 102</td>
</tr>
<tr>
<td>AUE_CLOSE</td>
<td>“close” on page 103</td>
</tr>
<tr>
<td>AUE_CORE</td>
<td>“process dumped core” on page 123</td>
</tr>
<tr>
<td>AUE_CREAT</td>
<td>“creat” on page 103</td>
</tr>
<tr>
<td>AUE_ENTERPROM</td>
<td>“enter prom” on page 103</td>
</tr>
<tr>
<td>AUE_EXEC</td>
<td>“exec” on page 104</td>
</tr>
<tr>
<td>AUE_EXECVE</td>
<td>“execve” on page 104</td>
</tr>
<tr>
<td>AUE_EXIT</td>
<td>“exit” on page 105</td>
</tr>
<tr>
<td>AUE_EXITPROM</td>
<td>“exit prom” on page 104</td>
</tr>
<tr>
<td>AUE_FCHDIR</td>
<td>“fchdir” on page 105</td>
</tr>
<tr>
<td>AUE_FCHMOD</td>
<td>“fchmod” on page 105</td>
</tr>
<tr>
<td>AUE_FCHOWN</td>
<td>“fchown” on page 106</td>
</tr>
<tr>
<td>AUE_FCHROOT</td>
<td>“fchroot” on page 106</td>
</tr>
<tr>
<td>AUE_FCNTL</td>
<td>“fcntl” on page 106</td>
</tr>
<tr>
<td>Audit Event</td>
<td>System Call</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>AUE_FORK</td>
<td>“fork” on page 107</td>
</tr>
<tr>
<td>AUE_FORK1</td>
<td>“fork1” on page 107</td>
</tr>
<tr>
<td>AUE_FSTATFS</td>
<td>“fstatfs” on page 108</td>
</tr>
<tr>
<td>AUE_GETAUDIT</td>
<td>“getaudit” on page 108</td>
</tr>
<tr>
<td>AUE_GETAUID</td>
<td>“getauid” on page 108</td>
</tr>
<tr>
<td>AUE_GETMSG</td>
<td>“getmsg” on page 109</td>
</tr>
<tr>
<td>AUE_GETPMSG</td>
<td>“getpmsg” on page 110</td>
</tr>
<tr>
<td>AUE_GETPORTAUID</td>
<td>“getportaudit” on page 110</td>
</tr>
<tr>
<td>AUE_IOCTL</td>
<td>“ioctl” on page 110</td>
</tr>
<tr>
<td>AUE_KILL</td>
<td>“kill” on page 111</td>
</tr>
<tr>
<td>AUE_LCHOWN</td>
<td>“lchown” on page 111</td>
</tr>
<tr>
<td>AUE_LINK</td>
<td>“link” on page 112</td>
</tr>
<tr>
<td>AUE_LSTAT</td>
<td>“lstat” on page 112</td>
</tr>
<tr>
<td>AUE_LXSTAT</td>
<td>“lxstat” on page 112</td>
</tr>
<tr>
<td>AUE_MEMCNTL</td>
<td>“memecltl” on page 113</td>
</tr>
<tr>
<td>AUE_MKDIR</td>
<td>“mkdir” on page 113</td>
</tr>
<tr>
<td>AUE_MKNOD</td>
<td>“mknodo” on page 113</td>
</tr>
<tr>
<td>AUE_MMAP</td>
<td>“mmap” on page 114</td>
</tr>
<tr>
<td>AUE_MODADDMAJ</td>
<td>“modctl: MODADDMAJ_BIND” on page 114</td>
</tr>
<tr>
<td>AUE_MODCONFIG</td>
<td>“modctl: MODCONFIG” on page 115</td>
</tr>
<tr>
<td>AUE_MODLOAD</td>
<td>“modctl: MODLOAD” on page 115</td>
</tr>
<tr>
<td>AUE_MODUNLOAD</td>
<td>“modctl: MODUNLOAD” on page 115</td>
</tr>
<tr>
<td>AUE_MOUNT</td>
<td>“mount” on page 116</td>
</tr>
<tr>
<td>AUE_MSGCTL_RMID</td>
<td>“msgctl: IPC_RMID” on page 116</td>
</tr>
<tr>
<td>AUE_MSGCTL_SET</td>
<td>“msgctl: IPC_SET” on page 116</td>
</tr>
<tr>
<td>AUE_MSGCTL_STAT</td>
<td>“msgctl: IPC_STAT” on page 117</td>
</tr>
<tr>
<td>AUE_MSGGET</td>
<td>“msgget” on page 117</td>
</tr>
</tbody>
</table>

*Table A-5  Event-to-System Call Translation  (3 of 6)*
<table>
<thead>
<tr>
<th>Audit Event</th>
<th>System Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_MSGRCV</td>
<td>“msgrcv” on page 117</td>
</tr>
<tr>
<td>AUE_MSGSND</td>
<td>“msgsnd” on page 118</td>
</tr>
<tr>
<td>AUE_MUNMAP</td>
<td>“munmap” on page 118</td>
</tr>
<tr>
<td>AUE_NICE</td>
<td>“nice” on page 118</td>
</tr>
<tr>
<td>AUE_OPEN_R</td>
<td>“open: read” on page 119</td>
</tr>
<tr>
<td>AUE_OPEN_RC</td>
<td>“open: read, create” on page 119</td>
</tr>
<tr>
<td>AUE_OPEN_RT</td>
<td>“open: read, truncate” on page 120</td>
</tr>
<tr>
<td>AUE_OPEN_RTC</td>
<td>“open: read,create,truncate” on page 119</td>
</tr>
<tr>
<td>AUE_OPEN_RW</td>
<td>“open: read, write” on page 120</td>
</tr>
<tr>
<td>AUE_OPEN_RWC</td>
<td>“open: read,write,create” on page 120</td>
</tr>
<tr>
<td>AUE_OPEN_RWT</td>
<td>“open: read,write,truncate” on page 121</td>
</tr>
<tr>
<td>AUE_OPEN_RWTC</td>
<td>“open: read,write,create,truncate” on page 120</td>
</tr>
<tr>
<td>AUE_OPEN_W</td>
<td>“open: write” on page 121</td>
</tr>
<tr>
<td>AUE_OPEN_WC</td>
<td>“open: write,create” on page 121</td>
</tr>
<tr>
<td>AUE_OPEN_WT</td>
<td>“open: write,truncate” on page 122</td>
</tr>
<tr>
<td>AUE_OPEN_WTC</td>
<td>“open: write,create,truncate” on page 122</td>
</tr>
<tr>
<td>AUE_OSETUID</td>
<td>“setuid” on page 132</td>
</tr>
<tr>
<td>AUE_PATHCONF</td>
<td>“pathconf” on page 122</td>
</tr>
<tr>
<td>AUE_PIPE</td>
<td>“pipe” on page 122</td>
</tr>
<tr>
<td>AUE_PRIOCNTLSYS</td>
<td>“priocntl” on page 123</td>
</tr>
<tr>
<td>AUE_PUTMSG</td>
<td>“putmsg” on page 123</td>
</tr>
<tr>
<td>AUE_PUTPMSG</td>
<td>“putpmsg” on page 124</td>
</tr>
<tr>
<td>AUE_READLINK</td>
<td>“readlink” on page 124</td>
</tr>
<tr>
<td>AUE_RENAME</td>
<td>“rename” on page 125</td>
</tr>
<tr>
<td>AUE_RMDIR</td>
<td>“rmdir” on page 125</td>
</tr>
<tr>
<td>AUE_SEMCTL_GETALL</td>
<td>“semctl: GETALL” on page 125</td>
</tr>
<tr>
<td>AUE_SEMCTL_GETNCNT</td>
<td>“semctl: GETNCNT” on page 126</td>
</tr>
</tbody>
</table>
Table A-5  Event–to–System Call Translation  (5 of 6)

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>System Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_SEMCTL_GETPID</td>
<td>“semctl: GETPID” on page 126</td>
</tr>
<tr>
<td>AUE_SEMCTL_GETVAL</td>
<td>“semctl: GETVAL” on page 126</td>
</tr>
<tr>
<td>AUE_SEMCTL_GETZCNT</td>
<td>“semctl: GETZCNT” on page 127</td>
</tr>
<tr>
<td>AUE_SEMCTL_RMICD</td>
<td>“semctl: IPC_RMID” on page 127</td>
</tr>
<tr>
<td>AUE_SEMCTL_SET</td>
<td>“semctl: IPC_SET” on page 127</td>
</tr>
<tr>
<td>AUE_SEMCTL_SETALL</td>
<td>“semctl: SETALL” on page 128</td>
</tr>
<tr>
<td>AUE_SEMCTL_SETVAL</td>
<td>“semctl: SETVAL” on page 128</td>
</tr>
<tr>
<td>AUE_SEMCTL_STAT</td>
<td>“semctl: IPC_STAT” on page 128</td>
</tr>
<tr>
<td>AUE_SEMGET</td>
<td>“semget” on page 129</td>
</tr>
<tr>
<td>AUE_SEMOP</td>
<td>“semop” on page 129</td>
</tr>
<tr>
<td>AUE_SETAUDIT</td>
<td>“setaudit” on page 129</td>
</tr>
<tr>
<td>AUE_SETAUID</td>
<td>“setauid” on page 130</td>
</tr>
<tr>
<td>AUE_SETEGID</td>
<td>“setegid” on page 130</td>
</tr>
<tr>
<td>AUE_SETEUID</td>
<td>“seteuid” on page 130</td>
</tr>
<tr>
<td>AUE_SETGIDS</td>
<td>“setgid” on page 131</td>
</tr>
<tr>
<td>AUE_SETGROUPS</td>
<td>“setgroups” on page 131</td>
</tr>
<tr>
<td>AUE_SETGPRP</td>
<td>“setgroups” on page 131</td>
</tr>
<tr>
<td>AUE_SETRLIMIT</td>
<td>“setrlimit” on page 132</td>
</tr>
<tr>
<td>AUE_SETUID</td>
<td>reported as AUE_OSETUID, see “setuid” on page 132</td>
</tr>
<tr>
<td>AUE_SHMAT</td>
<td>“shmat” on page 132</td>
</tr>
<tr>
<td>AUE_SHMCTL_RMICD</td>
<td>“shmctl: IPC_RMID” on page 133</td>
</tr>
<tr>
<td>AUE_SHMCTL_SET</td>
<td>“shmctl: IPC_SET” on page 133</td>
</tr>
<tr>
<td>AUE_SHMCTL_STAT</td>
<td>“shmctl: IPC_STAT” on page 133</td>
</tr>
<tr>
<td>AUE_SHMDT</td>
<td>“shmdt” on page 134</td>
</tr>
<tr>
<td>AUE_SHMGET</td>
<td>“shmget” on page 134</td>
</tr>
<tr>
<td>AUE_SOCKACCEPT</td>
<td>“getmsg: socket accept” on page 109</td>
</tr>
<tr>
<td>AUE_SOCKCONNECT</td>
<td>“putmsg: socket connect” on page 124</td>
</tr>
</tbody>
</table>
### Table A-5  Event–to–System Call Translation  (6 of 6)

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>System Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_SOCKRECEIVE</td>
<td>“getmsg: socket receive” on page 109</td>
</tr>
<tr>
<td>AUE_SOCKSEND</td>
<td>“putmsg: socket send” on page 124</td>
</tr>
<tr>
<td>AUE_STAT</td>
<td>“stat” on page 134</td>
</tr>
<tr>
<td>AUE_STATFS</td>
<td>“statfs” on page 135</td>
</tr>
<tr>
<td>AUE_STATVFS</td>
<td>“statvfs” on page 135</td>
</tr>
<tr>
<td>AUE_STIME</td>
<td>“stime” on page 135</td>
</tr>
<tr>
<td>AUE_SYMLINK</td>
<td>“symlink” on page 136</td>
</tr>
<tr>
<td>AUE_SYSINFO</td>
<td>“sysinfo” on page 136</td>
</tr>
<tr>
<td>AUE_SYSTEMBOOT</td>
<td>“system booted” on page 136</td>
</tr>
<tr>
<td>AUE_UMOUNT</td>
<td>“umount: old version” on page 136</td>
</tr>
<tr>
<td>AUE_UNLINK</td>
<td>“unlink” on page 137</td>
</tr>
<tr>
<td>AUE_UTIME</td>
<td>“utime” on page 137</td>
</tr>
<tr>
<td>AUE_UTIMES</td>
<td>“utimes” on page 137</td>
</tr>
<tr>
<td>AUE_UTSSYS</td>
<td>“utssys - fusers” on page 138</td>
</tr>
<tr>
<td>AUE_VFORK</td>
<td>“vfork” on page 138</td>
</tr>
<tr>
<td>AUE_VTRACE</td>
<td>“vtrace” on page 138</td>
</tr>
<tr>
<td>AUE_XMKNOD</td>
<td>“xmknod” on page 138</td>
</tr>
<tr>
<td>AUE_XSTAT</td>
<td>“xstat” on page 139</td>
</tr>
</tbody>
</table>

### Table A-6  Event–to–Command Translation

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUE_allocate_succ</td>
<td>“allocate: device allocate” on page 139</td>
</tr>
<tr>
<td>AUE_allocate_fail</td>
<td>“allocate: device allocate failure” on page 140</td>
</tr>
<tr>
<td>AUE_deallocate_succ</td>
<td>“allocate: deallocate device” on page 140</td>
</tr>
<tr>
<td>AUE_deallocate_fail</td>
<td>“allocate: deallocate device failure” on page 140</td>
</tr>
<tr>
<td>AUE_listdevice_succ</td>
<td>“allocate: list device” on page 140</td>
</tr>
<tr>
<td>Audit Event</td>
<td>Command</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AUE_listdevice_fail</td>
<td>“allocate: list device failure” on page 141</td>
</tr>
<tr>
<td>AUE_at_create</td>
<td>“at: create crontab” on page 141</td>
</tr>
<tr>
<td>AUE_at_delete</td>
<td>“at: delete atjob” on page 141</td>
</tr>
<tr>
<td>AUE_at_perm</td>
<td>“at: at-permission” on page 141</td>
</tr>
<tr>
<td>AUE_crontab_create</td>
<td>“crontab: crontab created” on page 142</td>
</tr>
<tr>
<td>AUE_crontab_delete</td>
<td>“crontab: crontab deleted” on page 142</td>
</tr>
<tr>
<td>AUE_cron_invoke</td>
<td>“crontab: cron-invoke atjob or crontab” on page 142</td>
</tr>
<tr>
<td>AUE_crontab_perm</td>
<td>“crontab: crontab-permission” on page 143</td>
</tr>
<tr>
<td>AUE_halt_solaris</td>
<td>“halt” on page 143</td>
</tr>
<tr>
<td>AUE_inetd_connect</td>
<td>“inetd” on page 143</td>
</tr>
<tr>
<td>AUE_ftpd</td>
<td>“in.ftpd” on page 143</td>
</tr>
<tr>
<td>AUE_login</td>
<td>“login: terminal login” on page 144</td>
</tr>
<tr>
<td>AUE_rlogin</td>
<td>“login: rlogin” on page 144</td>
</tr>
<tr>
<td>AUE_telnet</td>
<td>“login: telnet login” on page 144</td>
</tr>
<tr>
<td>AUE_logout</td>
<td>“login: logout” on page 144</td>
</tr>
<tr>
<td>AUE_mountd_mount</td>
<td>“mountd: NFS mount” on page 145</td>
</tr>
<tr>
<td>AUE_mountd_umount</td>
<td>“mountd: NFS unmount request” on page 145</td>
</tr>
<tr>
<td>AUE_passwd</td>
<td>“passwd” on page 145</td>
</tr>
<tr>
<td>AUE_reboot_solaris</td>
<td>“reboot” on page 145</td>
</tr>
<tr>
<td>AUE_rexd</td>
<td>“rpc.rexd” on page 146</td>
</tr>
<tr>
<td>AUE_rexecd</td>
<td>“in.rexecd” on page 146</td>
</tr>
<tr>
<td>AUE_rshd</td>
<td>“in.rshd” on page 146</td>
</tr>
<tr>
<td>AUE_su</td>
<td>“su” on page 147</td>
</tr>
</tbody>
</table>
BSM Reference

BSM brings a number of additional utilities to the Solaris operating environment. The utilities are listed here in four sections, each of which has a table below. Each table gives utility names and a short description of the task performed by each utility. The sections are identified by the man page suffix.

Table B-1  Section 1M—Maintenance Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocate(1M)</td>
<td>Allocate a device</td>
</tr>
<tr>
<td>audit(1M)</td>
<td>Control the audit daemon</td>
</tr>
<tr>
<td>audit_startup(1M)</td>
<td>Initialize the audit subsystem</td>
</tr>
<tr>
<td>audit_warn(1M)</td>
<td>Run the audit daemon warning script</td>
</tr>
<tr>
<td>auditconfig(1M)</td>
<td>Configure auditing</td>
</tr>
<tr>
<td>auditd(1M)</td>
<td>Control audit trail files</td>
</tr>
<tr>
<td>auditreduce(1M)</td>
<td>Merge and select audit records from audit trail files</td>
</tr>
<tr>
<td>auditstat(1M)</td>
<td>Display kernel audit statistics</td>
</tr>
<tr>
<td>bsmconv(1M)</td>
<td>Enable a Solaris system to use the Basic Security Module</td>
</tr>
<tr>
<td>bsmunconv(1M)</td>
<td>Disable the Basic Security Module and return to the Solaris operating environment (see the bsmconv(1M) man page)</td>
</tr>
<tr>
<td>deallocate(1M)</td>
<td>Deallocate a device</td>
</tr>
<tr>
<td>Command</td>
<td>Task</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>dminfo(1M)</td>
<td>Report information about a device entry in a device maps file</td>
</tr>
<tr>
<td>list_devices(1M)</td>
<td>List allocatable devices</td>
</tr>
<tr>
<td>praudit(1M)</td>
<td>Print contents of an audit trail file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Call</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit(2)</td>
<td>Write a record to the audit log</td>
</tr>
<tr>
<td>auditon(2)</td>
<td>Manipulate auditing</td>
</tr>
<tr>
<td>auditsvc(2)</td>
<td>Write audit log to specified file descriptor</td>
</tr>
<tr>
<td>getaudit(2)</td>
<td>Get process audit information</td>
</tr>
<tr>
<td>getauid(2)</td>
<td>Get user audit identity</td>
</tr>
<tr>
<td>setaudit(2)</td>
<td>Get process audit information (see getaudit(2))</td>
</tr>
<tr>
<td>setauid(2)</td>
<td>Get user audit identity (see getauid(2))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Library Call</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>au_open(3), au_close(3), au_write(3)</td>
<td>Construct and write audit records</td>
</tr>
<tr>
<td>au_preselect(3)</td>
<td>Preselect an audit event</td>
</tr>
<tr>
<td>au_to_arg(3), au_to_attr(3), au_to_data(3), au_to_groups(3), au_to_in_addr(3), au_to_ipc(3), au_to_ipc_perm(3), au_to_iport(3), au_to_me(3), au_to_opaque(3), au_to_path(3), au_to_process(3), au_to_return(3), au_to_socket(3), au_to_text(3)</td>
<td>Create audit record tokens (see au_to(3) for all of these functions)</td>
</tr>
<tr>
<td>au_user_mask(3)</td>
<td>Get user’s binary preselection mask</td>
</tr>
<tr>
<td>getacinfo(3), getacdir(3), getacflg(3), getacmin(3), getacna(3), setac(3), endac(3)</td>
<td>Get audit control file information</td>
</tr>
</tbody>
</table>
### Table B-3 Section 3—C Library Functions (Continued)

<table>
<thead>
<tr>
<th>Library Call</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>getauclassent(3), getauclassnam(3), setauclass(3),</td>
<td>Get audit_class entry</td>
</tr>
<tr>
<td>endauclass(3), getauclassnam_r(3), getauclassent_r(3)</td>
<td></td>
</tr>
<tr>
<td>getauditflags(3), getauditflagsbin(3), getauditflagschar(3)</td>
<td>Convert audit flag specifications</td>
</tr>
<tr>
<td>getauevent(3), getauevnam(3), getauevnum(3),</td>
<td>Get audit_user entry</td>
</tr>
<tr>
<td>getauevnonam(3), setauevent(3), endauevent(3),</td>
<td></td>
</tr>
<tr>
<td>getauevent_r(3), getauevnam_r(3), getauevnum_r(3)</td>
<td></td>
</tr>
<tr>
<td>getau usernam(3), getauuserent(3), setauuser(3),</td>
<td>Get audit_user entry</td>
</tr>
<tr>
<td>endauuser(3)</td>
<td></td>
</tr>
<tr>
<td>getfauditflags(3)</td>
<td>Generate the process audit state</td>
</tr>
</tbody>
</table>

### Table B-4 Section 4—Headers, Tables, and Macros

<table>
<thead>
<tr>
<th>Files</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit.log(4)</td>
<td>Gives format for an audit trail file</td>
</tr>
<tr>
<td>audit_class(4)</td>
<td>Gives audit class definitions</td>
</tr>
<tr>
<td>audit_control(4)</td>
<td>Controls information for system audit daemon</td>
</tr>
<tr>
<td>audit_data(4)</td>
<td>Holds current information on the audit daemon</td>
</tr>
<tr>
<td>audit_event(4)</td>
<td>Holds audit event definition and class mapping</td>
</tr>
<tr>
<td>audit_user(4)</td>
<td>Holds per-user auditing data file</td>
</tr>
<tr>
<td>device_allocate(4)</td>
<td>Contains physical device information</td>
</tr>
<tr>
<td>device_maps(4)</td>
<td>Contains physical device information</td>
</tr>
</tbody>
</table>
Index

Symbols
- audit flag prefix, 10 to 11
# for comments in files, 65, 67
* in device_allocate file, 67, 68
+ audit flag prefix, 10 to 11
\ ending file lines, 65, 67
^- audit flag prefix, 11
^+ audit flag prefix, 11

A
-a option of auditreduce command, 58
 to 59
acct audit record, 96
ad audit flag, 9
adding devices, 74
adjtime audit record, 96
administering auditing
 See also audit records; audit tokens;
audit trail
audit administration account, 35 to
36
audit classes
 auditconfig command
 options, 38
 changing definitions, 40
 flags and definitions, 9 to 10
 mapping events, 7, 40
 overview, 7 to 8
 selecting for auditing, 7
 audit events
 audit tokens, 46
 auditconfig command
 options, 37, 38
 C library functions, 156
categories, 8
event–to–system call translation
table, 147 to 152
including in audit trail, 7
kernel events, 7, 8, 37, 38, 46
mapping to classes, 7, 40
numbers, 8
overview, 7 to 8
preselecting, 156
record formats and, 45
user-level events, 8, 38, 46
audit files, 26 to 30
 auditreduce command, 20 to
 22
 combining, 20 to 22, 26
 copying login/logout messages
to single file, 57 to 58
directory locations, 27, 31
displaying in entirety, 57
file token, 50, 83
managing size of, 18
minimum free space for file
systems, 12
names, 27 to 29
nonactive files marked not_terminated, 29 to 30, 58
order for opening, 12
overview, 26 to 27
permissions, 32
printing, 57
reducing, 20 to 22, 26
reducing storage-space
requirements, 23 to 24, 25
switching to new file, 17
time stamps, 28
audit flags, 8 to 11
audit_control file line, 12
audit_user file, 13 to 14
auditconfig command
options, 38
C library functions, 157
definitions, 9 to 10
machine-wide, 8, 12
overview, 8
policy flags, 39
prefixes, 11
process preselection mask, 15
syntax, 10
audit partitions, 30 to 32
audit records, 8
audit trail creation, 16 to 18
audit daemon's role, 17
audit_data file, 16
directory suitability, 17
managing audit file size, 18
overview, 16
audit trail overflow prevention, 36 to 37
audit_control file
audit_user file
modification, 13
C library functions, 156
overview, 11 to 12
prefixes in flags line, 11
problem with contents, 20
audit_user file audit fields, 13 to 14
audit_warn script, 16, 18 to 20
auditreduce command, 20 to 22, 56
to 59
-a option, 58 to 59
-b option, 58 to 59
capabilities, 56
cleaning not_terminated
files, 29 to 30, 58
-d option, 57
described, 20 to 21, 44, 56
distributed systems, 56
examples, 57 to 58
-o option, 26, 30, 57 to 58
options, 20, 58 to 59
time stamp use, 28
without options, 20 to 22
configuration
audit trail overflow
prevention, 36 to 37
auditconfig command, 37 to 39
overview, 32 to 33
planning, 33 to 36
setting audit policies, 39
cost control, 22 to 24
analysis, 23
processing time, 23
storage, 23 to 24
efficiency, 25 to 26
normal users, 25
overview, 5 to 6
process audit characteristics, 14 to 15
audit ID, 15
audit session ID, 15
process preselection mask, 15, 23
to 24
terminal ID, 15
startup, 6
administrative audit class, 9
all
audit class, 10
audit flag
cautions for using, 10
described, 10
in user audit fields, 14
allhard string with audit_warn script, 19
allocatable devices, See device allocation
allocate audit record
allocate-list device success, 140
deallocate device, 140
deallocate device failure, 140
device allocate failure, 140
device allocate success, 139
allocate command
See also device allocation
how the allocate mechanism works, 71 to 73
options, 63
using, 75 to 76
allocate error state, 64
allocating devices, See device allocation
allsoft string with audit_warn script, 19
always-audit flags
described, 13 to 14
process preselection mask, 15
analysis, 43 to 60
audit record format, 45 to 55
auditting features, 43 to 44
auditreduce command, 45, 56 to 59
costs, 23
praudit command, 45, 59 to 60
tools, 44 to 45
ap audit flag, 9
application audit class, 9
arbitrary token, 48 to 49, 79
Archive tape drive clean script, 67
arg token, 49, 81
arge policy
exec_env token and, 82
flag, 39
argv policy
eexec_args token and, 82
flag, 39
asterisk (*) in device_allocate file, 67, 68
at audit record
at-create crontab, 141
at-delete atjob, 141
at-permission, 141
attr token, 49 to 50, 81
audio devices
See also device allocation
device-clean scripts, 70
audio_clean script, 70
AUDIO_DRAIN ioctl system call, 70
AUDIO_SETINFO ioctl system call, 70
AUDIOGETREG ioctl system call, 70
AUDIODETREG ioctl system call, 70
audit administration account, 35 to 36
audit attributes. See audit tokens
audit audit record, 96
audit classes
auditconfig command options, 38
changing definitions, 40
flags and definitions, 9 to 10
mapping events, 7, 40
overview, 7 to 8
selecting for auditing, 7
audit daemon
audit trail creation, 16, 17
audit_startup file, 6
audio_clean script
conditions invoking, 18 to 20
described, 16, 18
execution of, 17
directories suitable to, 17
enabling auditing, 6
functions, 17
order audit files are opened, 12
rereading the audit_control file, 12
terminating, 16
audit events
See also audit classes
audit_event file
audit event type, 45
overview, 7 to 8
C library functions, 156
categories, 8
event-to–system call translation table, 147 to 152
including in audit trail, 7
kernel events
  audit tokens, 46
  auditconfig command options, 37, 38
described, 7
mapping to classes, 7, 40
numbers, 8
overview, 7 to 8
preselecting, 156
record formats and, 45
user-level events
  audit tokens, 46
  auditconfig command options, 38
described, 8
audit files, 26 to 30
  See also audit trail; directories
auditreduce command, 20 to 22
combining, 20 to 22, 26
copying login/logout messages to single file, 57 to 58
directory locations, 27, 31
displaying in entirety, 57
file token, 50, 83
managing size of, 18
minimum free space for file systems, 12
names, 27 to 29
closed files, 29
form, 27 to 28
still-active files, 28 to 29
time stamps, 28
use, 28
nonactive files marked not terminated, 29 to 30, 58
order for opening, 12
overview, 26 to 27
permissions, 32
printing, 57
reducing, 20 to 22, 26
reducing storage-space requirements, 23 to 24, 25
switching to new file, 17
time stamps, 28
audit flags, 8 to 11
  audit_control file line, 12
  audit_user file, 13 to 14
  auditconfig command options, 38
C library functions, 157
definitions, 9 to 10
machine-wide, 8, 12
overview, 8
policy flags, 39
prefixes, 10 to 11
process preselection mask, 15
syntax, 10
audit ID, 6, 15, 44
audit log files, See audit files
audit -n command, 17
audit partitions, 30 to 32
audit policies
  See also audit flags
  auditconfig options, 39
setting, 39
audit records, 77 to 147
  See also audit tokens; specific audit records
audit directories full, 17, 19, 94
C library functions, 156 to 157
converting to human-readable format, 8, 20, 45, 59 to 60
displaying, 45
format or structure, 45 to 55, 78, 95
kernel-level generated, 95 to 139
overview, 8
policy flags, 39
reducing audit files, 26
selecting, 44
self-contained records, 44
tools, 44 to 45
user-level generated, 139 to 147
audit -s command
  preselection mask for existing processes, 12
  rereading audit files, 17
  resetting directory pointer, 17
Index

audit server mount-point path names, 31
audit session ID, 15, 44
audit -t command, 16
audit threshold, 12
audit tokens
  arbitrary token, 48 to 49, 79
  arg token, 49, 81
  attr token, 49 to 50, 81
  audit record format, 45 to 55, 78
  C library functions, 156
described, 8
  exec_args token, 82
  exec_env token, 82
  exit token, 50, 82
  file token, 50, 83
  groups token, 50 to 51, 84
  header token, 46, 47 to 48, 84 to 85
  in_addr token, 51, 85
  ip token, 51, 85
  ipc token, 51 to 52, 86 to 87
  ipc_perm token, 52, 87
  ipport token, 52, 88
  newgroups token, 88
  opaque token, 52 to 53, 89
  order in audit record, 46
  path token, 53, 89
  policy flags, 39
  process token, 53, 90
  return token, 54, 90 to 91
  seq token, 54, 91
  socket token, 54 to 55, 91 to 92
  socket-inet token, 92
  subject token, 55, 92
  table of, 78
  text token, 55, 93
  trailer token, 46, 48, 94
  types, 45 to 46
audit trail
  See also audit files; audit records; audit tokens
  analysis, 43 to 60
  audit record format, 45 to 55
  auditing features, 43 to 44
  auditreduce command, 45, 56 to 59
costs, 23
  praudit command, 45, 59 to 60
tools, 44 to 45
creating, 16 to 18, 26
  audit daemon's role, 16, 17
  audit_data file, 16
described, 16
  directory suitability, 17
  managing audit file size, 18
  overview, 16
directory locations, 27, 31
  events included, 7
  merging all files, 20 to 22
  monitoring in real time, 25
  overflow prevention, 36 to 37
audit_control file
  audit daemon rereading after editing, 12
audit_user file modification, 13
C library functions, 156
dir: line
described, 12
  examples, 13, 32
  files subdirectory, 31
  examples, 13, 32
  flags: line
described, 12
  prefixes in, 11
  process preselection mask, 15
  minfree: line
    audit_warn condition, 19
described, 12
  naflags: line, 12
  overview, 11 to 12
  prefixes in flags line, 11
  problem with contents, 20
audit_data file, 16
audit_event file
  See also audit events
  audit event type, 45
  overview, 7 to 8
audit_startup file, 6
audit_user file
  prefixes for flags, 11
  process preselection mask, 15
  user audit fields, 13 to 14
audit_warn script, 18 to 20
  allhard string, 19
  allsoft string, 19
  audit daemon execution of, 17
  auditsvc string, 19
  conditions invoking, 18 to 20
  described, 16, 18
  ebusy string, 19
  hardstring, 19
  postsigterm string, 19
  soft string, 18
  tmpfile string, 19
auditconfig command
  audit flags as arguments, 9
  options, 37 to 39
  prefixes for flags, 11
  reducing storage-space requirements, 24
auditd daemon
  audit trail creation, 16, 17
  audit_startup file, 6
  audit_warn script
    conditions invoking, 18 to 20
    described, 16, 18
    execution of, 17
    directories suitable to, 17
    enabling auditing, 6
    functions, 17
    order audit files are opened, 12
    rereading the audit_control file, 12
    terminating, 16
auditing, See administering auditing
  audit trail
auditon audit record
  A_GETCAR command, 97
  A_GETCLASS command, 97
  A_GETCOND command, 97
  A_GETCWD command, 97
  A_GETKMASK command, 98
  A_GETPOLICY command, 98
  A_GETQCRTL command, 98
  A_GETSTAT command, 98
  A_SETCLASS command, 99
  A_SETCOND command, 99
  A_SETKMASK command, 99
  A_SETPOLICY command, 100
  A_SETQCRTL command, 101
  A_SETSMASK command, 99 to 100
  A_SETSTAT command, 100
  A_SETUMASK command, 100
auditreduce command, 20 to 22
  -a option, 58 to 59
  -b option, 58 to 59
  capabilities, 56
  cleaning not_terminated files, 29 to 30, 58
  -d option, 57
  described, 20 to 21, 44, 56
  distributed systems, 56
  examples, 57 to 58
  -m option, 59
  -o option, 26, 30, 57 to 58
  options, 20, 58 to 59
  time stamp use, 28
  without options, 20 to 22
auditsvc
  audit record, 101
  system call
    fails, 19, 94
  AUE_... names, 7, 8
    event-to-system call translation table, 147 to 152
  automatically enabling auditing, 6
B
  -b option of auditreduce command, 58 to 59
backslash (\) ending file lines, 65, 67
Basic Security Module (BSM)
  client-server relationships, 2
  disabling, 2
  enabling, 2
  installing, 1 to 3
  packages, 1
binary audit record format, 45
BSM, See Basic Security Module (BSM)
bsmconv script
Index

devicemaps file creation, 64
  enabling BSM, 2
bsmunconv script, 2

C
  C library functions, 156 to 157
  C2 TCSEC features, 61
  carat (^) in audit flag prefixes, 11
  cartridge tape drives, See tape drives
  CD-ROM drives
    See also device allocation
    device-clean scripts, 69
  change password audit record, 145
  chdir audit record, 101
  -chkconf option of auditconfig
    command, 37
  chmod audit record, 102
  chown audit record, 102
  chroot audit record, 102
  cl audit flag, 9
  classes
    auditconfig command options, 38
    changing definitions, 40
    flags and definitions, 9 to 10
    mapping events, 7, 40
    overview, 7 to 8
    selecting for auditing, 7
  clean scripts, See device-clean scripts
  cleaning not_terminated files, 29 to 30,
    58
  clients, enabling BSM for, 3
  close audit record, 103
  cnt policy, 35 to 36
  flag, 39
  combining audit files, 26
    auditreduce command, 20 to 22
  commands
    See also specific commands
    device-allocation utilities, 63 to 64
    maintenance commands, 155 to 156
  comments
    device_allocate file, 67
    device_maps file, 65
  -conf option of auditconfig
    command, 37
  configuring
    audit trail overflow prevention, 36 to
      37
    auditconfig command, 37 to 39
    overview, 32 to 33
    planning, 33 to 36
    setting audit policies, 39
  converting audit records to human-
    readable format, 8, 20, 45, 59 to
    60
  copying login/logout messages to single
    file, 57 to 58
  cost control, 22 to 24
    analysis, 23
    processing time, 23
    storage, 23 to 24
  creat audit record, 103
  creating the audit trail, 16 to 18
    audit daemon’s role, 17
    audit_data file, 16
    directory suitability, 17
    managing audit file size, 18
    overview, 16
  cron job, 18
  crontab audit record
    cron-invoke at job or
      crontab, 142
    crontab-crontab created, 142
    crontab-crontab deleted, 142
    crontab-permission, 143

D
  -d option
    auditreduce command, 57
    praudit command, 59
  daemon, audit, See audit daemon
  date-time auditreduce command
    options, 58 to 59
  deallocate command
    allocate error state, 64
device allocation, 61 to 76
adding devices, 74
allocatable devices, 66, 67, 74
allocate command
how the allocate mechanism works, 71 to 73
options, 63
using, 75 to 76
allocate error state, 64
allocating a device, 75 to 76
components of the allocation mechanism, 62
deallocate command
allocate error state, 64
described, 63, 75
task clean scripts and, 70
using, 76
debugging sequence number, 54, 91
defaults
audit policies, 39
audit_startup file, 6
machine-wide, 8
praudit output format, 59, 60
header token, 48
device_allocate file
format, 67 to 68
overview, 66 to 68
device_maps file
format, 65 to 66
overview, 64
device-clean scripts
adding devices, 74
audio devices, 70
CD-ROM drives, 69
described, 68
diskette drives, 69
options, 70
tape drives, 67, 69
writing new scripts, 70
done
See also device allocation
adding, 74
lock files, 70 to 73
managing, 74
dir: line in audit_control file
described, 12
example, 13, 32
for files subdirectory, 31
directories
audit daemon pointer, 17
audit directories full, 17, 19, 94
audit directory locations, 27, 31
audit partitions, 30 to 32
audit_control file definitions, 12
diskfull machines, 27, 30
files subdirectory, 31
mounting audit directories, 27
permissions, 32
suitable to audit daemon, 17
disabling BSM, 2
diskette drives
See also device allocation
device-clean scripts, 69
diskfull machines' audit directory, 27, 30
diskless clients, enabling BSM for, 3
disk-space requirements, 23 to 24
displaying
audit log in entirety, 57

166

SunSHIELD Basic Security Module Guide—November 1995
audit records, 45
distributed systems' auditreduce command use, 56
dminfo command, 64
drives, See device allocation

E

ebusy string and audit_warn script, 19
efficiency, 25 to 26
eject command, 69
enabling
   auditing, 6
   BSM, 2
ending
   disabling BSM, 2
   signal received during auditing shutdown, 19
   terminating audit daemon, 16
enter prom audit record, 103
errors
   allocate error state, 64
   audit directories full, 17, 19, 94
   internal errors, 19
   /etc/security directory, 31
   /etc/security/audit directory, 27, 31
   /etc/security/audit/bsmconv script enabling BSM, 2
devicemaps file creation, 64
   /etc/security/audit/bsmunconv script, 2
   /etc/security/audit/audit_control file, See audit_control file
   /etc/security/audit_data file, 16
   /etc/security/audit_event file
      audit event type, 45
      overview, 7 to 8
      See also audit events
   /etc/security/audit_startup file, 6
   /etc/security/audit_warn script, 16, 18 to 20
   /etc/security/dev lock files, 70 to 73
event modifier field flags (header token), 85
event numbers, 8
events
   See also audit classes
   C library functions, 156
categories, 8
event-to-system call translation table, 147 to 152
including in audit trail, 7
kernel events
   audit tokens, 46
   auditconfig command options, 37, 38
described, 7
mapping to classes, 7, 40
numbers, 8
overview, 7 to 8
preselecting, 156
record formats and, 45
user-level events
   audit tokens, 46
   auditconfig command options, 38
described, 8
ex audit flag, 10
exec audit class, 10
exec audit record, 104
exec_args token, 82
exec_env token, 82
execve audit record, 104
exit audit record, 105
exit prom audit record, 104
exit token, 50, 82
export list, 27

F

-F option
   allocate command, 63
deallocate command, 63
   st_clean script, 70
fa audit flag, 9
failure
human-readable audit record format
  See also audit tokens
  converting audit records to, 8, 20, 45,
  59 to 60
  described, 45 to 55

I
  -I option
    deallocate command, 64
    st_clean script, 70
IDs
  audit, 6, 15, 44
  audit session, 15, 44
  audit user, 44
  auditconfig command options, 38
  terminal, 15
in.ftpd audit record, 143
in.rexedc audit record, 146
in.rshd: rshd access
denials/grants audit
  record, 146 to 147
in_addr token, 51, 85
inetd: inetd service request audit
  record, 143
installing BSM, 1 to 3
Internet-related tokens
  in_addr token, 51, 85
  ip token, 51, 85
  iport token, 52, 88
  socket token, 54 to 55, 91 to 92
  socket/inet token, 92
io audit flag, 9
ioctl audit class, 9
ioctl system calls, 9, 70
ioctl: ioctl to special devices
  audit record, 110 to 111
ip audit flag, 9
ip token, 51, 85
ipc audit class, 9
ipc token, 51 to 52, 86 to 87
ipc type field values (ipc token), 87
ipc_perm token, 52, 87
iport token, 52, 88
item size field values (arbitrary
  token), 80

K
kernel events
  See also audit events
  audit records, 95 to 139
  audit tokens, 46
  auditconfig command options, 37,
  38
  described, 7
  kill audit record, 111

L
  -l option
  praudit command, 59
  lchown audit record, 111 to 112
  libraries, C functions, 156 to 157
  link audit record, 112
  list_devices command, 64, 75
  lo audit flag, 9
  lock files
    how the allocate mechanism
    works, 71 to 73
    setting up, 71
  log files, See audit files
  login audit record
    logout, 144
    rlogin, 144
    telnet login, 144
    terminal login, 144
  login/logout messages, copying to single
    file, 57 to 58
  login_logout audit class, 9
  -lsevent option of auditconfig
    command, 38
  -lspolicy option of auditconfig
    command, 39
  lstat audit record, 112
  lxstat audit record, 112
M

-m option of auditreduce command, 59
machine halt audit record, 143
machine reboot audit record, 145
macros, 157
maintenance commands, 155 to 156
managing devices, 74
mappings, class, 7, 40
mask, process preselection
    auditconfig command options, 38
    C library functions, 156
described, 15
machine-wide, 12
    reducing storage costs, 23 to 24
memcntl audit record, 113
minfree: line in audit_control file
    audit_warn condition, 18, 19
described, 12
determining space needed, 34
minus (-) audit flag prefix, 10 to 11
mkdir audit record, 113
mknod audit record, 113
mmap audit record, 114
modctl audit record
    MODADDMAJBIND command, 114
    MODCONFIG command, 115
    MODLOAD command, 115
    MODUNLOAD command, 115
modems, See device allocation
monitoring audit trail in real time, 25
mount audit record, 116
mountd audit record
    NFS mount request, 145
    NFS unmount request, 145
mounting audit directories, 27
msgctl audit record
    IPC_RMID command, 116
    IPC_SET command, 116 to 117
    IPC_STAT command, 117
msgget audit record, 117
msgrcv audit record, 117
msgsnd audit record, 118
mt command, device-cleanup option, 69
munmap audit record, 118

N

na audit flag, 9
naflags: line in audit_control file, 12
names
    audit classes, 9 to 10
    audit files
        closed files, 29
        form, 27 to 28
        still-active files, 28 to 29
        time stamps, 28
        use, 28
    audit flags, 9 to 10
device names
    device_allocate file, 67
device_maps file, 65
IDs
    audit, 6, 15
    audit session, 15, 44
    auditconfig command
        options, 38
terminal, 15
kernel events, 7
mount-point path names on audit
    servers, 31
user-level events, 8
network audit class, 9
never-audit flags, 13 to 14
newgroups token, 88
NFS mount request audit record, 145
NFS unmount request audit
    record, 145
nice audit record, 118
no audit flag, 9
no_class audit class, 9
non_attrib audit class, 9
nonattributable flags in audit_control
    file, 12
normal users, auditing, 25
not_terminated files, cleaning, 29 to 30, 58
nt audit flag, 9
null audit class, 9
numbers, event, 8

O
-O option of auditreduce command, 26, 30, 57 to 58
object-reuse requirement, 61, 68 to 70
device-clean scripts
  adding devices, 74
  audio devices, 70
  CD-ROM drives, 69
described, 68
diskette drives, 69
tape drives, 67, 69
writing new scripts, 70
opaque token, 52 to 53, 89
open audit record
read, 119
read, create, 119
read, create, truncate, 119
read, truncate, 120
read, write, 120
read, write, create, 120
read, write, create, truncate, 120
read, write, truncate, 121
write, 121
write, create, 121
write, create, truncate, 122
write, truncate, 122
ot audit flag, 10
other audit class, 10
overflow prevention for audit trail, 36 to 37

P
partitions, audit, 30 to 32
passwd audit record, 145
path policy flag, 39
path token, 53, 89
pathconf audit record, 122
pc audit flag, 9
permissions for audit file systems, 32
pipe audit record, 122 to 123
plus (+) audit flag prefix, 10 to 11
policies
  See also audit flags
  auditconfig options, 39
  setting, 39
postsigterm string and audit_warn script, 19
pound sign (#) for comments in files, 65, 67
praudit command
  See also audit tokens
  converting audit records to human-readable format, 8, 20
described, 45
human-readable format, 46 to 55
output formats, 59 to 60
piping auditreduce output to, 57
using, 59 to 60
prefixes in audit flags, 10 to 11
preselection mask
  auditconfig command options, 38
  C library functions, 156
described, 15
machine-wide, 12
reducing storage costs, 23 to 24
primary audit directory, 12, 30
print format field values (arbitrary token), 80
printing audit log, 57
priocnt audit record, 123
process audit characteristics, 14 to 15
  audit ID, 15
  audit session ID, 15
  process preselection mask, 15, 23 to 24
  terminal ID, 15
process audit class, 9
process dumped core audit record, 123
process groups tokens
groups token, 50 to 51, 84
newgroups token, 88
process preselection mask
auditconfig command options, 38
described, 15
reducing storage costs, 23 to 24
process token, 53, 90
processing time costs, 23
putmsg audit record, 123
socket connect, 124
socket send, 124
putpmsg audit record, 124

R
-r praudit output format, 59, 60
header token, 48
raw praudit output format, 59, 60
header token, 48
readlink audit record, 124
reallocating devices, 63
reboot: machine reboot audit record, 145
records, See audit records
reducing audit files, 26
auditreduce command, 20 to 22
storage-space requirements, 23 to 24, 25
rename audit record, 125
return token, 54, 90 to 91
rewoff1 option of mt command, 69
risks associated with device use, 62
rmdir audit record, 125
rpc.rexd audit record, 146
rshd access denials/grants audit record, 146 to 147

S
-S option of st_clean script, 70
-s praudit output format, 59
header token, 48
SCSI devices
See also device allocation
st_clean script, 67
secondary audit directory, 12, 30
security risks associated with device use, 62
selecting audit records, 44
semctl audit record
GETALL command, 125
GETNCNT command, 126
GETPID command, 126
GETVAL command, 126
GETZCNT command, 127
IPC_RMDI command, 127
IPC_SET command, 127
IPC_STAT command, 128 to 129
SE1ALL command, 128
SE1VAL command, 128
semget audit record, 129
semop audit record, 129
seq policy flag, 40
seq token, 54, 91
servers, enabling BSM for clients, 3
session ID, 15, 44
setaudit audit record, 129 to 130
setauid audit record, 130
-setclass option of auditconfig command, 38
-setcond option of auditconfig command, 38
setegid audit record, 130
seteuid audit record, 130
-setgid audit record, 131
setgroups audit record, 131
setpgrp audit record, 131
-setmask option of auditconfig command, 38
-setpolicy option of auditconfig command, 39
setrlimit audit record, 132
-setsmask option of auditconfig command, 38
-setuid audit record, 132
-setumask option of auditconfig command, 38
SHIELD Basic Security Module, See Basic Security Module (BSM)
shmct1 audit record
  IPC_RMD command, 133
  IPC_SET command, 133
  IPC_STAT command, 133 to 134
shmdt audit record, 134
shmget audit record, 134
short praudit output format, 59
header token, 48
shutting down, See terminating signal received during auditing shutdown, 19
size
  managing audit files, 18
  reducing audit files, 26
    auditreduce command, 20 to 22
  storage-space requirements, 23 to 24, 25
socket accept audit record, 109
socket connect audit record, 124
socket receive audit record, 109
socket send audit record, 124
socket token, 54 to 55, 91 to 92
socket-inet token, 92
soft limit
  audit_warn condition, 18
  determining space needed, 34
    minfree: line described, 12
soft string with audit_warn script, 18
Solaris SHIELD Basic Security Module, See Basic Security Module (BSM)
sr_clean script, 69
st_clean script for tape drives, 67, 69
standard cleanup, 70
starting, See enabling
stat audit record, 134
statfs audit record, 135
statvfs audit record, 135
stime audit record, 135
storage costs, 23 to 24
storage overflow prevention, 36 to 37
su audit record, 147
subject token, 55, 92
success
  audit flag prefix, 10
  turning off audit flags for, 11
SUNWcar package, 1
SUNWcsr package, 1
SUNWcsu package, 1
SUNWhea package, 1
SUNWman package, 1
symlink audit record, 136
sysinfo audit record, 136
system booted audit record, 136
system calls
  arg token, 49, 81
  auditsvc fails, 19, 94
  close, 9
  event numbers, 7
  event–to–system call translation table, 147 to 152
  exec_args token, 82
  exec_env token, 82
  ioctl, 9, 70
  return token, 54, 90 to 91
  table, 156
System V IPC
  ipc audit class, 9
  ipc token, 51 to 52, 86 to 87
  ipc_perm token, 52, 87
T
tables, 157
tail command, 25
tape drives
  See also device allocation
device-clean scripts, 69
risks associated with use, 62
st_clean script, 67
TCP address, 52, 88
TCSEC (Trusted Computer System Evaluation Criteria) C2 features, 61
temporary file cannot be used, 19
terminal ID, 15
terminals, See device allocation terminating
audit daemon, 16
signal received during auditing shutdown, 19
text token, 55, 93
time stamps in audit files, 28
time-date auditreduce command options, 58 to 59
tmpfile string and audit_warn script, 19
tokens, See audit tokens
trail policy flag, 40
trail, See audit trail
trailer token
described, 48, 94
fields, 48
format, 94
order in audit record, 46, 94
praudit display, 48
Trusted Computer System Evaluation Criteria (TCSEC) C2 features, 61

U
-U option
allocate command, 63
list_devices command, 64
UDP address, 52, 88
umount: old version audit record, 136
unlink audit record, 137
user audit fields, 13 to 14
user ID (audit ID), 6, 15, 44
user-level events
See also audit events
audit records, 139 to 147
audit tokens, 46
auditconfig command options, 38
described, 8
/usr/bin/at audit record
at-create crontab, 141
at-delete atjob, 141
at-permission, 141
/usr/bin/crontab audit record
cron-invoke atjob or crontab, 142
crontab-crontab created, 142
crontab-crontab deleted, 142
crontab-permission, 143
/usr/bin/login audit record
logout, 144
rlogin, 144
telnet login, 144
terminal login, 144
/usr/bin/passwd: change password
audit record, 145
/usr/bin/su audit record, 147
/usr/lib/nfs/mountd audit record
NFS mount request, 145
NFS unmount request, 145
/usr/sbin/allocate audit record
allocate-list device
success, 140
deallocate device, 140
deallocate device failure, 140
device allocate failure, 140
device allocate success, 139
/usr/sbin/auditd daemon, See audit daemon
/usr/sbin/halt audit record, 143
/usr/sbin/in.ftpd audit record, 143
/usr/sbin/in.rexecd audit record, 146
/usr/sbin/in.rshd audit record, 146
to 147
/usr/sbin/inetd audit record, 143
/usr/sbin/reboot audit record, 145
Index

/usr/sbin/rpc.rexd audit record, 146
utilities
  C library functions, 156 to 157
  device allocation, 63 to 64
  headers, tables, and macros, 157
  maintenance commands, 155 to 156
  system calls, 156
utime audit record, 137
utimes audit record, 137
utssys - fusers audit record, 138

V
vfork audit record, 138
viewing, See displaying
vnode token, 49 to 50, 81
vtrace audit record, 138

W
writing new device-clean scripts, 70

X
xmknod audit record, 138
xstat audit record, 139
Xylogics tape drive clean script, 67

Z