Managing Auditing in Oracle® Solaris 11.4

Part No: E61027
November 2020

Managing Auditing in Oracle Solaris 11.4

Part No: E61027

Contents

Using This Documentation

- **Overview** – Describes how to administer auditing on Oracle Solaris systems.
- **Audience** – System administrators who must implement security on the enterprise.
- **Required knowledge** – Familiarity with security concepts and terminology.

Product Documentation Library

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

Feedback

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

♦ ♦ ♦ CHAPTER 1

About Auditing in Oracle Solaris

The auditing subsystem of Oracle Solaris keeps a record of how the system is being used. The audit service includes tools to assist with the analysis of the auditing data. This chapter introduces how auditing works in Oracle Solaris and also covers "Using Oracle Audit Vault and Database Firewall for Storage and Analysis of Audit Records" on page 29.

Current users of Oracle Solaris auditing should review "What's New in the Audit Service in Oracle Solaris 11.4" on page 13.

If you are new to Oracle Solaris auditing, you should review this chapter.

What's New in the Audit Service in Oracle Solaris 11.4

The Oracle Solaris 11.4 release includes auditing enhancements to add ease of use, greater flexibility, and more fine-grained control to your auditing options.

This section points to the new or enhanced auditing features in this release:

- "New Feature – Auditing Events Temporarily" on page 51
- "New Feature – Refreshing the `auditset` SMF Service After Changing Event-Class Mappings" on page 52
- "New Feature – Auditing Verified Boot" on page 52
- "New Feature – `auditstat` Command Extended" on page 53
- "New Feature – Annotating Reason for Access in the Audit Record" on page 65
- "New Feature – Per-Object Logging of Audit Events" on page 73
- "New Feature – Per-Privilege Logging of Audit Events" on page 73
- "New Feature – Restricting Access to Audit Records With File Labeling" on page 88
- "New Feature – Flexible Per-Plugin Configuration of Audit Classes" on page 89
- "New Feature – Reviewing Multiple Audit Events" on page 121
- "New Feature – Listing the Available Audit Classes" on page 120

- "New Feature – Filtering Audit Records by Functional Area" on page 121
- "New Feature – Viewing a Summary of Audit Records" on page 122
- "New Feature – Displaying Auditing Data Graphically" on page 123
- "New Feature – Listing Audit Events by Audit Class" on page 120

What Is Auditing?

Auditing is the collecting of data about the use of system resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility for actions that take place on a system.

Successful auditing starts with identification and authentication. At each login, after a user supplies a user name and PAM (pluggable authentication module) authentication succeeds, a unique and *immutable audit user ID* is generated and associated with the user, and a unique *audit session ID* is generated and associated with the user's "Process Audit Characteristics" on page 143. The *audit session ID* is inherited by every process that is started during that login session. When a user assumes a role or switches to another user, all original user actions are tracked with the same *immutable audit user ID*. For more details about assuming a role (switching identity), see the `su`(8) man page. Note that by default, certain actions such as booting and shutting down the system are always audited.

The audit service enables the following operations:

- Monitoring security-relevant events that take place on the system
- Recording the events in a network-wide audit trail
- Detecting misuse or unauthorized activity
- Reviewing patterns of access and the access histories of individuals and objects
- Discovering attempts to bypass the protection mechanisms
- Discovering extended use of privilege that occurs when a user changes identity

Note - To maintain security, audited events do not include sensitive information such as passwords. For more details, see "Audit Records and Audit Tokens" on page 23.

How Does Auditing Work?

Auditing generates audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown

- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of rights
- Identification actions and authentication actions
- Permission changes by a process or user
- Administrative actions, such as installing a package
- Site-specific applications

Audit records are generated from three sources:

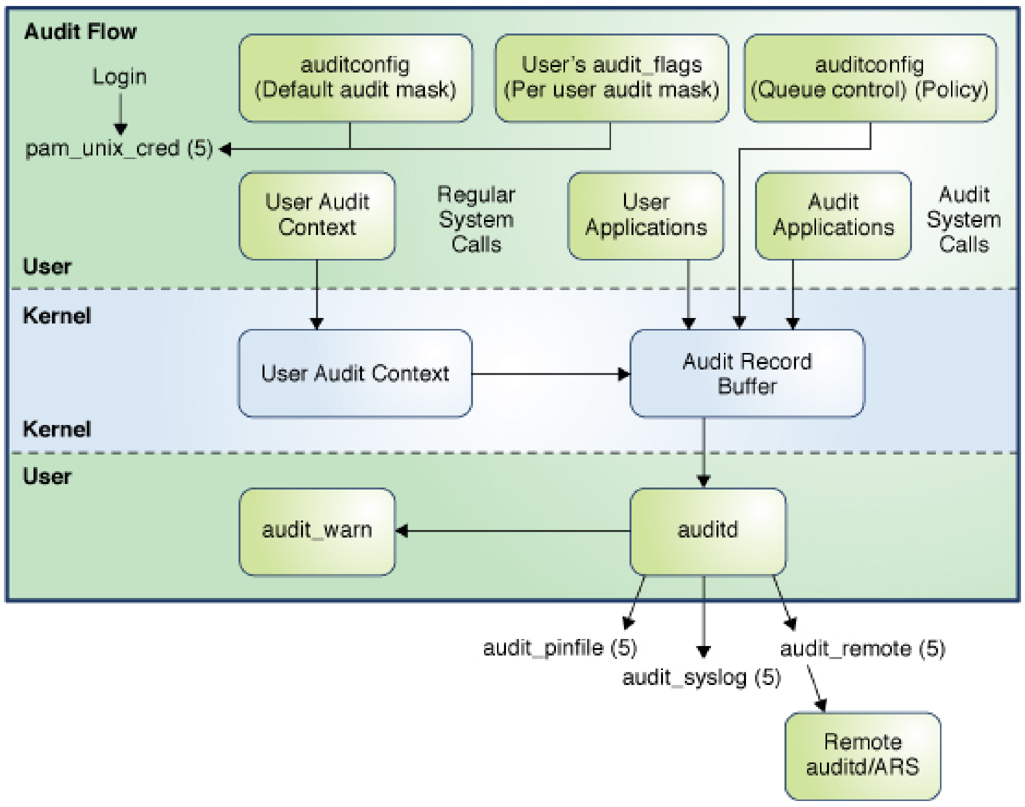
- By an application
- As a result of an asynchronous audit event
- As a result of a process system call

After the relevant event information has been captured, the information is formatted into an audit record. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information. This record is then placed in an audit queue and sent to the active *plugins* for storage. At least one plugin must be active or the remote audit server must be configured, although all plugins can be active. Plugins are described in "How Is Auditing Configured?" on page 15 and "Audit Plugin Modules" on page 24.

How Is Auditing Configured?

During system configuration, you *preselect* which classes of audit records to monitor. You can also fine-tune the degree of auditing that is done for individual users. The following figure shows details of the flow of auditing in Oracle Solaris.

FIGURE 1 Flow of Auditing



You use the `auditconfig` command to specify the audit parameters for your system. In addition, you can modify the audit flags to preselect which audit classes are to be audited for a process.

The audit service, `auditd`, tracks auditable actions that occur on a system. See "Audit Service" on page 136 and the `auditd(8)` man page.

After audit data is collected in the kernel, plugins distribute the data to the appropriate locations.

- The `audit_binfile` plugin places binary audit records in `/var/audit`. By default, the `audit_binfile` plugin is active. Post-selection tools enable you to examine interesting parts of the audit trail.

Audit files can be stored in one or more ZFS pools. The collection of audit files that are linked together is considered an *audit trail*.

- The `audit_remote` plugin sends binary audit records across a protected link to a remote audit service that is running an audit remote server (ARS). Records received by the remote server can only be sent to a local binary file. See "Managing the Audit Remote Server" on page 28.
- The `audit_syslog` plugin sends text summaries of audit records to the `syslog` utility.

Note - In addition to forwarding audit data to the plugins as described in this section, by default, audit records in the `sstore` meta-class are automatically forwarded to the Oracle Solaris StatsStore. For more information, see "`sstore` Audit Meta-Class" on page 48.

Systems that install non-global zones can audit all zones identically from the global zone. These systems can also be configured to collect different records in the non-global zones. For more information, see "Auditing on a System With Oracle Solaris Zones" on page 18.

How Is Auditing Related to Security?

Auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities.

To protect a computer system, especially a system on a network, requires mechanisms that control activities before system processes or user processes begin. Security requires tools that monitor activities as the activities occur. Security also requires reports of activities after the activities have happened.

Set audit parameters before users log in or system processes begin, because most audit activity involves monitoring current events and reporting the events that meet the specified parameters. How the audit service monitors and reports these events is discussed in detail in Chapter 2, "Planning for Auditing" and Chapter 3, "Managing the Audit Service".

Auditing cannot prevent hackers from unauthorized entry. However, the audit service can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and user name. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the audit service provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

Auditing on a System With Oracle Solaris Zones

A zone is a virtualized operating system environment that is created within a single instance of the Oracle Solaris OS. The audit service audits the entire system, including activities in zones. A system that has installed non-global zones can run a single audit service in the global zones to audit all zones identically. Or, it can run one audit service per non-global zone, including an audit service for the global zone. These audit services would be administered separately.

Sites can run a single audit service in the global zone in the following circumstances:

- The site requires a single-image audit trail.
- The non-global zones are used to contain and isolate applications. The zones are part of one administrative domain. That is, no non-global zone has customized naming service files.

  If all the zones on a system are within one administrative domain, the `zonename` audit policy can be used to distinguish audit events that are configured in different zones.
- Administrators want low audit overhead. The global zone administrator audits all zones identically. Also, the global zone's audit daemon serves all zones on the system.

Sites can run one audit service per non-global zone in the following circumstances:

- The site does not require a single-image audit trail.
- The non-global zones have customized naming service files. These separate administrative domains typically function as servers.
- Individual zone administrators want to control auditing in the zones that they administer. In per-zone auditing, zone administrators can decide to enable or to disable auditing for the zone that they administer.

The advantages of per-zone auditing are a customized audit trail for each zone, and the ability to disable auditing on a zone-by-zone basis. These advantages can be offset by the administrative overhead. Each zone administrator must administer auditing. Each zone runs its own audit daemon, and has its own audit queue and audit logs. These audit logs must be managed.

Audit Terminology and Concepts

This section provides an introduction to auditing terminology and concepts. Some definitions include pointers to more complete descriptions.

audit class	A grouping of audit events. Audit classes provide a way to select a group of events to be audited.

	For more information, see "Audit Classes and Preselection" on page 22, and the `audit_flags(7)`, `audit_class(5)`, and `audit_event(5)` man pages.
audit file system	A repository of audit files in binary format. For more information, see "Audit Logs" on page 24 and the `audit.log(5)` man page.
audit event	A security-related system action that is auditable. For ease of selection, events are grouped into audit classes. For more information, see "Audit Events" on page 21 and the `audit_event(5)` man page.
audit flag	An audit class that is supplied as an argument to a command or keyword. Audit flags preselect which audit classes are to be audited for a process. For information about using audit flags, see "Audit Class Syntax" on page 139 and the `audit_flags(7)` man page.
audit plugin	A module that transfers the audit records in the queue to a specified location. The `audit_binfile` plugin creates binary audit files. Binary files comprise the audit trail, which is stored on audit file systems. The `audit_remote` plugin sends binary audit records to a remote repository. The `audit_syslog` plugin summarizes audit records and writes them to the system log using `syslog`(3C). For more information, see "Audit Plugin Modules" on page 24 and the module man pages, `audit_binfile(7)`, `audit_remote(7)`, and `audit_syslog(7)`.
audit policy	A set of auditing options that you can enable or disable at your site. You can specify policies such as the following: ■ Whether to record certain kinds of audit data ■ How much information to include in the audit content ■ How to handle certain types of files ■ How to handle a full audit queue For more information, see "Understanding Audit Policy" on page 40 and the `auditconfig(8)` man page.
audit record	Audit data that is collected in the audit queue. An audit record describes a single audit event. Each audit record is composed of audit tokens.

	For more information, see "Audit Records and Audit Tokens" on page 23 and the `audit.log(5)` man page.
audit token	A field of an audit record. Each audit token describes an attribute of an audit record, such as a user, a group, a program, or other object. For more information, see "Audit Token Formats" on page 146 and the `audit.log(5)` man page.
audit trail	A collection of one or more audit files that store the audit data from all audited systems that use the default plugin, `audit_binfile`. For more information, see "Audit Trail" on page 144.
local auditing	The collecting of audit records that are generated on the local system. The records can be generated in the global zone or in non-global zones, or both. For more information, see "Audit Plugin Modules" on page 24.
post-selection	The choice of which audit events to examine from the preselected audit trail. The default active plugin, `audit_binfile`, creates the audit trail. A post-selection tool, the `auditreduce` command, selects records from the audit trail. For more information, see the `auditreduce(8)` and `praudit(8)` man pages.
preselection	The initial choice of which audit classes to monitor. The audit events of preselected audit classes are collected in the audit queue. Audit classes that are not preselected are not audited, so their events do not appear in the queue.

Note - Only audit events that are preselected are available for further post-selection review using the `auditreduce` command.

	For more information, see "Audit Classes and Preselection" on page 22 and the `audit_flags(7)` and `auditconfig(8)` man pages.
public object	A file that is owned by the `root` account and readable by the world. For example, some files in the `/etc` directory and the `/usr/bin` directory are public objects. Public objects are not audited for read-only events. For example, even if the `file_read` (`fr`) audit class is preselected, the

	reading of public objects is not audited. You can override the default by changing the `public` audit policy option.
remote auditing	The audit remote server (ARS) that receives and stores audit records from a system that is being audited and is configured with an active `audit_remote` plugin. To distinguish an audited system from an ARS, the audited system can be referred to as the "locally audited system". For more information, see the `-setremote` options on the `auditconfig`(8) man page, the `ars`(7) man page, and "Audit Remote Server" on page 141.

Audit Events

Audit events represent auditable actions on a system. Audit events are listed in the `/etc/security/audit_event` file. Each audit event is connected to a system call or user command, and is assigned to one or more audit classes. For a description of the format of the `audit_event` file, see the `audit_event`(5) man page.

For example, the `AUE_EXECVE` audit event audits the `execve()` system call. In the following example, when you preselect either the audit class `ps` or the audit class `ex`, then every `execve()` system call is recorded in the audit queue.

```
$ auditrecord -e execve
execve
system call execve                See execve(2)
event ID    23                    AUE_EXECVE
class       ps,ex                 (0x0000000080100000)
header
path
[attribute]                   omitted on error
[exec_arguments]              output if argv policy is set
[exec_environment]            output if arge policy is set
subject
[use_of_privilege]
return
```

Auditing handles *attributable* and *non-attributable* events. Audit policy divides events into *synchronous* and *asynchronous* events, as follows:

- **Attributable events –** Events that can be attributed to a user. The `execve()` system call can be attributed to a user, so the call is considered an attributable event. All attributable events are synchronous events.
- **Non-attributable events –** Events that occur at the kernel-interrupt level or before a user is authenticated. The `na` audit class handles audit events that are non-attributable. For

example, booting the system is a non-attributable event. Most non-attributable events are asynchronous events. However, non-attributable events that have associated processes, such as a failed login, are synchronous events.

- **Synchronous events –** Events that are associated with a process in the system. Synchronous events are the majority of system events. If synchronous events cannot be queued, the process is blocked until they can be queued.
- **Asynchronous events –** Events that are not associated with any process, so no process is available to be blocked and later started. Initial system boot and PROM enter and exit events are examples of asynchronous events.

In addition to the audit events that are defined by the audit service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications. Vendors need to contact their Oracle Solaris representative to reserve event numbers and obtain access to the audit interfaces.

Audit Classes and Preselection

Each audit event belongs to an *audit class*. Audit classes are convenient containers for large numbers of audit events. When you *preselect* a class to be audited, all the events in that class are recorded in the audit queue. For example, when you preselect the `ps` audit class, `execve()`, `fork()`, and other system calls are recorded.

You can preselect for events on a system and for events initiated by a particular user.

- **System-wide preselection –** Specify the system-wide defaults for auditing by using the `-setflags` and `-setnaflags` options to the `auditconfig` command.

  Note - If the `perzone` policy is set, default audit classes can be specified in every zone. For `perzone` auditing, the defaults are zone-wide, not system-wide.

  Also, note that, to use the `auditconfig` command, you must be an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

- **User-specific preselection –** Specify additional audit flags to audit for individual users to be included along with the system-wide auditing defaults. The `useradd`, `roleadd`, `usermod`, and `rolemod` commands place the `audit_flags` security attribute in the `user_attr` database. The `profiles` command places audit flags for rights profiles in the `prof_attr` database.

  The audit preselection mask determines which classes of events are audited for a user. For a description of the user preselection mask, see "Process Audit

Characteristics" on page 143. For the configured audit flags that are used, see "Order of Search for Assigned Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Audit classes are defined in the `/etc/security/audit_class` file. Each entry contains the audit mask for the class, the name for the class, and a descriptive name for the class. For example, the `lo` and `ps` class definitions appear in the `audit_class` file as follows:

```
0x0000000000001000:lo:login or logout
0x0000000000100000:ps:process start/stop
```

The audit classes include the two global classes: `all` and `no`. The audit classes are described in the `audit_class(5)` man page. For the list of classes, read the `/etc/security/audit_class` file.

The mapping of audit events to classes is configurable. You can remove events from a class, add events to a class, and create a new class to contain selected events. For the procedure, see "How to Change an Audit Event's Class Membership" on page 63. To view the events that are mapped to a class, use the `auditrecord -c` *class* command.

Audit Records and Audit Tokens

Each *audit record* records the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. The following example shows a `login` audit record with three tokens, `header`, `subject`, and `return`:

```
header,69,2,login - local,,example_system,2010-10-10 10:10:20.020 -07:00
subject,jdoe,jdoe,staff,jdoe,staff,1210,4076076536,69 2 example_system
return,success,0
```

The type of information that is saved for each audit event is defined by a set of *audit tokens*. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. In the preceding example, each line begins with the name of the audit token. The content of the audit token follows the token name. Together, the `header`, `subject`, and `return` audit tokens comprise the `login - local` audit record. To display the tokens that comprise an audit record, use the `auditrecord -e` *event* command.

Note - Files with the `sensitive` system attribute do not have their contents or content changes included in the audit record. The attribute ensures that no sensitive information in specific files, such as passwords, PINs, keys, and so on, is accessible to anyone. For more details, refer to the `pfedit(8)` man page.

For a detailed description of the structure of each audit token with an example of `praudit` output, see "Audit Token Formats" on page 146. For a description of the binary stream of audit tokens, see the `audit.log`(5) man page.

Audit Plugin Modules

The audit plugin modules direct the audit records from the audit queue to a file or repository. At least one plugin must be active or ARS must be configured. By default, the `audit_binfile` plugin is active. You configure plugins with the `auditconfig -setplugin` *plugin-name* command.

Note - To use the `auditconfig` command, you must be an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

The audit service provides the following plugins:

- `audit_binfile` plugin – Handles delivery of the audit queue to the binary audit files. For more information, see the `audit.log`(5) man page.
- `audit_remote` plugin – Handles secure delivery of binary audit records from the audit queue to a configured remote server. The `audit_remote` plugin uses the `libgss()` library to authenticate the server. The transmission is protected for privacy and integrity. For information, see "Managing the Audit Remote Server" on page 28.
- `audit_syslog` plugin – Handles delivery of selected records from the audit queue to the `syslog` logs.

For information about how to configure a plugin, see the `auditconfig`(8) man page. For examples of plugin configuration, see the tasks in "Configuring Local Audit Logs" on page 90. For information about the plugins, see the `audit_binfile`(7), `audit_remote`(7), and `audit_syslog`(7) man pages.

Audit Logs

Audit records are collected in audit logs. The audit service provides three output modes for audit records.

- Logs that are called *audit files* store audit records in binary format. The set of audit files from a system or site provides a complete audit record. The complete audit record is called the *audit trail*. These logs are created by the `audit_binfile` plugin or the Audit Remote Server, and can be reviewed by the `praudit` and `auditreduce` post-selection commands.

- The `audit_remote` plugin streams audit records to a remote repository. The repository is responsible for maintaining an audit trail and supplying post-selection tools.
- The `syslog` utility collects and stores text summaries of the audit record. A `syslog` record is not complete. The following example shows a `syslog` entry for a `login` audit record:

```
Feb 5 11:54:57 example_system audit: [ID 702911 audit.notice] \
login - login ok  session 2870512630 by user as user:staff
```

Note - Although the default installation of `syslog` is `syslog`, you can use the `rsyslog` utility, a reliable and extended `syslog` implementation whose modular design supports features such as filtering, TCP, encryption, high-precision time-stamps, and output control.

A site can configure auditing to collect audit records in all formats. You can configure the systems at your site to use binary mode locally, to send binary files to a remote repository, and to use `syslog` mode. The following table compares binary audit records with `syslog` audit records.

TABLE 1 Comparison of Binary, Remote, and `syslog` Audit Records

Feature	Binary and Remote Records	`syslog` Records
Protocol	Binary – Writes to the file system Remote – Streams to a remote repository	If you use `rsyslog`, you can configure it to use UDP or TCP or RELP)
Data type	Binary	Text
Record length	No limit	Up to 1024 characters per audit record
Location	Binary – Stored in a `zpool` on the system Remote – Remote repository	Stored in a location that is specified in the `/etc/syslog.conf` file
How to configure	Binary – Set the `p_dir` attribute on the `audit_binfile` plugin Remote – Set the `p_hosts` attribute on the `audit_remote` plugin and make the plugin active	Make the `audit_syslog` plugin active and configure the `syslog.conf` file
How to read	Binary – Typically, in batch mode, browser output in XML Remote – Repository dictates the procedure	In real time or searched by scripts that you have created for `syslog` Plain text output
Completeness	Guaranteed to be complete and to appear in the correct order	A summary that is not guaranteed to be complete
Time stamp	Coordinated Universal Time (UTC)	Time on the system that is being audited

For more information about plugins and audit logs, refer to the following resources:

- `audit_binfile`(7) man page
- `audit_syslog`(7) man page
- `audit.log`(5) man page
- `ars`(7) man page
- "Configuring Audit Space for the Audit Trail and Audit Files" on page 93
- "How to Configure `syslog` for the `audit_syslog` Plugin" on page 96

About Binary Records

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria (`https://www.commoncriteriaportal.org`) audit requirements.

The `audit_binfile` plugin writes the records to a file system that you protect from snooping. On a single system, all binary records are collected and displayed in order. The UTC time stamp on binary logs enables accurate comparison when systems on one audit trail are distributed across time zones. The `praudit` command enables you to view the records in a browser in XML. You can also use scripts to parse the XML output.

The `audit_remote` plugin writes the records to a remote repository. The repository handles storage and post-selection.

The audit remote server also yields binary.

About `syslog` Audit Records

The default syslog implementation is `syslog`.

If you use `rsyslog`, it has its own configuration file, `/etc/rsyslog.conf`. The `rsyslog` records might provide greater convenience and flexibility than other logging options. For example, you can collect the `rsyslog` data from a variety of sources. Also, when you monitor `audit.notice` events in the `rsyslog.conf` file, the `rsyslog` utility logs an audit record summary with the current time stamp. You can use the same management and analysis tools that you have developed for `syslog` messages from a variety of sources, including workstations, servers, firewalls, and routers. The records can be viewed in real time, and can be stored on a remote system.

By using `rsyslog.conf` to store audit records remotely, you protect log data from alteration or deletion by an attacker. However, consider the following drawbacks to the `rsyslog` mode:

- `syslog` records sent using UDP or TCP are susceptible to network attacks such as denial of service and spoofed source addresses. When using TCP, `rsyslog` can be configured to use TLS which prevents spoofing of source addresses or GSS-API.
- The UDP protocol can drop packets or deliver packets out of order. Therefore, you should configure `rsyslog` to use the TCP protocol, either plain TCP `rsyslog` or RELP.
- The default maximum message size for `rsyslog` is 2048 characters. This maximum could cause some audit records to be truncated in the log. This size can be increased by modifying `MaxMessageSize` in the `rsyslog` configuration. See the `rsyslog` documentation at `https://www.rsyslog.com/doc/master/index.html` for further information.
- On a single system, `rsyslog` messages might not be displayed in order.
- Each audit record is stamped with the local system's date and time. Thus, you cannot rely on the time stamp to construct an audit trail for several systems.

Storing and Managing the Audit Trail

When the `audit_binfile` plugin is active, an *audit file system* holds audit files in binary format. A typical installation uses `/var/audit` and can use additional file systems. The contents of all audit file systems comprise the *audit trail*. Audit records are stored in these file systems in the following order:

- **Primary audit file system–** `/var/audit` functions as the default file system for audit files for a system

  ---

  Note - `/var/audit` is actually a symbolic link to the `/var/share` file system, which enables shared access as part of a root pool. However, Oracle Solaris treats `/var/audit` like a file system.

- **Secondary audit file systems –** File systems where the audit files for a system are placed at administrator discretion

The file systems are specified as arguments to the `p_dir` attribute of the `audit_binfile` plugin. A file system is not used until a file system that is earlier in the list is full. For an example with a list of file system entries, see “How to Create ZFS File Systems for Audit Files” on page 91.

Placing the audit files in the default audit root directory assists the audit reviewer when reviewing the audit trail. The `auditreduce` command uses the audit root directory to find all files in the audit trail. The `/var/audit` functions as the default audit root directory.

You can use the following options with the `auditreduce` command:

- `-M` option – Used to specify the audit files from a specific machine.
- `-S` option – Used to specify a different audit file system.

For examples of the use of the `auditreduce` command, see "How to Merge Audit Files From the Audit Trail" on page 116. For more information, see the `auditreduce`(8) man page.

The audit service provides commands to combine and filter files from the audit trail. The `auditreduce` command can merge audit files from the audit trail. The command can also filter files to locate particular events. The `praudit` command reads the binary files. Options to the `praudit` command provide output that is suitable for scripting and for browser display.

Ensuring Reliable Time Stamps

When you merge audit logs from several systems, the date and time on those systems must be accurate. Similarly, when you send audit logs to a remote system, the recording system and the repository system must have accurate clocks. The Network Time Protocol (NTP) keeps system clocks accurate and coordinated. For more information, see "Time-Related Services Key Tasks" in *Introduction to Oracle Solaris 11.4 Network Services* and the `ntpd`(8) man page.

Managing the Audit Remote Server

The audit remote server (ARS) is the counterpart of the `audit_remote` plugin. Data sent by the plugin can be captured, processed, and stored by ARS according to the server configuration.

After the `audit_remote` plugin is configured, the audit remote server (ARS) receives audit records. The audit records stream to the ARS over a protected connection and can be stored similarly to how they are stored locally.

ARS is delivered as a disabled audit component in Oracle Solaris systems. You must configure ARS before it can be used to process a remote audit trail. To configure ARS:

- Configure the underlying security mechanisms used for secure audit data transport. See the `audit_remote`(7) man page.
- Configure the audit remote subsystem using the `auditconfig -setremote` command. The configuration includes both a server configuration and a connection group configuration. A connection group is the sets of hosts sharing the same local storage parameters. For information and examples, see the `ars`(7) man page and the `auditconfig`(8) man page.

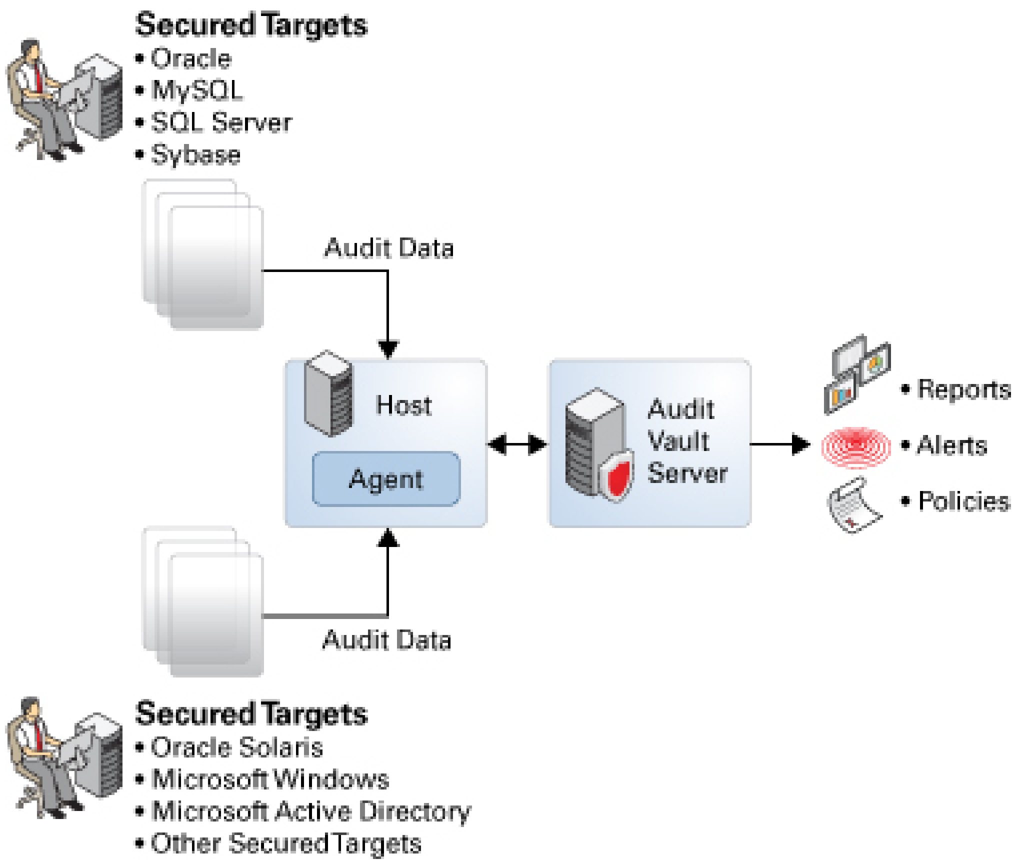
For more information, see "Audit Remote Server" on page 141 and the `ars(7)` man page. See also "How to Configure a Remote Repository for Audit Files" on page 99 and "How to Send Audit Files to a Remote Repository" on page 104.

Using Oracle Audit Vault and Database Firewall for Storage and Analysis of Audit Records

Audit records from an Oracle Solaris system can plug in to Oracle Audit Vault and Database Firewall, beginning with Oracle Audit Vault and Database Firewall release 12.1.0.0. Oracle Audit Vault and Database Firewall automates the consolidation and monitoring of audit data from Oracle and non-Oracle databases. You can then use Oracle Audit Vault and Database Firewall for analysis and reports of audited events on Oracle Solaris systems. For more information, see Oracle Audit Vault and Database Firewall (`https://www.oracle.com/database/technologies/security/audit-vault-firewall.html`).

The following figure shows how Oracle Audit Vault and Database Firewall collects Oracle Solaris audit records from designated secured targets. A secured target is any system that stores audit records or data.

FIGURE 2 Oracle Solaris and Audit Vault



A host system is designated to run the AV agent that communicates with Oracle Audit Vault and Database Firewall. The agent enables Oracle Audit Vault and Database Firewall to receive and process audit data from secured targets. The agent reads the audit records from a designated audit trail on the secured target. These audit records are encoded in the native binary format. The agent converts the data to a format parseable by Oracle Audit Vault and Database Firewall. Oracle Audit Vault and Database Firewall receives the data and generates reports for administrators and security managers as required.

The agent can be installed on a secured target instead of on a separate system. Multiple hosts with agents can also be configured to connect to the Audit Vault server. However, when registering secured targets, indicate a specific system with which the AV server communicates to obtain audit data.

To configure Oracle Audit Vault and Database Firewall to accept audit records from both Oracle Solaris secured targets and non Oracle Solaris secured targets, ensure that the agent is installed and activated on the designated host system. For more information, find the documentation at the bottom of the Oracle Audit Vault and Database Firewall (`https://www.oracle.com/database/technologies/security/audit-vault-firewall.html`) web site.

♦ ♦ ♦ CHAPTER 2

Planning for Auditing

This chapter describes how to plan the customization of the audit service for your Oracle Solaris installation. It covers the following topics:

Concepts in Planning Auditing

You want to be selective about what kinds of activities are audited. At the same time, you want to collect useful audit information. You also need to carefully plan who to audit and what to audit. If you are using the default `audit_binfile` plugin, note that audit files can quickly grow to fill the available space.

Planning an Audit Trail

If all your systems are within a single administrative domain,you can setup an audit policy that treats all the systems as a single-system image for auditing purposes.

To create a single-system image audit trail for these systems, follow these requirements:

- Use the same naming service for all systems.

  For correct interpretation of the audit records, `passwd`, `group`, and `hosts` must be consistent.
- Configure the audit service identically on all systems. For information about displaying and modifying the service settings, see the `auditconfig`(8) man page.
- Use the same `audit_warn`, `audit_event`, and `audit_class` files for all systems.

Note - Implementing an audit trail within a single administrative domain applies only to the `audit_binfile` plugin.

Refer to "How to Plan Who and What to Audit" on page 36 for additional considerations for enabling auditing on the systems.

Planning Auditing in Zones

If your system contains non-global zones, the zones can be audited as the global zone is audited, or the audit service for each non-global zone can be configured, enabled, and disabled separately. For example, you could audit only the non-global zones and not audit the global zone.

For a discussion of the trade-offs, see "Auditing on a System With Oracle Solaris Zones" on page 18.

The options described in this section are available when implementing auditing in zones.

Implementing One Audit Service for All Zones

A system that has installed non-global zones can run a single audit service in the global zones to audit all zones identically. Auditing all zones identically can create a single-image audit trail. A single-image audit trail occurs when you are using the `audit_binfile` or the `audit_remote` plugin, and all zones on a system are part of one administrative domain. The audit records can then be easily compared because the records in every zone are preselected with identical settings.

This configuration treats all zones as part of one system. The global zone runs the only audit service on a system and collects audit records for every zone. You customize the `audit_class` and `audit_event` files only in the global zone, then copy these files to every non-global zone.

Use the following guidelines when configuring a single audit service for all the zones:

- Use the same naming service for every zone.

  Note - If naming service files are customized in non-global zones, and `perzone` policy is not set, then careful use of the audit tools is required to select usable records. A user ID in one zone can refer to a different user from the same ID in a different zone.

- Enable the audit records to include the name of the zone.

  To include the zone name as part of the audit record, set the `zonename` policy in the global zone. The `auditreduce` command can then select audit events by zone from the audit trail. For an example, see the `auditreduce`(8) man page.

To plan a single audit service in the global zone to audit all zones identically, refer to "How to Plan Who and What to Audit" on page 36. Start with the first step. The global zone administrator must also set aside storage, as described in "How to Plan Disk Space for Audit Records" on page 38.

Implementing One Audit Service Per Zone

Choose to configure per-zone auditing if different zones use different naming service databases, or if zone administrators want to control auditing in their zones.

Note - To audit non-global zones, the `perzone` policy must be set but the audit service does not have to be enabled in the global zone. Non-global zone auditing is configured and its audit service is enabled and disabled separately from the global zone.

- When you configure per-zone auditing, you set the `perzone` audit policy in the global zone. If per-zone auditing is set before a non-global zone is first booted, auditing begins at the zone's first boot. To set audit policy, see "How to Configure Per-Zone Auditing" on page 85.
- Each zone administrator configures auditing for the zone.

  A non-global zone administrator can set all policy options except `perzone` and `ahlt`.
- Each zone administrator can enable or disable auditing in the zone.
- To generate records that can be traced to their originating zone during review, set the `zonename` audit policy.

Planning Auditing

The following outlines the major tasks that are required for planning disk space and which events to record.

- Determine who and what to audit – "How to Plan Who and What to Audit" on page 36
- Plan storage space for the audit trail – "How to Plan Disk Space for Audit Records" on page 38

- Plan transmission of the audit trail to a remote server – "How to Prepare to Stream Audit Records to Remote Storage" on page 39

▼ How to Plan Who and What to Audit

Before You Begin If you are implementing non-global zones, review "Planning Auditing in Zones" on page 34 before using this procedure.

To configure auditing, you must become an administrator who is assigned the Audit Configuration rights profile. To enable the audit service, you must become an administrator who is assigned the Audit Control rights profile . For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Determine the audit policy.**

   By default, only the `cnt` policy is enabled.

   Use the `auditconfig -lspolicy` command to see a description of available policy options.

   - For the effects of the policy options, see "Understanding Audit Policy" on page 40.
   - For the effect of the `cnt` policy, see "Audit Policies for Asynchronous and Synchronous Events" on page 142.
   - To set audit policy, see "How to Change Audit Policy" on page 57.

2. **Determine whether you want to add event-to-class mappings.**

   In almost all situations, the default mapping is sufficient. However, if you add new classes, you should add them to event-to-class mappings.

   For an example, see "How to Change an Audit Event's Class Membership" on page 63.

3. **Determine which audit classes to preselect.**

   The best time to add audit classes or to change the default classes is before users log in to the system.

   The audit classes that you preselect with the `-setflags` and `-setnaflags` options to the `auditconfig` command apply to all users and processes. You can preselect a class for success, for failure, or for both.

   For the list of audit classes, read the `/etc/security/audit_class` file.

4. **Determine user modifications to the system-wide preselections.**

   If you decide that some users should be audited differently from the system, you can modify the `audit_flags` security attribute for individual users or for a rights profile. Modify the user preselection mask for users whose audit flags are explicitly set or who are assigned a rights profile with explicit audit flags.

For the procedure, see "How to Configure a User's Audit Characteristics" on page 54. For which audit flag values are in effect, see "Order of Search for Assigned Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

5. **Decide how to manage the `audit_warn` email alias.**

   The `audit_warn` script is run whenever the audit system detects a situation that requires administrative attention. By default, the `audit_warn` script sends email to an `audit_warn` alias and sends a message to the console.

   To set up the alias, see "How to Configure the `audit_warn` Email Alias" on page 60.

6. **Decide in which format and where to collect audit records.**

   You have three choices.

   - By default, store binary audit records locally. The default storage directory is `/var/audit`. To further configure the `audit_binfile` plugin, see "How to Create ZFS File Systems for Audit Files" on page 91.
   - Stream binary audit records to a remote protected repository by using the `audit_remote` plugin. You must have a receiver for the records. For the requirements, see "Managing the Audit Remote Server" on page 28. For the procedure, see "How to Send Audit Files to a Remote Repository" on page 104.
   - Send audit record summaries to `syslog` by using the `audit_syslog` plugin. For the procedure, see "How to Configure `syslog` for the `audit_syslog` Plugin" on page 96.

     For a comparison of binary and `syslog` formats, see "Audit Logs" on page 24.

7. **Determine when to warn the administrator about shrinking disk space.**

   Note - This step applies only to the `audit_binfile` plugin.

   When disk space on an audit file system drops below the minimum free space percentage, or soft limit, the audit service switches to the next available audit directory. The service then sends a warning that the soft limit has been exceeded.

   To see how to set a minimum free space percentage, see Example 23, "Setting a Soft Limit for Warnings," on page 96.

8. **Decide what action to take when all the audit directories are full.**

   Note - This step applies only to the `audit_binfile` plugin.

In the default configuration, the `audit_binfile` plugin is active and the `cnt` policy is set. In this configuration, when the kernel audit queue is full, the system continues to work. The system counts the audit records that are dropped but does not record the events. For greater security, you can disable the `cnt` policy and enable the `ahlt` policy. The `ahlt` policy stops the system when an asynchronous event cannot be placed in the audit queue.

However, if the `audit_binfile` queue is full and the queue for another active plugin is not full, then the kernel queue will continue to send records to the plugin that is not full. When the `audit_binfile` queue can again accept records, the audit service will resume sending records to it.

For a discussion of the `cnt` and `ahlt` policy options, see "Audit Policies for Asynchronous and Synchronous Events" on page 142. To see how to configure these policy options, see Example 7, "Setting the `ahlt` Audit Policy Option," on page 59.

Note - The `cnt` or `ahlt` policy is not triggered if the queue for at least one plugin is accepting audit records.

Planning Disk Space for Audit Records

The `audit_binfile` plugin creates an audit trail. The audit trail requires dedicated file space. This space must be available and secure. The system uses the `/var/audit` file system for initial storage. You can configure additional audit file systems for audit files. The following procedure covers the issues that you must resolve when you plan for audit trail storage.

▼ How to Plan Disk Space for Audit Records

Before You Begin If you are implementing non-global zones, complete "Planning Auditing in Zones" on page 34 before using this procedure.

This procedure assumes that you are using the `audit_binfile` plugin.

1. **Determine how much auditing your site needs.**

   Balance your site's security needs against the availability of disk space for the audit trail.

   For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see "Controlling Auditing Costs" on page 43 and "Auditing Efficiently" on page 45.

For practical steps, see "Volume of Audit Records Is Large" on page 129, "How to Compress Audit Files on a Dedicated File System" on page 80, and Example 32, "Combining and Reducing Audit Files," on page 110.

2. **Determine which systems are to be audited and configure their audit file systems.**
   Create a list of all the file systems that you plan to use. For configuration guidelines, see "Storing and Managing the Audit Trail" on page 27 and the `auditreduce`(8) man page. To specify the audit file systems, see "Configuring Audit Space for the Audit Trail and Audit Files" on page 93.

3. **Synchronize the clocks on all systems.**
   For more information, see "Ensuring Reliable Time Stamps" on page 28.

Preparing to Stream Audit Records to Remote Storage

The `audit_remote` plugin sends the binary audit trail to an ARS in the same format as the `audit_binfile` plugin writes to the local audit files. The `audit_remote` plugin uses the `libgss` library to authenticate the ARS, and a GSS-API mechanism to protect the transmission with privacy and integrity. For reference, see "How the Kerberos Service Works" in *Managing Kerberos in Oracle Solaris 11.4*.

The only currently supported GSS-API mechanism is `kerberosv5`. For more information, see the `mech`(5) man page.

▼ How to Prepare to Stream Audit Records to Remote Storage

Note - If you have a Kerberos realm configured with an identified Audit Remote Server (ARS) and all audited systems within the realm, you can skip this procedure. The steps to configure the ARS and the audited systems are covered in "How to Configure a Remote Repository for Audit Files" on page 99 and "How to Send Audit Files to a Remote Repository" on page 104.

To verify whether a Kerberos realm is configured, issue the following command. The sample output indicates that Kerberos is not installed on the system.

```
$ pkg info kerberos-5/kdc
pkg: info: no packages matching these patterns are installed on the system.
```

Before You Begin This procedure assumes that you are using the `audit_remote` plugin. Also, you must be assigned the Software Installation rights profile. By default, the `root` role has this profile.

1. **Install the master KDC (Key Distribution Center) package.**

   You can use the system that will serve as the ARS, or you can use a nearby system. The ARS sends a significant amount of authentication traffic to the master KDC.

   ```
   $ pfexec pkg install pkg://solaris/security/kerberos-5/kdc
   ```

   On the master KDC, you use the Kerberos `kdcmgr` and `kadmin` commands to manage the realm. For more information, see the `kdcmgr(8)` and `kadmind(8)` man pages.

2. **On every audited system that will send audit records to the ARS, install the master KDC package.**

   ```
   $ pfexec pkg install pkg://solaris/security/kerberos-5/kdc
   ```

   This package includes the `kclient` command. On these systems, you run the `kclient` command to connect with the KDC. For more information, see the `kclient(8)` man page.

3. **Synchronize the clocks in the KDC realm.**

   If the clock skew is too big between the audited systems and the ARS, the attempt at connection will fail. After a connection is established, the local time on the ARS determines the names of the stored audit files, as described in “Conventions for Binary Audit File Names” on page 144.

   For more information about the clocks, see “Ensuring Reliable Time Stamps” on page 28.

Understanding Audit Policy

Audit policy determines the characteristics of the audit records for the local system. You use the `auditconfig` command to set these policies. For more information, see the `auditconfig(8)` man page.

To use the `auditconfig` command, you must be an administrator who is assigned the Audit Configuration rights profile. For more information, see “Using Your Assigned Administrative Rights” in *Securing Users and Processes in Oracle Solaris 11.4*.

Most audit policy options are disabled by default to minimize storage requirements and system processing demands. These options are properties of the audit service and determine the policies that are in effect at system boot. For more information, see the `auditconfig(8)` man page.

Use the following table to determine whether the needs of your site justify the additional overhead that results from enabling one or more audit policy options.

TABLE 2 Effects of Audit Policy Options

Policy Name	Description	Policy Considerations
`ahlt`	This policy applies to asynchronous events only. When disabled, this policy allows the event to complete without an audit record being generated. When enabled, this policy stops the system when the audit queue is full. Administrative intervention is required to clean up the audit queue, make space available for audit records, and reboot. This policy can be enabled only in the global zone. The policy affects all zones.	The disabled option is preferable when system availability is more important than security. The enabled option is preferable in an environment where security is paramount. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 142.
`arge`	When disabled, this policy omits environment variables of an executed program from the `execve` audit record. When enabled, this policy adds the environment variables of an executed program to the `execve` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 68. The enabled option is preferable when you are auditing a few users. The option is also useful when you are unsure about the environment variables that are being used in programs in the `ex` audit class.
`argv`	When disabled, this policy omits the arguments of an executed program from the `execve` audit record. When enabled, this policy adds the arguments of an executed program to the `execve` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. For a comparison, see "How to Audit All Commands by Users" on page 68. The enabled option is preferable when you are auditing a few users. The option is also useful when you have reason to believe that unusual programs in the `ex` audit class are being run.
`cnt`	When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because the audit queue is full. When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.	The disabled option is preferable in an environment where security is paramount. The enabled option is preferable when system availability is more important than security. For a fuller discussion, see "Audit Policies for Asynchronous and Synchronous Events" on page 142.
`group`	When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.	The disabled option usually satisfies requirements for site security. The enabled option is preferable when you need to audit the supplemental groups to which the subject belongs.

Policy Name	Description	Policy Considerations
`labeled_only`	When disabled, this policy has no effect. When enabled, this policy audits labeled files for read operations and prevents audit records from being written for read operations on unlabeled or `ADMIN_LOW` files.	The disabled option usually satisfies requirements for site security. The enabled option adds the label of labeled files and file systems to the audit record. Oracle Solaris labels are described in Chapter 3, "Labeling Files for Data Loss Protection" in *Securing Files and Verifying File Integrity in Oracle Solaris 11.4*.
`path`	When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.	The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a `path` token.
`perzone`	When disabled, this policy maintains a single audit configuration for a system. One audit service runs in the global zone. Audit events in specific zones can be located in the audit record if the `zonename` audit token was preselected. When enabled, this policy maintains a separate audit configuration, audit queue, and audit logs for each zone. An audit service runs in each zone. This policy can be enabled in the global zone only.	The disabled option is useful when you have no special reason to maintain a separate audit log, queue, and daemon for each zone. The enabled option is useful when you cannot monitor your system effectively by simply examining audit records with the `zonename` audit token.
`public`	When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit classes that contain read-only events include `fr`, `fa`, and `cl`. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.	The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.
`seq`	When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The `sequence` token holds the sequence number.	The disabled option is sufficient when auditing is running smoothly. The enabled option is preferable when the `cnt` policy is enabled. The `seq` policy enables you to determine when data was discarded. Alternatively, you can use the `auditstat` command to view dropped records.
`trail`	When disabled, this policy does not add a `trailer` token to audit records. When enabled, this policy adds a `trailer` token to every audit record.	The disabled option creates a smaller audit record. The enabled option clearly marks the end of each audit record with a `trailer` token. The `trailer` token is often used with the `sequence` token. The `trailer` token aids in the recovery of damaged audit trails.

Policy Name	Description	Policy Considerations
`zonename`	When disabled, this policy does not include a `zonename` token in audit records.	The disabled option is useful when you do not need to track audit behavior per zone.
	When enabled, this policy includes a `zonename` token in every audit record.	The enabled option is useful when you want to isolate and compare audit behavior across zones by post-selecting records according to zone.

Controlling Auditing 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 costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data

If you are using the default plugin, `audit_binfile`, you must also consider the storage cost of audit data.

Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant cost of auditing. Auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. Also, if you are using the `audit_binfile` plugin, audit administrators can move the post-selection tasks from the audited system to systems that are dedicated to analyzing audit data. Finally, unless kernel events are preselected, the audit service has no measurable impact on system performance.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records.

For records that are collected by the `audit_binfile` plugin, cost also includes the time that is required to archive the records and their supporting name service databases, and to keep the records in a safe place. Supporting databases include `groups`, `hosts`, and `passwd`.

The fewer records that you generate, the less time that is required to analyze the audit trail. The sections “Cost of Storage of Audit Data” on page 44 and “Auditing

Efficiently" on page 45 describe ways to audit efficiently. Efficient auditing reduces the amount of audit data while still providing enough coverage to achieve your site's security goals.

Cost of Storage of Audit Data

If you are using the `audit_binfile` plugin, storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of systems
- Amount of use
- Degree of traceability and accountability that is required

Because these factors vary from site to site, no formula can predetermine the amount of disk space to set aside for audit data storage. Use the following information as a guide.

- Understand the audit classes

  Before you configure auditing, you should understand the types of events that the classes contain. You can change the audit event-class mappings to optimize audit record collection.
- Preselect audit classes judiciously to reduce the volume of records that are generated.

  Full auditing, that is, with the `all` class, fills disk space quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute.

  For example, by omitting the `file_read` audit class, `fr`, you can significantly reduce audit volume. By choosing to audit for failed operations only, you can at times reduce audit volume. For example, by auditing for failed `file_read` operations, `-fr`, you can generate far fewer records than by auditing for all `file_read` events.
- If you are using the `audit_binfile` plugin, efficient audit file management is also important. For example, you can compress a ZFS file system that is dedicated to audit files.
- Develop a philosophy of auditing for your site.

  Base your philosophy on measures such as the amount of traceability that your site requires, and the types of users that you administer.

Auditing Efficiently

The following techniques can help you achieve your organization's security goals while auditing more efficiently.

- For as many audit classes as possible, preselect those classes only for users and roles, not system-wide.
- Randomly audit only a certain percentage of users at any one time.
- If the `audit_binfile` plugin is active, reduce the disk storage requirements for audit files by filtering, merging, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors.
  - `audit_syslog` plugin – You can extend management and analysis tools that you have already developed to handle the audit records in `syslog` files.
  - `audit_binfile` plugin – You can set up procedures to monitor the audit trail for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain systems in response to detection of unusual events.

    For example, you could write a script that does the following:

    1. Monitors the creation of audit files on the audited systems.
    2. Processes the audit files with the `tail` command.

       The piping of the output from the `tail -0f` command through the `praudit` command can yield a stream of audit records as the records are generated. For more information, see the `tail(1)` man page.
    3. Analyzes this stream for unusual message types or other indicators, and delivers the analysis to the auditor.

       Alternatively, the script can be used to trigger automatic responses.
    4. Constantly monitors the audit file systems for the appearance of new `not_terminated` audit files.
    5. Terminates outstanding `tail` processes when their files are no longer being written to.

♦ ♦ ♦ CHAPTER 3

Managing the Audit Service

This chapter provides procedures to help you configure and manage auditing on an Oracle Solaris system. The chapter covers the following tasks:

- "Default Configuration of the Audit Service" on page 47
- "Configuring the Audit Service" on page 50
- "Selecting What Is Audited" on page 68
- "Configuring the Audit Service in Zones" on page 83
- "Example: Configuring Oracle Solaris Auditing" on page 86

For an overview of the audit service, see Chapter 1, "About Auditing in Oracle Solaris". For planning suggestions, see Chapter 2, "Planning for Auditing". For reference information, see Chapter 7, "Auditing Reference".

Default Configuration of the Audit Service

The audit service has a default configuration and is immediately operational on the global zone after you install Oracle Solaris. No additional action is required to enable or configure the service to become usable. Run the following command to determine your default configuration:

```
$ auditconfig -getflags
configured user default audit flags = sstore(0x200031000,0x200031000)
active user default audit flags = sstore(0x200031000,0x200031000)
```

Because the service's default configuration has no performance impact on the system, disabling the service on performance grounds is not required.

Provided that you have the appropriate audit-related rights, such as those in the Audit Review Rights profile, you can review the audit logs. The logs are stored in `/var/audit`. You view these files by using the `praudit` and `auditreduce` commands. For more information, see "Displaying Audit Trail Data" on page 107.

`sstore` Audit Meta-Class

`sstore` is the audit meta-class which represents the group of events consumed by the Oracle Solaris StatsStore. The `sstore` meta-class is the default system-wide audit preselection mask.

For more information, see the `audit_class`(5) man page.

Displaying Audit Service Defaults

The audit service is regulated by the following parameters:

- Classes of attributable and non-attributable events
- Audit policy
- Audit plugins
- Queue controls

To display the audit service defaults, you typically use the `auditconfig -get*` subcommands. These subcommands display the current configuration of the parameter that is represented by the asterisk (*), such as `-getflags`, `-getpolicy`, or `-getqctrl`. To display information about classes for non-attributable events, use the `auditconfig -getnaflags` subcommand.

For more information about the `auditconfig` command, see the `auditconfig`(8) man page.

Note - To display the audit service configuration, you must become an administrator who is assigned the Audit Configuration or Audit Control rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

The following examples show the appropriate command syntax to use to display the default audit configuration settings.

EXAMPLE 1 Displaying the Default Class for Events

In this example, two subcommands display the preselected classes for attributable and non-attributable events respectively. To see which events are assigned to a class, and therefore which events are being recorded, run the `auditrecord -c` *class* command.

Display the preselected classes for attributable events.

```
$ pfbash ; auditconfig -getflags
active user default audit flags = sstore(0x1000,0x1000)
configured user default audit flags = sstore(0x1000,0x1000)
```

`sstore` is the audit meta-class. See "`sstore` Audit Meta-Class" on page 48. The format of the mask output is (*success*,*failure*).

Display the preselected class for non-attributable events.

```
$ auditconfig -getnaflags
active non-attributable audit flags = lo(0x1000,0x1000)
configured non-attributable audit flags = lo(0x1000,0x1000)
```

EXAMPLE 2 Displaying the Default Audit Policy

```
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
```

The *active* policy is the current policy, but the policy value is not being stored by the audit service. The *configured* policy is stored by the audit service, so the policy is restored when you restart the audit service.

EXAMPLE 3 Displaying the Default Audit Plugins

```
$ auditconfig -getplugin
Plugin: audit_binfile
Attributes: p_age=0h;p_dir=/var/audit;p_fsize=4M;p_minfree=1;

Plugin: audit_syslog (inactive)
Attributes: p_flags=;

Plugin: audit_remote (inactive)
Attributes: p_hosts=;p_retries=3;p_timeout=5;
```

The `audit_binfile` plugin is active by default.

Enabling and Disabling the Audit Service

The audit service is enabled by default. If the `perzone` audit policy is set, zone administrators must enable, refresh, or disable the audit service in each non-global zone as desired. If the `perzone` audit policy is not set, enabling, refreshing, or disabling the audit service from the global zone is effective for all non-global zones.

To disable or enable the audit service, you must become an administrator who is assigned the Audit Control rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

To disable the audit service, use the following command:

```
$ pfbash ; audit -t
```

To enable the audit service, use the following command:

```
$ audit -s
```

To verify that the audit service is running, use the following command:

```
$ auditconfig -getcond
audit condition = auditing
```

If the `perzone` audit policy is set, then you must perform this verification in the non-global zones where you enabled auditing.

For more information, see the `audit`(8) and `auditd`(8) man pages.

Configuring the Audit Service

Before you enable auditing on your network, you can modify the defaults to satisfy your site auditing requirements. Best practice is to customize your audit configuration as much as possible before the first users log in.

Configuring Audit With the `auditconfig` Subcommands

To configure the audit service, you typically use `auditconfig` subcommands. The configuration that is set with these subcommands applies to the whole system.

- `auditconfig -get*` displays the current configuration of the parameter that is represented by the asterisk (*), as shown in the examples of "Displaying Audit Service Defaults" on page 48.
- `auditconfig -set*` assigns a value to the parameter that is represented by the asterisk (*), such as `-setflags`, `-setpolicy`, or `-setqctrl`. To configure classes for non-attributable events, you use the `auditconfig setnaflags` subcommand.

- `auditconfig -conf` configures kernel audit event to class mappings. Runtime class mappings are changed to match those in the audit event to class database file.

Auditing Per User or Rights Profile

You can also customize auditing to apply to users or profiles rather than to the entire system. Audit class preselections for each user are specified by the `audit_flags` security attribute. These user-specific values, plus the preselected classes for the system, determine the user's audit mask, as described in "Process Audit Characteristics" on page 143.

By preselecting classes on a per user basis rather than on a per system basis, you can sometimes reduce the impact of auditing on system performance. Also, you might want to audit specific users slightly differently from the system.

To configure auditing that applies to users or profiles, you use the following commands:

- `userattr` displays the `audit_flags` value that is set for users. By default, users are audited for the system-wide settings only.
- `usermod -K` sets flags that apply to users.
- `profile` sets flags that apply to profiles.

For a description of the `userattr` command, see the `userattr`(1) man page. For a description of the `audit_flags` keyword, see the `user_attr`(5) and `audit_flags`(7) man pages.

Auditing Zones

If you have implemented zones, you can choose to audit all zones from the global zone or to audit non-global zones individually. For an overview, see "Auditing and Oracle Solaris Zones" on page 138. For planning, see "Planning Auditing in Zones" on page 34. For procedures, see "Configuring the Audit Service in Zones" on page 83.

New Feature – Auditing Events Temporarily

The `-t` option for the `auditconfig` command displays or sets the active values of the running system, without displaying or setting the persistent values of the audit service. This release provides the following new subcommands for the `-t` option:

- `-t -getflags` – Displays the user default audit preselection flags.
- `-t -getnaflags` – Displays the non-attributable audit flags.

- `-t -setflags audit_flags` – Sets the default user audit preselection flags. The default preselection flags are combined with the user's specific audit flags to form the user's audit preselection mask.
- `-t -setnaflags audit_flags` – Sets the non-attributable audit flags. Non-attributable audit flags define which classes of events are to be audited when the action cannot be attributed to an authenticated user. Failed logins, for example, are non-attributable.

For more information, see the `auditconfig`(8) and `audit_flags`(7) man pages.

New Feature – Refreshing the `auditset` SMF Service After Changing Event-Class Mappings

When you make changes to the `/etc/security/audit_event` or `/etc/security/audit_class` configuration files for the audit service, you must refresh the `svc:/system/auditset:default` service. Administrators who are assigned the Audit Configuration rights profile and the Service Configuration rights profile first configure auditing then refresh the service as follows:

```
$ pfbash ; auditconfig -conf
$ svcadm refresh auditset:default
```

New Feature – Auditing Verified Boot

The audit service tracks whether verified boot is enabled and working on your system.

- Auditing for the boot event indicates whether verified boot is enabled by checking whether `boot_policy` has one of the following verified boot-specific values specified:
  - `none (1)` – No `elfsign` signature verification is performed when a module is loaded.
  - `warning (2)` – If an `elfsign` signature is invalid or missing, prints warning but permits the module to load.
  - `enforce (3)` – If an `elfsign` signature is invalid or missing, prints error and prevents the module from loading.
- Auditing for the module load event, `AUE_MODLOAD`, indicates whether verified boot verified the module, and if that verification passed or failed. If the output of the `auditreduce | praudit` command includes notifications about `AUE_MODLOAD` audit events, verified boot is working.

For information about verified boot, see "Using Verified Boot" in *Securing Systems and Attached Devices in Oracle Solaris 11.4*.

New Feature – `auditstat` Command Extended

The `auditconfig -getstat` and `auditconfig -setstat` subcommands have been removed in this release. The `auditstat` command provides the same information. See the `auditstat(8)` man page.

Audit Configuration Task Map

The following task map points to the procedures for configuring auditing. All tasks are optional.

Task	Description	For Instructions
Select which events are audited.	Preselects system-wide audit classes. If an event is attributable, then all users are audited for this event.	"How to Preselect Audit Classes" on page 53
Select which events are audited for specific users.	Sets user-specific differences from the system-wide audit classes.	"How to Configure a User's Audit Characteristics" on page 54
Specify audit policy.	Defines additional audit data that your site requires.	"How to Change Audit Policy" on page 57
Create the `audit_warn` email alias.	Defines who receives email warnings when the audit service needs attention.	"How to Configure the `audit_warn` Email Alias" on page 60
Configure audit logs.	Configures the location of audit records for each plugin.	"Configuring Local Audit Logs" on page 90
Add audit classes.	Reduces the number of audit records by creating a new audit class to hold critical events.	"How to Add an Audit Class" on page 61
Change event-to-class mappings.	Reduces the number of audit records by changing the event-class mapping.	"How to Change an Audit Event's Class Membership" on page 63

▼ How to Preselect Audit Classes

You can preselect audit classes that contain the events that you want to monitor. Events that are not in preselected classes are not recorded.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Determine the current preselected classes.**

```
$ pfbash ; auditconfig -getflags
...

$ auditconfig -getnaflags
,,,
```

For an explanation of the output, see "Displaying Audit Service Defaults" on page 48.

2. **Preselect the attributable classes.**

   For example, the following command audits the events in the `login/logout`, `process start/stop`, and `file write` classes for success and for failure.

   ```
   $ auditconfig -setflags lo,ps,fw
   user default audit flags = ps,lo,fw(0x101002,0x101002)
   ```

   Note - The `auditconfig -setflags` command *replaces* the current preselection, so you must specify all classes that you want to preselect.

3. **Preselect the non-attributable classes.**

   The `na` class contains PROM, boot, and non-attributable mounts, among other events.

   ```
   $ auditconfig -setnaflags lo,na
   non-attributable audit flags = lo,na(0x1400,0x1400)
   ```

   `lo` and `na` are the only useful arguments to the `-setnaflags` option.

   Note - The `auditconfig -setnaflags` command *replaces* the current preselection, so you must specify all classes that you want to preselect.

▼ How to Configure a User's Audit Characteristics

The user-specific audit characteristics that you set with this procedure are combined with the preselected classes for the system. Together they determine the user's audit mask, as described in "Process Audit Characteristics" on page 143.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **(Optional) Display the audit classes that are currently selected for existing users.**

a. **Display the list of users.**

```
$ who
adoe    pts/1        Oct 10 10:20 (:0.0)
adoe    pts/2        Oct 10 10:20 (:0.0)
jdoe    pts/5        Oct 13 13:20 (:0.0)
jdoe    pts/6        Oct 13 13:20 (:0.0)
...
```

b. **Display each user's `audit_flags` attribute value.**

```
$ userattr audit_flags adoe
$ userattr audit_flags jdoe
```

2. **Set the audit flags in the `user_attr` or in the `prof_attr` database.**

   For example, you can create a rights profile that defines the rights of a subset of your users. Users who are assigned that rights profile are audited identically.

   - **To set audit flags for a user, use the `usermod` command.**

     ```
     # usermod -K audit_flags=fw:no jdoe
     ```

     The format of the `audit_flags` keyword is *always-audit*:*never-audit*.

always-audit	Lists the audit classes that are audited for this user.
never-audit	Lists the audit classes that are never audited for the user, even if these audit events are audited system-wide.

     To specify multiple audit classes, separate the classes with commas. For more information, see the `audit_flags(7)` man page.

   - **To set audit flags for a rights profile, use the `profiles` command.**

     ```
     # profiles -p "System Administrator"
     profiles:System Administrator> set name="Audited System Administrator"
     profiles:Audited System Administrator> set always_audit=fw,as
     profiles:Audited System Administrator> end
     profiles:Audited System Administrator> exit
     ```

     When you assign the Audited System Administrator rights profile to a user or a role, that user or role is audited for those flags, subject to search order as described in "Order of Search for Assigned Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Example 4 Changing Which Events Are Audited for One User

This example shows the audit preselection mask for all users.

```
$ auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No user except the administrator is logged in.

To lessen the impact of the `AUE_PFEXEC` audit event on system resources, the administrator does not audit this event at the system level. Rather, the administrator preselects the `pf` class for a user, `jdoe`. The `pf` class is created in Example 11, "Creating a New Audit Class," on page 63.

```
# usermod -K audit_flags=pf:no jdoe
```

The `userattr` command shows the addition.

```
$ userattr audit_flags jdoe
pf:no
```

When the user `jdoe` logs in, `jdoe`'s audit preselection mask is a combination of the `audit_flags` values with the system default values. `289` is the PID of `jdoe`'s login shell.

```
$ auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = ss,pf,lo(0x0100000008011000,0x0100000008011000)
terminal id (maj,min,host) = 242,511,example1(192.0.2.171)
audit session id = 103203403
```

Example 5 Modifying Audit Preselection Exception for One User

This example shows the audit preselection mask for all users.

```
$ auditconfig -getflags
active user default audit flags = ss,lo(0x11000,0x11000)
configured user default audit flags = ss,lo(0x11000,0x11000)
```

No users except the administrator are logged in.

The administrator decides not to collect failed `ss` events for the `jdoe` user.

```
# usermod -K audit_flags=^-ss:no jdoe
```

The `userattr` command shows the exception.

```
$ userattr audit_flags jdoe
^-ss:no
```

When the user jdoe logs in, jdoe's audit preselection mask is a combination of the audit_flags values with the system default values. 289 is the PID of jdoe's login shell.

```
$ auditconfig -getpinfo 289
audit id = jdoe(1234)
process preselection mask = +ss,lo(0x11000,0x1000)
terminal id (maj,min,host) = 242,511,example1(192.0.2.171)
audit session id = 103203403
```

Example 6 Creating a Rights Profile for a Group of Users

The administrator wants all administrative rights profiles at the site to explicitly audit the pf class. For every rights profile that is going to be assigned, the administrator creates a site-specific version in LDAP that includes audit flags.

First, the administrator clones an existing rights profile, then changes the name and adds audit flags.

```
# profiles -p "Network Wifi Management" -S ldap
profiles: Network Wifi Management> set name="Wifi Management"
profiles: Wifi Management> set desc="Audited wifi management"
profiles: Wifi Management> set audit_always=pf
profiles: Wifi Management> exit
```

After repeating this procedure for every rights profile that is going to be used, the administrator lists the information in the Wifi Management profile.

```
$ profiles -p "Wifi Management" -S ldap info
name=Wifi Management
desc=Audited wifi management
auths=solaris.network.wifi.config
help=RtNetWifiMngmnt.html
always_audit=pf
```

▼ How to Change Audit Policy

You might change default audit policy to record detailed information about audited commands, to add a zone name to every record, or to satisfy other site security requirements.

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **View the current audit policy.**

```
$ pfbash
$ auditconfig -getpolicy
configured audit policies = argv,cnt
active audit policies = argv,cnt
```

For an explanation of the output, see "Displaying Audit Service Defaults" on page 48.

2. **View the available policy options.**

```
$ auditconfig -lspolicy
policy string    description:
ahlt             halt machine if it can not record an async event
all              all policies for the zone
arge             include exec environment args in audit recs
argv             include exec command line args in audit recs
cnt              when no more space, drop recs and keep a cnt
group            include supplementary groups in audit recs
none             no policies
path             allow multiple paths per event
perzone          use a separate queue and auditd per zone
public           audit public files
seq              include a sequence number in audit recs
trail            include trailer token in audit recs
zonename         include zonename token in audit recs
```

Note - The `perzone` and `ahlt` policy options can be set only in the global zone. For the trade-offs to using a particular policy option, see "Understanding Audit Policy" on page 40.

---

3. **Enable or disable selected audit policy options.**

```
$ auditconfig [ -t ] -setpolicy [prefix]policy[,policy...]
```

`-t`	Optional. Creates a temporary, or *active*, policy. You might set a temporary policy for debugging or testing purposes. A temporary policy is in effect until the audit service is refreshed, or until the policy is modified by the `auditconfig -setpolicy` command, or the system is rebooted.
prefix	A *prefix* value of + adds the list of policies to the current policy. A *prefix* value of - removes the list of policies from the current policy. Without a prefix, the audit policy is reset. This option enables you to retain current audit policies.
policy	Selects the policy to be enabled or to be disabled.

Example 7 Setting the `ahlt` Audit Policy Option

In this example, strict site security requires the `ahlt` policy.

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

The plus sign (+) before the `ahlt` policy adds the policy to current policy settings. Without the plus sign, the `ahlt` policy replaces all current audit policies.

Example 8 Setting a Temporary Audit Policy

In this example, the `ahlt` audit policy is configured. For debugging, the administrator adds the `trail` audit policy to the active policy (`+trail`) temporarily (`-t`). The `trail` policy aids in the recovery of damaged audit trails.

```
$ auditconfig -setpolicy ahlt
$ auditconfig -getpolicy
configured audit policies = ahlt
active audit policies = ahlt
$ auditconfig -t -setpolicy +trail
$ auditconfig -getpolicy
configured audit policies = ahlt
active audit policies = ahlt,trail
```

The administrator disables the `trail` policy when the debugging is completed.

```
$ auditconfig -setpolicy -trail
$ auditconfig -getpolicy
configured audit policies = ahlt
active audit policies = ahlt
```

Refreshing the audit service by running the `audit -s` command also removes this temporary policy, plus any other temporary values in the audit service.

Example 9 Setting the `perzone` Audit Policy

In this example, the `perzone` audit policy is added to the existing policy in the global zone. The `perzone` policy setting is stored as a permanent property, so `perzone` policy is in effect during the session and when the audit service is restarted. For the zones, the policy is available at the next zone boot, or when the administrator runs `zlogin` and then runs the `audit -s` command.

```
$ auditconfig -getpolicy
configured audit policies = cnt
active audit policies = cnt
```

```
$ auditconfig -setpolicy +perzone
$ auditconfig -getpolicy
configured audit policies = perzone,cnt
active audit policies = perzone,cnt
```

Example 10 Collecting Audit Records for External Auditors

In this example, the administrator is collecting audit records to satisfy external auditors' requirements. The administrator decides to use an Audit Remote Server (ARS) to collect information about administrative activities. The administrator also collects actions that cannot be attributed to a user, such as booting.

The administrator sets up ARS. In addition to auditing the `cusa` class, the administrator adds policies to the audit configuration.

```
$ auditconfig -setflags cusa
user default audit flags = ex,xa,ua,as,ss,ap,lo,ft(0x80475080,0x80475080)
$ auditconfig -setpolicy ahlt,argv,arge
$ auditconfig -getpolicy
configured audit policies = ahlt,arge,argv
active audit policies = ahlt,arge,argv
$ auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

When the administrator enables the `audit_remote` plugin and refreshes the audit service, the records are collected.

▼ How to Configure the `audit_warn` Email Alias

The `/etc/security/audit_warn` script generates mail to notify the administrator of audit incidents that might need attention. You can customize the script and you can send the mail to an account other than `root`.

Note - If the `perzone` policy is set, the non-global zone administrator must configure the `audit_warn` email alias in the non-global zone.

Before You Begin The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator who is assigned the `solaris.admin.edit/etc/security/audit_warn` authorization can modify the alias.

- An administrator with the Mail Management rights profile can run the `newaliases` command.

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

- **Replace or redirect the `audit_warn` alias:**

  - **Replace the `audit_warn` email alias in the `ADDRESS` line with another email account in the `audit_warn` script.**

    For example:

```
#ADDRESS=audit_warn            # standard alias for audit alerts
ADDRESS=audadmin               # role alias for audit alerts
```

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 139.

  - **Redirect the `audit_warn` email to another mail account.**

    a. **You could add the alias to the local `/etc/mail/aliases` file or to the `mail_aliases` database in the name service.**

    In the following sample `/etc/mail/aliases` entry, the `root` and `audadmin` email accounts were added as members of the `audit_warn` email alias.

```
audit_warn: root,audadmin
```

    b. **Rebuild the random access database for the `aliases` file.**

```
$ pfexec newaliases
/etc/mail/aliases: 14 aliases, longest 10 bytes, 156 bytes total
```

▼ How to Add an Audit Class

When you create your own audit class, you can place into it just those audit events that you want to audit for your site. This strategy can reduce the number of records that are collected and reduce noise in your audit trail.

When you add the class on one system, copy the change to all systems that are being audited. Best practice is to create audit classes before the first users log in.

For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 139.

Tip - In Oracle Solaris you can create your own package that contains files and replace the Oracle Solaris packages with your site-customized files. When you set the `preserve` attribute to `true` in your package, the `pkg` subcommands, such as `verify`, `fix`, `revert`, and so on, will run relative to your packages. For more information, see the `pkg`(1) and `pkg`(7) man pages.

Before You Begin The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator who is assigned the `solaris.admin.edit/etc/security/audit_class` authorization can modify the file.
- An administrator with the Audit Configuration rights profile can run the `auditconfig` command.
- An administrator with the Service Configuration rights profile can run the `svcadm` command.

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

1. **(Optional) Save a backup copy of the `audit_class` file.**

```
$ cp /etc/security/audit_class /etc/security/audit_class.orig
```

2. **Choose free bits in the upper 8 bits for your unique entry.**

3. **Verify which bits are available for customer use in the `/etc/security/audit_class` file.**

4. **Add new entries to the `audit_class` file.**

   Each entry has the following format:

   `0x`*64bitnumber*`:`*flag*`:`*description*

   For a description of the fields, see the `audit_class(5)` man page. For the list of existing classes, read the `/etc/security/audit_class` file.

5. **Refresh the audit service and update the runtime mappings.**

```
$ pfbash ; svcadm refresh svc:/system/auditset:default
$ auditconfig -conf
```

The first command refreshes the audit service. The second command changes the runtime class mappings to match those in the audit event to class database file.

Example 11 Creating a New Audit Class

This example creates a class to hold administrative commands that are executed in a role. The added entry to the `audit_class` file is as follows:

```
0x0100000000000000:pf:profile command
```

The entry creates the new `pf` audit class. Example 12, "Mapping Existing Audit Events to a New Class," on page 64 shows how to populate the new audit class.

Troubleshooting If you have customized the `audit_class` file, make sure that any audit flags that are assigned directly to users or rights profiles are consistent with the new audit classes. Errors occur when an `audit_flags` value is not a subset of the `audit_class` file.

▼ How to Change an Audit Event's Class Membership

You might want to change an audit event's class membership to reduce the size of an existing audit class, or to place the event in a class of its own.

For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 139.

Caution - Never comment out events in the `audit_event` file. This file is used by the `praudit` command to read binary audit files. Archived audit files might contain events that are listed in the file. Also, never remove any existing audit classes.

When you reconfigure audit event-class mappings on one system, copy the change to all systems that are being audited. Best practice is to change event-class mappings before the first users log in.

Tip - In Oracle Solaris you can create your own package that contains files and replace the Oracle Solaris packages with your site-customized files. When you set the `preserve` attribute to `true` in your package, the `pkg` subcommands, such as verify, fix, revert, and so on, will run relative to your packages. For more information, see the `pkg(1)` and `pkg`(7) man pages.

Before You Begin You must become an administrator who is assigned the Audit Configuration and the Service Configuration rights profiles. You must also be assigned the `solaris.admin.edit/etc/security/audit_event` authorization. By default, only the `root` role has these rights. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **(Optional) Save a backup copy of the `audit_event` file.**

```
$ cp /etc/security/audit_event /etc/security/audit_event.orig
```

2. **Change the class to which particular events belong by changing the *class-list* of the events.**
Each entry has the following format:

number:*name*:*description*:*class-list*

number	The audit event ID.
name	The name of the audit event.
description	Typically, the system call or executable that triggers the creation of an audit record.
class-list	A comma-separated list of audit classes.

3. **Refresh the kernel events.**

```
$ pfbash ; auditconfig -conf
Configured 283 kernel events.
```

4. **Refresh the audit service.**

```
$ svcadm refresh system/auditset:default
```

Example 12 Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new `pf` class that was created in Example 11, "Creating a New Audit Class," on page 63. By default, the `AUE_PFEXEC` audit event is mapped to several audit classes. By creating the new class, the administrator can audit `AUE_PFEXEC` events without auditing the events in the other classes.

```
$ grep pf /etc/security/audit_class
0x0100000000000000:pf:profile command
$ grep AUE_PFEXEC /etc/security/audit_event
116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as
$ pfedit /etc/security/audit_event
```

```
#116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as
116:AUE_PFEXEC:execve(2) with pfexec enabled:pf
$ auditconfig -setflags lo,pf
user default audit flags = pf,lo(0x0100000000001000,0x0100000000001000)
```

New Feature – Annotating Reason for Access in the Audit Record

In Oracle Solaris 11.4, system administrators can configure and track annotations in audit records, annotations that explain why actions were performed in the system.

Administrators can configure users and roles to provide annotations when:

- Logging into a system
- Assuming a particular role
- Executing a rights profile, including authenticated rights profiles

Annotations should explain the reason for activity performed on a system. For example, a user might, during login, provide an annotation that states "Fix Ticket 134567". This annotation is recorded in each audit record associated with that particular authentication.

Note - Annotation is sometimes referred to as Session Annotation.

Configuring Annotation

Annotation may be configured for specific users and roles, or for all users on a system. To configure annotations, an administrator must be granted the User Security rights profile.

Configuring Annotation for Specific Users

To configure annotation for specific users or roles, administrators use the new `annotation` extended user attribute in the Oracle Solaris rights model.

TABLE 3 Annotation Attribute Added to Rights Model

Description	Annotation	Rights Man Page
Manage users and roles interactively	Supports `-K annotation=` option for CLI, also supports `Session annotation` options in interactive menu	`useradm`(8)

Description	Annotation	Rights Man Page
Annotate per user	Supports `-K annotation=` option	`useradd`(8), `usermod`(8)
Annotate per a role	Supports `-K annotation=` option	`roleadd`(8), `rolemod`(8)
Annotation described in extended user attributes database	Includes key value pair for `annotation`	`user_attr`(5)
Annotate per rights profiles	`annotation` value can be set. Annotation can be used for authenticated rights profiles.	`profiles`(1)

The annotation values can be set to *yes*, *no*, or *optional*. The default value is *no*.

For further information, see Chapter 3, "Assigning Rights in Oracle Solaris" in *Securing Users and Processes in Oracle Solaris 11.4*.

EXAMPLE 13 Adding an Annotation Requirement

The following command modifies a user, `bob`, adding a requirement for annotation:

```
$ pfexec usermod -K annotation=yes bob
```

Given the new requirement, the user logins as follows:

```
login: bob
Password:*******
Session Annotation: Customer Ticket 134567
```

The annotation, `Customer Ticket 134567`, is associated with all audited actions performed during that session and is included in those audit records.

If you changed `annotation=yes` to `annotation=optional` in this example, the user would be prompted but not required to annotate their login.

Configuring Annotation for All Users

Administrators can configure annotation for all authentication actions in an Oracle Solaris instance by adding a key value pair, `annotation=yes/no/optional`, in the `policy.conf` file. This default applies to all users who were not configured with an explicit `annotation` extended attribute. See the `policy.conf`(5) man page and "policy.conf File" in *Securing Users and Processes in Oracle Solaris 11.4*.

PAM Supports Annotation of Logins

The pluggable authentication module (PAM) supports annotation customization on a per-service basis by providing options to the `pam_unix_cred` service module. This module allows administrators to customize or suppress the default `Session Annotation:` prompt. See the `pam_unix_cred`(7) man page and *Managing Authentication in Oracle Solaris 11.4*.

Tracking Annotations in an Audit Trail

Auditing includes a new `annotation` token. When a user provides annotation entries, that annotation is included in audit records for any auditable actions that the user or role performed during that session. See the annotation token definition in the `audit.log`(5) man page.

For example, the following audit record includes an annotation.

```
header,116,2,su,,system1,2016-02-05 11:41:36.100-08:00
    subject,jand,up,staff,up,staff,101736,2438860677,61323 22 lethe
    return,success,0
    annotation,Customer Ticket 134567
    zone,global
```

Using the `-o annotation=text` option to the `auditreduce` command, administrators can select only those records that include specified annotation text. The text can be a regular expression. For information about regular expressions, see the `regex`(7) man page.

EXAMPLE 14 Filtering Audit Records for Annotations

The following command filters the audit records for any records that contain an annotation.

```
$ pfbash ; cd /var/audit/audit_summary
$ auditreduce -o annotation='[.]*'
```

This example uses a regular expression, `'[.]*'`, that searches for any entries. Alternately, you could filter for any records that includes a specific annotation such as "Ticket 134567".

```
$ cd /var/audit/audit_summary
$ auditreduce -o annotation="Ticket 134567"
```

See the `auditreduce`(8) man page and "Selecting Audit Events to Be Displayed" on page 109.

Selecting What Is Audited

The following task map points to procedures to configure auditing that is specific to your needs.

TABLE 4 Customizing Auditing Task Map

Task	Description	For Instructions
Audit everything that a user does on the system.	Audit one or more users for every command.	"How to Audit All Commands by Users" on page 68
Audit significant events.	Audit administrative commands, system access, and other significant events.	"How to Audit Significant Events in Addition to Login/ Logout" on page 70
Change the audit events that are being recorded and have the change affect existing sessions.	Update a user's preselection mask.	"How to Update the Preselection Mask of Logged In Users" on page 77
Locate modifications to particular files in audited events.	Audit file modifications, then use the `auditreduce` command to find particular files.	"How to Find Audit Records of Changes to Specific Files" on page 71
Specify files or directories to be audited.	Set per-object logging for audit events.	"Specifying Files or Directories to Be Audited" on page 74
Use less file system space for audit files.	Use ZFS quotas and compression.	"How to Compress Audit Files on a Dedicated File System" on page 80
Remove audit events from the `audit_event` file.	Correctly update the `audit_event` file.	"How to Prevent the Auditing of Specific Events" on page 78

▼ How to Audit All Commands by Users

As part of site security policy, some sites require audit records of all commands that are run by the `root` account and administrative roles. Some sites can require audit records of all commands by all users. Additionally, sites can require that the command arguments and environment be recorded.

Before You Begin To preselect audit classes and set audit policy, you must become an administrator who is assigned the Audit Configuration rights profile. By default, the `root` role has these rights. Only the `root` role can change how users are audited.

1. **Display user level event information for `lo` and `ex` classes.**

   The `ex` class audits all calls to the `exec()` and `execve()` functions.

The `lo` class audits logins, logouts, and screen locks. The following sample output lists the events in the `ex` and `lo` classes.

```
$ auditconfig -lsevent | egrep " lo |,lo|lo,"
AUE_login                       6152 lo login - local
AUE_logout                      6153 lo logout
AUE_telnet                      6154 lo login - telnet
AUE_rlogin                      6155 lo login - rlogin
AUE_rshd                        6158 lo rsh access
AUE_su                          6159 lo su
...


$ auditconfig -lsevent | egrep " ex |,ex |ex,"
AUE_EXECVE                        23 ex,ps execve(2)
AUE_PFEXEC                        116, ex,ps,ua,as execve(2) with pfexec enabled
```

2. **Audit the `cusa` class for administrators.**

   - **To audit these classes for administrative roles, modify the roles' security attributes.**

     In the following example, `root` is a role. The site has created three roles, `sysadm`, `auditadm`, and `netadm`. All roles are audited for the success and failure of events in the `cusa` class.

     ```
     # rolemod -K audit_flags=cusa:no root

     # rolemod -K audit_flags=cusa:no sysadm

     # rolemod -K audit_flags=cusa:no auditadm

     # rolemod -K audit_flags=cusa:no netadm
     ```

   - **To audit these classes for all users, set the system-wide flags.**

     ```
     # auditconfig -setflags lo,ex
     ```

     The output appears similar to the following:

     ```
     header,129,2,AUE_EXECVE,,mach1,2010-10-14 12:17:12.616 -07:00
     path,/usr/bin/ls
     attribute,100555,root,bin,21,320271,18446744073709551615
     subject,jdoe,root,root,root,root,2486,50036632,82 0 mach1
     return,success,0
     ```

3. **Specify additional information to be recorded about command use.**

- **To record the arguments to commands, add the `argv` policy.**

  ```
  $ pfbash ; auditconfig -setpolicy +argv
  ```

  The `exec_args` token records the command arguments. The following example wraps the lines for display purposes.

  ```
  header,151,2,AUE_EXECVE,,mach1,2010-10-14 12:26:17.373 -07:00
  path,/usr/bin/ls
  attribute,100555,root,bin,21,320271,18446744073709551615
  exec_args
  ,2,ls,/etc/security
  subject,jdoe,root,root,root,root,2494,50036632,82 0 mach1
  return,success,0
  ```

- **To record the environment in which the command is run, add the `arge` policy.**

  ```
  $ auditconfig -setpolicy +arge
  ```

  The `exec_env` token records the command environment. The following example wraps lines for display purposes.

  ```
  header,1460,2,AUE_EXECVE,,mach1,2010-10-14 12:29:39.679 -07:00
  path,/usr/bin/ls
  attribute,100555,root,bin,21,320271,18446744073709551615
  exec_args,2,ls,/etc/security
  exec_env
  ,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
  LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
  HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8,
  PRINTER=example-dbl,...,_=/usr/bin/ls
  subject,jdoe,root,root,root,root,2502,50036632,82 0 mach1
  return,success,0
  ```

▼ How to Audit Significant Events in Addition to Login/Logout

Use this procedure to audit administrative commands, system access, and other significant events as specified by your site security policy.

Note - The examples in this procedure might not be sufficient to satisfy your security policy.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Audit all uses of privileged commands by users who are assigned administrative rights profiles and roles by adding the `cusa` audit class to their preselection mask.**

   ```
   # usermod -K audit_flags=cusa:no username
   ```

   ```
   # rolemod -K audit_flags=cusa:no rolename
   ```

   The audit classes that the `cusa` meta-class includes are listed in the `/etc/security/audit_class` file.

2. **Record the arguments to audited commands.**

   ```
   # auditconfig -setpolicy +argv
   ```

3. **(Optional) Record the environment in which audited commands are executed.**

   ```
   # auditconfig -setpolicy +arge
   ```

   Note - This policy option can be useful when troubleshooting.

See Also An alternative to this procedure is to audit all successful or failed events that use privilege. For information, see "New Feature – Per-Privilege Logging of Audit Events" on page 73.

▼ How to Find Audit Records of Changes to Specific Files

If your goal is to log file writes against a limited number of files, such as `/etc/passwd` and the files in the `/etc/default` directory, you can use the `auditreduce` command to locate the files.

Note - Use this task to search for specific files from existing audited events. To pre-select files to directories to be audited, see "Specifying Files or Directories to Be Audited" on page 74.

Before You Begin The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator with the Audit Configuration rights profile can run the `auditconfig` command.

- An administrator with the Audit Review rights profile can run the `auditreduce` command.
- Only the `root` role can assign audit flags.

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

1. **Perform one of the following steps to audit file changes.**
   - Audit the `fw` class.

     Adding the `fw` class to the audit flags of a user or role generates fewer records than adding this class to the system-wide audit preselection mask. Perform one of the following steps:
     - Add the `fw` class to specific roles.

       ```
       # rolemod -K audit_flags=fw:no root
       # rolemod -K audit_flags=fw:no sysadm
       # rolemod -K audit_flags=fw:no auditadm
       # rolemod -K audit_flags=fw:no netadm
       ```
     - Add the `fw` class to the system-wide flags.

       ```
       # auditconfig -getflags
       active user default audit flags = lo(0x1000,0x1000)
       configured user default audit flags = lo(0x1000,0x1000)

       # auditconfig -setflags lo,fw
       user default audit flags = lo,fw(0x1002,0x1002)
       ```

     ```
     # rolemod -K audit_flags=fw:no root
     # rolemod -K audit_flags=fw:no sysadm
     # rolemod -K audit_flags=fw:no auditadm
     # rolemod -K audit_flags=fw:no netadm
     ```
   - Audit successful file-writes.

     Auditing successes generates fewer records than auditing failures and successes. Perform the following steps:

     Add the `+fw` flag to specific roles.

     ```
     # rolemod -K audit_flags=+fw:no root
     # rolemod -K audit_flags=+fw:no sysadm
     # rolemod -K audit_flags=+fw:no auditadm
     # rolemod -K audit_flags=+fw:no netadm
     ```

2. **Obtain the audit records for specific files with the `auditreduce` command.**

   ```
   # auditreduce -o file=/etc/passwd,/etc/default -O filechg
   ```

The `auditreduce` command searches the audit trail for all occurrences of the `file` argument. The command creates a binary file with the suffix `filechg` which contains all records that include the path of the files of interest. See the `auditreduce(8)` man page for the syntax of the `-o file=` *pathname* option.

3. **Read the `filechg` file with the `praudit` command.**

```
# praudit *filechg
```

New Feature – Per-Object Logging of Audit Events

Starting in this release, administrators with the appropriate privileges can set ACL entries to audit access attempts for specific files or specific directories.

Previously, administrators could determine every time a particular user or group accessed files by looking for those files in the audit logs, but they could not specify in advance which files or directories were to be audited. Now, administrators can configure a specific set of files or directories to be audited based on the access attempts for those files.

In addition, previously when an NFS client or SMB client accessed files on a remote Oracle Solaris server, this activity could not be audited. Starting in this release, local, NFS, and SMB access or failed access to a ZFS file or directory may result in the generation of an audit record depending on that file or directory's audit ACL entries.

For information, see "Specifying Files or Directories to Be Audited" on page 74.

New Feature – Per-Privilege Logging of Audit Events

The `admhist` command uses a new `pe` audit class that automatically tracks, through the lifetime of a process, the successful use of privilege. The audit record shows the privileges that enabled the process to change the system's configuration.

Note - For information about the `admhist` command, see "New Feature – Viewing a Summary of Audit Records" on page 122.

By default, only successful use of privilege is monitored.

To include failed use of privilege in your audit reports, use the following `auditconfig` subcommands to modify which privileges are monitored and whether to monitor success or failure:

- `-setfprivs` – Sets the privileges to monitor for failure
- `-setsprivs` – Sets the privileges to monitor for success
- `-getfprivs` – Displays the privileges to monitor for failed use
- `-getsprivs` – Displays the privileges to monitor for successful use

Note - You can set the privileges temporarily with the `-t` option as follows:

```
$ auditconfig -setsprivs privileges -t
```

Similarly, to display only the temporary settings, add `-t` to the `getsprivs` or `getfprivs` options.

To add specific privileges to the existing settings, include a plus as follows:

```
...setsprivs +privileges
```

To remove specific privileges, include a minus as follows:

```
...setsprivs -privileges
```

Use the following command to view successful events and failed events that you are auditing with `pe`:

```
$ auditreduce -m AUE_CMD_PRIVS | praudit
```

For more information, see the `auditconfig(8)` and `auditreduce(8)` man pages.

Specifying Files or Directories to Be Audited

Administrators with the appropriate privileges can set ACL entries to audit access attempts for specific files or specific directories.

The following rights profiles enable administrators to view and modify audit ACLs:

- Object Access Management
- Media Backup
- Media Restore

Local, NFS, and SMB access or failed access to a ZFS file or directory may result in the generation of an audit record, depending on that file or directory's audit ACL entries.

You cannot view or modify the Audit ACEs of a ZFS ACL without the `file_audit` privilege.

For remote callers to be able to view or set audit ACL entries:

- NFS client callers must be root and must have root access on the server, for example, where the client host has been given root access on the server NFS share. Note that this NFS option has security vulnerabilities and is not recommended.
- SMB client callers must be in an SMB group, with a SID, which has an Administrator, Backup, or Restore privilege. Backup and restore operations may require reading and setting ACL's which contain audit ACEs.

Caution - Unprivileged users can perform some tasks that affect the ACL for files and could lose audit information for those files. For example, users could rename another file to replace an existing file that has an audit ACL, thus losing the ACL information. Given this limitation, administrators might want to create default audit ACLs on directories in order to better monitor files in those directories.

In addition, some interfaces may update files without preserving the ACL, including the `passwd` command and other commands that affect password management, such as the `useradd` command. Administrators should use the appropriate commands when updating system files in order to preserve the ACL, using commands such as `chmod`, `pfedit`, `remove`, `rename`, or `vim`.

Setting Audit ACL Entries

The procedures for setting audit ACL entries have been built into existing procedures for modifying ACL entries.

Use the following command syntax to set Trivial ACL audit entries:

```
chmod [options] A[index]{+|=}owner@ |group@ |everyone@: \
  access-permissions/...[:inheritance-flags]: \
successful_access|failed access:audit  file ...

chmod [options] A-owner@, group@, everyone@: \
  access-permissions/...[:inheritance-flags]: \
successful_access|failed access:audit  file ...
```

As shown in this syntax, the `chmod` command includes the `audit` option. The command also includes a variable specifying that successful access attempts, failed access attempts, or both will be audited.

Note - Audit ACL entries can be set using either trivial ACL format or non-trivial format. For a detailed explanation of these formats, see "ACL Entry Descriptions" in *Securing Files and Verifying File Integrity in Oracle Solaris 11.4*.

EXAMPLE 15 Setting Audit ACL Entries

In this example, a trivial ACL exists on a directory named `dir1`. Run the following command to set an audit requirement for that directory.

```
# chmod A+everyone@:read_data/read_attributes/read_acl:failed_access:audit dir1
```

This command specifies that whenever a person in the `everyone` group tries to access `dir1` and fails, that access failure is recorded in the audit log.

To view revised ACL settings for this file, run the following command.

```
# ls -v
total 1
drwxr-xr-x   2 foo     staff          2 Feb  1 19:28 dir1
     0:everyone@:list_directory/read_data/read_attributes/read_acl
         :failed_access:audit
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/read_xattr/write_xattr/execute/delete_child
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
         /synchronize:allow
     2:group@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
     3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

Viewing Per-Object Logs

Almost all of the records that can be generated by per-object auditing are existing events in existing classes, so you can search as you usually do by class and then scan for per-file events. Use the `auditreduce` command to select records by audit classes, including the syntax described in the `audit_flags(7)` man page to select failed events or successful events. To search for audit records related to a specific file, use the `auditreduce` command with the `-o` option and the `file=pathname` variable. For instructions, see "Selecting Audit Events to Be Displayed" on page 109.

Note - The only new events for per-object auditing are events for reading file attributes and writing file attributes. These events are not included in any audit class.

For further information, see the auditreduce(8) and audit_flags(7) man pages.

▼ How to Update the Preselection Mask of Logged In Users

This procedure describes how to audit users who are already logged in for changes to the system-wide audit preselection mask. You can typically accomplish this task by instructing the users to log out and to log back in. Alternatively, in a role that is assigned the Process Management rights profile, you can manually terminate active sessions with the `kill` command. The new sessions will inherit the new preselection mask.

However, terminating user sessions could be impractical. As an alternative, you can use the `auditconfig` command to dynamically change each logged-in user's preselection mask.

This procedure assumes that you changed the system-wide audit preselection mask from `lo` to `lo,ex` by running the following command:

```
$ auditconfig -setflags lo,ex
```

Before You Begin You must become an administrator who is assigned the Audit Configuration rights profile. To terminate user sessions, you must become an administrator who is assigned the Process Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **List the regular users who are logged in and their process IDs.**

```
$ who -a
jdoe   - vt/2        Jan 25 07:56  4:10   1597   (:0)
jdoe   + pts/1       Jan 25 10:10   .     1706   (:0.0)
...
jdoe   + pts/2       Jan 25 11:36  3:41   1706   (:0.0)
```

2. **For later comparison, display each user's preselection mask.**

```
$ auditconfig -getpinfo 1706
audit id = jdoe(1234)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 9426,65559,mach1(192.0.2.234)
audit session id = 103203403
```

3. **Modify the appropriate preselection mask by running one or more of the following commands:**

- **For a specific process:**

  ```
  $ auditconfig -setpmask 1706 lo,ex
  ```

- **For a specific user:**

  ```
  $ auditconfig -setumask jdoe lo,ex
  ```

- **For a specific session:**

  ```
  $ auditconfig -setsmask 103203403 lo,ex
  ```

4. **Verify that the preselection mask for the user has changed.**

   For example, check a process that existed before you changed the mask.

   ```
   $ auditconfig -getpinfo 1706
   audit id = jdoe(1234)
   process preselection mask = ex,lo(0x40001000,0x40001000)
   terminal id (maj,min,host) = 9426,65559,mach1(192.0.2.234)
   audit session id = 103203403
   ```

▼ How to Prevent the Auditing of Specific Events

For maintenance purposes, sometimes you want to prevent user-level events from being audited. The event number must be greater than 2047.

Before You Begin The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator who is assigned the `solaris.admin.edit/etc/security/audit_event` authorization can modify the file.
- An administrator with the Audit Configuration rights profile can run the `auditconfig` command.
- An administrator with the Service Configuration rights profile can run the `svcadm` command.

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

1. **Change the class of the event to the `no` class.**

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 139.

For example, the *STAT events are kernel events. Events 6214 and 6215 belong to the user administration, ua, class.

```
## audit_event file
...
87:AUE_MSGCTL_STAT:msgctl(2) - IPC_STAT command:ip
94:AUE_SHMCTL_STAT:shmctl(2) - IPC_STAT command:ip
101:AUE_SEMCTL_STAT:semctl(2) - IPC_STAT command:ip
...
6214:AUE_kadmind_auth:authenticated kadmind request:ua
6215:AUE_kadmind_unauth:unauthenticated kadmind req:ua
...
```

Change these events to the no class.

```
## audit_event file
...
87:AUE_MSGCTL_STAT:msgctl(2) - IPC_STAT command:no
94:AUE_SHMCTL_STAT:shmctl(2) - IPC_STAT command:no
101:AUE_SEMCTL_STAT:semctl(2) - IPC_STAT command:no
...
6214:AUE_kadmind_auth:authenticated kadmind request:no
6215:AUE_kadmind_unauth:unauthenticated kadmind req:no

...
```

If the ip and ua classes are currently being audited, existing sessions will still audit these events. To stop these events from being audited, you must update the users' preselection masks by following the instructions in "How to Update the Preselection Mask of Logged In Users" on page 77.

Caution - Never comment out events in the audit_event file. This file is used by the praudit command to read binary audit files. Archived audit files might contain events that are listed in the file.

---

2. **Refresh the kernel events.**

```
$ pfexec auditconfig -conf
Configured 283 kernel events.
```

3. **Refresh the audit service.**

```
$ pfexec svcadm refresh system/auditset:default
```

See Also

- "How to Change an Audit Event's Class Membership" on page 63
- Example 12, "Mapping Existing Audit Events to a New Class," on page 64

▼ How to Compress Audit Files on a Dedicated File System

Audit files can grow large. You can set an upper limit to the size of a file, as shown in Example 20, "Limiting File Size for the `audit_binfile` Plugin," on page 95. In this procedure, you use compression to reduce the size.

Before You Begin The `root` role can perform every task in this procedure.

If administrative rights are distributed in your organization, note the following:

- An administrator who is assigned the `solaris.admin.edit/etc/security/audit_class` authorization can modify the file.
- An administrator with the Audit Control rights profile can refresh the audit service.
- An administrator with the ZFS File System Management rights profile can configure ZFS file systems.
- An administrator with the ZFS Storage Management rights profile can create storage pools.

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

1. **Dedicate a ZFS file system for audit files.**

   For the procedure, see "How to Create ZFS File Systems for Audit Files" on page 91.

2. **Compress the ZFS storage pool.**

   You can compress the audit file system in two different ways. After the audit service is refreshed, the compression ratio is displayed.

   In the following examples, the ZFS pool `auditp/auditf` is the dataset.

   - **Use the default compression algorithm.**

     ```
     $ pfbash ; zfs set compression=on auditp/auditf
     $ audit -s
     ```

```
$ zfs get compressratio auditp/auditf
NAME           PROPERTY       VALUE  SOURCE
auditp/auditf  compressratio  4.54x  -
```

- **Use a higher compression algorithm.**

  ```
  $ zfs set compression=gzip-9 auditp/auditf
  $ zfs get compression auditp/auditf
  NAME           PROPERTY     VALUE     SOURCE
  auditp/auditf  compression  gzip-9    local
  ```

  The `gzip-9` compression algorithm results in files that occupy one-third less space than the default compression algorithm, `lzjb`. For more information, see Chapter 7, "Managing Oracle Solaris ZFS File Systems" in *Managing ZFS File Systems in Oracle Solaris 11.4*.

3. **Refresh the audit service.**

   ```
   $ audit -s
   ```

4. **(Optional) Verify the new compression setting.**

   For example, if you used the higher compression algorithm, the information would be similar to the following:

   ```
   $ zfs get compressratio auditp/auditf
   NAME           PROPERTY       VALUE   SOURCE
   auditp/auditf  compressratio  16.89x  -
   ```

▼ How to Audit FTP and SFTP File Transfers

The FTP service creates logs of its file transfers. The SFTP service, which runs under the `ssh` protocol, can be audited by preselecting the `ft` audit class. Logins to both services can be audited.

● **Perform one of the following depending on whether you want to audit SFTP or FTP.**

- To log `sftp` access and file transfers, edit the `ft` class.

  The `ft` class includes the following SFTP transactions:

  ```
  $ auditrecord -c ft
  file transfer: chmod ...
  file transfer: chown ...
  ```

```
file transfer: get ...
file transfer: mkdir ...
file transfer: put ...
file transfer: remove ...
file transfer: rename ...
file transfer: rmdir ...
file transfer: session start ...
file transfer: session end ...
file transfer: symlink ...
file transfer: utimes
```

- To record access to the Professional File Transfer Protocol (FTP) server, ensure that you are auditing the `lo` class.

  As the following sample output indicates, logging in to and out of the `proftpd` daemon generates audit records.

  ```
  $ auditrecord -c lo | more
  ...
  FTP server login
  program     proftpd              See in.ftpd(8)
  event ID    6165                 AUE_ftpd
  class       lo                   (0x0000000000001000)
  header
  subject
  [text]                       error message
  return

  FTP server logout
  program     proftpd              See in.ftpd(8)
  event ID    6171                 AUE_ftpd_logout
  class       lo                   (0x0000000000001000)
  header
  subject
  return
  ...
  ```

See Also For information about how to log FTP commands and file transfers, use the `man` command to view the `proftpd`(8) man page.

For the available logging options, read ProFTPD Logging (`http://www.proftpd.org/docs/howto/Logging.html`).

Configuring the Audit Service in Zones

The audit service audits the entire system, including audit events in zones. A system that has installed non-global zones can audit all zones identically, or can configure auditing per zone. For more information, see "Planning Auditing in Zones" on page 34.

When you audit the non-global zones exactly as the global zone is audited, the non-global zone administrators might not have access to the audit records. Also, the global zone administrator can modify the audit preselection masks of users in non-global zones.

When you audit the non-global zones individually, the audit records are visible to the non-global zone and to the global zone from the non-global zone root.

▼ How to Configure All Zones Identically for Auditing

This procedure enables audits of every zone identically. This method requires the least computer overhead and administrative resources.

Before You Begin You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Configure the global zone for auditing.**

   Complete the tasks in "Configuring the Audit Service" on page 50, with the following exceptions:

   - Do not enable the `perzone` audit policy.
   - Set the `zonename` policy. This policy adds the name of the zone to every audit record.

     ```
     # auditconfig -setpolicy +zonename
     ```

2. **If you modified audit configuration files, copy them from the global zone to every non-global zone.**

   If you modified the `audit_class` or `audit_event` file, copy it in one of two ways:

   Note - The non-global zone must be running.

   - **Mount the changed `audit_class` and `audit_event` files as a loopback file system (`lofs`).**

a. **From the global zone, halt the non-global zone.**

```
# zoneadm -z non-global-zone halt
```

b. **Create a read-only loopback mount for every audit configuration file that you modified in the global zone.**

```
# zonecfg -z non-global-zone
zone: add fs
zone/fs: set special=/etc/security/audit-file
zone/fs: set dir=/etc/security/audit-file
zone/fs: set type=lofs
zone/fs: add options [ro,nodevices,nosetuid]
zone/fs: commit
zone/fs: end
zone: exit
#
```

c. **To make the changes effective, boot the non-global zone.**

```
# zoneadm -z non-global-zone boot
```

Later, if you modify an audit configuration file in the global zone, you reboot each zone to refresh the loopback-mounted files in the non-global zones.

- **Copy the files.**

a. **From the global zone, list the `/etc/security` directory in each non-global zone.**

```
# ls /zone/zonename/root/etc/security/
```

b. **Copy the changed `audit_class` and `audit_event` files to each zone's `/etc/security` directory.**

```
# cp /etc/security/audit-file /zone/zonename/root/etc/security/audit-file
```

Later, if you change one of these files in the global zone, you must copy the changed file to the non-global zones.

Example 16 Mounting Audit Configuration Files as Loopback Mounts in a Zone

In this example, the system administrator has modified the `audit_class`, `audit_event`, and `audit_warn` files.

The `audit_warn` file is read in the global zone only, so does not have to be mounted into the non-global zones.

On this system, `machine1`, the administrator has created two non-global zones, `machine1–webserver` and `machine1–appserver`. The administrator has finished modifying the audit configuration files. If the administrator later modifies the files, the zone must be rebooted to re-read the loopback mounts.

```
# zoneadm -z machine1-webserver halt
# zoneadm -z machine1-appserver halt
# zonecfg -z machine1-webserver
webserver: add fs
webserver/fs: set special=/etc/security/audit_class
webserver/fs: set dir=/etc/security/audit_class
webserver/fs: set type=lofs
webserver/fs: add options [ro,nodevices,nosetuid]
webserver/fs: commit
webserver/fs: end
webserver: add fs
webserver/fs: set special=/etc/security/audit_event
webserver/fs: set dir=/etc/security/audit_event
webserver/fs: set type=lofs
webserver/fs: add options [ro,nodevices,nosetuid]
webserver/fs: commit
webserver/fs: end
webserver: exit
#

# zonecfg -z machine1-appserver
appserver: add fs
appserver/fs: set special=/etc/security/audit_class
appserver/fs: set dir=/etc/security/audit_class
appserver/fs: set type=lofs
appserver/fs: add options [ro,nodevices,nosetuid]
appserver/fs: commit
appserver/fs: end
appserver: exit
```

When the non-global zones are rebooted, the `audit_class` and `audit_event` files are read-only in the zones.

▼ How to Configure Per-Zone Auditing

This procedure enables separate zone administrators to control the audit service in their zone. For the complete list of policy options, see the `auditconfig(8)` man page.

Before You Begin To configure auditing, you must become an administrator who is assigned the Audit Configuration rights profile. To enable the audit service, you must become an administrator who is assigned the Audit Control rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **In the global zone, configure auditing.**

   a. **Complete the tasks in "Configuring the Audit Service" on page 50.**

   b. **Add the `perzone` audit policy.**

      For the command, see Example 9, "Setting the `perzone` Audit Policy," on page 59.

   Note - You are not required to enable the audit service in the global zone.

2. **In each non-global zone that you plan to audit, configure the audit files.**

   a. **Complete the tasks in "Configuring the Audit Service" on page 50.**

   b. **Do not add the `perzone` or `ahlt` policy to the non-global zone.**

3. **Enable auditing in your zone.**

```
myzone$ pfexec audit -s
```

Example 17 Disabling Auditing in a Non-Global Zone

This example works if the `perzone` audit policy is set. The zone administrator of the `noaudit` zone disables auditing for that zone.

```
noauditzone$ pfexec auditconfig -getcond
audit condition = auditing
noauditzone$ pfexec audit -t
noauditzone$ pfexec auditconfig -getcond
audit condition = noaudit
```

Example: Configuring Oracle Solaris Auditing

This section provides an example of how you configure and implement Oracle Solaris auditing. The example assumes you have distributed the rights to manage auditing to different administrators.

The example begins with the configuration of different attributes of the service according to specific needs and requirements. After configuration is completed, the audit service is started to implement the configuration settings. Each time that you need to revise an existing audit configuration to accommodate new requirements, follow the same sequence of actions in this example.

1. Configure the audit parameters.
2. Refresh the audit service.
3. Verify the new audit configuration.

- Add a temporary policy.

  ```
  $ auditconfig -t -setpolicy +zonename
  $ auditconfig -getpolicy
  configured audit policies = ahlt,arge,argv,perzone
  active audit policies = ahlt,arge,argv,perzone,zonename
  ```
- Specify the plugin attributes.
  - For the `audit_binfile` plugin, remove the `qsize` value.

    ```
    $ auditconfig -getplugin audit_binfile
    Plugin: audit_binfile
    Attributes: p_dir=/audit/sys1.1,/var/audit;
    p_minfree=2;p_fsize=4G;
    Queue size: 200
    $ auditconfig -setplugin audit_binfile  "" 0
    $ auditconfig -getplugin audit_binfile
    Plugin: audit_binfile
    Attributes: p_dir=/audit/sys1.1,/var/audit
    p_minfree=2;p_fsize=4G;
    ```
  - For the `audit_syslog` plugin, specify that the successful login and logout events and failed executables be sent to `syslog`. The `qsize` for this plugin is set to 150.

    ```
    $ auditconfig -setplugin audit_syslog active p_flags=+lo,-ex 150
    $ auditconfig -getplugin audit_syslog
    auditconfig -getplugin audit_syslog
    Plugin: audit_syslog
    Attributes: p_flags=+lo,-ex;
    Queue size: 150
    ```
  - Do not configure or use the `audit_remote` plugin.
- Refresh the audit service and verify the configuration.
  - The temporary `zonename` policy is no longer set.

    ```
    $ audit -s
    ```

```
$ auditconfig -getpolicy
configured audit policies = ahlt,arge,argv,perzone
active audit policies = ahlt,arge,argv,perzone
```

- The `audit_binfile` plugin does not have a specified queue size. The `audit_syslog` plugin has a specified queue size.

```
$ auditconfig -getplugin
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;

Plugin: audit_syslog
Attributes: p_flags=+lo,-ex;
Queue size: 150
...
```

New Feature – Restricting Access to Audit Records With File Labeling

The Oracle Solaris 11.4 release enables you to configure systems that enforce company security policy in software by using labels. Using an Oracle Solaris label policy enables you to assign these labels to existing or new file systems that contain sensitive data, and to assign a set of trusted users the ability to access the files based on the users' clearances. For an introduction to labeling,see Chapter 3, “Labeling Files for Data Loss Protection” in *Securing Files and Verifying File Integrity in Oracle Solaris 11.4*.

This labeling feature can increase security during auditing. Administrators can use labeling to restrict access to audit files. A labeled audit trail reduces access to the audit trail, including reducing access to the contents of higher-labeled processes. For further information and instructions, see “How to Create a Labeled Audit Trail” in *Securing Files and Verifying File Integrity in Oracle Solaris 11.4*.

♦ ♦ ♦ CHAPTER 4

Configuring the Formats of Audit Logs and Where They Are Stored

This chapter provides procedures to help you configure the format of audit logs and their location. This chapter covers the following topics:

- "New Feature – Flexible Per-Plugin Configuration of Audit Classes" on page 89
- "Configuring Local Audit Logs" on page 90
- "Configuring Remote Repositories for Audit Logs" on page 99

For examples about filtering the audit logs and locating events in them from an Oracle Solaris system, see Chapter 5, "Viewing Audit Records".

New Feature – Flexible Per-Plugin Configuration of Audit Classes

In the Oracle Solaris 11.4 release, administrators have more flexibility regarding how and where they want their audit data recorded. Administrators can now specify which system-wide audit classes are written to each configured audit plugin. Although this feature was already available for some plugins. All plugins now have this option.

Administrators can filter out unwanted events from the configured system-wide defaults on a per-plugin basis by using the `p_flags` option to the `auditconfig -setplugin` command.

For example, the following command sends `lo` events only to `syslog`:

```
$ auditconfig -setplugin audit_syslog p_flags=lo
```

In addition, the Audit Remote Server (ARS) now supports specifying a set of audit flags on a per-connection group basis. By using the `p_flags` option to the `auditconfig -setremote` command, administrators can specify which audit classes, on a per-connection group basis, are audited.

For example, the following command specifies only the audit events from the `lo` and `ex` audit classes:

```
$ auditconfig -setremote group foo p_flags=lo,ex
```

For more information, see the `auditconfig`(8) man page.

Configuring Local Audit Logs

Two audit plugins, `audit_binfile` and `audit_syslog`, can create local audit logs. The following tasks explain how to configure these logs.

Note - For information about using the `audit_remote` plugin, see "Configuring Remote Repositories for Audit Logs" on page 99.

Configuring logs for the `audit_binfile` plugin is optional. The logs for other plugins must be configured by an administrator.

Configuring the `audit_binfile` and `audit_syslog` Plugins

The following task map points to the procedures for configuring audit logs for the `audit_binfile` and `audit_syslog` plugins.

TABLE 5 Configuring Audit Logs Task Map

Task	Description	For Instructions
Add local storage for the `audit_binfile` plugin.	Creates additional disk space for the audit files and protects them with file permissions.	"How to Create ZFS File Systems for Audit Files" on page 91
Assign storage for the `audit_binfile` plugin.	Identifies directories for binary audit records.	"Configuring Audit Space for the Audit Trail and Audit Files" on page 93
Configure storage for the `audit_syslog` plugin.	Enables you to stream audit events in text format to `syslog`.	"How to Configure `syslog` for the `audit_syslog` Plugin" on page 96

▼ How to Create ZFS File Systems for Audit Files

This procedure shows how to create a ZFS pool for audit files, as well as the corresponding file systems and mount point. By default, `/var/audit` holds audit files for the `audit_binfile` plugin.

Before You Begin You must become an administrator who is assigned the ZFS File System Management and ZFS Storage Management rights profiles. The latter profile enables you to create storage pools. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Determine the amount of disk space that is required.**

   How much auditing you require dictates the disk space requirements.

   Note - The default class preselection creates files in `/var/audit` that grow by about 80 bytes for every recorded instance of an event in the `lo` class, such as a login, logout, or role assumption.

   ---

2. **Create a mirrored ZFS storage pool.**

   The `zpool create` command creates a storage pool, that is, a container for the ZFS file systems. For more information, see Chapter 1, "Introducing the Oracle Solaris ZFS File System" in *Managing ZFS File Systems in Oracle Solaris 11.4*.

   For example, create the `auditp` pool from two disks, `c3t1d0` and `c3t2d0`, and mirror them.

   ```
   $ zpool create auditp mirror c3t1d0 c3t2d0
   ```

3. **Create a ZFS file system and mount point for the audit files.**

   You create the file system and mount point with one command. At creation, the file system is mounted.

   For example, create the `/audit` mount point for the `auditf` file system.

   ```
   $ zfs create -o mountpoint=/audit auditp/auditf
   ```

4. **Create a ZFS file system for the audit files.**

   For example, create an unencrypted ZFS file system for the `sys1` system.

   ```
   $ zfs create -p auditp/auditf/sys1
   ```

5. **(Optional) Create additional file systems for audit files.**

   One reason to create additional file systems is to prevent audit overflow. You can set a ZFS quota per file system, as shown in Step 7. The `audit_warn` email alias notifies you when each quota is reached. To free space, you can move the closed audit files to a remote server.

For example:

```
$ zfs create -p auditp/auditf/sys1.1

$ zfs create -p auditp/auditf/sys1.2
```

6. **Compress the audit files in the pool.**

   Typically, compression is set in ZFS at the file system level. However, in this example, because all the file systems in this pool contain audit files, compression is set at the top-level dataset for the pool.

   ```
   $ zfs set compression=on auditp
   ```

   See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in *Managing ZFS File Systems in Oracle Solaris 11.4*.

7. **Set quotas.**

   You can set quotas at the parent file system, the descendant file systems, or both. If you set a quota on the parent audit file system, quotas on the descendant file systems impose an additional limit.

   a. **Set a quota on the parent audit file system.**

      In the following example, when both disks in the `auditp` pool reach the quota, the `audit_warn` script notifies the audit administrator.

      ```
      $ zfs set quota=510G auditp/auditf
      ```

   b. **Set a quota on the descendant audit file systems.**

      In the following example, when the quota for the `auditp/auditf/`*system* file system is reached, the `audit_warn` script notifies the audit administrator.

      ```
      $ zfs set quota=170G auditp/auditf/sys1

      $ zfs set quota=170G auditp/auditf/sys1.1

      $ zfs set quota=165G auditp/auditf/sys1.2
      ```

8. **For a large pool, limit the size of the audit files.**

   By default, an audit file can grow to the size of the pool. For manageability, limit the size of the audit files. See Example 20, "Limiting File Size for the `audit_binfile` Plugin," on page 95.

Example 18 Creating an Encrypted File System for Audit Archiving

To comply with site security requirements, the administrator performs the following steps:

1. Creates, if necessary, a new ZFS pool to store the encrypted audit logs.
2. Generates an encryption key.
3. Creates the audit file system with encryption turned on to store the audit logs, as well as sets the mount point.
4. Configures auditing to use the encrypted directory.
5. Refreshes the audit service to apply the new configuration settings.

```
$ zpool create auditp mirror disk1  disk2


$ pktool genkey keystore=file outkey=/filename keytype=aes keylen=256

$ zfs create -o encryption=aes-256-ccm \
-o keysource=raw,file:///filename \
-o compression=on -o mountpoint=/audit auditp/auditf

$ auditconfig -setplugin audit_binfile p_dir=/audit/

$ audit -s
```

You must back up and protect the file where the key is stored, such as *filename* in the example.

When the administrator creates additional file systems under the `auditf` file system, these descendant file systems are also encrypted.

Configuring Audit Space for the Audit Trail and Audit Files

You can use the attributes for the `audit_binfile` plugin to assign additional disk space to the audit trail.

You must become an administrator who is assigned the Audit Configuration rights profile to set these attribute values. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

The attributes to the `audit_binfile` plugin are described in `OBJECT ATTRIBUTES` section of the `audit_binfile`(7) man page.

`p_dir` attribute	Specifies where the audit files will be created. The directories are listed in the order in which they are to be used.
`p_minfree` attribute	Defines the percentage of free space that the audit system requires before the audit daemon invokes the audit_warn script.
`p_fsize` attribute	Defines the maximum size that an audit file can become before it is automatically closed and a new audit file is opened. The format of the `p_fsize` value can be specified as an exact value in bytes or in a human-readable form with a suffix of B, K, M, G, T, P, E, Z (for bytes, kilobytes, megabytes, gigabytes, terabytes, petabytes, exabytes, or zettabytes, respectively). Suffixes of KB, MB, GB, TB, PB, EB, and ZB are also accepted.

See the following examples showing how to configure the audit space.

Caution - When you have finished making configuration change, you must refresh the audit service.

The `auditconfig -setplugin` command sets the *configured* value. This value is a property of the audit service, so it is restored when the service is refreshed or restarted. The configured value becomes *active* when the audit service is refreshed or restarted. For information about configured and active values, see the `auditconfig(8)` man page.

```
$ audit -s
```

EXAMPLE 19 Adding Directories to the Audit Trail

In this example, the `p_dir` attribute is used to add directories.

`/var/audit` functions as the default file system.

```
$ auditconfig -setplugin audit_binfile p_dir=/audit/sys1.1,/var/audit
```

The preceding command sets the `/audit/sys1.1` file system as the primary directory for audit files and the default `/var/audit` as the secondary directory. In this scenario, `/var/audit` functions as the directory of last resort. For this configuration to succeed, the `/audit/sys1.1` file system must exist.

A similar file system is created in "How to Create ZFS File Systems for Audit Files" on page 91.

EXAMPLE 20 Limiting File Size for the `audit_binfile` Plugin

In this example, the size of a binary audit file is set to a specific size. The size is specified in megabytes.

```
$ auditconfig -setplugin audit_binfile p_fsize=4M

$ auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_age=0h;p_dir=/var/audit;p_minfree=1;p_fsize=4M;
```

By default, an audit file can grow without limit. To create smaller audit files, the administrator specifies a file size limit of 4 MB. The audit service creates a new file when the size limit is reached. The file size limit goes into effect after the administrator refreshes the audit service.

```
$ audit -s
```

EXAMPLE 21 Specifying Time for Log Rotation

In this example, a time limit is set for an audit file. The time limit is specified in terms of hours, days, weeks, months, or years.

```
$ auditconfig -setplugin audit_binfile p_age=1w

$ auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_minfree=1;p_fsize=4M;p_age=1w;
Queue size: 200
```

By default, an audit file has no time limit. The file remains open indefinitely until an external operation causes a file rotation. The administrator sets the file's time limit to one week, beyond which a new audit file is opened. To implement the new time limit, the administrator refreshes the audit service.

```
$ audit -s
```

EXAMPLE 22 Specifying Several Changes to an Audit Plugin

In this example, the administrator on a system with high throughput and a large ZFS pool changes the binary file size, and the soft limit warning for the `audit_binfile` plugin. The administrator allows audit files to grow to 4 GB, is warned when 2 percent of the ZFS pool remains. The audit file is also set to have a time limit of 2 weeks.

```
$ auditconfig -getplugin audit_binfile
Plugin: audit_binfile
```

```
Attributes: p_dir=/var/audit;p_fsize=2G;p_minfree=1;

$ auditconfig -setplugin audit_binfile \
      "p_minfree=2;p_fsize=4G;p_age=2w" 200

$ auditconfig -getplugin audit_binfile
Plugin: audit_binfile
Attributes: p_dir=/var/audit;p_fsize=4G;p_minfree=2;p_age=2w
```

The changed specifications go into effect after the administrator refreshes the audit service.

```
$ audit -s
```

EXAMPLE 23 Setting a Soft Limit for Warnings

In this example, the minimum free-space level for all audit file systems is set so that a warning is issued when two percent of the file system is still available.

```
$ auditconfig -setplugin audit_binfile p_minfree=2
```

The default percentage is one (`1`). For a large ZFS pool, choose a reasonably low percentage. For example, 10 percent of a 16 TB pool is around 16 GB, which would warn the audit administrator when plenty of disk space remains. A value of `2` sends the `audit_warn` message when about two GB of disk space remains.

The `audit_warn` email alias receives the warning. To set up the alias, see "How to Configure the `audit_warn` Email Alias" on page 60.

For a large pool, the administrator also limits the file size to 3 GB.

```
$ auditconfig -setplugin audit_binfile p_fsize=3G
```

The `p_minfree` and `p_fsize` specifications for the plugin go into effect after the administrator refreshes the audit service.

```
$ audit -s
```

▼ How to Configure `syslog` for the `audit_syslog` Plugin

You can instruct the audit service to copy some or all of the audit records in the audit queue to the `syslog` utility. If you record both binary audit data and text summaries, the binary data provide a complete audit record, while the summaries filter the data for real-time review.

Before You Begin To configure the `audit_syslog` plugin, you must become an administrator who is assigned the Audit Configuration rights profile. To configure the `syslog` utility and create the `auditlog` file, you must assume the `root` role.

1. **Select audit classes to be sent to the `audit_syslog` plugin, and make the plugin active.**

   Note - `p_flags` audit classes must be preselected as either system defaults or in the audit flags of a user or a rights profile. Records are not collected for a class that is not preselected.

   ```
   $ auditconfig -setplugin audit_syslog \
        active p_flags=lo,+as,-ss
   ```

2. **Configure the `syslog` utility.**

   a. **Add an `audit.notice` entry to the `syslog.conf` file that includes the location of the log file.**

   ```
   # cat /etc/syslog.conf

   …
   audit.notice       /var/adm/auditlog
   ```

   b. **Create the log file.**

   ```
   # touch /var/adm/auditlog
   ```

   c. **Set the log file's permissions to 640.**

   ```
   # chmod 640 /var/adm/auditlog
   ```

   d. **Check which system-log service instance is running on the system.**

   ```
   # svcs system-log

   STATE        STIME      FMRI
   online       13:14:36   svc:/system/system-log:default
   disabled     13:13:36   svc:/system/system-log:rsyslog
   ```

   Note - The default system log daemon is `syslog`.

   ---

   e. **Refresh the configuration information for the active `syslog` service instance.**

   ```
   # svcadm refresh system/system-log:default
   ```

3. **Refresh the audit service.**

   The audit service reads the changes to the audit plugin upon refresh.

   ```
   # audit -s
   ```

4. **Regularly archive the syslog log files.**

   The audit service can generate extensive output. To manage the logs, see the logadm(8) and syslog.conf(5) man pages>.

Example 24 Specifying Audit Classes for syslog Output

In this example, the syslog utility collects a subset of the preselected audit classes. The pf class is created in Example 11, "Creating a New Audit Class," on page 63.

```
$ auditconfig -setnaflags lo,na

$ auditconfig -setflags lo,ss

# usermod -K audit_flags=pf:no jdoe

$ auditconfig -setplugin audit_syslog \
    active p_flags=lo,+na,-ss,+pf
```

The arguments to the auditconfig command instruct the system to collect all login/logout, non-attributable, and change of system state audit records. The audit_syslog plugin entry instructs the syslog utility to collect all logins, successful non-attributable events, and failed changes of system state.

For the jdoe user, the binary utility collects successful and failed calls to the pfexec command. The syslog utility collects successful calls to the pfexec command.

Example 25 Putting syslog Audit Records on a Remote System

You can change the audit.notice entry in the syslog.conf file to point to a remote system. In this example, the name of the local system is sys1.1. The remote system is remote1.

```
sys1.1 # cat /etc/syslog.conf

…
audit.notice       @remote1
```

The audit.notice entry in the syslog.conf file on the remote1 system points to the log file.

```
remote1 # cat /etc/syslog.conf
```

```
...
audit.notice        /var/adm/auditlog
```

Configuring Remote Repositories for Audit Logs

You can configure the audit remote server (ARS) to receive and store audit records from one or more audited systems.

You configure the audit remote server (ARS) to receive and store audit records from one or more audited systems. The configuration is twofold. First, you configure the underlying security mechanisms to securely transport the audit data, that is, you configure the KDC. Second, you configure the audit service on both the audited system and the ARS.

Then, you activate the audit daemon on the remote server.

- "How to Configure a Remote Repository for Audit Files" on page 99
- "How to Send Audit Files to a Remote Repository" on page 104

▼ How to Configure a Remote Repository for Audit Files

This procedure illustrates a scenario with one audited client and one ARS, where the ARS and the KDC are on the same server. More complex scenarios can be configured similarly. The first four steps describe the configuration of the KDC, while the final step describes the configuration of the audit service.

Before You Begin Ensure that you have completed the following:

- You have assumed the `root` role.
- You have installed the Kerberos packages, as described in "How to Prepare to Stream Audit Records to Remote Storage" on page 39.
- You are working with an administrator who has configured the system that is sending audit records, as described in "How to Send Audit Files to a Remote Repository" on page 104.

1. **If your site has not yet configured a KDC, configure one.**

   You need a KDC on a system that both the audited system and the ARS can use, a host principal for each system, and an `audit` service principal. The following example illustrates a KDC configuration strategy:

```
arstore # kdcmgr -a audr/admin -r EXAMPLE.COM create master
```

This command uses the administrative principal `audr/admin` to create a master KDC in the `EXAMPLE.COM` realm, enables the master KDC, and starts the Kerberos service.

2. **Verify that the KDC is available.**

   For more information, see the `kdcmgr`(8) man page.

```
# kdcmgr status

KDC Status Information
--------------------------------------------
svc:/network/security/krb5kdc:default (Kerberos key distribution center)
State: online since Wed Feb 29 01:59:27 2012
See: man -M /usr/share/man -s 8 krb5kdc
See: /var/svc/log/network-security-krb5kdc:default.log
Impact: None.

KDC Master Status Information
--------------------------------------------
svc:/network/security/kadmin:default (Kerberos administration daemon)
State: online since Wed Feb 29 01:59:28 2012
See: man -M /usr/share/man -s 8 kadmind
See: /var/svc/log/network-security-kadmin:default.log
Impact: None.

Transaction Log Information
--------------------------------------------

Kerberos update log (/var/krb5/principal.ulog)
Update log dump :
Log version # : 1
Log state : Stable
Entry block size : 2048
Number of entries : 13
First serial # : 1
Last serial # : 13
First time stamp : Wed Feb 29 01:59:27 2012
Last time stamp : Mon Mar 5 19:29:28 2012


Kerberos Related File Information
--------------------------------------------
(Displays any missing files)
```

3. **Add the `audit` service principal to the KDC keytab file.**

You can add the principal by typing the `kadmin.local` command on the KDC system or remotely add the principal by using the `kadmin` command and providing a password. In this example, the `arstore` system is running the KDC.

```
# kadmin -p audr/admin

kadmin: addprinc -randkey audit/arstore.example.com@EXAMPLE.COM

kadmin: ktadd audit/arstore.example.com@EXAMPLE.COM
```

4. **On each audited system, add keys.**

   The receiver and the sender must have keys.

```
enigma # kclient

.. Enter the Kerberos realm:
EXAMPLE.COM

.. KDC hostname for the above realm:
arstore.example.com

.. Will this client need service keys ? [y/n]:
y
```

5. **Configure the audit service on the ARS.**

   a. **Create and name a connection group that accepts audit records from any audited system in the Kerberos realm.**

   For example:

```
$ auditconfig -setremote group create Bank_A
```

   `Bank_A` is a connection group. Because the `hosts` attribute is not defined, this group accepts all connections, which means that it is a *wildcard* group. Any audited system in this Kerberos realm whose `audit_remote` plugin is correctly configured can reach this ARS.

   b. **To limit connections to this group, specify the audited systems that can use this repository.**

```
$ auditconfig -setremote group Bank_A "hosts=enigma.example.com"
```

   Connection group `Bank_A` now accepts only connections from the `enigma` system. A connection from any other host is refused.

c. **To prevent an audit file in this group from growing too large, set a maximum size.**

```
$ auditconfig -setremote group Bank_A "binfile_fsize=4GB"

$ auditconfig -getremote
Audit Remote Server
Attributes: listen_address=;login_grace_time=30;max_startups=10;listen_port=0;
Connection group: Bank_A (inactive)
Attributes: binfile_dir=/var/audit;binfile_fsize=4GB;binfile_minfree=1;
hosts=enigma.example.com;
```

6. **Configure the audit service on the audited system.**

   To specify the ARS, use the `p_hosts` attribute.

```
enigma $ auditconfig -setplugin audit_remote \
     active p_hosts=arstore.example.com

enigma $ auditconfig -getplugin audit_remote
Plugin: audit_remote
Attributes: p_retries=3;p_timeout=5;p_hosts=arstore.example.com;
```

7. **Refresh the audit service.**

   The audit service reads the audit plugin change upon refresh.

```
$ audit -s
```

   The KDC now manages the connection between the audited system `enigma` and the ARS.

Example 26 Streaming Audit Records to Different File Locations on the Same ARS

This example extends the example in the procedure and separates audit records by host on the ARS by creating two connection groups.

Audit files from the `audsys1` stream to the `Bank_A` connection group on this ARS.

```
arstore $ auditconfig -setremote group create Bank_A

arstore $ auditconfig -setremote group active Bank_A "hosts=audsys1" \
   "hosts=audsys1;binfile_dir=/var/audit/audsys1;binfile_fsize=4M;"
```

Audit files from the `audsys2` stream to the `Bank_B` connection group.

```
arstore $ auditconfig -setremote group create Bank_B
arstore $ auditconfig -setremote group active Bank_B \
"hosts=audsys2;binfile_dir=/var/audit/audsys2;binfile_fsize=4M;"
```

For easier maintenance, set other attribute values identically.

```
arstore $ auditconfig -getremote
Audit Remote Server
Attributes: listen_address=;login_grace_time=30;max_startups=10;listen_port=0;

Connection group: Bank_A
Attributes: binfile_dir=/var/audit/audsys1;binfile_fsize=4M;binfile_minfree=1;
hosts=audsys1

Connection group: Bank_B
Attributes: binfile_dir=/var/audit/audsys2;binfile_fsize=4M;binfile_minfree=1;
hosts=audsys2
```

Example 27 Placing the ARS on a Different System From the KDC

In this example, the ARS is places on a different system from the KDC. First, create and configure the master KDC.

```
kserv# kdcmgr -a audr/admin -r EXAMPLE.COM create master

kserv# kadmin.local -p audr/admin

kadmin: addprinc -randkey \
audit/arstore.example.com@EXAMPLE.COM

kadmin: ktadd -k /var/user/root/krb5.keytab.audit \
     audit/arstore.example.com@EXAMPLE.COM
```

After securely transmitting the `/tmp/krb5.keytab.audit` file to the ARS, `arstore`, move the file to the correct location.

```
arstore # chown root:root krb5.keytab.audit

arstore # chmod 600 krb5.keytab.audit

arstore # mv krb5.keytab.audit /etc/krb5/krb5.keytab
```

Rather than rewrite the file, you also have the option to use the `ktutil` command on the ARS to merge the KDC `krb5.keytab.audit` file with existing keys in the `arstore` 's `/etc/krb5/krb5.keytab` file.

Finally, generate keys on the audited system.

```
enigma # kclient
```

```
.. Enter the Kerberos realm: EXAMPLE.COM

.. KDC hostname for the above realm: kserv.example.com

.. Will this client need service keys ? [y/n]: y
```

▼ How to Send Audit Files to a Remote Repository

In this procedure, you use attributes of the `audit_remote` plugin to send the audit trail to a remote audit repository. To configure a remote repository on an Oracle Solaris system, see "How to Configure a Remote Repository for Audit Files" on page 99.

Before You Begin You must have a receiver of audit files at your remote repository. You must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Determine the attributes of the `audit_remote` plugin.**

   Read the `OBJECT ATTRIBUTES` section of the `audit_remote(7)` man page.

   ```
   $ man audit_remote

   ...
   OBJECT ATTRIBUTES
   The p_hosts attribute specifies the remote servers.
   You can also specify the port number and the GSS-API
   mechanism.

   The p_retries attribute specifies the number of retries for
   connecting and sending data. The default is 3.

   The p_timeout attribute specifies the number of seconds
   in which a connection times out.
   ```

   The default port is the `solaris_audit` IANA-assigned port, `16162/tcp`. The default mechanism is `kerberos_v5`. The timeout default is `5` seconds. You can also specify a queue size for the plugin.

2. **To specify the remote receiving system, use the `p_hosts` attribute.**

   In this example, the receiving system uses a different port.

   ```
   $ pfbash ; auditconfig -setplugin audit_remote \
   ```

```
    p_hosts=ars.example.com:16088:kerberos_v5
```

3. **Specify other attributes of the plugin that you want to change.**

   For example, the following command specifies values for all optional attributes:

   ```
   $ auditconfig -setplugin audit_remote "p_retries=;p_timeout=3" 300
   ```

4. **Verify the values, then activate the plugin.**

   For example, the following commands specify and verify the values of the plugin:

   ```
   $ auditconfig -getplugin audit_remote
   Plugin: audit_remote (inactive)
   Attributes: p_hosts=ars.example.com:16088:kerberos_v5;p_retries=5;p_timeout=3;
   Queue size: 300

   $ auditconfig -setplugin audit_remote active
   ```

5. **Refresh the audit service.**

   The audit service reads the audit plugin change upon refresh.

   ```
   $ audit -s
   ```

Example 28 Tuning the Audit Queue Buffer Size

In this example, the audit queue is full behind the `audit_remote` plugin. This audited system is configured to audit many classes and is transmitting across a high-traffic, slow network. The administrator enlarges the plugin's buffer size to enable the audit queue to grow and not exceed the buffer's limit before records are removed from the queue.

```
audsys1 $ pfbash ; auditconfig -setplugin audit_remote "" 1000

audsys1 $ audit -s
```

♦ ♦ ♦ CHAPTER 5

Viewing Audit Records

This chapter provides procedures to help you view audit data that are generated from different local systems. This chapter covers the following topics:

- "Displaying Audit Trail Data" on page 107
- "Managing Audit Records on Local Systems" on page 115

Displaying Audit Trail Data

The default plugin, `audit_binfile`, creates an audit trail. The trail can contain large amounts of data. The following sections describe how to work with this data.

Displaying Audit Record Definitions

To display audit record definitions, use the `auditrecord` command. The definitions provide the audit event number, audit class, selection mask, and record format of an audit event.

```
$ auditrecord -options
```

The screen output generated by the command depends on the option that you use. Some common options are:

- `-p` – Displays the audit record definitions of a program.
- `-c` – Displays the audit record definitions of an audit class.
- `-a` – Lists all audit event definitions.
- `-h` – Generates the output in an HTML file. When you display the HTML file in a browser, use the browser's Find tool to find specific audit record definitions. See Example 31, "Printing Audit Record Definitions to a File," on page 109.

For more information, see the `auditrecord(8)` man page.

EXAMPLE 29 Displaying the Audit Record Definitions of a Program

This example displays the definition of all audit records that are generated by the `login` program. Login programs include `rlogin`, `telnet`, `newgrp`, and the Secure Shell feature of Oracle Solaris.

```
$ auditrecord -p login
...
login: logout
program     various              See login(1)
event ID    6153                 AUE_logout
class       lo                  (0x0000000000001000)
...
newgrp
program     newgrp               See newgrp login
event ID    6212                 AUE_newgrp_login
class       lo                  (0x0000000000001000)
...
rlogin
program     /usr/sbin/login      See login(1) - rlogin
event ID    6155                 AUE_rlogin
class       lo                   (0x0000000000001000)
...
/usr/lib/ssh/sshd
program     /usr/lib/ssh/sshd    See login - ssh
event ID    6172                 AUE_ssh
class       lo                   (0x0000000000001000)
...
telnet login
program     /usr/sbin/login      See login(1) - telnet
event ID    6154                 AUE_telnet
class       lo                   (0x0000000000001000)
…
```

EXAMPLE 30 Displaying the Audit Record Definitions of an Audit Class

This example displays the definitions of all audit records in the `pf` class that was created in Example 11, “Creating a New Audit Class,” on page 63.

```
$ auditrecord -c pf
pfexec
system call pfexec               See execve(2) with pfexec enabled
event ID    116                  AUE_PFEXEC
class       pf                   (0x0100000000000000)
header
```

```
path                    pathname of the executable
path                    pathname of working directory
[privileges]            privileges if the limit or inheritable set are changed
[privileges]            privileges if the limit or inheritable set are changed
[process]               process if ruid, euid, rgid or egid is changed
exec_arguments
[exec_environment]      output if arge policy is set
subject
[use_of_privilege]
return
```

The `use_of_privilege` token is recorded whenever privilege is used. The `privileges` tokens are recorded if the limit or inheritable set is changed. The `process` token is recorded if an ID is changed. No policy option is required for these tokens to be included in the record.

EXAMPLE 31 Printing Audit Record Definitions to a File

In this example, the `-h` option is added to put all the audit record definitions to a file in HTML format. When you display the HTML file in a browser, use the browser's Find tool to find specific audit record definitions.

```
$ auditrecord -ah > audit.events.html
```

Selecting Audit Events to Be Displayed

As an administrator who is assigned the Audit Review rights profile, you can filter audit records for examination by using the `auditreduce` command. This command can eliminate the less interesting records as it combines the input files.

auditreduce -*option argument* **[***filename***]**

Some commonly used record selection options and their corresponding arguments are:

`-c`	Selects an audit class where *argument* is an audit class, such as `ua`.
`-d`	Selects all of the events on a particular date. The date format is *yyymmdd*. Other date options such as `-b` and `-a` select events before and after a particular date, respectively.
`-u`	Selects all of the events attributable to a particular user. For this option, you specify a user name. Another user option, `-e`, selects all of the events attributable to an effective user ID.

`-g`	Selects all of the events attributable to a particular group. For this option, specify a group name.
`-m`	Selects all of the instances of a particular audit event.
`-o`	Selects by object type. Use this option to select by file, group, file owner, FMRI, PID, and other object types.
filename	The name of an audit file.

The command also uses file selection options that determine which files are to be processed and certain types of special treatment. They are all in upper case as shown in the following examples. For the full list of options, see the `auditreduce`(8) man page.

EXAMPLE 32 Combining and Reducing Audit Files

In this example, only the login and logout records in audit files that are over a month old are retained. The example assumes that the current date is Sept 27. If you need to retrieve the complete audit trail, you could recover the trail from backup media. The `-O` option directs the command's output to a file named `lo.summary`. Records are included if they occurred before the date specified by the `-b` option.

```
$ cd /var/audit/audit_summary
$ auditreduce -O lo.summary -b 20100827 -c lo; compress *lo.summary
```

EXAMPLE 33 Copying One User's Audit Records to a Summary File

In this example, the records in the audit trail that contain the name of a particular user are merged. The `-e` option finds the effective user. The `-u` option finds the login user. The `-O` option directs the output to the file `tamiko`.

```
$ cd /var/audit/audit_summary
$ auditreduce -e tamiko -O tamiko
```

You can further narrow the displayed information. In this next example, the following data is filtered and printed to a file called `tamikolo`.

- Time of user login and logout, specified by the `-c` option.
- Date of Sept 7, 2013, specified by the `-d` option. The short form of the date is *yyyymmdd*.
- User name of `tamiko`, specified by the `-u` option.
- Name of system, specified by the `-M` (Machine) option.

```
$ auditreduce -M system1 -O tamikolo -d 20130907 -u tamiko -c lo
```

EXAMPLE 34 Merging Selected Records to a Single File

In this example, login and logout records for a particular day are selected from the audit trail. The records are merged into a target file. The target file is written in a file system other than the file system that contains the audit root directory.

```
$ auditreduce -c lo -d 20130827 -O /var/audit/audit_summary/logins

$ ls /var/audit/audit_summary/*logins
/var/audit/audit_summary/20130827183936.20130827232326.logins
```

Viewing the Contents of Binary Audit Files

As an administrator who is assigned the Audit Review rights profile, you can view the contents of binary audit files by using the `praudit` command.

```
$ praudit options
```

The following options are commonly used. You can combine any of these options with the `-l` option to display each record on one line. For a complete list of options, see the `praudit(8)` man page.

`-s`	Displays audit records in a short format, one token per line.
`-r`	Displays audit records in their raw format, one token per line.
`-x`	Displays audit records in XML format, one token per line. This option is useful for further processing. You can reformat records in the XML file to become readable in any browser by using the `xsltproc` tool. This tool applies stylesheet definitions to the file contents. See Example 38, "Making Audit Records in XML Format Readable in a Browser," on page 112.

You can also use the `auditreduce` and `praudit` commands together by piping the `praudit` output from the `auditreduce` command.

You have the option to process output from the `praudit` command as lines of text.

EXAMPLE 35 Displaying Audit Records in a Short Format

In this example, login and logout events that are extracted by the `auditreduce` command display in short format.

```
$ auditreduce -c lo | praudit -s

header,69,2,AUE_screenlock,,mach1,2010-10-14 08:02:56.348 -07:00
subject,jdoe,root,staff,jdoe,staff,856,50036632,82 0 mach1
return,success,0
sequence,1298
```

EXAMPLE 36 Displaying Audit Records in Raw Format

In this example, login and logout events that are extracted by the `auditreduce` command are displayed in raw format.

```
$ auditreduce -c lo | praudit -r

21,69,2,6222,0x0000,192.0.2.45,1287070091,698391050
36,26700,0,10,26700,10,856,50036632,82 0 192.0.2.45
39,0,0
47,1298
```

EXAMPLE 37 Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
$ praudit -x 20100827183214.20100827215318.logins > 20100827.logins.xml
```

Similarly, you can display audit records filtered by the `auditreduce` command in XML format.

```
$ auditreduce -c lo | praudit -x
<record version="2" event="screenlock - unlock" host="mach1"
iso8601="2010-10-14 08:28:11.698 -07:00">
<subject audit-uid="jdoe" uid="root" gid="staff" ruid="jdoe
rgid="staff" pid="856" sid="50036632" tid="82 0 mach1"/>
<return errval="success" retval="0"/>
<sequence seq-num="1298"/>
</record>
```

The contents of the file can be operated on by a script to extract the relevant information.

EXAMPLE 38 Making Audit Records in XML Format Readable in a Browser

In this example the `xsltproc` tool is used to reformat records in the XML file to become readable in any browser. This tool applies stylesheet definitions to the file contents. To put the reformatted contents in a separate file, you would type the following:

```
$ auditreduce -c lo | praudit -x | xsltproc - > logins.html
```

In a browser, the contents of `logins.html` would be displayed in a format similar to the following:

```
                     Audit Trail Data

File: time: 2013-11-04 12:54:28.000 -08:00

Event: login - local
time: 2013-11-04 12:54:28.418 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: jdoe gid: staff ruid: jdoe rgid: staff
      pid: 1534 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: connect to RAD
time: 2013-11-04 12:54:52.029 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: jdoe gid: staff ruid: jdoe rgid: staff
      pid: 1835 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: role login
time: 2013-11-08 08:42:52.286 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: root gid: root ruid: root rgid: root
      pid: 4265 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: role logout
time: 2013-11-08 08:43:37.125 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jdoe uid: root gid: root ruid: root rgid: root
      pid: 4265 sid: 3583012893 tid: 0 0 host
RETURN errval: success retval: 0

Event: login - ssh
time: 2013-12-23 12:24:37.292 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: jsmith gid: staff ruid: jsmith rgid: staff
      pid: 2002 sid: 39351741 tid: 14632 202240 host.example.com
RETURN errval: success retval: 0

Event: role login
time: 2013-12-23 12:25:07.345 -08:00 vers: 2 mod: fe host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2023 sid: 39351741 tid: 14632 202240 host.example.com
RETURN errval: failure retval: Permission denied

Event: su
time: 2013-12-23 17:19:24.031 -08:00 vers: 2 mod: na host: host
RETURN errval: success retval: 0

Event: su logout
```

```
time: 2013-12-23 17:19:24.362 -08:00 vers: 2 mod: na host: host
RETURN errval: success retval: 0

Event: login - ssh
time: 2013-12-23 17:27:21.306 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: jsmith gid: staff ruid: jsmith rgid: staff
      pid: 2583 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0

Event: role login
time: 2013-12-23 17:27:28.361 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2593 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0

Event: role logout
time: 2013-12-23 17:30:39.029 -08:00 vers: 2 mod: host: host
SUBJECT audit-uid: jsmith uid: root gid: root ruid: root rgid: root
      pid: 2593 sid: 3401970889 tid: 13861 5632 host.example.com
RETURN errval: success retval: 0
```

Other events

EXAMPLE 39 Displaying `pfedit` Records Only

You can use filters to extract and view only specific records from the audit trail. In this example, records that capture the use of the `pfedit` command are filtered. Suppose that the summary file is `20130827183936.20130827232326.logins`. Use of the `pfedit` command generates the `AUE_admin_edit` event. Therefore, to extract `pfedit` records, run the following command:

```
auditreduce -m AUE_admin_edit 20130827183936.20130827232326.logins | praudit
```

EXAMPLE 40 Printing the Entire Audit Trail

With a pipe to the print command, the output for the entire audit trail goes to the printer. For security reasons, the printer has limited access.

```
$ auditreduce | praudit | lp -d example.protected.printer
```

EXAMPLE 41 Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
$ cd /var/audit/audit_summary/logins

$ praudit 20100827183936.20100827232326.logins | more
```

EXAMPLE 42 Processing `praudit` Output With a Script

In this example, you process output from the `praudit` command as lines of text. If, for example, you want to select records that the `auditreduce` command cannot select, you can use a simple shell script to process the output of the `praudit` command. The following sample script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form.

```
#!/bin/sh
#
## This script takes an argument of a user-specified string.
#  The sed command prefixes the header tokens with Control-A
#  The first tr command puts the audit tokens for one record
#  onto one line while preserving the line breaks as Control-A
#
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \\
| tr '\\012\\001' '\\002\\012' \\
| grep "$1" \\
```
Finds the user-specified string

```
| tr '\\002' '\\012'
```
Restores the original newline breaks

Note that the `^a` in the script is Control-A, not the two characters `^` and `a`. The prefix distinguishes the `header` token from the string `header` that might appear as text.

A message similar to the following indicates that you do not have enough privilege to use the `praudit` command.

```
praudit: Can't assign 20090408164827.20090408171614.sys1.1 to stdin.
```

Run the `praudit` command in a profile shell. You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Managing Audit Records on Local Systems

This section covers procedures for selecting, analyzing, and managing audit records.

TABLE 6 Managing Audit Records on Local Systems Task Map

Task	Description	For Instructions
Merge audit records.	Combines audit files from several audit directories into one audit trail.	"How to Merge Audit Files From the Audit Trail" on page 116
Clean up incorrectly named audit files.	Provides an end time stamp to audit files that were inadvertently left open by the audit service.	"How to Clean Up a `not_terminated` Audit File" on page 118
Prevent audit trail overflow.	Prevents the audit file systems from becoming full.	"Preventing Audit Trail Overflow" on page 119

▼ How to Merge Audit Files From the Audit Trail

By merging the audit files from all the audit directories, you can analyze the contents of the entire audit trail.

Note - Because the time stamps in the audit trail are in Coordinated Universal Time (UTC), the date and hour must be translated to the current time zone to be meaningful. Keep this point in mind whenever you manipulate these files with standard file commands rather than with the `auditreduce` command.

Before You Begin You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Create a file system for storing merged audit files.**

   To lessen the chance of reaching the limit of disk space, this file system should be in a *different zpool* from the file systems that you created in "How to Create ZFS File Systems for Audit Files" on page 91 to store the original files.

2. **Merge the audit records in the audit trail.**

   Go to the directory for storing merged audit files. From this directory, merge the audit records into a file with a named suffix. All directories in the audit trail on the local system are merged and placed in this directory.

   ```
   $ cd audit-storage-directory
   $ auditreduce Uppercase-option -O suffix
   ```

   The uppercase options of the `auditreduce` command manipulate files in the audit trail. The uppercase options include the following:

`-A`	Selects all of the files in the audit trail.

`-C`	Selects complete files only.
`-D`	Selects files to be deleted.
`-M`	Selects files with a particular suffix. The suffix can be a physical machine name or a suffix that you have specified for a summary file.
`-O` *suffix*	Creates an audit file with 14-character time stamps for both the start time and the end time, with the suffix *suffix* in the current directory.
`-R` *pathname*	Specifies to read audit files in *pathname*, an alternate audit root directory.
`-S` *server*	Specifies to read audit files from the specified server.

For the full list of options, see the `auditreduce`(8) man page.

Example 43 Copying Audit Files to a Summary File

In this example, an administrator who is assigned the System Administrator rights profile copies all files from the audit trail into a merged file on a different file system. The `/var/audit/storage` file system is on a separate disk from `/var/audit` which functions as the audit root file system.

```
$ cd /var/audit/storage
$ auditreduce -A -O All
$ ls /var/audit/storage/*All
20100827183214.20100827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file. The complete path is specified as the value of the `-0` option. The last component of the path, `Complete`, is used as the suffix.

```
$ auditreduce -C -O /var/audit/storage/Complete
$ ls /var/audit/storage/*Complete
20100827183214.20100827214217.Complete
```

In the following example, by adding the `-D` option, the original audit files are deleted.

```
$ auditreduce -C -O daily_sys1.1 -D sys1.1

$ ls *sys1.1
20100827183214.20100827214217.daily_sys1.1
```

Cleaning Up Non-Terminated Audit Files

When anomalous system interruptions occur, the audit service exits while its audit file is still open. Or, a file system becomes inaccessible and forces the system to switch to a new file system. In such instances, an audit file remains with the string `not_terminated` as the end time stamp, even though the file is no longer used for audit records. Use the `auditreduce -O` command to give the file the correct time stamp.

Note - The file remains. To archive an audit file prior to removing the file, see "Preventing Audit Trail Overflow" on page 119.

▼ How to Clean Up a `not_terminated` Audit File

This procedure leaves the repaired audit file in place. To archive the file prior to removing it, see "Preventing Audit Trail Overflow" on page 119.

Before You Begin You must become an administrator who is assigned the Audit Review rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **List the files with the `not_terminated` string on your audit file system in order of creation.**

   ```
   $ ls -1t audit-directory */* | grep not_terminated
   ```

`-t`	Lists files from most recent to oldest.
`-1`	Lists the files in one column.

2. **Clean up the old `not_terminated` file.**
   Specify the name of the old file to the `auditreduce -O` command.

   ```
   $ auditreduce -O system-name old-not-terminated-file
   ```

3. **Remove the old `not_terminated` file.**

   ```
   $ rm system-name old-not-terminated-file
   ```

Example 44 Cleaning Up Closed `not_terminated` Audit Files

In this example, `not_terminated` files are found, renamed, then the originals are removed.

```
ls -R1t */* | grep not_terminated
```

```
…/egret.1/20100908162220.not_terminated.egret
…/egret.1/20100827215359.not_terminated.egret

$ cd */egret.1
$ auditreduce -O egret 20100908162220.not_terminated.egret
$ ls -1t
20100908162220.not_terminated.egret      Current audit file

20100827230920.20100830000909.egret      Cleaned-up audit file

20100827215359.not_terminated.egret      Input (old) audit file

$ rm 20100827215359.not_terminated.egret
$ ls -1t
20100908162220.not_terminated.egret      Current audit file

20100827230920.20100830000909.egret      Cleaned-up audit file
```

The start time stamp on the new file reflects the time of the first audit event in the `not_terminated` file. The end time stamp reflects the time of the last audit event in the file.

Preventing Audit Trail Overflow

If your security policy requires that all audit data be saved, prevent audit record loss by observing the following practices.

- Set a minimum free size on the `audit_binfile` plugin by using the `p_minfree` attribute.

  The `audit_warn` email alias sends a warning when the disk space fills to the minimum free size. See Example 23, "Setting a Soft Limit for Warnings," on page 96.
- Set up a schedule to regularly archive audit files.

  Archive audit files by backing up the files to offline media. You can also move the files to an archive file system.

  If you are collecting text audit logs with the `syslog` utility, archive the text logs. For more information, see the `logadm`(8) man page.
- Set up a schedule to delete the archived audit files from the audit file system.
- Save and store auxiliary information.

  Be sure to archive information that is necessary to interpret audit records along with the audit trail. Minimally, you should save `passwd`, `group`, and `hosts`. You also might archive `audit_event` and `audit_class`.
- Keep records of which audit files have been archived.
- Store the archived media appropriately.

- Reduce the amount of file system capacity that is required by enabling ZFS compression.

  On a ZFS file system that is dedicated to audit files, compression shrinks the files considerably. For an example, see "How to Compress Audit Files on a Dedicated File System" on page 80.

  See also "Interactions Between ZFS Compression, Deduplication, and Encryption Properties" in *Managing ZFS File Systems in Oracle Solaris 11.4*.
- Reduce the volume of audit data that you store by creating summary files.

  You can extract summary files from the audit trail by using options to the `auditreduce` command. The summary files contain only records for specified types of audit events. To extract summary files, see Example 32, "Combining and Reducing Audit Files," on page 110 and Example 34, "Merging Selected Records to a Single File," on page 111.

New Feature – Listing the Available Audit Classes

Using the new `-lsclass` option for the `auditconfig` command, users can view configured audit classes:

```
$ auditconfig -lsclass
no       invalid class
fr       file read
fw       file write
fa       file attribute access
...
am       administrative (meta-class)
         ss,as,ua
ad       old administrative (meta-class)
         ss,as,ua,aa
...
```

Note - The name of the class is followed by its description. Meta-classes are followed by a list of their constituent classes.

For more information, see the `auditconfig`(8) man page.

New Feature – Listing Audit Events by Audit Class

The `auditconfig -lsevent` command takes an *audit_flags* argument that displays the audit events in the specified classes.

The following example specifies the `as` and `lo` audit flags:

```
$ auditconfig -lsevent as,lo
AUE_UMOUNT                        12 as umount(2) - old version
AUE_ACCT                          18 as acct(2)
AUE_ADJTIME                       50 as adjtime(2)
AUE_MOUNT                         62 as mount(2)

...

AUE_login                       6152 lo login - local
AUE_logout                      6153 lo logout
AUE_telnet                      6154 lo login - telnet
AUE_rlogin                      6155 lo login - rlogin
AUE_rshd                        6158 lo rsh access
AUE_su                          6159 lo su
```

See the `auditconfig`(8) and `audit_flags`(7) man pages.

New Feature – Filtering Audit Records by Functional Area

System administrators can filter common sets of audit records by functional areas such as `cpu`, `dev`, `file`, or `net`. These functional areas are called **audit tags** and defined in the `audit_tags` database. Also, administrators can create and use their own audit tags.

- List the audit tags – `auditconfig -lstags`
- Verify the audit tags – `auditconfig -chktags`
- View events in the audit record by audit tag – `auditreduce -t` *audit-tag*
- View summaries of privileged events in the audit record by audit tag – `admhist -t` *audit-tag*

For further information, see the `auditconfig`(8), `auditreduce`(8), and `admhist`(8) man pages.

New Feature – Reviewing Multiple Audit Events

The `auditreduce` command can review several audit events with one iteration of the command. The `auditreduce -m` command accepts a one or more audit events, separated by a commas. The following example specifies three audit events from the audit trail.

```
$ auditreduce -m AUE_SYSTEMBOOT,6166,6153 | praudit | more
```

```
file,2014-02-14 08:02:25.000-08:00,
header,64,2,system booted,na,system1,2014-02-14 08:02:25.472-08:00
text,booting kernel
header,54,2,init(8),na,system1,2014-02-14 08:05:36.440-08:00
text,booted
return,success,0
header,52,2,system booted,na,system1.example.com,2014-02-14 08:31:29.330-08:00
text,booting kernel
header,42,2,init(8),na,system1.example.com,2014-02-14 08:32:35.109-08:00
text,booted
return,success,0
header,69,2,logout,,system1.example.com,2014-02-14 08:32:49.940-08:00
subject,jandoe,jandoe,staff,jandoe,staff,1075,2258250201,60120 5632 dhcp-whq-twvpn-2-
vpnpool-10-159-180-35.vpn.example.com
return,success,0
...
```

For more information, see the `auditreduce(8)` man page.

New Feature – Viewing a Summary of Audit Records

Using the new `admhist` command, administrators can view a summary of successful privileged execution audit records in a helpful, easy-to-understand format. The `admhist` command extracts from the audit trail those commands that successfully used an administrative privilege, so the output displays commands that are more likely to have modified the system.

Note - Users must be assigned the Audit Review rights profile to use the `admhist` utility.

The `admhist` command pulls successful audit records from the privileged execution (`pe`) audit class and displays them in summary form. See also "New Feature – Per-Privilege Logging of Audit Events" on page 73.

Using the `admhist` command options, you can view events that occurred during a specified time period, events from a specified zone name, or events in one or more functional areas labeled with audit tags as described in "New Feature – Filtering Audit Records by Functional Area" on page 121.

For example, the following command displays successful events that occurred in `zone1`.

```
$ pfexec ; admhist -z zone1
```

The following command displays successful administrative commands that occurred on the system in the last six hours.

```
$ admhist -a "last 6 hours"
```

Note - The `admhist` facility does not record events that are run by `root`. Use the `auditreduce` and `praudit` commands to view audit events that `root` is responsible for, such as device attachment and detachment. The events must be in the preselection mask. Audit events that are in the `other` audit class are not in the preselection mask by default. To enable the events to be audited, see "How to Add an Audit Class" on page 61.

For further information, see the `admhist(8)` man page.

New Feature – Displaying Auditing Data Graphically

In addition to the auditing tools that are described in Chapter 5, "Viewing Audit Records", administrators can use two new tools, the Oracle Solaris StatsStore and Oracle Solaris Analytics.

The Oracle Solaris StatsStore collects data from multiple sources such as SMF services, FMA, and the auditing system, and collects statistics from sources such as `kstat`, `dtrace`, and some applications. A related new tool, Oracle Solaris Analytics, provides graphic diagnosis tools that you can use to analyze data from the StatsStore to monitor and troubleshoot system activity.

Access to auditing data in the StatsStore and in Oracle Solaris Analytics requires the `solaris.sstore.read.audit` authorization. This authorization is granted to the Audit Review rights profile.

Viewing Audit Data in the Statistics Store

By default, audit records in the `sstore` meta-class are automatically forwarded to the StatsStore.

The `sstore` meta-class has replaced the `lo` class as the default set of audit flags. Audit classes included in the `sstore` meta-class are: `lo`, `ss`, `as`, and `pe`.

The total number of written audit records is provided as a statistic in the StatsStore. In addition, each configured audit class as defined in the `audit_class` file, has a corresponding total number of records that is generated as a statistic in the StatsStore, along with the total number of failed events and passed events for each audit class.

If auditing was configured for all audit records to be recorded only in the global zone, only the global zone's StatsStore receives audit class statistics. If auditing was configured so that each zone records audits independently, each zone's StatsStore receives audit class statistics.

Application developers who want to customize content in the StatsStore should see *Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface*.

Analytics' Auditing Sheet

The Oracle Solaris Analytics tool provides problem diagnosis tools that use the data from the Oracle Solaris StatsStore, including statistics about audit events. In the Analytics auditing sheet, administrators can review the amount of activity by audit class for specific time periods, and they can drill down in each graphic to view specific audit events.

The Analytics auditing sheet provides graphical summaries of audit events by class that cover:

- Monitoring system administration and access activity
- Monitoring other audit event categories, such as networking and application activity
- Monitoring file access activity

The Analytics auditing sheet uses descriptive class names that map to existing audit class names. The mapping is very similar to the mapping in the `/etc/security/audit_class` file.

For more information about the StatsStore and the Analytics tool, review *Using Oracle Solaris 11.4 StatsStore and System Web Interface*.

♦ ♦ ♦ C H A P T E R 6

Analyzing and Resolving Audit Issues

This chapter provides information about the following topics:

- "Troubleshooting the Audit Service" on page 125
- "Best Practices for Auditing Core System Files" on page 133

Troubleshooting the Audit Service

This section covers various auditing error messages, preferences, and the auditing that is provided by other tools to help you debug audit problems.

Typically, different notices are sent to alert you of errors in the audit service. Review your email and the log files if you think that problems exist with the audit service.

- Read the email sent to the `audit_warn` alias.

  The `audit_warn` script sends alert messages to the `audit_warn` email alias. In the absence of a correctly configured alias, the messages are sent to the `root` account.
- The audit service provides error information to its corresponding SMF log file, `/var/svc/log/system-auditd:default.log`. Review this log for possible issues related to the SMF processes.
- Review the log files for the audit service.

  The output from the `svcs -x auditd` command lists the full path to the audit service log files.
- Review the system log files.

  The `audit_warn` script logs messages to the `daemon` facility at the `alert` level.

Note - Administrators configure how the `syslog` daemon handles `daemon.alert` activity on their systems. See the system log service configuration to identify where such messages are recorded.

After you locate and fix the problems, restart by using one of the following options.

- If the audit service is in maintenance mode due to SMF-related issues, run the following command to start the audit service:

  ```
  # svcadm clear auditd
  ```

  For more information, see the `svcadm`(8) man page.
- Enable or restart the audit service.

  ```
  # audit -s
  ```

The following sections describe possible problem cases and the steps to resolve them.

Note - Before you perform any troubleshooting tasks, ensure that you have the proper authorization. For example, to configure auditing, you must become an administrator who is assigned the Audit Configuration rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Audit Records Are Not Being Logged

Auditing is enabled by default. If you believe that auditing has not been disabled, but no audit records are being sent to the active plugin, the causes might be one or a combination of the following factors discussed in this section. Note that to modify a system file, you must be assigned the `solaris.admin.edit/`*path-to-system-file* authorization. By default, the root role has this authorization.

Audit Service Not Running

To check whether auditing is running, use any of the following methods:

- Verify the current audit condition.

  The following output indicates that auditing is not running:

  ```
  # auditconfig -getcond
  audit condition = noaudit
  ```

  The following output indicates that auditing is running:

  ```
  # auditconfig -getcond
  audit condition = auditing
  ```

- Verify that the audit service is running.

  The following output indicates that auditing is not running:

  ```
  # svcs -x auditd

  svc:/system/auditd:default (Solaris audit daemon)
  State: disabled since Tues Jan 5 10:10:10 2016
  Reason: Disabled by an administrator.
  See: http://support.oracle.com/msg/SMF-8000-05
  See: auditd(8)
  See: audit(8)
  See: auditconfig(8)
  See: audit_warn(8)
  See: ars(7)
  See: audit_flags(7)
  See: audit_binfile(7)
  See: audit_syslog(7)
  See: audit_remote(7)
  See: audit_sstore(7)
  See: http://www.oracle.com/pls/topic/lookup?ctx=solaris&id=OSMAA
  See: /var/svc/log/system-auditd:default.log
  See: /var/adm/messages
  Impact: This service is not running.
  ```

  The following output indicates that the audit service is running:

  ```
  # svcs auditd
  STATE          STIME    FMRI
  online         10:10:10 svc:/system/auditd:default
  ```

If the audit service is not running, enable it. For the procedure, see "Enabling and Disabling the Audit Service" on page 49.

No Audit Plugin Active

Use the following command to check if any plugins are active. At least one plugin must be active for the audit service to work.

```
# audit -v
audit: no active plugin found
```

If no plugin is active, make one active.

```
# auditconfig -setplugin audit_binfile active
```

```
# audit -v
configuration ok
```

Audit Class Undefined

You might be attempting to use an audit class that has not been defined. For a description of creating the `pf` class, see "How to Add an Audit Class" on page 61.

For example, the following list of flags contains the `pf` class, which Oracle Solaris software did not deliver:

```
# auditconfig -getflags
active user default audit flags = pf,lo(0x0100000000000000,00x0100000000001000)
configured user default audit flags = pf,lo(0x0100000000000000,00x0100000000001000)
```

If you do not want to define the class, run the `auditconfig -setflags` command with valid values to reset the current flags. Otherwise, ensure the following when defining a class:

- The audit class is defined in the `audit_class` file.

  ```
  # grep pf /etc/security/audit_class
  ```
  Verify class exists

  ```
  0x0100000000000000:pf:profile
  ```
- The mask is unique. If it is not unique, replace the mask.

  ```
  # grep 0x0100000000000000 /etc/security/audit_class
  ```
  Ensure mask is unique

  ```
  0x0100000000000000:pf:profile
  ```

No Assigned Events to Audit Class

The customized class that you are using, although defined, might not have any events assigned to the class.

To verify whether events are assigned to the customized class, use one of the following methods:

```
# auditconfig -lsevent | egrep " pf|,pf|pf,"
AUE_PFEXEC      116 pf execve(2) with pfexec enabled

# auditrecord -c pf
```

List of audit events assigned to pf class

If events are not assigned to the class, assign the appropriate events to this class.

Volume of Audit Records Is Large

After you have determined which events must be audited at your site, use the following suggestions to create audit files with just the information that you require. Note that to assign flags to users, roles, and rights profiles, you must assume the `root` role.

- Specifically, avoid adding events and audit tokens to the audit trail. The following policies increase the size of the audit trail.

`arge`	Adds environment variables to `execv` audit events. Although auditing `execv` events can be costly, adding variables to the audit record is not.
argv	Adds command parameters to `execv` audit events. Adding command parameters to the audit record is not costly.
group	Adds a group token to audit events that include an optional `newgroups` token.
path	Adds a `path` token to audit events that include an optional `path` token.
public	If file events are being audited, adds an event to the audit trail every time an auditable event happens to a public object. File classes include `fa`, `fc`, `fd`, `fm`, `fr`, `fw`, and `cl`. For the definition of a public file, see "Audit Terminology and Concepts" on page 18.
seq	Adds a sequence token to every audit event.
trail	Adds a trailer token to every audit event.
zonename	Adds the zone name to every audit event. If the global zone is the only configured zone, adds the string `zone, global` to every audit event.

The following audit record shows the use of the `ls` command. The `ex` class is being audited and the default policy is in use:

```
header,129,2,AUE_EXECVE,,mach1,2010-10-14 11:39:22.480 -07:00
```

```
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
subject,jdoe,root,root,root,root,2404,50036632,82 0 mach1
return,success,0
```

The following is the same record when all policies are turned on:

```
header,1578,2,AUE_EXECVE,,mach1,2010-10-14 11:45:46.658 -07:00
path,/usr/bin/ls
attribute,100555,root,bin,21,320271,18446744073709551615
exec_args,2,ls,/etc/security
exec_env,49,MANPATH=/usr/share/man,USER=jdoe,GDM_KEYBOARD_LAYOUT=us,EDITOR=gedit,
LANG=en_US.UTF-8,GDM_LANG=en_US.UTF-8,PS1=#,GDMSESSION=gnome,SESSIONTYPE=1,SHLVL=2,
HOME=/home/jdoe,LOGNAME=jdoe,G_FILENAME_ENCODING=@locale,UTF-8, PRINTER=example-dbl,
...
path,/lib/ld.so.1
attribute,100755,root,bin,21,393073,18446744073709551615
subject,jdoe,root,root,root,root,2424,50036632,82 0 mach1
group,root,other,bin,sys,adm,uucp,mail,tty,lp,nuucp,daemon
return,success,0
zone,global
sequence,197
trailer,1578
```

- Use the `audit_syslog` plugin to send some audit events to `syslog`.

  Do not send those audit events to the `audit_binfile` or `audit_remote` plugin. This strategy works only if you are not required to keep binary records of the audit events that you send to the `syslog` logs.
- Set fewer system-wide audit flags and audit individual users.

  Reduce the amount of auditing for all users by reducing the number of audit classes that are audited system-wide.

  Use the `audit_flags` keyword to the `roleadd`, `rolemod`, `useradd`, and `usermod` commands to audit events for specific users and roles. For examples, see Example 24, "Specifying Audit Classes for `syslog` Output," on page 98 and the `usermod`(8) man page.

  Use the `always_audit` and `never_audit` properties of the `profiles` command to audit events for specific rights profiles. For information, see the `profiles`(1) man page.

Note - Like other security attributes, audit flags are affected by search order. For more information, see "Order of Search for Assigned Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

- Create your own customized audit class.

You can create audit classes at your site. Into these classes, put only those audit events that you need to monitor. For the procedure, see "How to Add an Audit Class" on page 61.

Note - For information about the effects of modifying an audit configuration file, see "Audit Configuration Files and Packaging" on page 139.

Binary Audit File Sizes Grow Without Limit

As an administrator who is assigned the Audit Review rights profile, you can limit the size of binary files to facilitate archiving and searching. You can also create smaller binary files from the original file by using one of the options described in this section.

- Use the `p_fsize` attribute to limit the size of individual binary audit files.

  For a description of the `p_fsize` attribute, see the OBJECT ATTRIBUTES section of the `audit_binfile`(7) man page.

  For an example, see Example 20, "Limiting File Size for the `audit_binfile` Plugin," on page 95.
- Use the `auditreduce` command to select records and write those records to a smaller file for further analysis.

  The `auditreduce -`*lowercase* options find specific records.

  The `auditreduce -`*Uppercase* options write your selections to a file. For more information, see the `auditreduce`(8) man page. See also "Displaying Audit Trail Data" on page 107.

Logins From Other Operating Systems Not Being Audited

The Oracle Solaris OS can audit all logins independent of source. If logins are not being audited, then the `lo` class for both attributable and non-attributable events is probably not set, This class audits logins, logouts, and screen locks. These classes are audited by default.

Note - To audit `ssh` logins, your system must be running the `ssh` daemon from Oracle Solaris. This daemon is modified for the audit service on an Oracle Solaris system. For more information, see *Managing Secure Shell Access in Oracle Solaris 11.4*.

EXAMPLE 45 Ensuring That Logins Are Audited

In this example, the output of the first two commands shows that the `lo` class for attributable and non-attributable events is not set. Then, the last two commands set the `lo` class to enable auditing of login events.

```
# auditconfig -getflags
active user default audit flags = as,st(0x20800,0x20800)
configured user default audit flags = as,st(0x20800,0x20800)

# auditconfig -getnaflags
active non-attributable audit flags = na(0x400,0x400)
configured non-attributable audit flags = na(0x400,0x400)

# auditconfig -setflags lo,as,st
user default audit flags = as,lo,st(0x21800,0x21800)

# auditconfig -setnaflags lo,na
non-attributable audit flags = lo,na(0x1400,0x1400)
```

`crontab` File Editing Fails With Audit Context Error

The following error indicates that you are not permitted to edit another user's `crontab` file:

```
not-sys# crontab -e sys
crontab: The audit context for your shell has not been set.
```

Even if audit logs are not being collected and the `crontab` process is not being audited, `crontab` only permits the `crontab` file owner, `sys` in the preceding command, to edit the file.

To verify that you are permitted to edit another user's `crontab` file, check the output of the following command, where *PID* is the process ID of your login shell. If this shell does not report a valid audit context, then `crontab` denies edits to the file.

```
auditconfig -getpinfo PID
```

To edit a `crontab` file that you do not own requires the `solaris.jobs.admin` authorization.

Best Practices for Auditing Core System Files

To satisfy a site security requirement to monitor and audit for changes to core Oracle Solaris system files, consider configuring security features in addition to the audit service. For example:

- Use the immutable zones feature – Enables you to configure system files to be read-only.

  See "Immutable Zones" in *Introduction to Oracle Solaris Zones*.
- Create and use rights profiles – Enables you to limit who can make configuration changes, and puts those changes in the audit record.

  See "Creating Rights Profiles and Authorizations" in *Securing Users and Processes in Oracle Solaris 11.4*.
- Use the `pfedit` command – Enables you to put the differences from an original system file and its edited version in the audit record.

Tip - Explicitly list the files an administrator is permitted to edit with `pfedit` in a rights profile.

- Use the Stop rights profile – Enables you to limit the commands a user or role can use to just those commands in the assigned rights profiles.

  See "Order of Search for Assigned Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.
- Use the `zfs diff` command – Enables you to view the differences between a ZFS dataset from one snapshot to the next snapshot.

  See "Identifying ZFS Snapshot Differences (zfs diff)" in *Managing ZFS File Systems in Oracle Solaris 11.4*.
- Use the `bart` command – Enables you to track differences in files between an initial `bart` report and subsequent `bart` reports.

  See Chapter 4, "Verifying File Integrity by Using BART" in *Securing Files and Verifying File Integrity in Oracle Solaris 11.4*.

♦ ♦ ♦ CHAPTER 7

Auditing Reference

This chapter describes the important components of auditing, and covers the following topics:

- "Audit Service" on page 136
- "Audit Service Man Pages" on page 136
- "Rights Profiles for Administering Auditing" on page 138
- "Auditing and Oracle Solaris Zones" on page 138
- "Audit Configuration Files and Packaging" on page 139
- "Audit Classes" on page 139
- "Audit Plugins" on page 140
- "Audit Remote Server" on page 141
- "Audit Policy" on page 141
- "Process Audit Characteristics" on page 143
- "Audit Trail" on page 144
- "Conventions for Binary Audit File Names" on page 144
- "Audit Record Structure" on page 144
- "Audit Token Formats" on page 146

For an overview of auditing, see Chapter 1, "About Auditing in Oracle Solaris". For planning suggestions, see Chapter 2, "Planning for Auditing". For procedures to configure auditing at your site, see the following chapters:

- Chapter 3, "Managing the Audit Service"
- Chapter 4, "Configuring the Formats of Audit Logs and Where They Are Stored"
- Chapter 5, "Viewing Audit Records"
- Chapter 6, "Analyzing and Resolving Audit Issues"

Audit Service

The audit service, `auditd`, is enabled by default. To find out how to enable, refresh, or disable the service, see "Enabling and Disabling the Audit Service" on page 49.

The audit service tracks auditable actions that occur on a system. These auditable actions are defined as "Audit Events" on page 21. Each audit event is connected to a system call or user command and is assigned to one or more "Audit Classes" on page 139.

Without customer configuration, the following defaults are in place:

- All login events are audited.

  Both successful and unsuccessful login attempts are audited.
- All users are audited for login and logout events, including role assumption and screen lock.
- The `audit_binfile` plugin is active. `/var/audit` stores audit records The size of an audit file is not limited.
- The `cnt` policy is set.

  When audit records fill the available disk space, the system tracks the number of dropped audit records. A warning is issued when one percent of available disk space remains.

To display the defaults, see "Displaying Audit Service Defaults" on page 48.

The audit service enables you to set temporary, or active, values. These values can differ from configured, or property, values.

- Temporary values are not restored when you refresh or restart the audit service.

  Audit policy accepts temporary values. Audit flags do not have a temporary value.
- Configured values are stored as property values of the service, so they are restored when you refresh or restart the audit service.

Rights profiles control who can administer the audit service. For more information, see "Rights Profiles for Administering Auditing" on page 138.

By default, all zones are audited identically. See "Auditing and Oracle Solaris Zones" on page 138.

Audit Service Man Pages

The following table summarizes the major administrative man pages for the audit service.

Man Page	Summary
`audit`(8)	Command that controls the actions of the audit service `audit -n` starts a new audit file for the `audit_binfile` plugin. `audit -s` enables and refreshes auditing. `audit -t` disables auditing. `audit -v` verifies that at least one plugin is active.
`audit_binfile`(7)	Default audit plugin, which sends audit records to a binary file. See also "Audit Plugins" on page 140.
`audit_remote`(7)	Audit plugin that sends audit records to a remote receiver.
`audit_syslog`(7)	Audit plugin that sends text summaries of audit records to the `syslog` utility.
`audit_class`(5)	File that contains the definitions of audit classes. The eight high-order bits are available for customers to create new audit classes. For more information about the effect of modifying this file on system upgrade, see "How to Add an Audit Class" on page 61.
`audit_event`(5)	File that contains the definitions of audit events and maps the events to audit classes. The mapping can be modified. For more information about the effect of modifying this file on system upgrade, see "How to Change an Audit Event's Class Membership" on page 63.
`audit_flags`(7)	Describes the syntax of audit class preselection, the prefixes for selecting only failed events or only successful events, and the prefixes that modify an existing preselection.
`audit.log`(5)	Describes the naming of binary audit files, the internal structure of a file, and the structure of every audit token.
`audit_warn`(8)	Script that notifies an email alias when the audit service encounters an unusual condition while writing audit records. You can customize this script for your site to warn of conditions that might require manual intervention or can specify how to handle those conditions automatically.
`auditconfig`(8)	Command that retrieves and sets audit configuration parameters. Issue this `auditconfig` with no options to display a list of parameters that can be retrieved and set.
`auditrecord`(8)	Command that displays the definition of audit events in the `/etc/security/audit_event` file. For sample output, see "Displaying Audit Record Definitions" on page 107.
`auditreduce`(8)	Command that post-selects and merges audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The records remain in binary format. Uppercase options affect file selection. Lowercase options affect record selection.
`auditstat`(8)	Command that displays kernel audit statistics. For example, the command can display the number of dropped records and the number of audit records that user processes produced in the kernel as a result of system calls.
`praudit`(8)	Command that reads audit records in binary format from standard input and displays the records in a presentable format. The input can be piped from the `auditreduce` command or from a single audit file or a list of audit files. Input can also be produced with the `tail -0f` command for a current audit file. For sample output, see "Viewing the Contents of Binary Audit Files" on page 111.

Rights Profiles for Administering Auditing

Oracle Solaris provides rights profiles for configuring the audit service, for enabling and disabling the service, and for analyzing the audit trail. You must have the privileges of `root` to edit an audit configuration file.

- **Audit Configuration –** Enables an administrator to configure the parameters of the audit service and to run the `auditconfig` command.
- **Audit Control –** Enables an administrator to start, refresh, and disable the audit service and to run the `audit` command to start, refresh, or stop the service.
- **Audit Review –** Enables an administrator to analyze audit records. This rights profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. This administrator can also run the `auditstat` command.
- **System Administrator –** Includes the Audit Review rights profile. An administrator with the System Administrator rights profile can analyze audit records.

To configure roles to handle the audit service, see "Creating a Role" in *Securing Users and Processes in Oracle Solaris 11.4*.

Auditing and Oracle Solaris Zones

Non-global zones can be audited exactly as the global zone is audited, or non-global zones can set their own flags, storage, and audit policy.

When all zones are being audited identically through the global zone, the `audit_class` and `audit_event` files provide the class-event mappings for auditing in the global zone and in every non-global zone. The `+zonename` policy option is useful for post-selecting records by zone name.

Zones can also be audited individually. When the policy option, `perzone`, is set in the global zone, each non-global zone runs its own audit service, handles its own audit queue, and specifies the content and location of its audit records. A non-global zone can also set most audit policy options. It cannot set policy that affects the entire system, so a non-global zone cannot set the `ahlt` or `perzone` policy. For further discussion, see "Auditing on a System With Oracle Solaris Zones" on page 18 and "Planning Auditing in Zones" on page 34.

To learn about zones, see *Introduction to Oracle Solaris Zones*.

Audit Configuration Files and Packaging

The audit configuration files in Oracle Solaris are marked in the package with the `preserve=renamenew` package attribute. This attribute enables the files to be modified, and the modifications are retained across package updates and package fixes. For information about the effects of the `preserve` values, see the `pkg`(7) man page.

These configuration files are also marked with the `overlay=allow` package attribute. This attribute enables you to create your own package that contains these files and replace the Oracle Solaris files with files from your package. When you set the `overlay` attribute to `true` in your package, the `pkg` subcommands, such as `verify`, `fix`, `revert`, and so on, will return results on your packages. For more information, see the `pkg`(1) and `pkg`(7) man pages.

Audit Classes

Oracle Solaris defines audit classes as convenient containers for large numbers of audit events.

You can reconfigure audit classes and make new audit classes. Audit class names can be up to 8 characters in length. The class description is limited to 72 characters. Numeric and non-alphanumeric characters are allowed. For more information, see the `audit_class`(5) man page and "How to Add an Audit Class" on page 61.

Caution - The `all` class can generate large amounts of data and quickly fill disks. Use the `all` class only if you have extraordinary reasons to audit all activities.

Audit Class Syntax

Events in an audit class can be audited for success, for failure, and for both.

- Without a prefix, a class of events is audited for success and for failure.
- With a plus (+) prefix, a class of events is audited for success only.
- With a minus (-) prefix, a class of events is audited for failure only.
- To modify a current preselection, add a caret (^) preceding a prefix or an audit flag. For example:
  - If `ot` is preselected for the system, and a user's preselection is `^ot`, that user is not audited for events in the `other` class.

- If `+ot` is preselected for the system, and a user's preselection is `^+ot`, that user is not audited for successful events in the `other` class.
- If `-ot` is preselected for the system, and a user's preselection is `^-ot`, that user is not audited for failed events in the `other` class.

To review the syntax of audit class preselection, see the `audit_flags(7)` man page.

The audit classes and their prefixes can be specified in the following commands:

- As arguments to the `auditconfig` command options `-setflags` and `-setnaflags`.
- As values for the `p_flags` attribute to the `audit_syslog` plugin. You specify the attribute as an option to the `auditconfig -setplugin audit_syslog active` command.
- As values for the `-K audit_flags=`*always-audit-flags*`:`*never-audit-flags* option to the `useradd`, `usermod`, `roleadd`, and `rolemod` commands.
- As values for the `always_audit` and `never_audit` properties of the `profiles` command.

Audit Plugins

Audit plugins specify how to handle the audit records in the audit queue. The audit plugins are specified by name: `audit_binfile`, `audit_remote`, and `audit_syslog`, as arguments to the `auditconfig -setplugin` command. The plugins can be further specified by the following attributes:

- `audit_binfile` plugin

  `p_dir` attribute – Where to send binary data

  `p_minfree` attribute – Minimum space remaining on a disk before the administrator is warned.

  `p_fsize` attribute – Maximum size of an audit file.
- `audit_remote` plugin

  `p_hosts` attribute– Remote authenticated audit server to which to send the binary audit data.

  `p_retries` attribute – Number of attempts to make to reach a remote authenticated audit server.

  `p_timeout` attribute – Number of seconds between attempts to reach a remote authenticated audit server.
- `audit_syslog` plugin

  `p_flags` attribute – Selection of text summaries of audit records to be sent to `syslog`

Refer to the `audit_binfile`(7), `audit_remote`(7), `audit_syslog`(7), and `auditconfig`(8) man pages.

Audit Remote Server

The audit remote server (ARS) receives audit records over a secure link from audited systems and stores the records.

The reception relies on the following being configured:

- A Kerberos realm with specific audit principals and a GSS-API mechanism
- The ARS with at least one configured and active *connection group*
- At least one audited system in the connection group and a configured and active `audit_remote` plugin

A connection group is specified in the `group` property of the ARS. For file management, `group` can limit the size of an audit file and specify the minimum free space. The primary reason to specify different connection groups is to specify different storage locations on the ARS, as shown in Example 26, "Streaming Audit Records to Different File Locations on the Same ARS," on page 102.

For more information about the ARS, see the `ars`(7) man page. For ARS configuration information, see the `-setremote` option in the `auditconfig`(8) man page.

To configure the audited systems, see the `audit_remote`(7) man page and the `-setplugin` option in the `auditconfig`(8) man page.

Audit Policy

Audit policy determines whether additional information is added to the audit trail.

The following policies add tokens to audit records: `arge`, `argv`, `group`, `path`, `seq`, `trail`, and `zonename`.

The remaining policies do not add tokens. The `public` policy limits auditing of public files. The `perzone` policy establishes separate audit queues for non-global zones. The `ahlt` and `cnt` policies determine what happens when audit records cannot be delivered. For details, see "Audit Policies for Asynchronous and Synchronous Events" on page 142.

The effects of the different audit policy options are described in "Understanding Audit Policy" on page 40. For a description of audit policy options, see the `-setpolicy` option in the `auditconfig(8)` man page. For a list of available policy options, run the command `auditconfig -lspolicy`. For the current policy, run the command `auditconfig -getpolicy`.

Audit Policies for Asynchronous and Synchronous Events

Together, the `ahlt` policy and the `cnt` policy govern what happens when the audit queue is full and cannot accept more events. The policies include specifics for an asynchronous audit event or a synchronous audit event.

Note - The `cnt` or `ahlt` policies are not triggered if the queue for at least one plugin can accept audit records.

The `cnt` and `ahlt` policies are independent and related. The combination of the policies has the following effects:

- `-ahlt +cnt` is the default policy that is shipped. This default allows the processing of an audited event even if the event cannot be logged.

  The `-ahlt` policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, the system will count the events and continue processing.

  The `+cnt` policy states that if a synchronous event arrives and the event cannot be placed in the kernel audit queue, the system will count the event and continue processing.

  The `-ahlt +cnt` configuration is generally used at sites where processing must continue, even if continued processing could result in a loss of audit records. The `auditstat drop` field shows the number of audit records that are dropped in a zone.
- The `+ahlt -cnt` policy states that processing halts when an asynchronous event cannot be added to the kernel audit queue.

  The `+ahlt` policy states that if an audit record of an asynchronous event cannot be placed in the kernel audit queue, all processing is stopped. The system will panic. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

  The `-cnt` policy states that if a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

The `+ahlt -cnt` configuration is generally used at sites where a record of every audit event takes precedence over system availability. The `auditstat` `wblk` field shows the number of times that threads were blocked.

However, if an asynchronous event occurs, the system will panic, leading to an outage. The kernel queue of audit events can be manually recovered from a saved crash dump. The asynchronous event will not be in the audit queue and must be recovered from pointers on the call stack.

- The `-ahlt -cnt` policy states that if an asynchronous event cannot be placed in the kernel audit queue, the event will be counted and processing will continue. When a synchronous event cannot be placed in the kernel audit queue, the thread that is attempting to deliver the event will be blocked. The thread is placed in a sleep queue until audit space becomes available. No count is kept. Programs might appear to hang until audit space becomes available.

  The `-ahlt -cnt` configuration is generally used at sites where the recording of all synchronous audit events takes precedence over some potential loss of asynchronous audit records. The `auditstat` `wblk` field shows the number of times that threads were blocked.
- The `+ahlt +cnt` policy states that if an asynchronous event cannot be placed in the kernel audit queue, the system will panic. If a synchronous event cannot be placed in the kernel audit queue, the system will count the event and continue processing.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask –** A combination of the system-wide audit mask and the user-specific audit mask, if a user audit mask has been specified. When a user logs in, the login process combines the preselected classes to establish the *process preselection mask* for the user's processes. The process preselection mask specifies the events that generate audit records.

  In addition, as described in the `audit_flags(7)` man page, the preselection can specify auditing only successful events, auditing only failed events, or auditing all events.

  The following algorithm describes how the system obtains the user's process preselection mask:

  `(system-wide default flags  +` *always-audit-classes*`) -` *never-audit-classes*

  Add the system-wide audit classes from the results of the `auditconfig -getflags` command to the classes from the *always-audit-classes* value for the user's `always_audit` keyword. Then, from the total, subtract the classes from the user's *never-audit-classes*. See also the `audit_flags`(7) man page.

- **Audit user ID –** A process acquires an *immutable audit user ID* (auid) when the user logs in. This ID is inherited by all child processes that were started by the user's initial process. The audit user ID helps enforce accountability. Even after a user assumes a role, the audit user ID remains the same. The audit user ID that is saved in each audit record enables you to always trace actions back to the login user.
- **Audit session ID –** The audit session ID is assigned at login. This ID is inherited by all child processes.
- **Terminal ID –** For a local login, the terminal ID consists of the local system's IP address, followed by a device number that identifies the physical device on which the user logged in. Most often, the login is through the console. The number that corresponds to the console device is `0,0`. For a remote login, the terminal ID consists of a the remote host's IP address followed by the remote port number and the local port number.

Audit Trail

The *audit trail* contains binary audit files. The trail is created by the `audit_binfile` plugin. The audit service collects the records in the audit queue and sends them to the plugin, which writes them to disk.

Conventions for Binary Audit File Names

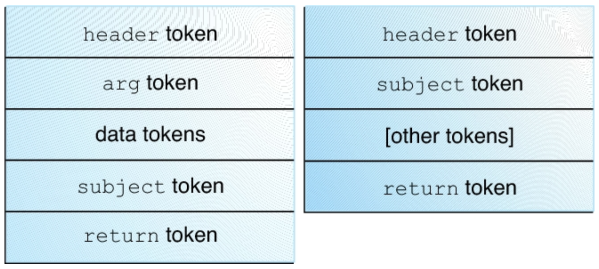
The `audit_binfile` plugin creates binary audit files. Each binary audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the system that generated them. The time stamps that indicate the time span are specified in Coordinated Universal Time (UTC) to ensure that they sort in proper order, even across time zones.

For more information, see the `audit.log`(5) man page. For examples of open and closed audit file names, see "How to Clean Up a `not_terminated` Audit File" on page 118.

Audit Record Structure

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

FIGURE 3 Typical Audit Record Structures



Audit Record Analysis

Audit record analysis involves post-selecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can use the `praudit` command. Options to the command provide different text output. For example, the `praudit` command provides XML for input into scripts and browsers. `praudit` output does not include fields whose sole purpose is to help to parse the binary data. Note that the order and format of `praudit` output is not guaranteed between Oracle Solaris releases.

  For examples of `praudit` output, see "Viewing the Contents of Binary Audit Files" on page 111.

  For examples of `praudit` output for each audit token, see the individual tokens in "Audit Token Formats" on page 146.
- You can write a program to parse the binary data stream. The program must take into account the variants of an audit record. For example, the `ioctl()` system call creates an audit record for "Bad file name". This record contains different tokens from the `ioctl()` audit record for "Invalid file descriptor".
  - For a description of the order of binary data in each audit token, see the `audit.log`(5) man page.
  - For manifest values, see the `/usr/include/bsm/audit.h` file.
  - To view the order of tokens in an audit record, use the `auditrecord` command. Output from the `auditrecord` command includes the different tokens for different

manifest values. Square brackets (`[]`) indicate that an audit token is optional. For more information, see the `auditrecord`(8) man page.

Audit Token Formats

Each audit record contains a series of audit tokens that describe the attributes of that record. Every audit record begins with a `header` token, which indicates where the audit record begins in the audit trail. The header token is followed by various data tokens that describe attributes such as user ID, process ID or session ID. Depending on the audit policy, optional tokens may be included in an audit record.

Oracle Solaris adds tokens to record sensitive information in new features. For example, in this release, the new tokens include `annotation` and `cipher`.

To view the list of tokens and their definitions, see the `audit.log`(5) man page.

Note - Obsolete tokens are maintained for compatibility with previous Solaris releases. For information about obsolete tokens, see the reference material for the release that included the token.

Audit Service Glossary

These glossary entries cover words that can be ambiguous because they are used differently in different parts of the operating system, or have meanings in Oracle Solaris that are distinct from other operating systems.

asynchronous audit event	Asynchronous events are the minority of audit events. These system events are not associated with any process, so no process is available to be blocked and later woken up. Initial system boot and PROM enter and exit events are examples of asynchronous events. See audit event.
attributable audit event	An audit event on a system that can be attributed to a user.
audit class	A convenient container for large numbers of audit events. See audit event.
audit event	Each audit event represents an auditable action on a system. Each audit event is connected to a system call or user command, and is assigned to one or more audit classes. An audit event can be either an asynchronous audit event or a non-attributable audit event. Also see attributable audit event.
audit files	Binary audit logs. Audit files are stored separately in an audit file system.
audit policy	The global and per-user settings that determine which audit events are recorded. The global settings that apply to the audit service typically affect which pieces of optional information are included in the audit trail. Two settings, `cnt` and `ahlt`, affect the operation of the system when the audit queue fills. For example, audit policy might require that a sequence number be part of every audit record.
audit trail	The collection of all audit files from all systems.
non-attributable audit event	An audit event whose initiator cannot be determined, such as the AUE_BOOT event.
policy	Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. For example, security

	policy might require that systems be audited, that devices must be allocated for use, and that passwords be changed every six weeks. See audit policy.
public object	A file that is owned by the `root` user and readable by the world, such as any file in the `/etc` directory.
rights	An alternative to the all-or-nothing superuser model. User rights management and process rights management enable an organization to divide up superuser's privileges and assign them to users or roles. Rights in Oracle Solaris are implemented as kernel privileges, authorizations, and the ability to run a process as a specific UID or GID. Rights can be collected in a rights profile and a role.
security policy	See policy.
single-system image	A single-system image is used in Oracle Solaris auditing to describe a group of audited systems that use the same naming service. These systems send their audit records to a central audit server, where the records can be compared as if the records came from one system.
synchronous audit event	The majority of audit events. These events are associated with a process in the system. A non-attributable event that is associated with a process is a synchronous event, such as a failed login. See audit event.

Index