eTrust® Access Control for UNIX and Linux

Administrator Guide

r8 SP1
This documentation and any related computer software help programs (hereinafter referred to as the “Documentation”) is for the end user’s informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the documentation for their own internal use, and may make one copy of the related software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the product are permitted to have access to such copies.

The right to print copies of the documentation and to make a copy of the related software is limited to the period during which the applicable license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user’s responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of any product referenced in the Documentation is governed by the end user’s applicable license agreement.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2006 CA. All rights reserved.
CA Product References

This document references the following CA products:

- eTrust® Access Control (eTrust AC)
- eTrust® Single Sign-On (eTrust SSO)
- eTrust® Web Access Control (eTrust Web AC)
- eTrust® CA-Top Secret®
- eTrust® CA-ACF2®
- eTrust® Audit
- Unicenter® TNG
- Unicenter® Network and Systems Management (Unicenter NSM)
- Unicenter® Software Delivery

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.
Contents

Chapter 1: Introduction ................................................................. 13
About this Guide ........................................................................ 13
Who Should Use this Guide ......................................................... 13
Command Notation Conventions .................................................. 13

Chapter 2: Basic Concepts ............................................................ 15
eTrust AC .................................................................................. 15
What Is Access Control? ............................................................. 16
  Why Does UNIX Need Protecting? ........................................... 16
  How Does This Work? ............................................................... 17
  What Is Protected? .................................................................. 17
  How Is It Protected? ............................................................... 20
What Is an Access Rule? ............................................................. 21
  Access Control Lists ............................................................... 21
  Default Access Field ............................................................. 21
  Default Record for Class ........................................................ 22
  UACC Class ........................................................................... 22
  Conditional Access Control Lists ......................................... 23
  Negative Access Control Lists (NACLs) ............................... 24
  Accumulative Group Rights .................................................... 24
  Security Levels ...................................................................... 25
  Security Categories ............................................................... 25
  Security Labels ....................................................................... 25
  Generic Access Rules ............................................................. 26
Graphical Access Summary .......................................................... 27

Chapter 3: Users, Groups, and Resources ...................................... 45
Introduction to Users and Groups .................................................. 45
  Users .................................................................................... 45
  Groups .................................................................................. 46
  Determining Access: A Summary ............................................. 48
Modifying User Records ............................................................. 48
  Examples: Create or Modify Users ......................................... 49
  Special Predefined Users ......................................................... 49
Modifying Group Records .......................................................... 50
  Nested Groups ...................................................................... 50
Special Predefined Groups ................................................................. 50
Examples: Create or Modify Group Records ................................................ 51
What Is a Resource? .............................................................................. 52
A Special Resource: Abstract Object ..................................................... 52
Records and ACLs ............................................................................... 53
Examples: Setting ACLs ....................................................................... 53
Classes ................................................................................................. 54
Predefined Classes ............................................................................... 54
User-Defined Classes .......................................................................... 57

Chapter 4: Protecting Accounts ................................................................. 59
Why Protect Accounts? ......................................................................... 59
Protecting Substitute User (su) Requests ................................................ 59
Setting Up the Surrogate DO Facility ...................................................... 62
Surrogating Safely with sesu ................................................................. 63
Preventing Password Attacks .............................................................. 64
serevu ................................................................................................. 64
pam_seos ............................................................................................ 65
Restrictions and Limitations ................................................................... 66
Checking User Inactivity ....................................................................... 66

Chapter 5: Password Control ................................................................. 69
Password Control .................................................................................. 69
Defining Password Policies ................................................................... 70
Setting Up Password Quality Checking ............................................... 71
Implementing Password Policy Rules ................................................... 72
Password Expiration and Grace Logins ............................................... 72
Specifying the Password Interval ......................................................... 73
Specifying Grace Logins ....................................................................... 74

Chapter 6: Protecting Files and Programs .................................................. 77
Restricting Access to Files and Directories .............................................. 77
How File Protection Works .................................................................... 80
Protecting Files .................................................................................... 81
Restricting File Access .......................................................................... 82
Blocking Trojan Horses with the _abspath Group .................................. 85
Synchronization with Native UNIX Security ......................................... 86
Example: Synchronization ..................................................................... 87
HP-UX Limitations ................................................................................ 88
Sun Solaris Limitations ......................................................................... 88
Monitoring Sensitive Files ................................................................................................. 89
Protecting setuid and setgid Programs ............................................................................. 90
  Conditional Access ........................................................................................................ 92
  Protecting the Login Command ..................................................................................... 92
Protecting Regular Programs ............................................................................................ 92
Protecting Binary Files from the kill Command ............................................................... 93

Chapter 7: Controlling Login Commands ........................................................................ 95
Controlling the Login Process .......................................................................................... 95
  Examples: LOGINAPPL ................................................................................................. 95
Controlling Generic Login Applications .......................................................................... 96
  Defining a Generic Login Application ......................................................................... 96
  Generic Login Program Interception .......................................................................... 97
Defining User Authority to Use Terminals ...................................................................... 97
  Restricting Terminals for Root Users .......................................................................... 99
  Recommended Restrictions ......................................................................................... 100
Password Checking and Login Restrictions .................................................................... 101
  Logon Checks ............................................................................................................. 102
Defining Time and Day Login Rules .............................................................................. 103
Disabling Concurrent Logins .......................................................................................... 103
Limiting Concurrent Logins for a User .......................................................................... 104
  Limiting Concurrent Logins Globally ........................................................................ 104
  Limiting Concurrent Logins Individually ................................................................... 105
Recognizing a Login Event .............................................................................................. 106

Chapter 8: Protecting TCP/IP Services .......................................................................... 107
Why Protect TCP/IP Services .......................................................................................... 107
Restricting TCP/IP Services ............................................................................................ 107
Using the TCP Class ........................................................................................................ 110
  Streams Module for Network Interception ................................................................. 111

Chapter 9: Managing Policies Centrally .......................................................................... 117
The Policy Model Database ............................................................................................. 117
  PMDB Location on Disk ............................................................................................. 118
  Managing Local PMDBs .............................................................................................. 118
  Managing Remote PMDBs ......................................................................................... 119
Architecture Dependency ................................................................................................. 120
Methods for Centrally Managing Policies ..................................................................... 122
  Automatic Rule-based Policy Updates ..................................................................... 122
  How Automatic Rule-based Policy Updates Work ..................................................... 123
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How You Can Set Up a Hierarchy</td>
<td>123</td>
</tr>
<tr>
<td>UID/GID Synchronization</td>
<td>129</td>
</tr>
<tr>
<td>Update Subscribers</td>
<td>130</td>
</tr>
<tr>
<td>Dual Control</td>
<td>138</td>
</tr>
<tr>
<td>Using the seagent and sepmdd Daemons</td>
<td>143</td>
</tr>
<tr>
<td>Advanced Policy Management and Reporting</td>
<td>144</td>
</tr>
<tr>
<td>Environment Architecture</td>
<td>144</td>
</tr>
<tr>
<td>How You Set Up a Hierarchy for Advanced Policy-based Management and Reporting</td>
<td>148</td>
</tr>
<tr>
<td>How Advanced Policy-based Management Works</td>
<td>151</td>
</tr>
<tr>
<td>How Advanced Policy Reporting Works</td>
<td>161</td>
</tr>
<tr>
<td>How Policy Deviation Calculations Work</td>
<td>168</td>
</tr>
<tr>
<td>Mainframe Password Synchronization</td>
<td>172</td>
</tr>
<tr>
<td>Chapter 10: General Security Features</td>
<td>173</td>
</tr>
<tr>
<td>Locking Idle Stations</td>
<td>173</td>
</tr>
<tr>
<td>Protecting Resources Using APIs</td>
<td>174</td>
</tr>
<tr>
<td>Protecting Against Stack Overflow: STOP</td>
<td>175</td>
</tr>
<tr>
<td>Starting and Stopping STOP</td>
<td>175</td>
</tr>
<tr>
<td>Defining Day and Time Access Rules for Resources</td>
<td>176</td>
</tr>
<tr>
<td>B1 Security Level Certification</td>
<td>176</td>
</tr>
<tr>
<td>Security Levels</td>
<td>176</td>
</tr>
<tr>
<td>Security Categories</td>
<td>177</td>
</tr>
<tr>
<td>Security Labels</td>
<td>179</td>
</tr>
<tr>
<td>Chapter 11: Auditing Events</td>
<td>181</td>
</tr>
<tr>
<td>Setting Audit Rules</td>
<td>181</td>
</tr>
<tr>
<td>Using the Warning Mode</td>
<td>182</td>
</tr>
<tr>
<td>Audit Logs</td>
<td>183</td>
</tr>
<tr>
<td>The System Auditor</td>
<td>184</td>
</tr>
<tr>
<td>Chapter 12: Remote Status View</td>
<td>187</td>
</tr>
<tr>
<td>Requirements</td>
<td>187</td>
</tr>
<tr>
<td>Installing and Starting RSV</td>
<td>187</td>
</tr>
<tr>
<td>Installation</td>
<td>188</td>
</tr>
<tr>
<td>Starting</td>
<td>188</td>
</tr>
<tr>
<td>Displaying</td>
<td>189</td>
</tr>
<tr>
<td>Status Categories</td>
<td>190</td>
</tr>
<tr>
<td>Security Status Summary</td>
<td>190</td>
</tr>
<tr>
<td>File Protection</td>
<td>191</td>
</tr>
<tr>
<td>Program Protection</td>
<td>192</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Account Protection</td>
<td>193</td>
</tr>
<tr>
<td>Password Control</td>
<td>193</td>
</tr>
<tr>
<td>Optional Services</td>
<td>194</td>
</tr>
<tr>
<td>Using RSV</td>
<td>194</td>
</tr>
<tr>
<td>RSV Security</td>
<td>195</td>
</tr>
<tr>
<td>Chapter 13: Scope of Administration Authority</td>
<td>197</td>
</tr>
<tr>
<td>Global Authorization Attributes</td>
<td>197</td>
</tr>
<tr>
<td>ADMIN Attribute</td>
<td>197</td>
</tr>
<tr>
<td>AUDITOR Attribute</td>
<td>198</td>
</tr>
<tr>
<td>OPERATOR Attribute</td>
<td>198</td>
</tr>
<tr>
<td>PWMANAGER Attribute</td>
<td>198</td>
</tr>
<tr>
<td>SERVER Attribute</td>
<td>199</td>
</tr>
<tr>
<td>IGN_HOL Attribute</td>
<td>199</td>
</tr>
<tr>
<td>Group Authorization</td>
<td>199</td>
</tr>
<tr>
<td>Parentage</td>
<td>200</td>
</tr>
<tr>
<td>Group Authorization Attributes</td>
<td>200</td>
</tr>
<tr>
<td>Ownership</td>
<td>203</td>
</tr>
<tr>
<td>File Ownership</td>
<td>204</td>
</tr>
<tr>
<td>Authorization Examples</td>
<td>204</td>
</tr>
<tr>
<td>Single Group Authorization</td>
<td>204</td>
</tr>
<tr>
<td>Parent and Child Groups</td>
<td>205</td>
</tr>
<tr>
<td>The ADMIN Class</td>
<td>206</td>
</tr>
<tr>
<td>Environmental Considerations</td>
<td>207</td>
</tr>
<tr>
<td>Remote Administration Restrictions</td>
<td>207</td>
</tr>
<tr>
<td>UNIX Environment</td>
<td>208</td>
</tr>
<tr>
<td>Chapter 14: Improving Performance</td>
<td>209</td>
</tr>
<tr>
<td>Using Global Access Check</td>
<td>210</td>
</tr>
<tr>
<td>How Does GAC Work?</td>
<td>211</td>
</tr>
<tr>
<td>Implementing GAC</td>
<td>211</td>
</tr>
<tr>
<td>GAC Restrictions</td>
<td>213</td>
</tr>
<tr>
<td>Troubleshooting GAC</td>
<td>214</td>
</tr>
<tr>
<td>Using the Resource Cache</td>
<td>214</td>
</tr>
<tr>
<td>Tuning Recommendations</td>
<td>215</td>
</tr>
<tr>
<td>Using the Network Cache</td>
<td>215</td>
</tr>
<tr>
<td>Using the Real Path Cache</td>
<td>215</td>
</tr>
<tr>
<td>Using Fork Synchronization</td>
<td>216</td>
</tr>
<tr>
<td>Using High Priority</td>
<td>216</td>
</tr>
<tr>
<td>Bypassing the Process File System</td>
<td>217</td>
</tr>
<tr>
<td>Bypassing Real Paths</td>
<td>217</td>
</tr>
</tbody>
</table>
Bypassing Trusted Process Authorization ............................................................. 218
Bypassing Ports for Network Activity ................................................................. 218
Reducing Audit and Trace Loads .................................................................... 218
Reducing Database Loads ................................................................................. 219
Improving PMDB Updates ................................................................................. 219
Improving Watchdog Performance .................................................................. 220
Improving Class Parameters ............................................................................ 220
  Class Activation ............................................................................................. 220
  Class Authorization ....................................................................................... 220
Resolving Names ............................................................................................... 221

Chapter 15: Using UNIX Exits ........................................................................ 223
UNIX Exits ....................................................................................................... 223
User or Group Record Update Exits ................................................................ 224
  How the Provided selang Exit Script Works ................................................ 224
  Arguments You Can Pass to selang Exits .................................................... 227
  Specify selang Exit Programs to Run ........................................................... 228
  Time Out and Other Failures ........................................................................ 229
  selang Exit Samples ..................................................................................... 229
eTrust AC Kernel Loader Exits ........................................................................ 229
  How the Kernel Loading Exits Work ........................................................... 230
  How the Kernel Unloading Exits Work ......................................................... 231

Chapter 16: Interacting with LDAP .................................................................. 233
Transferring User Names .................................................................................. 233
ldap2seos ......................................................................................................... 234
seos2ldap ......................................................................................................... 236
S50CREATE_u_LdapE ....................................................................................... 238

Chapter 17: Unicenter Security Migration and Integration ............................... 239
Installing Unicenter Security Integration Tools .............................................. 239
Unicenter Security Integration Features .......................................................... 239
  SSF/EMSec API Support .............................................................................. 240
  eTrust AC to Unicenter Security Synchronization Utility ............................. 241
Unicenter Security Data Migration Features .................................................... 244
  Unicenter Security Options Migration ........................................................ 244
  Unicenter Security Database Migration ....................................................... 245
Unicenter TNG User Exit Support .................................................................... 247
  Use a PMDB with Unicenter Security Objects .......................................... 248
Unicenter TNG Calendar ................................................................................... 249
Certification with Unicenter TNG and Unicenter NSM .................................................. 251
Audit Events Integration ............................................................................................... 251

**Appendix A: NIS Configuration** 253

- Installation Notes ........................................................................................................ 253
- Name Resolution .......................................................................................................... 254
  - Name Resolution on an NIS/DNS Client ............................................................... 254
  - Name Resolution on a Server: Deadlock ............................................................... 255
  - Name Resolution on Sun Solaris: Deadlock ........................................................... 255
- Avoiding Deadlocks: The Lookaside Database .......................................................... 256
  - Storing Resolution Tables on Disk ...................................................................... 256
  - Setting Up the Lookaside Database ....................................................................... 256
  - How the Lookaside Database Works ..................................................................... 258
  - Implementing the Lookaside Database .................................................................. 258
  - Updating the Hosts Lookaside Table .................................................................... 258
- Configuration Tokens: The seos.ini File ................................................................. 259

**Appendix B: Password Synchronization with Mainframes** 261

- Password Synchronization Support ........................................................................... 261
- Password Policy Model Methods ............................................................................... 261
- Installing Password Synchronization ......................................................................... 262
  - Installation Requirements on the Mainframe ...................................................... 262
  - Installation Requirements on UNIX ....................................................................... 263
  - Checking the Installation ....................................................................................... 264
  - Configuring the mfsd ............................................................................................. 264
  - Configuring the Translation File .......................................................................... 265
  - Defining Exit Functions .......................................................................................... 266
  - Completing the Policy Model Configuration ....................................................... 267
- Starting Mainframe Synchronization ......................................................................... 270
- The CAICCI Configuration File ................................................................................. 273

**Appendix C: eTrust Audit Integration** 275

- eTrust Audit Integration ............................................................................................. 275
- eTrust AC Logs ............................................................................................................ 276
- Configuring eTrust AC ............................................................................................... 276
  - For UNIX .................................................................................................................. 276
  - eTrust Audit Configuration ...................................................................................... 277
- Collecting eTrust AC for UNIX Logs into Audit ....................................................... 278
- Collecting eTrust AC for UNIX logs, syslogs, and sulogs into eTrust Audit ............ 278
- Collecting eTrust AC for Windows Logs into eTrust Audit ...................................... 279
Chapter 1: Introduction

This section contains the following topics:

About this Guide (see page 13)
Who Should Use this Guide (see page 13)
Command Notation Conventions (see page 13)

About this Guide

This guide describes the concepts used by eTrust AC for UNIX—a product that provides a total security solution for open systems. The guide emphasizes the eTrust AC database.

Who Should Use this Guide

This guide was written for security and system administrators who are implementing and maintaining an eTrust AC-protected environment.

Command Notation Conventions

The eTrust AC documentation uses a few special conventions when explaining command syntax and user input:

<table>
<thead>
<tr>
<th>Format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono-spaced font</td>
<td>Code or program output</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Placeholder for information that you must supply</td>
</tr>
<tr>
<td><strong>Bold</strong></td>
<td>Elements that you must type exactly as shown</td>
</tr>
<tr>
<td>Between square brackets ([])</td>
<td>Optional items</td>
</tr>
<tr>
<td>Between braces ({}) choices separated by pipe (</td>
<td>)</td>
</tr>
<tr>
<td>Space and a backslash at end of line ( \ )</td>
<td>Command continues on the following line</td>
</tr>
</tbody>
</table>
**Notes:**

- Bold text is also used for simple emphasis. For example:
  
  You should **never** tape your password to the monitor.

- Sometimes a command does not fit on a single line in this guide. In these cases, a space followed by a backslash (\) at the end of a line indicates that the command continues on the following line.

  **Note:** Avoid copying the backslash character as it is not needed in the actual command syntax.

- A pipe (|) separates mutually exclusive items. The set of items is enclosed in braces ({}), which you are **not** intended to type when you type one of the items. For example, the following means **either** a user name **or** a group name:
  
  {username|groupname}

**Example: command notation conventions**

The following code illustrates how command conventions are used in this guide:

```ruler
className [props({all}|{propertyName1[,propertyName2]})]
```

In this example:

- The command name (ruler) is shown in bold as it must be typed as shown.
- The `className` option is in italic as it is a placeholder for a class name (for example, USER).
- You can run the command without the second part enclosed in square brackets, which is optional.
- When using the optional parameter (props), you can choose the keyword *all* or, specify one or more property names separated by a comma.
Chapter 2: Basic Concepts

This section contains the following topics:

- **eTrust AC** (see page 15)
- **What Is Access Control?** (see page 16)
- **What Is an Access Rule?** (see page 21)
- **Graphical Access Summary** (see page 27)

**eTrust AC**

eTrust AC is a software product that is an active, comprehensive security software solution for Open Systems, tied dynamically to the operating system. Each time a user requests a security-sensitive operation—such as opening a file, substituting a user ID, or obtaining a network service—eTrust AC can intercept the event in real time and evaluate its validity before passing control to the standard operating system (OS) functions.
What Is Access Control?

eTrust AC protects the information assets of computer centers. It checks whether users who request services from the host operating system are authorized to access those services. Because eTrust AC is conceptually similar in structure to mainframe access control products, information centers can keep well-established security procedures, regulations, and policies while integrating UNIX-based operating systems.

The eTrust AC package includes the Security Administrator, a graphical user interface (GUI) from which you can define users, groups, and access rules in both the native eTrust AC database and databases of other security platforms in use at the site. You can also use it to monitor and audit events of interest.

Why Does UNIX Need Protecting?

Many operating systems have built-in access control, using one technique or another. IBM's MVS, a well-established and mature mainframe operating system, includes the System Authorization Facility (SAF)-a set of calls issued by the operating system itself to verify a user's authorization.

Access control software in an MVS environment sets a return code for the SAF call and MVS grants or denies access according to the code. The decision of what return code to set is based on the access rules and policies defined in the security database by the security administrator.

Other operating systems, such as OS/2, provide similar techniques for access control. The OS/2 access control module, called Security Enabling Services (SES), is based on the same concept as MVS's SAF.
Unfortunately, UNIX-based operating systems were not designed this way. Authorization decisions are made mainly for file accesses and are performed by the operating system itself using the nine bits (rwx-rwx-rwx) in the file’s *inode* entry. Unlike SAF, no exit point for event interception is provided. Therefore, further security is necessary to perform functions that are more complex than those of mainframe-type security packages.

**How Does This Work?**

In addition to supplying the regular security functions—such as an access rule database, an audit log, and administration tools—eTrust AC intercepts the operating system events that are to be protected. Since eTrust AC has to work with many different operating systems, it intercepts events in memory. No changes are made to system files, and the operating system is not modified at all.

**What Is Protected?**

eTrust AC protects the following entities:

- **Files**
  
  Is a user authorized to access a particular file?

  eTrust AC restricts a user’s ability to access a file. You can give a user one or more types of access, such as READ, WRITE, EXECUTE, DELETE, and RENAME. The access can be specified regarding an individual file or to a set of similarly named files.

- **Terminals**
  
  Is a user authorized to use a particular terminal?

  This check is done during the login process. Individual terminals and groups of terminals can be defined in the eTrust AC database, with access rules that state which users, or groups of users, are allowed to use the terminal or terminal group. Terminal protection ensures that no unauthorized terminal or station can be used to log into the accounts of powerfully authorized users.

- **Signon time**
  
  Is a user authorized to log on at a particular time on a particular day?

  Most users use their stations only on weekdays and only during work hours; the time-of-day and day-of-week login restrictions, as well as holiday restrictions, provide protection from hackers and from other unauthorized accessors.
What Is Access Control?

- **TCP/IP**
  
  Is another station authorized to receive TCP/IP services from the local computer? Is another station authorized to supply TCP/IP services to the local computer? Is another station permitted to receive services from every user of the local station?

  The advantage of an open system—a system in which both the computers and the networks are open—is also a disadvantage. Once a computer is connected to the outside world, one can never be sure who enters the system and what damage an alien user may do, whether intentionally or by mistake. eTrust AC includes "firewalls" that prevent local stations and servers from providing services to unknown stations.

- **Multiple login privileges**
  
  Is the user permitted to log in from a second terminal?

  The term *concurrent logins* refers to a user's ability to be logged onto the system from more than one terminal. eTrust AC can prevent a user from logging in more than once. This prevents intruders from logging into the accounts of users who are already logged in.

- **User-defined entities**
  
  You can define and protect both regular entities (such as TCP/IP services and terminals) and functional entities (known as *abstract objects*; such as performing a transaction and accessing a record in a database). The Application Programmer's Interface (API) that is used to define and protect abstract objects is described in the *SDK Developer Guide*.

- **Aspects of administrator authority**
  
  eTrust AC provides the means to both delegate administrator authorities to operators and restrict root itself.

  eTrust AC provides the means to delegate administrator authorities to operators and to restrict the administrator itself.

- **Substitute-user**
  
  Are users authorized to substitute their user IDs?

  The UNIX *setuid* system call, one of the most sensitive services provided by the operating system, is intercepted by eTrust AC to check whether the user is authorized to perform the substitution. The substitute-user authority check includes program pathing—users are permitted to substitute their user IDs only through specific programs. This is especially important in controlling who can substitute to root and thereby gain root access.
Substitute-group
Is a user authorized to issue the newgrp (substitute-group) command?
Substitute-group protection is similar to substitute-user protection.

Setuid and setgid programs
Can a particular setuid or setgid program be trusted? Is the user authorized to invoke it?
The security administrator can test programs that are marked as setuid or setgid executables to ensure that they do not contain any security loopholes that can be used to gain unauthorized access. Programs that pass the test and are considered safe are defined as trusted programs. The eTrust AC Self-Protection Module (also referred to as the eTrust AC watchdog) knows which program is in control at a particular time and checks whether the program has been modified or moved since it was classified as trusted. If a trusted program is modified or moved, the program is no longer considered trusted and eTrust AC does not allow it to run.

In addition, eTrust AC protects against various deliberate and accidental threats, including:

Kill attempts
eTrust AC can be used to protect critical servers and services or daenins against kill attempts.

Password Attack
eTrust AC protects against various types of password attacks, enforces the password-definition policies of your site, and detects break-in attempts.

Password Delinquency
eTrust AC policies delineate rules that force users to create and use passwords of sufficient quality. To ensure that users create and use acceptable passwords, eTrust AC can set maximum and minimum lifetimes for passwords, restrict certain words, prohibit repetitive characters, and enforce other restrictions. Passwords are not permitted to last too long.

Account Management
eTrust AC policies ensure that dormant accounts are dealt with appropriately.

Domain Management
eTrust AC can implement password protection and enforce security across NIS and non-NIS domains.
How Is It Protected?

eTrust AC protects resources and services transparently, by intercepting user requests and either permitting or denying access, based on rules set up by the system administrator.

The Process

The eTrust AC process can start immediately after the operating system finishes its initialization. eTrust AC places hooks in system services that require protection. In this way, control is passed to eTrust AC before the service is performed. eTrust AC then decides whether the service should be granted to the user.

For example, suppose a user attempts to access a protected resource. This access request generates a system call to the kernel to open the resource. eTrust AC intercepts this system call and determines whether to grant access. If permission is granted, control is passed to the regular service of the system; that is, the system call proceeds without interference. If, on the other hand, eTrust AC denies permission, it returns the standard "permission denied" error code to the program that activated the system call—the system call does not proceed further.

The Database

The decision to grant or deny access is based on access rules and policies defined in the eTrust AC database. The security administrator defines most of the records in the database.

eTrust AC is an object-oriented security system. The database describes two types of objects: accessors and resources. Users and groups are accessors. Objects to be protected, such as files and programs, are resources. Each record in the database describes either an accessor or a resource.

Each object belongs to a class—a collection of objects of the same type. For example, TERMINAL is a class containing objects that are terminals protected by eTrust AC. The concept of classes or types of resources is well known to mainframe security administrators.

In earlier versions of eTrust AC, the information on CLASS status (that is, whether the resource class was active or inactive) was held in the database. Every attempt to access a resource was intercepted by SEOS_syscall and passed to seosd, which checked the status in the database. If the class was inactive, access was allowed without further checking for authorization.

Now overhead is improved by passing a list of active classes to SEOS_syscall when seosd starts up, and every time a user changes the CLASS activity status. If a class is inactive, access to the resource is not intercepted.
Each user is represented by an accessor-element, which is an in-memory reflection of the user record in the database. eTrust AC builds the accessor-element during the login process. The accessor-element is associated with the user’s process and with any subprocess that is forked from the user’s process. Whenever the process requests a system service that is protected, or that issues an implicit request to access a resource, eTrust AC reads the resource record. It then determines whether information in the previously created accessor-element—such as the user’s security level, mode, and groups—means that the user is allowed to access the resource.

The following section describes the basic concept of an access rule and how eTrust AC decides whether to grant or deny access.

**What Is an Access Rule?**

An access rule is an item of information governing the permission of one or more given accessors to work with one or more given resources. The rule may allow zero or more types of access.

Relevant access types vary according to resource type. They may include READ, EXECUTE, and others as appropriate. A special access type called NONE is used for allowing zero access.

You can define access rules in several ways, as described in the following sections:

**Access Control Lists**

The record for a resource can include an Access Control List (ACL) field. An ACL is a list of accessors paired with access types. The ACL in a resource record is used as the second highest-priority access rule for that resource (after the negative access control list).

**Default Access Field**

The record for a resource can include a Default Access Field (commonly known as defaccess and sometimes abbreviated defacc in commands). The defaccess specifies the access type for accessors that do not appear in the resource ACL.

Suppose the user Smith tries to kill the process store_acct. eTrust AC retrieves the store_acct record from the PROCESS class in the database and searches for “Smith” in the record’s ACL. If it appears there with READ access, then Smith is permitted to kill the process.
On the other hand, users Jones, Doe, and Roe cannot kill that process because their access type is NONE. However, a user named Henderson—a name not included in the ACL list—can kill it, because defaccess for store_acct is READ.

Default Record for Class

Most classes can include a default record (_default) specifying access types for resources of that class that are defined to UNIX but do not have database records of their own.

Like other resource records, the _default record can include an ACL and a defaccess field. You can create a _default record for all classes except USER, GROUP, CATEGORY, SECLABEL, and SEOS.

UACC Class

Some earlier versions of eTrust AC used a separate class, called UACC, for records resembling the _default records of other classes. The UACC class is no longer recommended, and if you use a _default record, the equivalent record in the UACC class is not checked. In future versions, the UACC class may no longer be supported.

For example, suppose user Henderson tries to kill process store_log. eTrust AC checks for authorization in the following order. The primary question is this: Is the process store_log defined in the database? eTrust AC searches the database for a record named store_log in the PROCESS class.

- If no such record can be found, the process is not defined to eTrust AC. In that case, eTrust AC therefore uses either the _default record of class PROCESS, or the PROCESS record in the UACC class, to determine whether Henderson is allowed to kill store_log.
  - If user Henderson appears in the _default record's ACL, the authority specified in it is applied.
  - If Henderson does not appear in the _default record's ACL, the authority specified in the defaccess property of the _default record is applied. This authority is applied to all users who do not appear explicitly in the _default ACL.

- If process store_log is defined in the database, then the question is whether user Henderson appears in the ACL for process store_log in the database.
  - If user Henderson appears in the ACL for process store_log, the authority specified there is applied.
  - If Henderson does not appear in the ACL, eTrust AC applies the authority specified in the default access property of the store_log resource. This authority is called the resource's default access.
**Note:** If the default access (defaccess) of _default is set to NONE, or if _default is not specified and the default of the corresponding resource in the UACC class is NONE, then any accessor attempting to access a resource not defined in the class is denied access to the resource.

If the default access of _default (or UACC) is set to the highest authority (ALL, or in some cases READ or EXECUTE), then any resource that is not explicitly protected is accessible to everyone.

### Conditional Access Control Lists

Conditional Access Control Lists (CACLs) provide an extension to ACLs. Authorities specified in a CACL are applied only if a particular condition is met. The more common type is the Program Access Control List (PACL). Each element in a CACL specifies the accessor, the accessor’s authority over the resource, and the condition under which the authority is assigned. Use CACLs to define program pathing rules.

For example, suppose that a program called secured_su performs user substitution only after requesting users have verified their identity using a smart-card device. To allow user sysadm1 to become superuser only if sysadm1 is using program secured_su, specify the following conditional access rule:

Authorize user sysadm1 to surrogate id to root only under the program secured_su

Using the eTrust AC language (selang), enter the following command to declare that rule:

```
eTrustAC> authorize SURROGATE user.root id(sysadm1) via(pgm(secured_su))
```

Other types of CACLs include TCP class ACLs and CALENDAR class ACLs.

**Note:** Generic PACL, a new feature, is an extension to PACL. By placing wildcard characters inside the program name in the PACL, a file that is protected by the PACL can be accessed, using a program that matches the mask created by wildcard characters. If a program matches several masks, the longest mask takes precedence.
What Is an Access Rule?

**Negative Access Control Lists (NACLs)**

A Negative Access Control List (NACL) lists specific access rights to deny to an object.

With the eTrust AC language (selang), use the following command to deny access:

```
eTrustAC> {authorize | auth} className resourceName [gid(group-name...)]  \ 
   [uid({user-name...|*})] [deniedaccess(accessvalue)]
```

**Accumulative Group Rights**

The Accumulative Group Rights (ACCGR) property specifies that the authority of a user belonging to more than one group is equal to the sum of all the authorities of the groups to which the user belongs. However, if any of the access types is NONE, then NONE always takes precedence over the access types from other groups. When eTrust AC is installed, the value of this property is set to yes.

In general, the most important information contained in a resource record is the list of the accessors that are authorized to access the resource (the ACL). If a user is a member of more than one group in this list, then:

- If any of the access types is NONE, eTrust AC denies the user access to the resource.
- If the accumulated group rights (ACCGR) option is set, the authority of a user belonging to more than one group is equal to the sum of all the authorities of the groups to which the user belongs. That is, the NACL authorities are checked, then the ACL authorities are checked, and finally the PACL authorities are checked.

To set the ACCGR option, enter the following command:

```
eTrustAC> setoptions accgrr
```

If the ACCGR option is turned off, eTrust AC selects only one of the authorities granted by the groups. The NACL authorities are checked first, then the ACL authorities, then the _default authorities, and finally (if at all) the PACL authorities. In practice, authorities designated in the PACL are rarely consulted in this case.

To turn off the ACCGR option, enter the following command:

```
eTrustAC> setoptions accgrr-
```
Security Levels

Accessors and resources in the database can be assigned a security level. The security level is an integer between 1 and 255. An accessor can gain access to a resource only if the accessor has a security level equal to or greater than the security level assigned to the resource.

A user with security level 100 cannot access a resource with security level higher than 100, even if the user is specifically permitted access to the resource in the resource's access control list.

To turn off security level checking for a resource, specify zero (0) instead of a security level. To prevent an accessor from accessing any resource that has security level checking enabled, give the accessor a zero value instead of a security level.

Security Categories

Accessors and resources in the database can belong to one or more security categories. An accessor can access a resource only if the accessor belongs to all of the security categories assigned to the resource.

If a file belongs to the categories ACCOUNTING and MANPOWER, a user who belongs only to the ACCOUNTING category is not able to access the file; the user must belong to both security categories in order to access the file.

Security Labels

A security label is a name that associates a particular security level with a set of zero or more security categories. Assigning a user to a security label gives the user both the security level and any security categories associated with the security label.

Suppose SYSHIGH is a security label that associates security level 255 with security categories MANAGEMENT and CONFIDENTIAL. Assigning user usr1 to security label SYSHIGH automatically gives usr1 a security level of 255 and assigns usr1 to categories MANAGEMENT and CONFIDENTIAL.
**Generic Access Rules**

Access rules can protect many resources—not just one! An access rule that protects many resources is called a *generic* access rule. Instead of the name of a single resource, the record for a generic access rule contains a wildcard pattern that identifies a group of resources. Any resource with a name matching that wildcard pattern is protected by the generic access rule. Should a resource match more than one generic access rule, the closest of the matches is used for that resource.

**Note:** Generic access rules are currently implemented for the FILE class, HOSTNP class, and user-defined classes only.

For example, a record in the FILE class (that protects files and directories) with the key `/usr/lpp/bin/*` protects all files, sub-directories, sub-sub-directories, and so forth, under the path `/usr/lpp/bin`. This allows you to protect many resources using a compact definition consisting of one access control list.

eTrust AC does **not** accept the following patterns in generic access rules:

- `/*`
- `/tmp/*`
- `/etc/*`
The following set of charts summarizes how eTrust AC allows or stops a particular user's attempt to access a particular resource.
Graphical Access Summary

From the database resource record, make a list of ACLs for all groups that the user belongs to.

- Deny access
- Is there an ACL with access NONE for the user's group?
  - Yes
  - Deny access
  - No
  - Is there a deny/allow verdict?
    - Yes
    - Deny access
    - No
    - Is there an ACL for the groups?
      - Yes
      - Does user's access match the first access in the list?
        - Yes
        - Allow access
        - No
        - Deny access
      - No
      - Are accumulated group rights ON?
        - Yes
        - Accumulate all access levels from all ACLs in the list
        - No
        - Deny access
Graphical Access Summary
Find the accessing program in the PROGRAM class of the database

Is the PROGRAM class active and the program trusted?

Yes

Is there a discrete record for the program in the PROGRAM class?

Yes → 11
No → J
Is there a generic PAACL that matches the program?

Yes

Repeat (H) for every PAACL, in matching order

No

K
Graphical Access Summary
Chapter 3: Users, Groups, and Resources

This section contains the following topics:

Introduction to Users and Groups (see page 45)
Modifying User Records (see page 48)
Modifying Group Records (see page 50)
What Is a Resource? (see page 52)
Examples: Setting ACLs (see page 53)
Classes (see page 54)

Introduction to Users and Groups

The eTrust AC concepts of users and groups are parallel to the UNIX concepts but are more detailed.

Information in a user record or a group record is stored in properties. A property is equivalent to a database field. For example, the user's first and last name are stored in the FULLNAME property of the user record.

Users

In eTrust AC, each action or access attempt is performed on behalf of a user who is held responsible for submitting the request. Every process in the system must therefore be associated with a certain user name. The user name identifies the user to eTrust AC. A user is generally a person who can log on and for whom access authorities should be assigned and checked.

Though typically an eTrust AC user name that you create should be identical to a login name recognized by UNIX, for some purposes you may want an eTrust AC user name that is not a UNIX login name. (Then the login command could not put that user to work, but another command such as sesu could.)

The user name associated with a daemon process is often not a user name of a person working in front of a terminal, but rather an abstract entity that identifies the process. Such an abstract user name is treated like any other user name.

The user record contains information about the user (person) associated with the user name, such as the user's full name, the times the user is allowed to log on, and whether the user is a security officer or an ordinary user. The user record also contains a list of the groups to which the user belongs.
Groups

It is often convenient to group users together to work on specific projects or in specific departments or divisions in the organization. eTrust AC lets you define groups of users. You assign authorities to groups just as you would assign authorities to users. Using groups can ease your workload and simplify maintenance of the security database, because:

- You can assign authorities once for the group rather than repetitiously assigning the same authorities to each user.
- Using a profile group, you can create a standard setup for a new user that specifies such things as the number of grace logins or the user's UNIX home directory.

The group record contains information about a group. The most important information stored in the group record is the list of users who are members of the group.

Note: You cannot set a group's access to expire, only a user's.

Priority of Permissions

If the permissions of a user conflict with the permissions of the group to which the user belongs, the user permissions override the group permissions. This allows you to give some users in a group of authorities that differ from the rest of the group, without having to repeat all the group's authorities in the user record. You need only specify those authorities that are different from the group to which the user belongs.

If a user belongs to more than one group, and one of the groups has no access to a particular resource, then the user does not receive access based on group membership.

Note: Do not assign a user to two groups that have different access rights to the same resource.

Profile Groups

Profile groups let administrators efficiently create a standard setup with specific permissions for any new user assigned to that group. This setup can specify such things as the user's UNIX home directory, the PMDB that defines the authorities, and a variety of password rules affecting a user who is a member of a profile group. Thus, password policy can now be controlled on a profile group level, as well as on a whole database level.

Note: For more information about setting password rules at the database level, see the setoptions command in the Reference Guide.
To assign a user to a profile group, add the user to the profile group. If properties are set in the profile group, but not in the user's record, the following properties for the user are derived from the profile group:

- `audit_mode`
- `authnmthd`
- `daytime`
- `expire_date`
- `gracelogin`
- `homedir`
- `inactive`
- `maxlogins`
- `min_time`
- `passwd_int`
- `passwdrules`
- `policymodel`
- `pwd_autogen`
- `pwd_sync`
- `pwpolicy`
- `resume_date`
- `shell`
- `suspend_date`
- `suspend_who`

The profile group can also be used when creating new users in UNIX. Use the following command:

```
eTrustAC> newusr username profile(groupname) unix
```

The profile group is now assigned to the eTrust AC user. The groupname, if it exists in the UNIX environment, is assigned as the user's primary group. The homedir and shellprog properties, taken from the profile group, are assigned to the UNIX user.

The rest of this chapter discusses the properties of user and group records and shows you how to add, modify, and delete these properties.
**Determining Access: A Summary**

A user can be given access to a resource for any of three reasons:

- The user is permitted to access as a person working with the computer system and associated with a specific user name.
- The user is permitted to access as a member of a group that has access authorities assigned to it.
- The user is permitted to access as someone running a production (daemon) process that is associated with a certain user name.
- The user is permitted to access as someone running a regular program (as long as it has matching record in the SPECIALPGM class) that is associated with a certain user name.

**Modifying User Records**

You can create, modify, delete, and list the properties of a user record by any of the following methods:

- selang, the eTrust AC command language. Use the following selang commands on user records:
  - **newusr** or **editusr** defines a new user record
  - **chusr** or **editusr** changes the properties of a user record
  - **rmusr** deletes a user record
  - **showusr** lists the properties of a user record.

  **Note:** For more information about selang, see the Reference Guide.

- Security Administrator, the UNIX-based interface for eTrust AC.

  **Note:** For more information about Security Administrator, see the User Guide.

- Policy Manager, the Windows-based interface.

  **Note:** For more information about Policy Manager, see the User Guide.
Examples: Create or Modify Users

The following examples show you how to use the selang command language to create or modify users.

1. To define a new user to eTrust AC, use the newusr command. For example, to add a new user “Terry,” whose full name is Teresa Smith and whose security level is 100, specify the following:

   eTrustAC> newusr Terry name('Teresa Smith') level(100)

2. To change the information contained in a user record, use the chusr command. For example, to change Terry's security level to 150 and give her the AUDITOR attribute (to allow her to perform certain security auditing functions), specify the following:

   eTrustAC> chusr Terry level(150) auditor

3. To set up Terry as a sub administrator with the authority to manage users, use the authorize command as follows:

   selang> authorize ADMIN USER uid(sub-admin) access(R,Modify,Del,Cre,Join,PW)

4. To remove user Terry from the database, use the rmusr command as follows:

   eTrustAC> rmusr Terry

Special Predefined Users

eTrust AC contains several users internally that you cannot modify or delete:

_seagent

_seagent is an internal logical user with the SERVER attribute. This user is defined internally and therefore cannot be removed from the database.

Note: _seagent is the defined logical user under which the PMDB daemon (sepmdd) and the deviation calculation daemon (devcalc) run; therefore, you cannot assign another logical user to sepmdd or to devcalc.

_undefined

The _undefined property represents all users that are undefined in eTrust AC, and enables including undefined users in PACLs.
Modifying Group Records

You can create, modify, delete, and list the properties of a group record by any of the following methods:

- **selang** - *selang* is the eTrust AC command language.
  
  **Note:** For more information about selang, see the *Reference Guide*.

The following selang commands operate on group records:

- **newgrp** or **editgrp** defines a new group record.
- **chgrp** or **editgrp** changes the properties of a group record.
- **rmgrp** deletes a group record.
- **showgrp** lists the properties of a group record.
- **join** adds a user to a group.
- **join** - removes a user from a group

- **Security Administrator**
  
  The UNIX-based interface for eTrust AC
  
  **Note:** For more information about Security Administrator, see the *User Guide*.

- **Policy Manager**
  
  The Windows-based interface for eTrust AC.
  
  **Note:** For more information about Policy Manager, see the *User Guide*.

**Nested Groups**

Using nested groups, you can add and delete super groups (parents) and member groups (children) from existing groups. Authorization rules defined to a group are passed down to its member groups.

**Special Predefined Groups**

eTrust AC comes with a few, special, predefined groups. By including a user in one of these groups, you give the user a specified characteristic.

**The _restricted Group**

For users in the _restricted group, all files are eTrust AC-protected. If no access rule with the name of a particular file, or with a file name pattern that matches that particular file, exists, then for _restricted users the file is covered by the _default record in the FILE class.
eTrust AC reads the list of _restricted users at startup and at runtime.

**Notes:**

- If a user is already logged in, and you add the user to the _restricted group, the user is not affected until the next login.
- Use _restricted users with caution. If a user is a member of the _restricted group, the FILE class's _default object has NONE as its default access type, and the database contains rather few FILE access rules, then a _restricted user can easily find himself unable to do anything.

Remember, a user needs EXEC permission to run executables, READ permission to load dynamic libraries, and often CREATE/WRITE/UTIME authorization for various log/audit/cfg files that the executable needs. If you plan to add users to the _restricted group with NONE as the default access type for the FILE class's _default object, consider using WARNING mode. Then the audit events show you what files your _restricted users need for their work. After a while, you can grant the appropriate authorizations and turn WARNING mode off.

**The _abspath Group**

If a user is in the _abspath group at login time, that user cannot use relative path names to invoke programs.

**The _surrogate Group**

If a user is a surrogate to a user in the _surrogate group, seosd sends a full trace of the user's actions as the new user to the audit trail.

**Examples: Create or Modify Group Records**

The following examples show you how to use the selang command language to create or modify group records.

1. To define a new group, use the **newgrp** command. For example, to add the new group “sales” whose full name is “Sales Department,” enter the following command:

   ```
   eTrustAC> newgrp sales name('Sales Department')
   ```

2. To change the definition of a group record, use the **chgrp** command. For example, to change the name of group “sales” to “West Coast Sales Dept,” enter the following command:

   ```
   eTrustAC> chgrp sales name('West Coast Sales Dept')
   ```
What Is a Resource?

For eTrust AC, a resource is an entity that can be accessed by users or groups. The most common type of resource is a file. You access a file when you read information from it or write information to it. Other types of resources, for example, include terminals. (Terminals are accessed when you log in.)

A Special Resource: Abstract Object

Many resources-files, directories, terminals, disk volumes, and the like-are analogous to physical objects. eTrust AC also supports another type of resource, called an abstract object. Consider a user who wants to display highly sensitive information contained in a database field, or a user who attempts to perform a restricted transaction such as transferring a large sum of money from one account to another. eTrust AC lets you associate these actions with an abstract object to which access rules and authorities can be assigned. You can, for example, associate the transfer of one million dollars with an abstract object as follows:

- Define the abstract object to eTrust AC
- Define the access rules for the abstract object
- Insert a call to the eTrust AC runtime routine in your application.
When a user attempts to transfer the money, your application calls eTrust AC, which uses the abstract object's ACL to determine whether the user is authorized to transfer the money. Your application decides whether to continue processing the request based on the information returned by eTrust AC.

**Note:** For information about about the eTrust AC API, see the *SDK Guide*.

### Records and ACLs

The properties of the protected resource are stored in the resource's *record*. A record is a collection of data consisting of the name and properties of a resource or accessor.

These properties tell who defined the record, the date when the record was defined, and more. In general, the most important information contained in a resource record is a list of the accessors that are authorized to access the resource. This list is referred to as the access control list (ACL).

### Examples: Setting ACLs

This section shows you examples on how to add accessors to access control lists using the selang command language.

- To add an accessor to an ACL, use the `authorize` command. For example, to add user user27 to terminal tty30 and give user27 READ access to the terminal, enter the following command:

  ```
  eTrustAC> authorize TERMINAL tty30 access(READ) uid(user27)
  ```

- To change the access type of an accessor in an ACL, use the `authorize` command. For example, to change user27's access to terminal tty30 to NONE, enter the following command:

  ```
  eTrustAC> authorize TERMINAL tty30 access(NONE) uid(user27)
  ```

- To delete an accessor from an ACL, use the `authorize-` command. For example, to remove user27 from terminal tty30's ACL, specify:

  ```
  eTrustAC> authorize- TERMINAL tty30 uid(user27)
  ```

**Note:** For more information about the `authorize` and `authorize-` commands, see the *Reference Guide*. 

A class is a group of similar records. For example, the TERMINAL class contains all objects that are of type terminal, such as tty1, tty2, and so on; the FILE class contains definitions for files and file-masks; and the PROGRAM class contains the records that protect trusted programs from being modified.

Within each class, each record lists values for the same set of properties: the properties appropriate to the type of resource or accessor that the class describes. For example, a record in the USER class includes such properties as the user's location and working hours, while a record in the HOSTNET class includes such properties as net services and IP address data.

eTrust AC contains four types of predefined classes. In addition, you can define new classes of your own as necessary.

Predefined Classes

The predefined classes include not only accessor classes and resource classes, but also definition and installation classes.

<table>
<thead>
<tr>
<th>Class Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessor</td>
<td>Defines objects that access resources.</td>
</tr>
<tr>
<td>Definition</td>
<td>Defines objects that define security entities, such as security labels and categories.</td>
</tr>
<tr>
<td>Installation</td>
<td>Defines objects that control the behavior of eTrust AC.</td>
</tr>
<tr>
<td>Resource</td>
<td>Defines objects that are protected by access rules.</td>
</tr>
</tbody>
</table>

The following table contains a full list of the predefined classes.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN</td>
<td>Definition</td>
<td>Use this class to delegate administrative responsibilities to users who do not have the ADMIN attribute. You give these users global authorization attributes (see page 197) and limit their administration authority scope.</td>
</tr>
<tr>
<td>AGENT</td>
<td>Resource</td>
<td>Each record in this class defines an object that is used as an agent by eTrust SSO or eTrust Web AC.</td>
</tr>
<tr>
<td>AGENT_TYPE</td>
<td>Resource</td>
<td>Each record in the AGENT_TYPE class defines an agent type used by eTrust SSO or eTrust Web AC.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Class Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>APPL</td>
<td>Resource</td>
<td>Each record in the APPL class defines an application used by eTrust SSO or eTrust Web AC.</td>
</tr>
<tr>
<td>AUTHHOST</td>
<td>Accessor</td>
<td>Each record in the AUTHHOST class defines an authentication host in eTrust SSO or eTrust Web AC.</td>
</tr>
<tr>
<td>CALENDAR</td>
<td>Resource</td>
<td>Each record in the CALENDAR class defines a Unicenter TNG calendar object for user, group, and resource enforced time restrictions in eTrust AC.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>Definition</td>
<td>Each record in this class defines a security category.</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Resource</td>
<td>The CONNECT class protects the outgoing connection. The records in this class define which users can access which Internet hosts. Before you activate the CONNECT class, be sure that the streams module is active.</td>
</tr>
<tr>
<td>CONTAINER</td>
<td>Resource</td>
<td>Each record in the CONTAINER class defines a group of objects from other resource classes, thus simplifying the job of defining access rules when a rule applies to several different classes of objects.</td>
</tr>
<tr>
<td>FILE</td>
<td>Resource</td>
<td>Each record in this class defines a file, a directory, or a file name mask.</td>
</tr>
<tr>
<td>GAPPL</td>
<td>Resource</td>
<td>Each record in the GAPPL class defines a group of applications used by eTrust Web AC or SSO.</td>
</tr>
<tr>
<td>GAUTHHOST</td>
<td>Definition</td>
<td>Each record in the GAUTHHOST class defines a group of authentication hosts used by eTrust Web AC or SSO.</td>
</tr>
<tr>
<td>GFILE</td>
<td>Resource</td>
<td>Each record in this class defines a group of files or directories. Grouping is accomplished by explicitly connecting files or directories (resources of the FILE class) to the GFILE resource in the same way users are connected to groups.</td>
</tr>
<tr>
<td>GHOST</td>
<td>Resource</td>
<td>Each record in this class defines a group of hosts. Grouping is accomplished by explicitly connecting hosts (resources of the HOST class) to the GHOST resource in the same way users are connected to groups.</td>
</tr>
<tr>
<td>GROUP</td>
<td>Accessor</td>
<td>Each record in this class defines a group of users.</td>
</tr>
<tr>
<td>GSUDO</td>
<td>Resource</td>
<td>Each record in this class defines a group of commands that one user can execute as if another user were executing it. The sesudo command uses this class.</td>
</tr>
<tr>
<td>GTERMINAL</td>
<td>Resource</td>
<td>Each record in this class defines a group of terminals.</td>
</tr>
</tbody>
</table>
### Classes

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HNODE</td>
<td>Definition</td>
<td>The HNODE class contains information about the organization's Policy Model hierarchy. That is, it will include the propagation tree structure (subscribers, parent PMDBs, and so on). Each record in the class represents a node in this tree (a hierarchy node), with the name of objects in this class being the actual host name for an end-point (for example, myHost.ca.com) or the PMDB name for a Policy Model node (for example, <a href="mailto:myPMD@myHost.ca.com">myPMD@myHost.ca.com</a>). This class is used to manage information uploaded from the various PMDBs and end-point machines and stored on the DMS.</td>
</tr>
<tr>
<td>HOLIDAY</td>
<td>Definition</td>
<td>Each record in this class defines one or more periods when users need extra permission to log in.</td>
</tr>
<tr>
<td>HOST</td>
<td>Resource</td>
<td>Each record in this class defines a host. The host is identified by either its name or its IP address. The object contains access rules that determine whether the local host can receive services from this host. Before you activate the HOST class, be sure that the streams module is active.</td>
</tr>
<tr>
<td>HOSTNET</td>
<td>Resource</td>
<td>Each record in this class is identified by an IP address mask and contains access rules. If a host requests a service that is not specified in the host's access rules as defined in class HOST or class GHOST, eTrust AC checks whether there exists in the HOSTNET class an object whose mask fits the accessor's IP address and whose access rules allow the requested access.</td>
</tr>
<tr>
<td>HOSTNP</td>
<td>Resource</td>
<td>Each record in this class defines a group of hosts, where the hosts belonging to the group all have the same name pattern. Each HOSTNP object's name contains a wildcard.</td>
</tr>
<tr>
<td>LOGINAPPL</td>
<td>Definition</td>
<td>Each record in the LOGINAPPL class defines a login application, identifies who can use the program to log in, and controls the way the login program is used.</td>
</tr>
<tr>
<td>MFTERMINAL</td>
<td>Definition</td>
<td>Each record in the MFTERMINAL class defines a Mainframe computer that is used to administer eTrust AC.</td>
</tr>
<tr>
<td>POLICY</td>
<td>Resource</td>
<td>Each record in the POLICY class defines the information required to deploy and remove a policy. It includes a link to the RULESET objects that contain a list of the selang commands for deploying and removing the policy.</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Resource</td>
<td>Each record in this class defines an executable file.</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Resource</td>
<td>Each record in this class defines a trusted program that can be used with conditional access rules. Trusted programs are setuid/setgid programs that are monitored by the Watchdog to ensure they are not tampered with.</td>
</tr>
<tr>
<td>PWPOLICY</td>
<td>Definition</td>
<td>Each record in the PWPOLICY class defines a password policy.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Class Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RESOURCE_DESC</td>
<td>Definition</td>
<td>Each record in the RESOURCE_DESC class defines all of the names that new user-defined class objects are allowed to access in eTrust Web AC.</td>
</tr>
<tr>
<td>RESPONSE_TAB</td>
<td>Definition</td>
<td>Each record in the RESPONSE_TAB class defines an eTrust Web AC response table to different authorization decisions.</td>
</tr>
<tr>
<td>RULESET</td>
<td>Resource</td>
<td>Each record in the RULESET class represents a set of rules which define a policy.</td>
</tr>
<tr>
<td>SECFILE</td>
<td>Definition</td>
<td>Each record in this class defines a file that must not be altered.</td>
</tr>
<tr>
<td>SECLABEL</td>
<td>Definition</td>
<td>Each record in this class defines a security label.</td>
</tr>
<tr>
<td>SEOS</td>
<td>Installation</td>
<td>The one record in this class specifies your active classes and password rules.</td>
</tr>
<tr>
<td>SPECIALPGM</td>
<td>Installation</td>
<td>Each record in the SPECIALPGM class registers backup, DCM, PBF and PBN functions in Windows or xdm, backup, mail, DCM, PBF, and PBN programs in UNIX or associates an application that needs special eTrust AC authorization protection with a logical user ID. This effectively allows setting access permissions according to what is being done rather than who is doing it.</td>
</tr>
<tr>
<td>SUDO</td>
<td>Resource</td>
<td>This class, used by the sesudo command, defines commands that one user (such as a regular user) can execute as if another user (such as root) were executing them.</td>
</tr>
<tr>
<td>SURROGATE</td>
<td>Resource</td>
<td>Each record in this class protects a user or group from surrogate requests issued by other users.</td>
</tr>
<tr>
<td>TCP</td>
<td>Resource</td>
<td>Each record in this class defines a TCP/IP service, such as mail or http or ftp.</td>
</tr>
<tr>
<td>TERMINAL</td>
<td>Resource</td>
<td>Each record in this class defines a terminal—a device from which a user can log in.</td>
</tr>
<tr>
<td>UACC</td>
<td>Resource</td>
<td>Defines default access rules for each resource class.</td>
</tr>
<tr>
<td>USER</td>
<td>Accessor</td>
<td>Each record in this class defines a user to eTrust AC.</td>
</tr>
<tr>
<td>USER_ATTR</td>
<td>Definition</td>
<td>Each record in the USER_ATTR class defines the valid user attributes of an eTrust Web AC user directory.</td>
</tr>
<tr>
<td>USER_DIR</td>
<td>Resource</td>
<td>Each record in the USER_DIR class defines an eTrust Web AC user directory.</td>
</tr>
</tbody>
</table>

Note: For more information about eTrust AC classes, USER and GROUP classes, see the Reference Guide.
User-Defined Classes

eTrust AC enables you to define new classes, so that you can protect abstract objects by creating appropriate records for them.

Sample User-Defined Class

Suppose, for example, that your system serves a bank and you want to protect transfers of over one million dollars between accounts. You may want to use the following outline to set up this security.

1. Define a class to contain the necessary records: records that describe transfers. You may call the class something like TFERs.

2. For each type of transfer that a user might be permitted or forbidden to perform, define a record in the TFERs class.
   
   For example, you might define records named Upto.$1K, Upto.$1M, Upto.$10M, and Over.$10M.
   
   Any other resources that you require for the control of transactions can also be defined as members of the TFERs class.

3. To give different users permission to perform different maximum transfers, grant or deny them access to the various records in the TFERs class.
   
   Finally, arrange a call from the bank’s money-transfer program to an eTrust AC API to check the user’s permission before allowing any transfer to proceed.
Chapter 4: Protecting Accounts

This section contains the following topics:

- Why Protect Accounts? (see page 59)
- Protecting Substitute User (su) Requests (see page 59)
- Setting Up the Surrogate DO Facility (see page 62)
- Surrogating Safely with sesu (see page 63)
- Preventing Password Attacks (see page 64)
- Checking User Inactivity (see page 66)

Why Protect Accounts?

User accounts are often the object of password attacks. Root account protection involves monitoring substitute user (su) requests and using the Surrogate DO (SUDO) facility, which solves the dilemma of superuser privileges. eTrust AC provides a two-level password protection system: serevu (revoke user daemon) and PAM (Pluggable Authentication Module). You can also protect accounts by specifying automatic lockouts after a period of user inactivity.

Protecting Substitute User (su) Requests

After an account logs in, you must monitor it to ensure that it performs only authorized functions on files. UNIX-based operating systems provide some degree of protection for files based on the accessor's user ID. To bypass that protection, a user must first su (substitute or surrogate) to another user ID. The default protection against unauthorized substitution is that the su command prompts the requesting user to specify the target user's password.

This scheme has many faults. A user who wants to substitute a user ID must memorize the target user's password, write it down, or ask the target user to use a trivial password. This violates several password policies. In addition, there is no effective accountability; you can never tell which user changed identification to a specific user. Moreover, once the password of the superuser is known to a user, any security is bypassed, and that user has unlimited access to the system.
eTrust AC uses a more advanced method for protecting substitution: a user can change the user ID to another user's ID only if a specific rule allows the change. For example, if user X executes the sesu command to substitute the user ID to user Y, user X does not have to know user Y's password (if you use the standard UNIX su command, you must supply the password). If user X has user Y's password and user X does not have permission to substitute to user Y, the sesu request is denied. This way, root's password is not enough for an intruder; there must also be a rule in the database allowing the intruder to become root.

Each user ID (UID) and group ID (GID) can have an access rule in the database. eTrust AC has assigned the SURROGATE class for this type of protection. In the initial stage you wish to grant access to any su request, use the following command:

    eTrustAC> newres SURROGATE _default defaccess(READ)

This command tells eTrust AC to allow access if a user makes a request to su (surrogate) to another user, and a record in the database does not explicitly protect the user substitution.

To protect against attempts to substitute the UID to that of the superuser, use the following command:

    eTrustAC> newres SURROGATE USER.root defaccess(NONE)

This command tells eTrust AC that the root user name is protected and that users not explicitly permitted to use it cannot su to root. To permit the security administrator to use root, you must explicitly specify it by using the following command:

    eTrustAC> authorize SURROGATE USER.root gid(SECADMIN)

Use this command to protect any other group that requires protection.

Notes:

- If a surrogate record of a user does not specifically permit a certain user to do the substitution, the user gets the default access of that record. In the previous example, the default is NONE, which means that users without permission cannot su to root.

- A record called USER._default represents all users who do not have their own records. Similarly, a record called GROUP._default represents all groups that do not have their own records. If no surrogate record for a certain accessor exists, a request for substitution to that accessor ends with the default specified in the SURROGATE USER._default record, the SURROGATE GROUP._default record, or the _default record (for both users and groups).
- The default value for the _default record is READ; undefined surrogate records imply permission to su to those users. This default meets the general rule of thumb during implementation, that "whatever is not defined in eTrust AC is not protected by eTrust AC." You can modify this rule after the implementation stage to the opposite rule: "whatever is not permitted in eTrust AC is automatically forbidden by eTrust AC."

- If you want to prevent loopback and local host bypasses of the surrogate access rules, you need to apply the recommended restrictions (see page 100). To prevent exec login bypasses of the surrogate access rules, you need to protect the login command (see page 92).
Setting Up the Surrogate DO Facility

Operators, production personnel, and end users often need to perform tasks that only the superuser can perform. These tasks include the following:

- Mounting a CD-ROM
- Using backup scripts
- Setting up a printer

The traditional solution is to supply all these users with the superuser's password, which compromises the security of the site. The secure alternative—keeping the password secret—results in the system administrator being overloaded with legitimate requests from users to perform routine tasks.

The Surrogate DO (sesudo) utility solves this dilemma. It allows users to perform actions that are defined in the SUDO class, where each record contains a script, specifies which users and groups can run the script, and lends them the necessary permissions for the purpose.

For example, to define a SUDO resource that mounts a CD-ROM as if the user were root, enter the following command:

```
eTrustAC> newres SUDO MountCd data('mount /usr/dev/cdrom /cdr') targuid(root)
```

This newres command defines MountCd as a protected action that some users may receive root authority to perform. This example uses the targuid(root) parameter to show that root is the ID of the target user—the user whose permissions are borrowed. In practice, the parameter would be unnecessary for this example because root is the default target ID for a SUDO record.

**Important!** In the data property, use a full absolute path name. A relative path name could accidentally execute a Trojan horse program planted in an unprotected directory.

In addition, users can be authorized to perform the MountCd action by using the authorize command. For example, to allow the user operator1 to mount the CD-ROM, enter the following command:

```
eTrustAC> authorize SUDO MountCd uid(operator1)
```

You can also explicitly prevent a user from performing the protected action by using the authorize command. For example, to prevent the user operator2 from mounting the CD-ROM, enter the command:

```
eTrustAC> authorize SUDO MountCd uid(operator2) access(None)
```

Executing the sesudo utility performs the protected action. For example, the user operator1 would mount the CD-ROM using the following command:
The sesudo utility first checks whether the user is authorized to perform the SUDO action and then, provided the user is authorized to the resource, executes the command script defined in the resource. In the case of our example, sesudo checks whether operator1 is authorized to perform the MountCd action and then invokes the command mount /usr/dev/cdrom /cdr.

If you would like sesudo to request the user's password before executing, define or modify the SUDO record with a command that includes the PASSWORD parameter. If you do not use that parameter, the user's ability to execute the command is based solely on the access rules for the SUDO object.

**Note:** For more information about the sesudo utility, see the *Utilities Guide*. For more information regarding the formatting of the SUDO record's data property, see the chres, editres, and newres commands in the *Reference Guide*.

### Surrogating Safely with sesu

A problem with the UNIX su command is that no record is kept of who invoked the command. Any user pretending to be the owner of an account is indistinguishable from the actual owner.

As an alternative to the su command, eTrust AC includes the sesu utility—an enhanced version of the UNIX su command—that enables a user to substitute to another user without knowing the target user's password. You can configure sesu to prompt users for their own passwords as means of authentication, rather than for the target user's password. The authorization process is based on the access rules defined in the SURROGATE class and, optionally, on the password of the user executing the command.

Unlike permission to su, permission to sesu does not depend on knowing the target user's password. Instead, it depends on permissions specified in the database; users remain accountable for their actions because their login identities are remembered.

**Notes:**
- If a user is a surrogate to one of the users in the _surrogate group, seosd sends a full trace of the user’s actions as the new user to the audit trail.
- Do not use the sesu command during implementation until all users are defined in the database. This prevents you from opening up the entire system to users who are not defined in eTrust AC. To protect against inadvertent use of this program, it is marked in the file system so that no one can run it. The security administrator must mark the program as executable and setuid to root before it can be used.
Preventing Password Attacks

The most common type of unauthorized access is that of hackers who guess passwords. eTrust AC provides two tools that detect and protect against password attacks: servu and pam_seos.

Another method of protecting against password attacks is controlling passwords (see page 69) used in your environment by setting password policy rules.

**servu**

The servu daemon locks the accounts of users who performed more than a specified number of login attempts. This prevents potential password attacks by rejecting further attempts to enter the account; it also prevents “dictionary attacks.”

Normally, the danger in using the user lockout utility is that it opens the system to service denial attacks. The most common type of service denial attack is an attempt to break into the system administrator’s account—after a few attempts, the system administrator is revoked and can no longer log in. If similar attacks are performed on other critical user accounts, the system may be rendered unusable, with no way of recovering the system (servu never revokes root, so the system is never locked out).

**Note:** For more information about the servu daemon, see the Utilities Guide.

To prevent this, the servu daemon provides the following two modes of operation:

- The account is revoked for a specified period of time, after which it is automatically restored.
- The account is permanently revoked.

**Note:** Take special care regarding the root user’s password to prevent successful dictionary attacks on root.
pam_seos

pam_seos is a Pluggable Authentication Module (PAM) that eTrust AC uses for advanced account management functions. eTrust AC calls pam_seos during the login procedure of any login program. The module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand.

You can configure pam_seos to perform three actions:

- Detect login failures
  The Account Management Component detects any failed login attempt and logs it to both the audit file and a special failed logins file. This module detects UNIX failures, not cases in which eTrust AC denies access.
  eTrust AC writes the failed login attempts to a special file. The serevu utility reads this file and uses the information to determine if and when user access should be revoked.

- Provides debug mode
  When eTrust AC denies a login, it usually does not show the reason for denial during the login session. If the pam_seos module's debug mode is set, eTrust AC gives a short description of the reason for login denial. For example, “grace logins” means that the user has no remaining logins.

- Checks for expired passwords and grace logins
  The Password Management Component invokes the segrace utility, which checks for a user's password expiration and the number of grace logins. If a user's password expires, and the user has no grace logins left, segrace invokes the sepass utility to allow the user to change the password.

**Note:** eTrust AC invokes segrace only when a password change is needed.

The installation program adds the relevant lines to the pam.conf configuration file, and stores the old configuration file as /etc/pam.conf.bak.

Configuration of the pam_seos modules is performed through the seos.ini file. Set the following tokens, located in the [pam_seos] section, according to the required functionality:

To use the Password Expiration and Grace Logins check, set the following token in the seos.ini file:

```
call_segrace = Yes
```

To use Login Debug Mode, set the following token in the seos.ini file:

```
debug_mode_for_user = Yes
```
Checking User Inactivity

To make serevu use pam_seos login failure detection, set the following token in the seos.ini file:

serevu_use_pam_seos = Yes

Restrictions and Limitations

The protection techniques described in this section have the following restrictions and limitations:

- On Sun Solaris, after five failed login attempts, serevu is notified.
- The pam_seos module is only implemented in the versions of Sun Solaris, HP-UX, and Linux that support PAM.

Checking User Inactivity

The inactivity feature protects the system from unauthorized access through accounts whose owners are away or no longer employed by the organization. An inactive day is a day in which the user does not log in. You can specify the number of inactive days that must pass before the user account is suspended and cannot log in. Once an account is suspended, you must manually reactivate it.

**Note:** Password changes count as activities, in terms of inactivity checks. If a user’s password changes, that user cannot become suspended due to inactivity.

You can set the number of inactive days with the inactive property of a USER class record or a GROUP class record. The latter affects only users that have that group as a profile group. You can also set inactivity for all users systemwide with the INACT property of the SEOS class.

Both selang and the Security Administrator provide the means for setting inactivity. In selang, use the following command to specify inactivity globally:

```
eTrustAC> setoptions inactive (numdays)
```

To set the number of days for a group (which overrides the systemwide inactive setting for that group), use the following command:

```
eTrustAC> {chgrp | editgrp | newgrp} groupName inactive (numdays)
```

To set the number of days for a user (which overrides group and systemwide settings for that user), use the following command:

```
eTrustAC> {chusr | editusr | newusr} userName inactive (numdays)
```
To reactivate a suspended user account, use the following command:

eTrustAC> {chusr | editusr} userName resume

To reactivate a suspended profile group, use the following command:

eTrustAC> {chgrp | editgrp} userName resume

To disable inactive login checking at the systemwide level, use the following command:

eTrustAC> setoptions inactive-

To disable inactive login checking for a group, use the following command:

eTrustAC> {chgrp | editgrp} groupName inactive-

To disable inactive login checking for a user, use the following command:

eTrustAC> {chusr | editusr} userName inactive-

**Note:** For information about setting inactivity in Security Administrator, see the *User Guide*.
Password Control

Passwords are the most popular device for authentication, but password protection has well-known problems:

- Trivial passwords are easy to guess.
- Passwords that last for years and cyclic passwords are eventually broken.
- Listeners can trap passwords that are sent in clear text over the network.
Defining Password Policies

The most important password rule is that users must not give out their passwords explicitly or indirectly (by using trivial passwords). The only way to achieve acceptable password security is by training and education. eTrust AC cannot replace education, but it can enforce rules and policies that force users to use passwords of a minimum quality. The rules that you can specify include the following:

- The new password cannot match previous passwords.
- The new password cannot contain the user name.
- The new password cannot contain the password that it is replacing.
- The new password cannot be contained by the password that it is replacing.
- The new password cannot match the password that it is replacing, regardless of case sensitivity.
- The new password must have at least the minimum number of alphanumeric characters, special characters, digits, lowercase characters, and uppercase characters.
- The new password must not have more repetitive characters.
- The new password cannot be one of the restricted words in the dictionary to which the Dictionary token in the seos.ini file points.
- Each password must have a maximum lifetime; that is, it must expire, forcing the user to choose a new password after a certain interval.
- Each password must have a minimum lifetime. (By specifying a minimum lifetime, you can prevent users from quickly and repeatedly changing passwords. By quickly changing passwords, they could overflow the password history list and then re-use a previous password.)
Setting Up Password Quality Checking

**To set up password quality checking:**

1. Turn on password quality checking with the following command:
   
   ```
   eTrustAC> setoptions class+ (PASSWORD)
   ```

2. Define the rules to be used for the password checks:

   ```
   eTrustAC> setoptions password(rules(rule))
   ```

   The `rule` parameter specifies one or more rules.

   **Note:** For the syntax of the rule settings, see the chapter Reference Guide.

3. Update the new passwords by using the sepass utility, as described in the following section.

   **Note:** For a detailed description of sepass, see the Utilities Guide.

Changing Passwords

eTrust AC includes the executable `eTrustACDir/bin/sepass` (where `eTrustACDir` is the installation directory for eTrust AC, by default `/opt/CA/eTrustAccessControl`), with which most users should change their passwords (instead of with `/bin/passwd`).

- Only sepass ensures that the new password matches eTrust AC password policies. And only sepass updates the database with the new password and the date on which the password was changed. In addition, sepass performs the same functions as `/bin/passwd`.
- The original `/bin/passwd` executable should not be used unless you choose to discard the password quality checks performed by eTrust AC. In this case, you can continue to use the original `/bin/passwd`, and eTrust AC accepts the system’s password without performing any quality checks on passwords.

You can also change passwords using selang. Enter the following command to assign a password to a user:

```
se\> {chusr | editusr | newusr} userName password(string)
```

**Note:** For information about changing passwords with Security Administrator, see the User Guide.

**Note:** If you change another user's password (as an administrator) and password checking is enabled, the user must change the password at the next login.
Implementing Password Policy Rules

Implementing Password Policy Rules

After deciding which password policies you wish to apply, you can use the setoptions and user commands to assign passwords and their rules, as shown in the following examples.

You can apply password policy rules from the selang command shell or Security Administrator. In selang, use the setoptions command to specify global password rules, and the chgrp command for group password rules, as illustrated in the following examples.

To activate the minimum-length check with a minimum of seven characters, enter the following command:

eTrustAC> setoptions password(rule(length(7)))

To set the password interval to 30 days for all users whose profile group is MailRoom, enter the following command:

eTrustAC> chgrp MailRoom interval(30)

To deactivate password policy checking globally, enter the following command:

eTrustAC> setoptions password(norules)

Note: For more information about the setoptions and user commands, see the Reference Guide.

The following sections discuss these rules in more detail.

Password Expiration and Grace Logins

The interval parameter sets the maximum number of days a password can be used. When the specified number of days passes, eTrust AC informs the user that the current password has expired. The user can then renew the password immediately, or continue using the old password until the number of grace logins is reached. In the latter case, the user cannot access the system and must contact the system administrator to select a new password.
Specifying the Password Interval

At the systemwide level, you use the setoptions command to specify the interval before the system prompts all users for a new password. If the segrace utility is part of the user’s login script or if you configure PAM to call segrace (if your native operating system supports PAM), eTrust AC informs the users that the current password has expired when the specified number of days is reached. The users can then immediately renew the password, or continue using the old password until the number of grace logins (see page 74) is reached. After reaching the number of grace logins, the users are denied access to the system and must contact the system administrator to select a new password.

To set or cancel the password interval at the systemwide level, use the following command:

```plaintext
eTrustAC> setoptions password({interval(NumDays)|interval-})
```

The value of NumDays must be zero or a positive integer. An interval of zero disables password interval checking for users. Set the interval to zero if you do not want passwords to expire. An interval of zero should only be used for users with low security requirements.

The interval- parameter cancels the password interval setting. If the user has a profile group with a value for this parameter, that value is used. Otherwise, the default set by the setoptions command is used. Only use this parameter with the chusr or editusr command.

Individual User or Group Password Intervals

You can also set the interval for specific users or profile groups. These settings override the systemwide interval for those users or groups. When the specified number of days is reached, eTrust AC informs the users that the current password has expired. The users can then immediately renew the password, or continue using the old password until the number of grace logins is reached. After reaching the number of grace logins, the users are denied access to the system and must contact the system administrator to select a new password.

To set or cancel the password interval for a user:

```plaintext
eTrustAC> {chusr | editusr | newusr}\ {interval(NumDays) | interval-}
```

To set or cancel the password interval for a group:

```plaintext
eTrustAC> {chgrp | editgrp | newgrp}\ password({interval(NumDays) | (interval-})
```
The value of NumDays must be zero or a positive integer. An interval of zero disables password interval checking. Set the interval to zero if you do not want a password to expire. An interval of zero should only be used for users with low security requirements.

The interval- parameter cancels the password interval setting. If it is canceled and a value for interval is set in the user record, the value in the user record is used. Otherwise, the default set by the setoptions command is used. Use this parameter with the setoptions, chgrp, or editgrp commands only.

Specifying Grace Logins

With password checking enabled, eTrust AC checks whether the user's password has expired each time a user attempts to log in. After the password expires, the user can be “graced” with the opportunity to log in a few more times, after which the user can no longer log in.

The grace login option sets the maximum number of logins that are permitted after password expiration before the user is suspended. The number of grace logins must be between 0 and 255. After the number of grace logins is reached, the user is denied access to the system and must contact the system administrator to select a new password. If grace is set to zero, the user cannot log in. The default number of grace logins is five.

You can use this method to force a user to change their password. Reset the user's password and give them one grace login whence they can change their password.

Tracking Grace Logins

To allow the end user to keep track of grace logins after the expiration, insert a call to the segrace utility in the user's .login, .profile, or .cshrc file. The segrace utility then displays a message to the user stating the number of remaining grace logins. You can also check whether a user's password has expired graphically with the segracex utility.

**Note:** For more information about the segrace and segracex utilities, see the *Utilities Guide*.

To set the systemwide default value for the number of grace logins, enter the following command:

```
eTrustAC> setoptions password(rule(grace(nLogins)))
```

To set or cancel grace logins for a specific user, enter the following command:

```
eTrustAC> chusr userName {grace(nLogins) | grace-}
```
To set or cancel grace logins for a profile group, enter the following command:

eTrustAC> chgrp groupName {grace(nLogins) | grace-}

The value set by the chusr or chgrp command overrides the system value for the users specified in that command.

**Note:** The grace property for a GROUP class and also the global grace login setting set the number of grace logins for a user **after** the user's password expires. However, the grace property in the USER class sets the password to expire immediately; the grace logins are automatically set up (using the GROUP record or the system default) after the user's password expires. You cannot set password expirations for a group, only for users.
Restricting Access to Files and Directories

eTrust AC leaves the UNIX system of permissions intact but adds a layer of enhanced access control to it.

eTrust AC intercepts each of the following file access operations and verifies that the user has authorization for the specific operation before returning control to UNIX. The access type is in parentheses.

- File create (create)
- File open for read or write (read, write)
- File execute (execute)
- File delete (delete)
- File rename (rename)
- Change permission bits (chmod)
- Change owner (chown)
- Change timestamp—for example, as a result of executing the touch command (utime)
- Edit native ACL—using the acledit command—for systems that support ACLs (sec)
- Change directory (chdir)
Restricting Access to Files and Directories

eTrust AC access checking differs from the native UNIX authorization in the following ways:

- eTrust AC bases its authorization checks on the original user ID of the user who logged in, not on the effective user ID (euid). For example, if userA invokes the su command to surrogate to another user, userA still only has access to those files to which userA is permitted. Surrogating to another user does not automatically give the original user access to the target user’s files as it does in UNIX.

- eTrust AC does not give the superuser (root) automatic access to every file on the system. The superuser is subject to authorization checking like all other users of the system.

- Authorization checking is based on the eTrust AC normal and conditional access lists, day and time restrictions, security levels, security categories, and security labels.

- If you do not specifically authorize a user to access a file, eTrust AC checks whether that user belongs to any group authorized to access the file.

- Each file access is audited through the normal eTrust AC audit procedures.

- When renaming or deleting a file, eTrust AC requires the user to have RENAME and DELETE access authority to the specified file, whereas UNIX requires the user to have WRITE authority for the parent directory.

- All users are given permanent READ access (as a minimum) to the files /etc/passwd and /etc/group, regardless of the default setting of these files. This prevents the possible hanging of the system.

- The owner of a FILE object in the eTrust AC database always has full access to the file protected by the object.

- The chdir access type controls the chdir command specifically, and does not execute, as UNIX does.

The following are the limits of the File Protection System:

- With respect to users who are not members of the special "_restricted" group, eTrust AC can protect only the following files and directories:
  - Files and directories that are defined by their individual names in the database
  - Files and directories that match a name pattern (for example, /etc/*) that is defined in the database

- With respect to users that belong to the _restricted group (see page 50), all remaining files can also be protected.
- eTrust AC maintains a table of all file names and directory names (including patterns using wildcards) that indicate resources that need protection. The amount of memory available for this table is limited. Normally, the maximum number of files and directories you can define by individual names in the database is 4096, and the maximum number of name patterns is 512.

- Some files receive protection even if no explicit access rules exist for them. These include the eTrust AC database files, audit logs, and configuration files.

  **Note:** For more information, see the FILE class in the *Reference Guide*.

eTrust AC supports the following access types for files.

- ALL
- CHDIR
- CHMOD
- CHOWN
- CONTROL
- CREATE
- DELETE
- EXECUTE
- NONE
- READ
- RENAME
- SEC
- UPDATE
- UTIME
- WRITE

**Note:** For more information about these access types, see the ADMIN class in the *Reference Guide*.

The File Protection System is useful for protecting selected sets of files that contain sensitive data. For example, you can use eTrust AC to protect the following files:

- /etc/passwd
- /etc/group
- /etc/hosts
- /etc/shadow
You should use eTrust AC to protect databases (access should be granted only to the server daemon) and all other sensitive files at your site.

Some files that always need access control are governed by rules even without you specifying them.

**Note:** For more information about these files, see the FILE class in the Reference Guide.

**How File Protection Works**

When the seosd daemon starts, it performs the UNIX stat command for each discrete file object defined in the database. It then builds a table in memory that contains an entry for each file object. In addition, for each discrete file, the table contains the file's inode and device; with this information, eTrust AC can also protect the hard links to the files because the protection is according to device and inode. The database does not keep information about a file's inode and device.

When creating a new file rule through eTrust AC:

- If the file exists in UNIX, eTrust AC first performs a stat command for the file and then adds a new entry to the file table with the file's inode and device information.
- If the file does not exist in UNIX, eTrust AC adds a new entry of the file's name to the file table (without inode and device information). This entry is the same as the entry for a generic file object. At the same time, the kernel keeps an indication in its internal tables that this file must be checked during creation for inode and device information. When the file is subsequently created, the kernel intercepts its creation and informs seosd of the file's inode and device information so that seosd can update the file's entry in the file table.

When you delete a file, eTrust AC deletes its entry in the seosd file table, but the entry remains in the eTrust AC database in case you create it again.
Protecting Files

To define a protected file in selang, enter the following command:

```
eTrustAC> newres FILE filename
```

For example, to register a file named /tmp/binary.bkup, enter the following command:

```
eTrustAC> newres FILE /tmp/binary.bkup
```

**Note:** When you define a file rule without specifying its access type, the default access of NONE is assigned. In this case, the file's owner is the only one who can access the file.

Most protected files should be protected from access by the superuser. Otherwise, any user who knows the superuser's password gains automatic access to the files. At the same time, you can prevent all other users except the file's owner from accessing the file.

To protect several similarly named files, use a file name pattern that includes a wildcard. The wildcards are `*` (which indicates zero or more characters) and `?` (which indicates any one character, other than `/`).

The pattern that you specify is matched against the file's full path name so that the pattern `/tmp/x*` matches files named `/tmp/x1`, `/tmp/xxx`, and even `/tmp/xdir/a`.

Patterns that eTrust AC does not let you specify are: `/`, `/tmp/*`, and `/etc/*`.

**Important!** Because file name patterns are such a powerful tool, you should not experiment freely with them.

For example, the following command defines as protected every file in the `/tmp` directory that has a name starting with a, and ending with b (this would include a file like `/tmp/axyz/axyzb`):

```
eTrustAC> newres FILE /tmp/a*b
```

To protect one or more files in the Security Administrator (the eTrust AC user interface for UNIX machines), select the relevant files on the Resources tab, and then choose Edit, Update.

**Note:** For more information about protecting files using Security Administrator, see resource administration in the *User Guide*. 
Restricting File Access

To restrict a file from access by the superuser in selang, use a longer version of the newres command. For example, to prevent the file /tmp/binary.bkup from being accessed by the superuser, as well as any other user except the user "myuser," use the following command:

eTrustAC> newres FILE /tmp/binary.bkup owner(myuser) defaccess(N)

This newres command does the following:

1. Defines /tmp/binary.bkup as a protected file.
2. Sets the user myuser as the owner of the file, granting myuser access to the file.
3. Sets the default access of the file to NONE, preventing any other user from accessing the file. To permit other users access to the file, you must explicitly define access rules for that file.

Important! If you invoke the selang command under root authority and then define FILE records without explicitly specifying another user as their owner, root becomes the owner of those files. As the owner, root (or any user who logs in as root) has complete and free access to the files.

Note: You can set the token use_unix_file_owner in the seos.ini file to "yes." This permits regular UNIX users to define access rules for the files they own.

To restrict file access in the Security Administrator (the eTrust AC user interface for UNIX machines), select the relevant files on the Resources tab, and then choose Edit, Update.

Note: For more information about Security Administrator, see the User Guide.
Preventing File Access

Sometimes it is convenient to define a FILE record that has no owner. To define a FILE record that does not have an owner in selang, use the special owner “nobody.”

For example, to define the file /tmp/binary.bkup as a protected file and prevent all users from accessing the file, enter the following command:

```
eTrustAC> newres FILE /tmp/binary.bkup owner(nobody) defaccess(N)
```

This newres command ensures that even the user who defined the command, whether root or otherwise, cannot access the file. After preventing all users from accessing a file, you must usually grant one or more users access to that file explicitly.

To explicitly permit a user access to a protected file, use the authorize command. For example, to grant the user "userJo" update access to all files in the /tmp directory beginning with Jo, enter the command:

```
eTrustAC> authorize FILE /tmp/Jo* uid(userJo) acc(Update)
```

You can prevent and permit file access in the Security Administrator by selecting the file on the Resources tab, and choosing Edit, Update.

**Note:** For more information, see resource administration in the User Guide.

**Note:** eTrust AC protects only those files defined in its database.

Viewing Default Access Authority

To view the default access of users in the _restricted group (when no matching records are found), use the selang showres command with the _default record of the class.

For example, to view the default access that users in the _restricted group have for files that are not in the eTrust AC database, use the showres command to display the _default resource of FILE class:

```
eTrustAC> showres FILE _default
```

**Note:** All other users have access defined by eTrust AC.

In the Security Administrator, select the class in left panel of the Resources tab, and double-click the Default Policy object in the right panel.
Using Conditional Access Control Lists

You can make access to a file conditional on the use of a particular program. Such conditioning is called program pathing.

**Note:** For program pathing that specifies access through a shell script, the shell script must have `#!/bin/sh` as its first line.

The following code is an example, allowing any process to update the file `/etc/passwd` under the control of the password change program `/bin/passwd.` All access attempts to the `/etc/passwd` file that do not originate from `/bin/passwd` are blocked.

```
eTrustAC> newres FILE /etc/passwd owner(nobody) defaccess(R)
eTrustAC> authorize FILE /etc/passwd gid(users) access(U) via(pgm(/bin/passwd))
```

The `newres` command defines the file `/etc/passwd` to eTrust AC and allows any user, including the file's owner, to read the file. The `authorize` command allows all users to access the file when the access is made under the program `/bin/passwd`. Once the password file is protected in this manner, any Trojan horse that inserts entries into the `/etc/passwd` file or any update to the password file by a user of the group "users" is blocked if the user is not using the `/bin/passwd` program.

Conditional access lists are also useful for controlling access to the files of a database management system (DBMS). Usually, you should permit users to access such files only through the programs and utilities supplied by the database vendor. Consider the following commands:

```
eTrustAC> authorize FILE /usr/dbms/xyz uid(*) \ via(pgm(/usr/dbms/bin/pgm1)) access(U)
eTrustAC> authorize FILE /usr/dbms/xyz uid(*) \ via(pgm(/usr/dbms/bin/pgm2)) access(U)
```

This set of authorize commands allows all eTrust AC users to access the file `xyz` of the DBMS system provided the access is made by either program pgm1 or program pgm2, which belong to the DBMS binaries directory. Note the use of the asterisk in the user operand. The asterisk specifies all users who are defined to eTrust AC. The use of the asterisk is similar in concept to the default access, except that default access also applies to users who are not defined to eTrust AC. Note that you can use the _undefined group for users not defined in the eTrust AC database.

You can also use the Unicenter TNG calendar (see page 249) ACL property to permit or deny access to specific users and groups for the current resource according to the Unicenter TNG calendar status. There are two types of ACL properties for Unicenter TNG calendars: regular and restrictive.
For example, the following command adds a user named george to a conditional access control list for a regular calendar named basecalendar:

```
eTrustAC> auth file file1 uid(george) calendar(basecalendar) access(rw)
```

And the following command removes a user named george from the Unicenter TNG calendar:

```
eTrustAC> auth- file file2 uid(george) calendar(basecalendar)
```

**Using Negative Access Control Lists**

You can deny a user or group specific access types using a Negative Access Control List (NACL).

With the eTrust AC language (selang), use the following command to deny access:

```
eTrust> {authorize | auth} className resourceName [gid(group-name...)]   \
   [uid({user-name...|*})] [deniedaccess(accessvalue)]
```

**Blocking Trojan Horses with the _abspath Group**

Any relative path names in the $PATH variable, but particularly the dot (.) path name meaning “current directory,” is a security weakness. Consider the following scenario:

- At the top of the PATH variable for root is the current (.) directory.
- A malicious user creates a destructive program—a Trojan horse—and stores it as /tmp/ls.
- In time, as the malicious user expects, root issues the ls command in the /tmp directory. Instead of running the usual ls command, root actually runs—with full administrative privileges—the Trojan horse that had been stored in the /tmp directory.

To eliminate this security weakness, eTrust AC provides a user group named _abspath. All members of the _abspath group are forbidden to use relative path names in invoking programs.

You can add a user to the _abspath group just as you add one to any other group. Effective at the next login, the user is forbidden to use relative path names when accessing programs.
Synchronization with Native UNIX Security

Although eTrust AC permissions are more complex than native UNIX permissions, you can synchronize your native UNIX permissions to your eTrust AC permissions. That is, you can make the permissions coincide. However, the synchronization is subject to some limitations:

- Synchronization is not retroactive. Once it is in effect, it can govern all newly issued eTrust AC authorization commands, but it does not govern pre-existing access rules.
- Permissions that you grant in eTrust AC can be passed to UNIX, but permissions granted in UNIX are not passed to eTrust AC.
- Because of limitations in its own system of permissions, UNIX may be unable to adopt more than a simplified form of the eTrust AC permissions. Even UNIX versions that feature access control lists (ACLs) may be unable to reflect all the complexity of the eTrust AC ACLs.

UNIX platforms with ACLs that can be synchronized to eTrust AC are Sun Solaris (see page 88), HP-UX (see page 88), and Tru64.

Without such ACLs, you can still synchronize the traditional UNIX rwx permissions to the eTrust AC permissions, to the extent possible.

Synchronization is controlled by the combination of the authorize command's UNIX option and the seos.ini file's SyncUnixFilePerms token:

- By including the UNIX option, the authorize command calls for implementation in UNIX as well as in eTrust AC. The command can even grant UNIX permission where permission did not exist before.
  
  (When the UNIX option is not used, selang commands have no effect on UNIX security. Moreover, where UNIX retains a prohibition, an eTrust AC permission is not effective. So the only way that selang can overcome a UNIX prohibition is with the UNIX option of the authorize command.)

- In the authorize command, the UNIX option works only when the SyncUnixFilePerms token is appropriately set in the [seos] section of the seos.ini file. The token has several permitted values:
  
  - **no** specifies not to synchronize ACL permissions. This is the default value.
  - **warn** specifies not to synchronize ACL permissions, but to issue a warning if the eTrust AC and native UNIX permissions conflict.
  - **traditional** specifies to adjust the rwx permissions for the group according to the eTrust AC ACL (and permissions for individual users are not copied to UNIX).
  - **acl** specifies to adjust the UNIX ACL according to the eTrust AC ACL.
force specifies to adjust the UNIX world access attribute according to the eTrust AC defaccess permissions.

Any change in the SyncUnixFilePerms token value takes effect only after you restart the seosd daemon.

Example: Synchronization

The following example involves a file named /var/temp/newdata and a user named fowler, and assumes that a record in the FILE class already represents the file.

1. Shut down the seosd daemon, so you can edit the seos.ini file:
   # secons -s

2. Logged in as a user with permission to edit the seos.ini file, edit the seos.ini file to make the SyncUnixFilePerms line, in the [seos] section, look like this:
   
   SyncUnixFilePerms = acl

   Remember, acl means that the UNIX option adjusts the UNIX ACL according to the eTrust AC ACL. The UNIX option will have this function as long as the token remains set to acl.

3. Restart the seosd daemon:
   
   # seosd

4. Invoke selang, then issue the following selang command:
   
   eTrustAC> authorize FILE /var/tmp/newdata uid(fowler) access(r w) unix

   The command gives fowler Read and Write access to the new data file and, by specifying the UNIX option, it grants the corresponding native UNIX permissions.
HP-UX Limitations

The ACL of HP-UX is limited in how it can reflect the ACL of eTrust AC.

- In HP-UX, the ACL assigns access per user and group combination. That is, the assigned access applies to the specified user only when the user's primary group is also specified.

  eTrust AC, on the other hand, assigns access per user or per group, but not per combination.

  Accordingly, eTrust AC permissions are mapped to HP-UX user/group combinations in which either the user or the group is set to the equivalent of “*” or “any.”

- HP-UX does not support ACLs on file-systems that are under control of the volume manager (LVM). Thus, some important HP-UX machines are likely to allow ACL synchronization only on the “root” file-system.

- The ACL of HP-UX is limited to 16 entries. eTrust AC synchronization uses the available entries as efficiently as possible, but 16 entries may not be enough to reflect every eTrust AC ACL completely.

Sun Solaris Limitations

Under Sun Solaris, native UNIX ACLs are not implemented in the /tmp directory.
Monitoring Sensitive Files

The Watchdog can protect the binaries of your setuid/setgid programs, as well as any other files you specify. The seoswd utility (the Watchdog daemon) continually checks two issues:

- Whether the seosd daemon is alive and responding. (If necessary, the watchdog daemon restarts the seosd daemon.)
- Whether a user has modified any trusted programs or files. (If so, seoswd prevents these files from executing.)

When the seosd daemon forks, it automatically executes the seoswd program to start the Watchdog.

**Note:** For more information about seoswd, see the *Utilities Guide*.

The seos.ini file contains several tokens that control the scanning and time-out values of the watchdog. It also contains the most up-to-date documentation on these values.

**Note:** For a description of the seos.ini file, see the *Reference Guide*. To learn more about the utilities and daemons that use the seos.ini file, see the *Utilities Guide*.

You can use the Watchdog to perform the same background checks as those made for the setuid and setgid programs on ordinary files, including generating audit records when these files are altered.

For example, consider a configuration where only the security administrator is allowed to modify the file /etc/inittab. To make eTrust AC monitor the file and generate an alert in any case of modification, use the following command in selang:

```
eTrustAC> newres SECFILE /etc/inittab
```

In the Security Administrator (the eTrust AC user interface for UNIX machines), select the System Resources category in the left panel of the Resources tab, and select Files and Directories. Find the /etc/inittab object in the right panel and double-click it.

**Note:** For more information about resource administration, see the *User Guide*.

The file /etc/inittab is now constantly monitored for modifications.
Protecting setuid and setgid Programs

Set user ID (setuid) programs are among the most frequently used programs at a UNIX site. A process that invokes a setuid program automatically acquires the identity of the owner of the setuid program. If the owner of a setuid program is root, then any regular user automatically becomes superuser by invoking the setuid program. When the setuid program starts, the process can do anything a superuser can do, so it is extremely important to make sure that setuid programs do exactly what they are supposed to do and nothing else. Back doors or shells within a setuid program grant the user access to everything on the system.

eTrust AC uses the PROGRAM class to protect setuid and setgid programs. Upon installation, eTrust AC permits any program execution by default. After defining trusted programs in the database, you can change the behavior of eTrust AC so that execution of a setuid or setgid program is prohibited unless the program is defined as a trusted program. For example, to allow /bin/ps (the process status program) to run as a setgid program (as it is supposed to), use the following selang command:

```
eTrustAC> newres PROGRAM /bin/ps defaccess(EXEC)
```

eTrust AC registers the program /bin/ps as a trusted program. It then calculates and stores its CRC, inode number, size, device number, owner, group, permission bits, last modification time, and, optionally, other digital signatures in a record in the PROGRAM class of the database.

The Watchdog periodically checks the program's CRC, size, inode, and the rest of the characteristics. If any of these values have changed, the Watchdog automatically asks seosd to remove the program from the trusted programs list and deny access to it. This ensures that no one can misuse the program by modifying or moving setuid programs. Note that the permission in the example newres command allows all users, including those not defined in the database, to run the /bin/ps command.

Untrusted setuid programs are possibly the most dangerous security loophole of UNIX-based operating systems. By using trusted programs' access rules, the security administrator can restrict the use of setuid to certain trusted programs that were tested and checked to ensure their integrity. However, any user cannot automatically start a trusted executable; the access rule must specify explicit users and groups that are granted access to that setuid program. For example, the following set of selang commands grants the execution of /bin/su only to the System Department users (group sysdept):

```
eTrustAC> newres PROGRAM /bin/su defaccess(NONE)
eTrustAC> authorize PROGRAM /bin/su gid(sysdept) access (EXEC)
```
Use an asterisk (*) to specify all users who are defined in the database. For example, to permit all users who are defined to eTrust AC to perform the su command, enter the following command:

```
eTrustAC> authorize PROGRAM /bin/su uid(*) access(EXEC)
```

This description is also true for setgid executables.

You can use the nr and er commands to register the setuid and setgid programs in the PROGRAM class. Important non setuid and setgid programs can be registered in the PROGRAM class similarly. Define a FILE rule for these programs to prevent unauthorized users from upgrading them. If you want to allow the program execution when it is untrusted (after upgrade, the program is executed without being retrusted), set the blockrun property to no.

- If the blockrun property is set to yes, the program is not executed until it is re-trusted and is not allowed to access any file that the relevant PACL would allow. The PACL is effectively disabled until the program is re-trusted.
- If the blockrun property is set to no, the program is executed and it is allowed to access any file that the relevant PACL would allow.

To set the value of the blockrun property to yes, use the following editres/newres command:

```
er program /bin/p blockrun
```

To set the value of the blockrun property to no, use the following editres/newres command:

```
er program /bin/p blockrun-
```

By default, for all the programs registered in the PROGRAM class, the blockrun property is set to yes. You can change this using the SetBlockRun token in the seos.ini file. Refer to the seos.ini file description for details.

### Defining setuid/setgid Programs Automatically

eTrust AC provides a way to define all your setuid and setgid programs automatically. Use the utility program /bin/seuidpgm to build the set of commands to define all the setuid programs and their permissions.

For example, to scan the entire file system for setuid and setgid programs and write the generated selang commands to the file /tmp/pgm_script, enter the following selang command:

```
# seuidpgm -qln / -x /home > /tmp/pgm_script
```

You can edit and modify the output file generated by seuidpgm according to your needs before submission.
Protecting Regular Programs

**Note:** For more information about the seuidpgm utility, see the *Utilities Guide*. To learn how to give similar protection to programs that are neither setuid nor setgid programs, see the SECFILE class in the *Reference Guide*.

### Conditional Access

Another sophisticated permissions technique is the conditional access rule (see page 84). For example, suppose you have a very secure version of the su command called securedSU that uses a fingerprint reader to verify the user's identity before allowing the user to become a superuser.

One way to ensure that UserX can become superuser only under that program is to set a conditional access rule as follows: (Before setting the rule, you must also set defaccess(none) for USER.root.)

```
eTrustAC> authorize SURROGATE USER.root uid(UserX) via(pgm(securedSU))
```

### Protecting the Login Command

We strongly recommend that you limit the use of /bin/login to the superuser only. Otherwise, any user who knows another user's password can log in as another user and supply the other user's password to bypass all surrogate and terminal restrictions.

To change the /bin/login permissions in selang, use the following command:

```
eTrustAC> chres LOGINAPPL /bin/login defaccess(N) owner(root)
```

In the Security Administrator (the eTrust AC user interface for UNIX machines):

1. Select the Login on Application item in left panel of the Resources tab.
2. Click the BIN_LOGIN icon in the right panel, and then choose Edit, Update.
3. In the Update field of the LOGINAPPL dialog, enter `root` in the Owner text box, select `None` in the Default Access check box, and then click OK.

**Note:** For more information about Security Administrator, see the *User Guide*.

### Protecting Regular Programs

eTrust AC can also protect regular programs in the same way it protects setuid and setgid programs (see page 90). To do this, set the blockrun property in the PROGRAM class to the value you choose.

**Note:** For more information about possible options, see the *Reference Guide*. 

You must protect mission-critical processes, such as database servers or application daemons, against denial of service attacks. The native UNIX security system bases its process protection on the process user ID. This implies that under native UNIX, root can do anything to any process. eTrust AC adds to UNIX process protection by defining rules based on the executable file running in the process. eTrust AC process protection is independent of the user ID of the process. A record in the PROCESS class must define every process that eTrust AC will protect.

For example, to protect the ASCII viewer /bin/more from being killed, follow this procedure:

1. Start selang.
2. Enter the following selang command:

   ```
   eTrustAC> newres PROCESS /bin/more defaccess(N) owner(nobody)
   ```

   This command defines /bin/more as a process to be protected from kill attempts; therefore the default access is `none` (N). The `owner(nobody)` setting ensures that even the user who defined this rule cannot kill the /bin/more process.

3. Exit selang.
4. Test the rules that Step 2 defined:
   a. Enter the command:

      ```
      eTrustAC> /bin/more /tmp/seosd.trace
      ```

   b. Assuming the file /tmp/seosd.trace is large enough to keep /bin/more from exiting immediately, press Ctrl+Z to suspend the /bin/more process.

   c. Try to kill the suspended job by entering the command:

      ```
      eTrustAC> kill %1
      ```

   Your attempt should fail, with eTrust AC displaying the "Permission denied" message.
To make an exception that permits a specific user to kill the /bin/more processes, enter the command:

eTrustAC> authorize PROCESS /bin/more uid(username)

**Note:** Use the same procedure to protect other binary executables on your system from being killed.

eTrust AC protects regular kill signals (SIGTERM) and the kill signals that an application cannot mask (SIGKILL and SIGSTOP). It passes other signals, such as SIGHUP or SIGUSR1, to the process to determine whether to ignore or react to the kill signal.
Chapter 7: Controlling Login Commands

This section contains the following topics:

- Controlling the Login Process (see page 95)
- Controlling Generic Login Applications (see page 96)
- Defining User Authority to Use Terminals (see page 97)
- Password Checking and Login Restrictions (see page 101)
- Defining Time and Day Login Rules (see page 103)
- Disabling Concurrent Logins (see page 103)
- Limiting Concurrent Logins for a User (see page 104)
- Recognizing a Login Event (see page 106)

Controlling the Login Process

eTrust AC provides two types of login protection: by terminal, and by application. Using the TERMINAL class, you can establish which users can log in from which terminals or hosts.

**Note**: For more information about the TERMINAL class, see the Reference Guide.

You can also control which user or group can log in using a certain login application (such as telnet, ftp, and rlogin) with the LOGINAPPL class. By establishing the access rules of the class, you define specific rules for each login application. For instance, you can define rules that permit all users to ftp to your host, a limited number of users to telnet to your system, and no one to rlogin to the system. Each record in the LOGINAPPL class defines access rules for a specific login application.

Examples: LOGINAPPL

For example, to permit only an anonymous user to use the ftp application, use the following procedure:

1. Change the ftp default access to none with the following command:
   
   ```
   eTrustAC> cr LOGINAPPL FTP defaccess(NONE) owner(nobody)
   ```

2. Permit the user anonymous to use ftp with the following command:
   
   ```
   eTrustAC> auth LOGINAPPL FTP uid(anonymous) access(X)
   ```
To restrict users from the group named account to use only telnet:

1. Block the use of rlogin and rsh with the following command:
   
   eTrustAC> auth LOGINAPPL(RLOGIN RSH) gid(account) access(N)

2. Permit the group named account to use telnet with the following command:

   eTrustAC> auth LOGINAPPL TELNET gid(account) acc(X)

**Note:** The previous example shows RLOGIN and RSH restrictions, but other login programs should be included as well.

Whenever you add or use a new login program, you must add a new LOGINAPPL record using selang or Policy Manager.

The login interception sequence always starts with setgid or setgroup events, which are called **triggers**. The sequence ends with a setuid event that changes the user's identity to the real user who logged in.

Login applications issue a variety of system calls, which eTrust AC uses to monitor login activity. These login sequences are preset for standard login applications. You can see them by studying the eTrust AC trace file.

**Note:** For more information about the LOGINAPPL class and setting a sequence, see the *Reference Guide*.

---

### Controlling Generic Login Applications

eTrust AC can also control and protect generic login applications; this means that you can protect groups of login applications that match a certain rule with a generic pattern. To define a generic login application, use the LOGINAPPL class.

### Defining a Generic Login Application

To define a generic login application with selang, use the same commands as setting regular login restrictions, except for the LOGINPATH parameter, which should include a generic path composed of a regular expression using one or more of the following characters: [, ], *, ?. For example, to define a generic telnet application, issue the following command:

```
er LOGINAPPL GENERIC_TELNET loginpath(/usr/sbin/in.tel*)
```
**Generic Login Program Interception**

With regular login restrictions, the activated rules are obvious; if a LOGINAPPL object that has the intercepted login program specified for the loginpath property exists in the database, the rules for that object would apply.

However, for generic LOGINAPPL objects, eTrust AC does the following:

1. seosd searches for an exact match for the intercepted login application. (A matching login path for the LOGINAPPL object.) If found, the rules for that object apply.
2. If not found, the search continues for a LOGINAPPL object with a generic login path that matches.
3. If there is more than one match, the rules for the object with the more specific match apply.

**Defining User Authority to Use Terminals**

One of the most effective ways to block intruders from accessing the system is by terminal protection, that is, the source of the login. The source can be the host or the terminal (such as an X terminal or a console) from which the user logs in.

In today's modern architecture, a terminal is no longer the teletype machine UNIX was developed for. On most sites, a "pseudo terminal" is allocated through the pseudo terminal server (PTS) or by the X window manager, and the terminal's name is meaningless symbol for the security system. eTrust AC protects what we understand as a terminal. eTrust AC implements terminal protection during the login stage, when eTrust AC defines a terminal in one of three ways:

- When the user logs in from an X terminal using the XDM login window, eTrust AC takes the IP address of the X terminal translated to host name (from /etc/hosts, NIS, or DNS) to be the terminal used for the login request. eTrust AC can also protect using the IP addresses if the translation to the host name fails or if you prefer to use IP addresses.

- When the user logs in from a dumb terminal, the TTY name identifies the terminal.

- When the user logs in from the network (through telnet, rlogin, rsh, and so on), the requesting IP address translated to the host name (through /etc/hosts, NIS, or DNS) is taken to be the terminal name.
You can define login rules for a specific host by defining this host in the TERMINAL class and adding the appropriate users and groups to the object's access list. For each login source, you can also limit the days and hours in which login from this host or terminal is allowed by setting the day and time restrictions for the TERMINAL object.

In most cases, highly authorized users such as the superuser or system administrators must be restricted to terminals that are located in secure places. Intruders and hackers who wish to enter the system as superuser are not able to do it from their own remote stations; they have to work from one of the authorized terminals, which should be in a secured location.

When logging in from the network, you cannot be certain that the user is indeed sitting in front of the host console. The user could be sitting in front of any terminal attached to that host or communicating from any other node in the network authorized to receive services from the requesting host. Permitting a user to log in from another host implies that we permit login to that user not only from that specific station but also from any other terminal authorized by that station. To ensure isolation between departments, define terminal groups and allow users of each department to work only from the terminal group of their department.

Unlike other resources, in terminal authorizations the more the user is authorized to access information, the lower the user's terminal authorization should be. The superuser must be the most restricted user in terminal access to ensure that nobody can log in as root from remote unsafe terminals.

When defining terminals, eTrust AC requires you to explicitly specify the owner of the terminal definition. The reason is that if root, as the security administrator, becomes the owner of the terminal by default, it makes the terminal eligible for superuser login. In most cases, this is not wanted. To guard you from making such mistakes that may unintentionally cause loopholes, eTrust AC makes you define an owner when defining the terminal.

To define the terminal tty34, use the following command:

```
eTrustAC> newres TERMINAL tty34 defaccess(none) owner(userA)
```

This command creates a record for the terminal tty34, sets its default access to NONE, and defines userA as its owner. Note that userA, as the owner of the terminal, is automatically allowed to enter the system through terminal tty34.

To prevent all users from logging in from the terminal tty34, specify "nobody" as the owner:

```
eTrustAC> newres TERMINAL tty34 defaccess(none) owner(nobody)
```

To permit a user to log in from a particular terminal, enter the following command:
Defining User Authority to Use Terminals

```
eTrustAC> authorize TERMINAL tty34 uid(USR1)
```

This command permits USR1 to log in from terminal tty34.

Permission to use a terminal can also be granted to a group. For example, the following command permits members of the group DEPT1 to use the terminal tty34:

```
eTrustAC> authorize TERMINAL tty34 gid(DEPT1)
```

To define a group of terminals (known as a terminal group), enter the following command:

```
eTrustAC> newres GTERMINAL TERM.DEPT1 owner(ADM1)
```

To add member terminals to terminal group TERM.DEPT1, enter the following command:

```
eTrustAC> chres GTERMINAL TERM.DEPT1 mem(tty34, tty35)
```

To authorize USR1 to use this terminal group, enter the following command:

```
eTrustAC> authorize GTERMINAL TERM.DEPT1 uid(USR1)
```

This grants USR1 the authority to use both tty34 and tty35.

Restricting Terminals for Root Users

Another issue to consider is the default rule of the TERMINAL class. At the initial implementation stages, the default is set to permit anything that is not defined. In the case of a TERMINAL, this could be a shortcoming.

Consider the following situation: A site has a few hundred terminals, and you want most users to be able to log in from any terminal, but you want root to be able to log in only from two predefined terminals.

First we consider that setting the default of the TERMINAL class to READ enables anyone-including root-to log in from any terminal that does not have a specific TERMINAL record in the database. You do not want the superuser to be able to log in from any terminal. But, we also consider that setting the default of the TERMINAL class to NONE forces you to define each terminal in the database, which may be impractical.
Defining User Authority to Use Terminals

To solve this problem, eTrust AC supports the definition of an access control list within the _default record of the TERMINAL class. The following commands show you how to restrict root to two terminals with minimum effort:

newres TERMINAL term1 defaccess(N) owner(root)
newres TERMINAL term2 defaccess(N) owner(root)
newres TERMINAL _default defaccess(R)
authorize TERMINAL _default uid(root) access(N)

The first two commands define term1 and term2 as terminals owned by root, so they are eligible for superuser login. The newres TERMINAL _default and chres commands set the default access to READ, so that any terminal not defined in the database is accessible to anyone. The authorize command explicitly denies access of the superuser to undefined terminals.

Note: The UACC class still exists; you can use it to specify the default access of a resource. However, using _default records to specify the default access of a resource is much easier.

Recommended Restrictions

You should restrict the use of the loopback terminals, local host terminals, and station host names if the default access for the TERMINAL class is READ. Allowing users to use these terminals permits all other users to substitute their own user IDs if they know the target user's password. For example, consider the following scenario:

- User U is allowed to work from terminal T.
- Terminal T is not allowed for superuser login.
- User U is not authorized to substitute user ID to root.
- User U managed to get the superuser password.
- All users are permitted to log in from terminal loopback.

User U can bypass this set of access rules by simply performing the command telnet loopback, specifying the user ID root, and supplying the password. Now a superuser session has started from terminal T, which is not supposed to allow superuser login. A user can similarly bypass access rules by exploiting the local host or the station's host name.

To restrict these three vulnerabilities, use the following definitions:

eTrustAC> newres TERMINAL loopback defaccess(N) owner(nobody)
eTrustAC> newres TERMINAL localhost defaccess(N) owner(nobody)
eTrustAC> chres TERMINAL hostname defacc(N) owner(nobody)
An alternative approach to preventing this security breach is to limit the TCP requests for telnet, ftp, and so forth from local host.

Yet another option is to set default access for the TERMINAL group to NONE, then specify TERMINAL and GTERMINAL rules.

**Password Checking and Login Restrictions**

eTrust AC does not replace the /bin/login executable. Even when eTrust AC is running, passwords continue to be checked against /etc/passwd, the shadow password file, or the NIS passwd map. But eTrust AC also performs additional checks, described in the following section.
Logon Checks

After the login process passes the authentication stage, eTrust AC intercepts the process and checks the following points:

- Has the password expired?
  If it has, the user receives a number of grace logins accompanied by warnings before being denied access. Following access denial, the security administrator must reassign the user's password. The number of grace logins is determined by the user password policy, which you can specify either globally with the setoptions command, or for a profile group with the chgrp command.

  **Note:** For more information about the setoptions command, see the *Reference Guide*.

  You can use the segrace utility to view the number of grace logins left for a user, the number of days remaining until the user's existing password expires, or the date and time the user last logged on and from which terminal.

  **Note:** For more information about the segrace command, see the *Utilities Guide*.

- Is the user logging on from an authorized terminal?
  If so, login proceeds normally to the next check; if not, the user cannot log in.

- Do the current time-of-day and day-of-week allow login (per the predefined restrictions)?
  If they do, login proceeds normally to the next check; otherwise, the user cannot log in.

- Was this user name unused for more than a predefined number of days?
  If it was, access is denied. (The default is 90 days; use the setoptions command to change it.)
Defining Time and Day Login Rules

Information security is most vulnerable in times of low activity. Late hours of the night and weekends are ideal times for breaking in, because fewer people are available to monitor the audit records. Setting up appropriate terminal authority rules forces an intruder to use a terminal that is in a protected location. Setting up days-of-week (DOW) and time-of-day (TOD) access rules forces the intruder to make break-in attempts during work hours when offices are open and active. This combination severely restricts alien break-ins.

Limiting the days and hours in which a user can log in is done on a user-by-user basis. To define the DOW and TOD login restrictions for a user, use the following command:

```
eTrustAC> chusr USR1 restrictions(days(Mon,Tue,Wed)time(800:1700))
```

This command permits user USR1 to log in only between 8:00 and 17:00 on Mondays, Tuesdays, and Wednesdays. USR1 cannot log in outside the specified time on the specified days, or on days other than those specified.

The days parameter also accepts the values ANYDAY (allow logins on all seven days of the week) and WEEKDAYS (allow logins Monday through Friday). The time parameter also accepts the value ANYTIME (allow logins at any time of the day).

**Note:** You can apply the DOW and TOD restrictions to many resources defined in the database. This feature is particularly useful for giving terminals and terminal groups limited periods of usability.

Disabling Concurrent Logins

Most UNIX-based operating systems allow concurrent logins. But if a user is permitted to log in from more than one terminal, there is a danger that while the user is logged in, other users can log in from elsewhere and masquerade as that user.

After you log in, eTrust AC allows you to disable your own concurrent login permission so that no one else can log in as you from another terminal. However, you can still log in repeatedly from the particular terminal that you are using. Use the secons command with the following switches:

```
# secons -d-  (disables concurrent login)
# secons -d+  (enables concurrent login)
```
Any user can issue the -d option. (All other options are only allowed for users with the ADMIN or OPERATOR attribute). Users who want to disable concurrent logins can use this command in their initial scripts. Although they are then able to open as many windows as they want, they cannot log in from a second terminal.

**Note:** If you use the secons -d- command to prevent concurrent logins, you must remember to use secons -d+ before logging out, to avoid being locked out of the system. If you forget to reinstate concurrent logins and try to log in again, eTrust AC allows you to log in provided no process with the same user ID is running.

### Limiting Concurrent Logins for a User

eTrust AC can control the number of concurrent logins in two ways:

**Administrator Level**

Set a systemwide definition in the database of the number of concurrent sessions a user can have. You can set this value globally, for a profile group, or for individual users.

**User Level**

Users individually control the number of concurrent logins allowed for them. This way, when logging in, users can block the option of more login sessions with their names, thus protecting themselves.

**Note:** The number of concurrent logins is independent of the number of sessions the user is running on a particular terminal. Multiple sessions on one terminal are considered as a single login. The concurrent-logins limit restricts the number of *terminals* a user can concurrently log in from, not the number of logins from each terminal.

### Limiting Concurrent Logins Globally

In selang, enter the following command:

```
eTrustAC> setoptions maxlogins(NumLogins)
```

In the Security Administrator, do the following:

1. Click Security Options on the toolbar.
2. Enter the value in the Maximum Number of Logged-in Terminals box and click OK. The Activity window opens.
3. Click Go to update the user or group.

**Note:** For information about the Activity window, see Security Administrator basics in the *User Guide*. 

104 Administrator Guide
Limiting Concurrent Logins Individually

In selang, enter the following command:

```
eTrustAC> chusr username maxlogsins(NumLogins)
```

In the Security Administrator, do the following for a single user or group:

1. Select the user in the Accounts tab.
2. Right-click and select Update.
3. For a user, click the Login tab on the right side of the dialog box; for a group, click the Profile login tab.
4. Enter the number of concurrent logins in the Max logins box.
5. Click OK; the Activity window opens.
6. Click Go to update the user or group.

**Note:** For information about the Activity window, see the Security Administrator basics in the *User Guide*.

In the Security Administrator, do the following for multiple users or groups:

1. Choose Login Protection Setup in the Tools menu.
2. Click the Restrictions tab in the dialog box.
3. In the Accessors to Protect section, click List to the right of the Users or Groups text box (you can also enter the names manually).
4. Select the users or groups in the left-hand box, and click the appropriate arrow to move them into the Selected box.
5. Click OK to return to the Log protection dialog box. The names you selected appear in the appropriate text box.
6. Enter the value in the Maximum Number of Concurrent Logins box and click OK. The Activity window opens.
7. Click Go to update the users or groups.

The concurrent logins limit set for a user overrides the systemwide limit. To prevent eTrust AC from enforcing the concurrent logins limit for a specific user, set the user's concurrent logins limit to zero. (Note that you cannot use selang if you set the maximum number of concurrent logins to one.)
Recognizing a Login Event

eTrust AC does not treat all attempts to change the user ID of a process as login events. Usually a program attempts to change its user ID with a setuid system call. The SURROGATE class controls these events, which are not necessarily considered login events, and do not necessarily change the user identity from the point of view of eTrust AC.

eTrust AC always preserves the original user identity—the identity with which the user logged in initially. Ordinary setuid system calls do not cause eTrust AC to register a change in user identity.

For eTrust AC to recognize the identity change, it must recognize this event as a login event. It recognizes login events using the following rules:

- The program that attempts to change the identity is defined as a login program. All programs in the LOGINAPPL class are login programs.
- The program executes a series of system calls corresponding to its definition in the LOGINAPPL class.

Note: Previous versions of eTrust AC used the configuration file loginpgms.init for the definition of login programs. This file is no longer used.

When you begin an administration session (in selang or the Security Administrator), eTrust AC performs a dummy login event. This is not a true login; rather, eTrust AC performs certain internal checks, which are similar to login checks.

Note: For more information, see the SEQUENCE property for the LOGINAPPL class in the Reference Guide.

At the start of an administration session, the user name is checked in the machine to be administered. You get access to this machine for administration only if you have WRITE access for the terminal from which you perform the session.

For example, if you are logged in to host Minerva and would like to administer eTrust AC on host Artemis, two conditions are necessary:

- A TERMINAL object called Minerva (or the relevant fully qualified name) is in the database record for Artemis.
- You are listed in the ACL of this object with WRITE permission.

These conditions are checked prior to any other user authority check. Note that you also need administrative authority in the database.
Chapter 8: Protecting TCP/IP Services

This section contains the following topics:

- Why Protect TCP/IP Services (see page 107)
- Restricting TCP/IP Services (see page 107)
- Using the TCP Class (see page 110)

Why Protect TCP/IP Services

Protecting TCP/IP services is most important for file servers that contain sensitive data. These servers must provide certain services only to trusted stations, and not to intruders or computers that are unknown to the host.

Restricting TCP/IP Services

In an open network, any station can request services from other computers on the network. The TCP/IP protocol can be used to supply many services. Some of these services, such as rlogin, rcp, rsh, ftp, telnet, and rexec, are common to all UNIX-based operating systems. Others are provided by in-house and third-party software.

eTrust AC intercepts the accept processes of TCP/IP at the host computer and determines whether the accept program should continue normally or be overridden. eTrust AC bases its decision on access rules governing hosts and services that you define. You can create TCP/IP access rules in the database to specify the computers and networks that are allowed to receive services such as file transfers, remote login, and remote shell from a specific computer.

The following examples show how TCP/IP access rules can be defined and set to efficiently block unwanted outsiders. If you have not yet had time to develop a complete database, you may want to let any station that is not defined in the database receive any service. If so, set the HOST record in the UACC class as follows:

```
eTrustAC> chres UACC HOST defaccess(READ)
```
A station that is to have access rules for TCP/IP services from the local host is defined in a record in the database under the HOST class. For each of these stations, the services allowed are listed in the record. For example, the following command sequence defines a record for station ws5 and denies it from receiving any TCP/IP service from the local host:

```
eTrustAC> newres HOST ws5
eTrustAC> authorize HOST ws5 service(*) access(NONE)
```

The following command allows ws5 to perform telnet to the local computer:

```
eTrustAC> authorize HOST ws5 service(telnet)
```

These settings allow users to telnet to the local computer, which means that the remote user must specify a user name and password before using the local system. To allow a station to receive all TCP/IP services from the local computer, you can use an asterisk in the service keyword. For example, the following command allows ws5 to invoke any TCP/IP service from the local computer:

```
eTrustAC> authorize HOST ws5 service(*)
```

The service can be specified in several ways, some of which involve the port number. The port number is an identification number for a service. All services have port numbers, and the port numbers are mapped to the services in the file /etc/services. You can specify a service in the following ways:

- By its name as defined in the file /etc/services
- By its port number
- As a range of port numbers
- As an RPC port that is listed in the /etc/rpc system file

For example, the following command permits ws5 to receive any TCP/IP service whose port number falls between 7045 and 7050:

```
eTrustAC> authorize HOST ws5 service(7045-7050)
```

In many cases, it is more economical to define a group of hosts and set its permissions once, instead of making permissions for each individual computer. eTrust AC provides the GHOST class, where each GHOST record defines a group of hosts. To define a GHOST record and add hosts to its member list, enter the following commands:

```
eTrustAC> newres GHOST gh1 mem(ws2, ws3, ws5)
eTrustAC> authorize GHOST gh1 service(ftp)
```

The newres command defines a group of hosts called gh1 that contain the members ws2, ws3, and ws5. The authorize command allows all three stations to receive ftp (file transfer) services.
Managing host groups is easier than managing individual stations, but to supply more flexibility, eTrust AC also supports the definition of network access rules. Networks are defined in the HOSTNET class. For example, consider the following set of commands:

```
eTrustAC> newres HOSTNET hn1 mask(255.555.0.0) match(192.168.0.0)
eTrustAC> authorize HOSTNET hn1 service(*) access(NONE)
eTrustAC> authorize HOSTNET hn1 service(ftp)
```

- In the first line, the `newres` command defines a network called `hn1`. With its mask and match values, it specifies that any computer with an IP address whose first two qualifiers are 192.168 is considered as coming from the `hn1` network.
- The combination of the second and third lines permits any station from the `hn1` network to perform `ftp`, but not any other service, in the host computer.

Another method eTrust AC provides for defining TCP/IP access rules is name-pattern access rules. eTrust AC supports the definition of generic records in the HOSTNP class (host name pattern) with wildcards.

**Note:** For information on how eTrust AC performs string matching, see the *Utilities Guide*.

For example, the following command sequence permits all hosts whose names start with the characters "lin" and end with the characters ".org.com" to receive all TCP/IP services on the local host:

```
eTrustAC> newres HOSTNP lin*.org.com
eTrustAC> authorize HOSTNP lin*.org.com service(*).
```

**Note:** Hosts that are managed by NIS must be identified by their official names that appear in a NIS map and not by their aliases. The chart in the following section summarizes the TCP/IP check flow.
Using the TCP Class

Alternatively, you can specify protection by service instead of by host, by using the TCP class.

Note: For more information about the TCP class, see the Reference Guide.

Use the TCP class to control incoming and outgoing services.

For example, the following commands create a record for the ftp service, with READ (meaning the service can be used) as default access type, but prevent hosts that match the name pattern PUBLIC* from receiving the service.

```
eTrustAC> newres TCP ftp defaccess(READ)
eTrustAC> authorize TCP ftp hostnp(PUBLIC*) access(N)
```

You can also specify that a particular user or group be only permitted to receive a particular service. For example, to allow all users to ftp to a host called hermes, but to specify that only members of the group called acctng can access hermes with telnet, enter the following commands:

```
eTrustAC> newres HOST hermes
eTrustAC> newres TCP ftp owner(nobody) defaccess(read)
eTrustAC> newres TCP telnet owner(nobody) defaccess(read)
eTrustAC> authorize TCP ftp uid(*) host(hermes) access(write)
eTrustAC> authorize TCP telnet gid(acctng) host(hermes) access(write)
```

Note: defaccess(read) disables outgoing services. defaccess(write) disables incoming services.

If the HOST class is active (that is, if it is used as a criterion for access), then the TCP class cannot effectively be active. You can use the command setoptions class- HOST to deactivate the HOST class; then use the command setoptions class+ TCP (if necessary) to activate the TCP class. Deactivating the HOST class automatically deactivates GHOST, HOSTNET, and HOSTNP as well.

Also, if the TCP class is active, use the setoptions command class- CONNECT to deactivate the CONNECT class.
Streams Module for Network Interception

By default, the TCP class is not active. Before you activate the TCP class, the CONNECT class, or the HOST class, be sure that the streams module is enabled.

To load the eTrust AC streams module on Solaris, complete the following steps:

1. Stop eTrust AC. Enter the following command:
   
   secons -s

2. Enter the following command:
   
   SEOS_load -s

3. Start eTrust AC. Enter the following command:
   
   seload

**Note:** If you attempt to activate the TCP class when the streams module is not loaded, an error appears:

**ERROR:** `<class>` class cannot be activated when streams are not loaded. Please use SEOS_load -s to load the streams.
The algorithm for incoming authorizations is:

1. Start by checking on IP address and service.
2. Is service specified in ACL for host?
   - Yes: Is service allowed?
     - Yes: Request to service granted.
     - No: Request to service denied.
   - No: Is host a member of GHOST record?
     - Yes: Is service specified in ACL for GHOST?
       - Yes: Request to service granted.
       - No: Request to service denied.
     - No: Request to service denied.
3. Is there a HOSTNET record for which IP&MASK=MATCH?
   - Yes: Is service specified in ACL for matching?
     - Yes: Request to service granted.
     - No: Request to service denied.
   - No: Does host name match a HOSTNET record?
     - Yes: Service allowed by the TCP's local address?
       - Yes: Request to service granted.
       - No: Request to service denied.
     - No: Request to service denied.
Using the TCP Class

Diagram A

Does service TCP record exist? yes
Use service record ACL

no
Use default record ACL

Does HOST record exist?

See the previous diagram
The algorithm for outgoing authorizations is:

1. Start NET check on IP address and service.
2. Does service TCP record exist?
   - Yes: Use service record ACL.
   - No: Use default record CACL.
3. Is user defined in SDO/F??
   - Yes: Does user appear in CACL?
     - Yes: Use diagram B for CACLS in which user appears.
     - No: Use diagram B for CACLS in which groups appear.
   - No: Does user appear in CACL?
     - Yes: Use diagram B for CACLS in which user appears.
     - No: Request to service granted.
4. Service allowed by the TCPs recent allowing.
   - Yes: Request to service granted.
   - No: Request to service denied.

The algorithm checks various conditions to determine whether a service request should be granted or denied.
Using the TCP Class

Protecting TCP/IP Services 115
Chapter 9: Managing Policies Centrally

This section contains the following topics:

- The Policy Model Database (see page 117)
- Architecture Dependency (see page 120)
- Methods for Centrally Managing Policies (see page 122)
- Automatic Rule-based Policy Updates (see page 122)
- Advanced Policy Management and Reporting (see page 144)
- Mainframe Password Synchronization (see page 172)

The Policy Model Database

Managing tens or hundreds of databases individually is not practical. eTrust AC supplies the Policy Model service, a component that lets you manage many databases from one central database. Using the Policy Model service is optional, but it greatly simplifies administration at large sites.

The Policy Model (PMD) service, uses a Policy Model database (PMDB). Like other eTrust AC databases, the PMDB contains users, groups, protected resources, and rules governing access to the resources. In addition, the PMDB contains a list of subscriber databases. Each subscriber is an eTrust AC database that resides on a separate computer, or another PMDB that resides on the same or another computer. A PMDB that updates a subscriber is the subscriber's parent.

The PMDB is a useful tool for managing many databases that have similar authority restrictions and access rules.

Note: For information about administrating a PMDB, see the sepmd utility in the Utilities Guide. For information about managing PMDBs remotely, see the Reference Guide.
PMDB Location on Disk

All PMDBs reside in a common directory (one per computer). The name of the directory is specified by the _pmd_directory_ token in the [pmd] section of the seos.ini file. The default value of _pmd_directory_ is eTrustACDir/policies, where eTrustACDir is the installation directory for eTrust AC (by default /opt/CA/eTrustAccessControl).

Each PMDB occupies a subdirectory in the common directory. The name of the subdirectory is the name of the Policy Model. The files in the subdirectory contain all the data required to define the Policy Model including the pmd.ini file.

Managing Local PMDBs

eTrust AC offers several utilities for administrating local PMDBs:

**sepmad**

A PMDB administration utility that lets you:

- Administer subscribers
- Truncate the update file
- Administer Dual Control
- Manage the Policy Model log file
- Perform other administrative tasks

**sepmadadm**

Creates PMDBs and configures them with the necessary settings for setting up your hierarchy.

**Note:** For a comprehensive discussion of the Policy Model utilities, see the *Utilities Guide.*
Managing Remote PMDBs

eTrust AC also offers you a range of selang commands that you can use in the pmd environment. These commands let you manage PMDBs remotely:

**createpmd**
- Creates a PMDB.

**deletepmd**
- Deletes a PMDB.

**findpmd**
- Displays the names of all PMDBs on the computer.

**listpmd**
- Lists the following information about a PMDB:
  - Subscribers and their status
  - PMDB description and its status
  - Commands in the update file and their offsets
  - Contents of the error log

**pmd**
- A PMDB administration command that lets you:
  - Remove a subscriber from the list of unavailable subscribers
  - Clear the Policy Model error log
  - Start and stop the Policy Model daemon
  - Truncate the update file
  - Reload the initialization files

**subs**
- A PMDB subscription command that lets you:
  - Add a subscriber to a parent PMDB
  - Assign a parent PMDB to a database (eTrust AC or another PMDB)

**subspmd**
- Assigns a parent PMDB to the local database.

**unsubs**
- Removes a subscriber from the PMDB.

**Note:** For a comprehensive discussion of selang commands you can use in the pmd environment, see the *Reference Guide*. 
When deploying eTrust AC, you should consider the hierarchy of your environment. At many sites, the network includes a variety of architectures. Some policy rules, such as the list of trusted programs, are architecture-dependent. On the other hand, most rules are independent of the system's architecture.

You can cover both kinds of rules by using a hierarchy. You can define a global database for architecture-independent rules, and give it subscriber PMDBs that define architecture-dependent rules.

**Note:** The root PMBD and all of its subscribers can reside on the same computer or on separate computers, depending on the physical needs of your environment.

**Example: A Two-tiered Deployment Hierarchy**

The following UNIX example also applies to a Windows architecture with small modifications.

In the example, the site consists of IBM AIX and Sun Solaris systems. Since the list of trusted programs on IBM AIX differs from the one on Sun Solaris, the PMDBs need to consider architecture dependency.

To set up a multiple-architecture PMDB, set up your PMDBs as follows:

1. Define a PMDB named whole_world, to contain the users, groups, and all other architecture independent policies.
2. Define a PMDB named pm_aix, to contain all the IBM AIX specific rules.
3. Define the PMDB pm_sol, to contain all the Sun Solaris specific rules.

   The PMDBs pm_aix and pm_sol are subscribers of the PMDB whole_world. All IBM AIX computers at the site are subscribers of pm_aix. All Sun Solaris computers at the site are subscribers of pm_sol. The concept is illustrated in the following chart.
4. When you enter platform-independent commands in whole_world, such as adding a user or setting a SURROGATE rule, all databases at the site are automatically updated.

5. When you add a trusted program to pm_aix, only IBM AIX computers are updated, without affecting the Sun Solaris systems.
Methods for Centrally Managing Policies

eTrust AC lets you manage several databases from a single computer in two ways:

- **Automatic rule-based policy updates**
  
  Regular rules you define in a central database (PMDB) are automatically propagated to databases in a configured hierarchy.

  **Note:** Dual control (see page 138) is only available with this method and on UNIX only.

- **Advanced policy management and reporting**
  
  Policies (group of rules) you deploy are propagated to all databases in a configured hierarchy. You can also undeploy (remove) policies and report on deployment status, deployment deviation, and deployment hierarchy. You need to install and configure additional components to use this functionality.

  **Note:** Advanced policy management and reporting offers additional functionality that is not available with automatic rule-based policy updates. However, to use advanced policy management you first need to configure your environment for automatic rule-based policy updates.

**Automatic Rule-based Policy Updates**

Single-rule policy updates (regular selang rules) you make in a central database are automatically propagated to the subscriber databases. By subscribing several computers to the same database, and by subscribing one database to another, you can create a hierarchy. You configure your environment for automatic rule-based policy updates after installation.

**Note:** This method of managing policies is limited to letting you make single-rule policy updates across your hierarchy. Other functionality is only available through implementing advanced policy management and reporting (see page 144).
How Automatic Rule-based Policy Updates Work

When you configure your environment for automatic rule-based policy updates, each rule you define in the central database is automatically propagated to all of its subscribers in the following way:

1. A rule is defined for any PMDB with at least one subscriber.
2. The PMDB sends the command to all subscriber databases.
3. The subscriber database applies the propagated command.
   a. If the subscriber database does not respond, the PMDB sends the command at a regular interval (by default, every 30 minutes) until the subscriber database has been updated.
   b. If a subscriber database is responding, but refuses to apply the command, the PMDB places the command in the Policy Model error log (see page 136).
4. If the subscriber database is a parent to other subscribers, it then sends the command to its subscribers.

Example: Removing a user from all computers in a hierarchy

If a user is deleted from a PMDB using the rmusr command, the same rmusr command is sent to all the subscriber databases. In this way, a single rmusr command can remove a user from many databases on a variety of computers.

How You Can Set Up a Hierarchy

eTrust AC uses the Policy Model service to propagate rule-based policy updates across the configured hierarchy. By subscribing several eTrust AC computers to the same PMDB, and by subscribing one PMDB to another, you create a hierarchy.

To enable automatic rule-based policy updates, do the following:

1. Create and configure the master PMDB (see page 124).
2. (Optional) Create and configure subscriber PMDBs (see page 126).
3. Define parent PMDBs for the subscribing computers (see page 128), called end-points.

Note: The following sections show how you set up a PMDB hierarchy. There are other ways of creating PMDBs and then setting their hierarchy. For a comprehensive discussion of the Policy Model utilities, see the Utilities Guide.
Create and Configure the Master PMDB

To let you manage policies from a central location, you first need to create and configure a master PMDB. To do this on a local host, you can use the `sepmdadm` command.

**Note:** The following procedure shows the interactive form of the `sepmdadm` command. For information about using the command-line parameters for all input, see the *Utilities Guide*.

**To create and configure the master PMDB**

1. In a command line, enter the following command:
   ```bash
   sepmdadm -i
   ```
   eTrust AC starts the Policy Model database administration script (`sepmdadm`) and displays a menu with options for you to choose from.

2. Enter 1, to select the first option (create a master PMDB and define its subscribers).
   The script is configured to ask you the relevant questions.

3. Press Enter to continue.
   The script continues to ask you the first question.
   **Note:** If eTrust AC is not running, the script issues a warning and lets you start eTrust AC before the script is rerun.

4. Enter a name for the Policy Model you want to create.
   The script registers the Policy Model name and continues.

5. Enter the name of the first subscriber computer you want to specify.
   The script registers the name of the first subscriber and then asks you to enter the name of the next subscriber.

6. Continue to enter subscriber names as necessary, then press Enter without entering a subscriber name.
   The script registers all subscriber names and continues.
   **Note:** You still must point each subscriber computer to its parent PMDB.

7. If you are running NIS, NIS+, or DNS, choose whether you want to update the NIS/DNS tables with PMDB changes.
   Updates are made to users and groups in the PMDB. The tables provide information on users and their characteristics. If you choose yes, a UNIX user or UNIX group updated through the Policy Model is also updated in the NIS passwd and group files.
   a. Enter `y` if you want to update the NIS/DNS tables.
      The script now asks you for the location of the NIS passwd and group files.
a. Enter the full path of the NIS password file.
   The script registers the full path and continues.

b. Enter the full path of the NIS group file.
   The script registers the full path and continues.

b. Enter n or press Enter if you want to update the NIS/DNS tables.
   The script registers your answer and continues.

8. Enter the users you want to give special attributes for the PMDB:
   a. Enter administrator names as necessary, then press Enter without
      entering an administrator's name.
      Administrators are authorized to change the properties of the PMDB.
      **Note:** At least one administrator must be defined in a PMDB (root is
      the default).

   b. Enter auditor names as necessary, then press Enter without entering
      an auditor's name.
      Auditors are authorized to view the PMDB's audit log files.

   c. Enter password manager names as necessary, then press Enter
      without entering a password manager's name.
      Password managers are authorized to change passwords in the PMDB.
      The script registers your answer and continues.

9. Enter administration terminals as necessary, then press Enter without
   entering an administration terminal.
   The script registers all administration terminals and then reports the
   selections you have made and asks you to confirm them.

10. Press Enter to confirm the selections you have made, or enter n to rerun
    the script with new inputs.
    If you confirm your selections, a new PMDB is created using the answers
    you supplied.
Create and Configure Subscriber PMDBs

Once you have a master PMDB configured, if you want to extend your hierarchy, you need to create and configure subscriber PMDBs. To do this on a local host, you can use the `sepmdadm` command.

**Note:** The following procedure shows the interactive form of the `sepmdadm` command. For information about using the command-line parameters for all input, see the *Utilities Guide*.

**To create and configure subscriber PMDBs**

1. In a command line, enter the following command:
   ```bash
   sepmdadm -i
   ```
   eTrust AC starts the Policy Model database administration script (`sepmdadm`) and displays a menu with options for you to choose from.

2. Enter 2, to select the second option (create a subsidiary PMDB and define its subscribers and parent.).
   The script is configured to ask you the relevant questions.

3. Press Enter to continue.
   The script continues to ask you the first question.
   **Note:** If eTrust AC is not running, the script issues a warning and lets you start eTrust AC before the script is rerun.

4. Enter a name for the Policy Model you want to create.
   The script registers the Policy Model name and continues.

5. Enter the name of the first subscriber computer you want to specify.
   The script registers the name of the first subscriber and then asks you to enter the name of the next subscriber.

6. Continue to enter subscriber names as necessary, then press Enter without entering a subscriber name.
   The script registers all subscriber names and continues.
   **Note:** You still must point each subscriber computer to its parent PMDB (see page 128).

7. Enter the name of the parent PMDB.
   The script registers the parent PMDB name and continues.
   **Note:** `sepmdadm` only lets you enter one parent for each subscribing database. You can, however, define multiple parents for each database. To do this, modify the `parent_pmd` token of the `pmd.ini` configuration file. For more information about using this token, see the *Reference Guide*.

8. If you are running NIS, NIS+, or DNS, choose whether you want to update the NIS/DNS tables with PMDB changes.
Updates are made to users and groups in the PMDB. The tables provide information on users and their characteristics. If you choose yes, a UNIX user or UNIX group updated through the Policy Model is also updated in the NIS passwd and group files.

a. Enter **y** if you want to update the NIS/DNS tables.
   
   The script now asks you for the location of the NIS passwd and group files.
   
   a. Enter the the full path of the NIS password file.
      
      The script registers the full path and continues.
   
   b. Enter the the full path of the NIS group file.
      
      The script registers the full path and continues.

b. Enter **n** or press Enter if you want to update the NIS/DNS tables.
   
   The script registers your answer and continues.

9. Enter the users you want to give special attributes for the PMDB:

   a. Enter administrator names as necessary, then press Enter without entering an administrator's name.
      
      Administrators are authorized to change the properties of the PMDB.
      
      **Note:** At least one administrator must be defined in a PMDB (root is the default).

   b. Enter auditor names as necessary, then press Enter without entering an auditor's name.
      
      Auditors are authorized to view the PMDB's audit log files.

   c. Enter password manager names as necessary, then press Enter without entering a password manager's name.
      
      Password managers are authorized to change passwords in the PMDB.

   The script registers your answer and continues.

10. Enter administration terminals as necessary, then press Enter without entering an administration terminal.

    The script registers all administration terminals and then reports the selections you have made and asks you to confirm them.

11. Press Enter to confirm the selections you have made, or enter **n** to rerun the script with new inputs.

    If you confirm your selections, a new PMDB is created using the answers you supplied.
Define Parent PMDBs for Subscribing Computers

To establish an end-point computer as a subscriber to a PMDB, you must do more than register the subscriber's name in the PMDB. You also need to complete a procedure at the subscriber computer.

To define parent PMDBs for subscribing computers

1. In a command line on the subscriber computer, start sepmdadm in interactive mode:
   ```
   sepmdadm -i
   ```
   eTrust AC starts the Policy Model database administration script (sepmdadm) and displays a menu with options for you to choose from.

2. Enter 3, to select the third option (define the parent and password PMDBs of the local host).
   The script is configured to ask you the relevant questions.

3. Press Enter to continue.
   The interactive script continues to ask you the first question.
   **Note:** If eTrust AC is running, the script issues a warning and lets you stop eTrust AC before the script is rerun.

4. Enter the name of the parent PMDB.
   The script registers the name of the parent PMDB name and continues.

5. Enter the name of the parent password PMDB.
   The script registers the name of the parent password PMDB name and then reports the selections you have made and asks you to confirm them.

6. Press Enter to confirm the selections you have made, or enter n to rerun the script with new inputs.
   If you confirm your selections, the subscriber computer is set up with these inputs.

**Note:** sepmdadm only lets you enter one parent for each subscribing database. You can, however, define multiple parents for each database. To do this, modify the parent_pmd token of the seos.ini configuration file. For more information about using this token, see the Reference Guide.
UID/GID Synchronization

As an administrator, you may receive messages that refer to users by UID and to groups by GID. You need to make sure that the UIDs and GIDs have the same meaning everywhere.

By default, the PMDB attempts to use the same UIDs and GIDs for new users and groups everywhere, but you can help by providing the necessary conditions from the start. Start with identical passwd files and identical group files, making sure that the synch_uid token in the pmd.ini file is set to yes. If your local database is a subscriber to your PMDB, and the PMDB is the only source of new users and new groups for your subscriber databases, then you can depend on compatibility between the UIDs and between the GIDs of your local database, your PMDB, and your PMDB subscribers.

If you create a new user with a UID that is already in use in the PMDB or in some other subscriber computer, the subscriber's individual update fails; but in all other subscriber computers where no such conflict exists, the update succeeds.

An alternative to synchronizing your passwd and group files is to explicitly specify the UID of each new user and the GID of each new group.

Synchronize Users and Groups

To ensure the lists of users and groups in your various databases correspond correctly at all times, you need an initial set of identical lists. Because the password and group files are so important, synchronize them before they begin accumulating local user and group information.

To synchronize users and groups

1. Copy your /etc/passwd file and /etc/group file to your Policy Model directory.
   
   This is a one-time procedure that destroys any previous passwd and group files in your Policy Model directory (see page 118).

   Note: If you are using a shadow file and want to synchronize passwords, we recommend using the secrepsw utility. For more information, see the Utilities Guide.

2. Copy the /etc/passwd file and /etc/group file to each subscriber computer so that they are identical to the ones on your own computer.

3. On the computer where the PMDB resides, ensure that the synch_uid token in your pmd.ini file is set to yes.

   By default, the value of the token synch_uid is yes. If you ever want a subscriber database to have independent default UIDs and default GIDs (that is, not necessarily attempting to match those of the PMDB), you can set synch_uid to no.
Specify UIDs Explicitly

Another way to send an identical UID or GID to the PMDB and to all its subscribers is to explicitly set it when you create a new user.

To specify UIDs explicitly use the userid or groupid parameter with each newusr command.

**Example: Create a new user with a specified UID**

If you want to establish 1234 explicitly as the UID of new user terry_jones (and assuming that no one else in the database has that UID yet), enter the command:

```bash
eTrustAC> newusr terry_jones unix (userid(1234))
```

If the specified UID is already being used in the PMDB, then the PMDB will not itself be updated, but the command will still propagate to the other subscriber databases. Among the other databases, wherever the particular UID is already in use, the subscriber's individual update will fail; but where no such conflict exists, the update succeeds.

Update Subscribers

When updating subscribers, the Policy Model performs the following actions:

1. The Policy Model tries to fully qualify subscriber names as they are added or deleted from the Policy Model.
2. The PMDB daemon, sepmdd, attempts to update a subscriber database for the amount of time defined by the token `_QD_timeout_`.
3. If the maximum time elapses and the daemon does not succeed in updating a subscriber, it skips that particular subscriber and tries to update the remainder of the subscribers on its list.
4. After it completes its first scan of the subscriber list, sepmdd then performs a second scan, in which it tries to update the subscribers that it did not succeed in updating during its first scan. During the second scan, it tries to update a subscriber until the connect system call times out (approximately 90 seconds).

**Note:** The token `_QD_timeout_` may be found in both the seos.ini and pmd.ini files. If the token exists in both files, sepmdd uses the value in the pmd.ini file.

**Note:** Whenever a PMDB encounters an error while propagating updates to subscribers, the sepmdd daemon creates an entry in the Policy Model error log file (see page 136). This file, ERROR_LOG, is located in the PMDB directory (see page 118).
Update a Policy Model Database

Working at the computer where the PMDB resides does not automatically update the PMDB itself. To update a PMDB, you need to specify it as your target database.

To specify a target database, use the hosts command in the selang command shell:

```
eTrustAC> hosts <pmd_name>@<pmd_host>
```

All selang commands now update the policy model database specified. The commands then automatically propagate to the active databases on this computer and of all subscriber computers.

Example: Specify a target PMDB

To set the target database to policy1 on myPMD_host, use the following command:

```
eTrustAC> hosts policy1@myPMD_host
```

If you now enter the newusr command, the new user is added to the policy1 database as well as the active databases on this computer and of all subscriber computers.

Clean Up the Update File

The sepmd utility automatically writes each update it receives in the updates.dat file. To prevent the file from growing too large, we recommend that you delete processed updates from the file periodically.

To clean up the update file, use the following command:

```
<eTrustAC_InstallDir>/bin sepmd -t pmdbName auto
```

sepmd calculates the offset of the first update entry that has not been propagated and deletes all the update entries before it.

Encrypt the Update File

After you create a PMDB, but before you start sepmdd, you can specify that information saved to the updates.dat file be encrypted.

To encrypt the update file, set the UseEncryption token to yes in the [pmd] section of the pmd.ini file.

To decrypt the updates.dat file, use the sepmd utility with the -de switch.

Note: For more information about sepmd, see the Utilities Guide.
**Exclude Subscribers**

You can skip subscribers so that they do not receive updates from parent PMDBs.

To exclude the local host, set the token exclude_localhost to yes in the pmd.ini file.

To add additional subscribers to the excluded list, set the token exclude_file (name-of-file).

To make a subscriber receive updates, remove the subscriber from the excluded list.

**Propagate Passwords**

When a user changes a password using the sepass utility, the new password is normally sent to the computer's parent PMDB. The parent PMDB is defined in the parent_pmd or the passwd_pmd token in the [seos] section of the seos.ini file or in both. However, if the user changes the password with the utility sepass, you can also specify that the user's new password should be sent to and propagated by a separate PMDB.

To send a new user's password to a separate PMDB, use the pmdb parameter with the newusr, chusr, or editusr command.

**Example: Specifying a separate PMDB for password propagation**

To specify that the new passwords created with sepass for the user Tony should be sent to and propagated by a separate PMDB pw_pmdb@name1.yourorg.com, enter the following command:

```
eTrustAC> editusr tony pmdb(pw_pmdb@name1.yourorg.com)
```
Remove a Subscriber

If you no longer want to propagate updates to a particular subscriber, you should remove it. Alternatively, you can exclude a subscriber from receiving updates (see page 132).

To remove a subscriber

1. Remove the computer from the subscription list:
   ```bash
   sepmd -u <PMDB_name> <computer_name>
   ```
   The computer is removed from the Policy Model subscription list.

2. Shut down seosd on the computer that you removed from the subscription list:
   ```bash
   secons -s
   ```
   The daemon seosd is shut down.

3. Delete the value of the parent_pmd token in the [seos] section of the seos.ini file on the computer you removed from the subscription list.
   The computer will stop accepting updates from the parent PMDB.

4. Restart seosd.
   The active database on the computer that you removed from the subscription list, is no longer a subscriber of the specified PMDB.

Note: Once the database is unsubscribed from the PMDB, the PMDB no longer sends commands.

Filter Updates

If you want your PMDB to update different subsets of data at different subscriber databases, you need to define which records are sent to subscriber databases.

To filter updates

1. Configure PMDBs to serve as parents to subsets of subscribers (see page 128).

2. Modify the filter token in the pmd.ini file of the parent PMDB, to point to a filter file you set up on the same computer.
   Updates to the subscriber databases are then limited to the records that pass the filter.
Policy Model Filter File

A filter file consists of lines, each with six fields. The fields contain information on:

- The form of access permitted or denied.
  For example, READ or MODIFY
- The environment affected:
  eTrust or UNIX
- The class of the record.
  For example, USER or TERMINAL
- The objects, within the class, that the rule covers.
  For example, User1, AuditGroup, or TTY1
- The properties that the record grants or cancels.
  For example, OWNER and FULL_NAME in the filter line means that any command having those properties is filtered. You must enter each property exactly as it appears in the Reference Guide.
- Whether such records should be forwarded to the subscriber database or not:
  PASS or NOPASS

The following rules apply to each line in the filter file:

- You can use an asterisk (*) to denote all possible values in any field.
- If more than one line covers the same records, the first applicable line is used.
- Spaces separate the fields.
- In fields with more than one value, semicolons separate the values.
- Lines beginning with # are considered a comment line.
- Empty lines are not allowed.

Example: Filter file

The following example describes a line from a filter file:

<table>
<thead>
<tr>
<th>CREATE</th>
<th>eTrust</th>
<th>USER</th>
<th>*</th>
<th>FULL_NAME;OBJ_TYPE</th>
<th>NOPASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>

form of access  environment  class  record name  properties  treatment
In this example, if we name the file with this line TTY1_FILTER and edit the pmd.ini file for PMDB TTY1 so that filter=/opt/CA/eTrustAccessControl/TTY1_FILTER, then PMDB TTY1 will not propagate to its subscribers any records that create new users with the FULL_NAME and OBJ_TYPE property.

Policy Model Error Log File

The Policy Model error log, which is organized chronologically, looks similar to this:

<table>
<thead>
<tr>
<th>Error Text</th>
<th>Error Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 Nov 03 11:56:07 (pmdb1): fargo nu u5 0 Retry ERROR: Login procedure failed (10068) ERROR: Cannot accept update from a non-parent PMDB (<a href="mailto:pmdb1@name.company.com">pmdb1@name.company.com</a>) (10104)</td>
<td>Configuration Errors</td>
</tr>
<tr>
<td>20 Nov 03 19:53:17 (pmdb1): fargo nu u5 0 Retry ERROR: Connection failed (10071) Host is unreachable (12296)</td>
<td>Connection Errors</td>
</tr>
<tr>
<td>20 Nov 03 11:57:06 (pmdb1): fargo nu u5 560 Cont ERROR: Failed to create USER u5 (10028) Already exists (-9)</td>
<td>Database Update Errors</td>
</tr>
<tr>
<td>20 Nov 03 11:57:06 (pmdb1): fargo nu u5 1120 Cont ERROR: Failed to create USER u5 (10028) Already exists (-9)</td>
<td></td>
</tr>
</tbody>
</table>

The Policy Model error log is in binary format; you can view it only by entering the following command:

```bash
<eTrustAC_InstallDir>/bin sepmd -e pmdname
```

**Note:** Do not manually delete an error log (for example, with the UNIX rm command). To delete the log, only use the following command:

```bash
<eTrustAC_InstallDir>/bin sepmd -c pmdname
```

**Important!** The error log in eTrust AC r5.1 and later versions has a format that is not compatible with the format of earlier versions. sepmd cannot handle error logs from these earlier versions. When you upgrade to a version that has this format, the old error log is copied to ERROR_LOG.bak; a new log file is created when you start sepmdd.
Example: PMDB update error message

The following example shows a typical error message:

```
```

- The top line always consists of the date, time, and subscriber. The command that generated the error appears next, followed by the offset (in decimal format), which indicates the location of the failed update inside the updates file. Lastly, the flag indicates whether the PMDB retries the update automatically or continues without it.

- The second line shows an example of a major level message (what type of error occurred) and its return code.

- The third line displays an example of a minor level message (why the error occurred), and its return code.

Example: Error message

A command may generate and display more than one error. Also, an error may consist of a major level message, a minor level message, or both.

The following error has only one message level:

```
```

This message occurs when sepm pull attempts to release a subscriber that is already available.
Dual Control

Dual Control is a way of operation that divides the process of updating the PMDB into two stages:

- Creating a transaction which consists of one or more commands.

  The maker - any user with the ADMIN attribute - enters one or more commands that update the PMDB. The transaction is given a unique ID number and placed in a file, where it waits to be processed before execution.

- Authorizing the transaction for execution.

  The checker - not the same user, but any other user with the ADMIN attribute - locks the commands in the transaction, checks the commands, and authorizes or rejects them. If the transaction is authorized, then the commands are executed in the PMDB. If the transaction is rejected, then the transaction is deleted and the PMDB is not updated. The checker cannot authorize some of the commands in a transaction and reject others; the transaction must be processed as a whole.

  **Note:** Only the find and show commands do not need the authorization of a checker.

Using the parameters in the sepmd utility, makers can list, retrieve and edit, or delete unprocessed transactions; checkers can lock transactions in order to authorize or reject them, and they can unlock transactions for processing at a later time or by a different checker.

When the sepmdd daemon receives the start_transaction command, it sends the child process a unique number. The child process tags any further commands with this identifying number, and the number is added to the new transaction and kept in the memory of the sepmdd daemon. When sepmdd receives the end_transaction command, the authorization algorithm is invoked. The authorization algorithm checks that none of the commands in the transaction pertain to the maker of the transaction, and none of the objects in the commands are already locked by another transaction that is waiting to be processed prior to execution.

You cannot use the same objects in different transactions before they are processed. If the check passes, then the relevant objects are locked, the transaction is assigned a unique sequential number, and the data is saved in a file. Each transaction is saved in a different file.

  **Note:** For more information about the sepmd utility or the sepmdd daemon, see the the *Utilities Guide*. 
Activate Dual Control

Dual Control lets you divide the duty of updating PMDBs between two people: a maker and a checker.

To activate Dual Control, set the `is_maker_checker` token, in the `pmd.ini` file and in the `[pmd]` section of the `seo.ini` file, to yes:

```
is_maker_checker=yes
```

**Note:** Create the Policy Model maker **before** setting these token values.

Create or Edit Transactions

When Dual Control is activated, the maker needs to create transactions before these are processed by a checker.

**To create a transaction**

1. Make sure the following is true:
   - You (as a maker) have the ADMIN authority.
   - None of the commands pertain to you. (You cannot enter commands that change yourself.)
   - None of the objects in the commands are already part of another transaction that has not been processed by a checker yet.
   - All the objects in the commands exist.
   - You are not editing an existing transaction that another maker invoked. (You can only edit your own transactions.)

2. Connect to the maker PMDB:

   ```
   hosts maker@
   ```

   The hosts command connects you to the PMDB (maker). When Dual Control is activated, the name of the PMDB is always “maker.” After you enter the hosts command, a message reports whether the connection to the host is successful or not.

3. Start the transaction:

   ```
   start_transaction transactionName
   ```

   Use `start_transaction` command as the first step when entering or updating a transaction. You can describe the transaction or give it any name you want, up to 256 alphanumeric characters.
4. Enter your transaction.
   This is a list of commands. For example:
   
   ```
   newusr mary owner(bob) audit(failure,loginfailure) 
   chres TERMINAL tty30 defaccess(read) \ 
   restrictions(days(weekdays)time(0800:1800))
   ```

5. End the transaction:
   ```end_transaction```

   The transaction is complete; you are presented with the unique ID number assigned to your transaction. The commands are placed in a file, where you can still access and change them until a checker, in preparation for processing, locks them.

   **Note:** Make sure you record the transaction ID number if you want to be able to edit the transaction later.

**To edit a transaction**

- When you enter the `end_transaction` command, an ID number displays. This is a unique number that identifies the transaction. If you want to overwrite your transaction later, then the process is the same as creating a new transaction, except that you add to the file the transaction's ID number after the name. You can enter to the file any changes you want to make. For example:

  ```
  hosts maker@
  start_transaction transactionName transactionId
  
  chusr mary category (FINANCIAL)
  ```

  You can then enter the appropriate commands to update the transaction:

  ```
  end_transaction
  ```

- View specific unprocessed transactions with the following parameters.

  Make sure you are in the `eTrustACDir/bin` path (where `eTrustACDir` is the installation directory for eTrust AC, by default `/opt/CA/eTrustAccessControl`).

<table>
<thead>
<tr>
<th>Command with Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sepmid -m l</code></td>
<td>Lists the unprocessed transactions of the user who invoked the parameter.</td>
</tr>
<tr>
<td><code>sepmid -m la</code></td>
<td>Lists all the transactions of all the makers that are waiting to be processed.</td>
</tr>
</tbody>
</table>
## Command with Parameter

<table>
<thead>
<tr>
<th>Command with Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sepmd -m lo</td>
<td>Lists the transactions of all the makers except those of the user who invoked the parameter. Each transaction in the list includes the name of the maker, the ID number of the transaction, and a description of the transaction, if the maker entered one.</td>
</tr>
</tbody>
</table>

- Retrieve a specific transaction to the standard output with the following command:
  ```
  sepmd -m r transactionId
  ```
- Delete a specific transaction with this command:
  ```
  sepmd -m d transactionId
  ```
Checking and Processing Transactions

When Dual Control is activated, the checker needs process transactions created by a maker.

To check a transaction

1. Make sure the following is true:
   - You (as the checker) have ADMIN authority.
   - Another Checker does not lock the transaction.
   - None of the commands pertain to you. (You cannot process commands that involve yourself.)

2. Navigate to the eTrustACDir/bin path
   where eTrustACDir is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl.

3. View the transactions that are waiting to be processed before execution:
   `sepmd -m la`
   Or, view all the transactions except the transactions that you yourself created:
   `sepmd -m lo`
   Each transaction includes the name of the maker, the ID number of the transaction, and the name or description of the transaction.

4. Lock the transactions before processing them:
   `sepmd -m r transactionId`
   
   *Note:* A locked transaction cannot be changed.

5. Process the transaction:
   `sepmd -m p transactionId code`
   where `code` is one of:
   - **0**-The transaction is rejected. In this case, all the commands in the transaction are deleted and no changes are implemented in the PMDB.
   - **1**-The transaction is authorized. The commands in the transaction are immediately implemented in the PMDB.
   - **2**-The transaction is unlocked. The transaction returns to the queue of waiting transactions and can be processed later, perhaps by a different checker.
   
   A message appears stating which commands were successful and which failed.

   *Note:* For more information on makers and checkers, see the sepmd utility in the Utilities Guide and the start_transaction command in the Reference Guide.
Using the seagent and sepmdd Daemons

The seagent daemon is responsible for accepting requests from remote computers and applying them to PMDBs; the seagent daemon also sends requests to seosd. The sepmdd daemon is the PMDB daemon. This section describes how these daemons work together in the PMDB environment.

The seagent Daemon

The seagent daemon waits for connections on the seoslang and seoslang2 TCP services (whose default values are 8890 and 8891, respectively). When a connection request arrives, seagent forks a child process to handle the communication on the connection and then continues waiting for new connections.

When a user enters the hosts command in selang, seagent forks a child process on the machine that the user is connected to. The child process then receives commands from the command language interface and passes them on to the sepmdd daemon.

The sepmdd Daemon

The sepmdd daemon performs the following functions:

- Administers the PMDB
- Administers the subscriber databases
- Propagates changes from the PMDB to the subscriber databases

The sepmdd daemon is automatically started by seagent when seagent has to access the PMDB. Normally you do not need to run sepmdd explicitly.

Note: sepmdd runs under the logical user _seagent (not under root) in the eTrust environment. To permit or restrict access to resources by sepmdd (for example, to restrict access to the PMDB directory), create the relevant rules for _seagent.
Using a Shadow File

Usually, sepmdd does not use a shadow file when updating a native environment. You can, however, set up a shadow file. To do this, set the UseShadow token in the [pmd] section of the pmd.ini file to yes.

If the UseShadow token is set to yes, sepmdd uses a default shadow file in the same directory as the PMDB. If you want to change the location of the shadow file, specify the new location with the YpServerSecure token in the [pmd] section of the pmd.ini.

If you change the location of the shadow file (with the YpServerSecure), to the local host’s shadow file (for example, /etc/shadow), sepmdd sets a token, UseSystemFiles, to yes.

**Important!** Do not change the UseSystemFiles token yourself. The sepmdd or seagent daemons change it automatically.

**Note:** For more information about the seagent or sepmdd daemons, see the seagent and sepmdd utilities in the *Utilities Guide*.

Advanced Policy Management and Reporting

Multiple-rule policies (script files) you create can be stored and then deployed in a configured hierarchy. Using this policy-based method, you can store policy versions and then deploy and undeploy those. You can also create reports on deployment status, deployment deviation, and deployment hierarchy.

**Note:** Dual control (see page 138) is not available with this method and is only available on UNIX.

Environment Architecture

To use advanced policy management and reporting, you need to install and configure the following additional components:

- A Deployment Map Server (DMS) (see page 146) on a central computer that is designated for this purpose.
- A Deployment Map Agent (DMA) (see page 146) on each computer which has at least one PMDB.

**Note:** Advanced policy management and reporting requires setting your environment for automatic rule-based policy updates. After installing the DMS and DMA on appropriate computers, configure your parent and subscriber databases (see page 123).
Example: A two-tiered hierarchy with a central DMS

Note: The following UNIX example also applies to a Windows architecture with small modifications.

In this example, the site consists of IBM AIX and Sun Solaris systems. Since the list of trusted programs on IBM AIX differs from the one on Sun Solaris, the PMDBs need to consider architecture dependency. For management and reporting purposes, we will set up the DMS and DMA to support the environment we created when we configured a multiple-PMDB environment (see page 120). The DMS stores all policy deployment and deviation information and eTrust AC lets you create reports from this information.
Deployment Map Server (DMS)

The DMS sits at the core of advanced policy management and reporting. The purpose of DMS is to keep an up-to-date map of the eTrust AC deployment hierarchy and the status of policies deployed on each computer. Having this data in a central location (rather than connecting to each database in a hierarchical manner) reduces the time required for generating reports. In addition, the DMS stores versions of your policies that you can later deploy and undeploy as required.

A DMS is a PMD node and it uses a PMDB as its data repository. It collects the data it receives from notifications from each PMD node it is configured for.

Deployment Map Agent (DMA)

The DMA is an agent that is responsible for PMD communication with the DMS. You should install a DMA on each PMD node (at least one PMDB). The DMA on the parent PMDB is responsible for notifying the DMS of policy deployment status and hierarchy changes. End-points send only policy deviation status directly to the DMS.

**Note:** Do not install a DMA on end-point computers.

The DMA uses standard eTrust AC communication and encryption mechanisms for communication with the DMS.

**Note:** You do not need to define the DMA as the parent of the DMS to enable DMA\DMS communication.
Advanced Policy Management Classes

In order for the DMS to keep an up-to-date map of the eTrust AC deployment hierarchy and the status of policies deployed on each computer, it uses specific eTrust AC classes.

HNODE

Each HNODE object represents a node in a hierarchy. It holds information about the particular node it represents, its subscribers, and the parent PMDBs. In addition, each HNODE object holds information about the policies that should be deployed on the node it represents, and the status of each policy (deployed, deployed with errors, and so on).

The name of the HNODE object depends on the type of node it represents:

- The actual host name for end-points.
  For example, myhost.mydomain.com
- The Policy Model name for a PMDB.
  For example, mypmd@hostB.domain.com

POLICY

Each POLICY object represents a version of a policy that may be deployed on any part of the HNODE hierarchy. It contains information about where the associated policy scripts are stored (in which RULESET object) and which nodes it should be deployed on.

The name of the object is the name of the policy, suffixed by a version number (policy_name#xx).

RULESET

Each RULESET object holds both the deployment and undeployment (removal) scripts that are associated with a policy version.

The name of the object is based on the respective POLICY object name.

Note: For more information about these classes, see the Reference Guide.
How You Set Up a Hierarchy for Advanced Policy-based Management and Reporting

eTrust AC uses a DMS to keep an up-to-date map of the eTrust AC deployment hierarchy and the status of policies deployed on each computer. By installing and configuring the appropriate components on each computer in your hierarchy, you enable policy-based management and reporting.

To enable policy-based management and reporting, do the following:

1. Install a DMS on a central computer.
   A DMS can be installed during eTrust AC installation or by the dmsmgr utility.

2. Install a DMA on each PMD node.
   A DMA can be installed during eTrust AC installation or by the dmsmgr utility.

3. Install advanced policy management and reporting functionality on each eTrust AC computer.
   This configures the deviation calculation for sending policy deviation status to the DMS.

4. Set up a hierarchy (see page 123).
   As you set up your hierarchy, HNODE objects representing each node in your hierarchy are added in the DMS.

   **Important!** When you uninstall eTrust AC from a computer or delete a PMDB, the HNODE object remains. You need to remove obsolete object nodes (see page 150). If you unsubscribe databases from the hierarchy, the HNODE object remains but its link to the parent node is removed. You do not need to remove this object but it will maintain links to policy objects that were previously deployed on that node.

**Note:** For information about installing a DMS, a DMA, and configuring advanced policy management and reporting functionality, see the *Implementation Guide*. 
DMS Notifications

When you configure your environment for advanced policy management and reporting, components in your hierarchy notify the DMS of status changes in three areas:

- **Hierarchy change.**
  A notification is sent when a subscriber (PMD node or end-point) is added or deleted.

- **Policy deployment and undeployment.**
  A notification is sent when a policy is being deployed or undeployed on a subscriber (PMD node or end-point). The policy details and the deployment status are then updated according to the result of the operation (succeeded, failed, and so on).

- **Deviation status.**
  A notification is sent when an eTrust AC end-point calculates policy deviation and sends the result (deviation found or not found).

**Note:** Hierarchy change and policy deployment and undeployment notifications are sent by the DMA from PMD nodes only. Deviation status notifications are sent by the deviation calculator from eTrust AC end-points only.

How Hierarchy and Policy Status Notifications Work

The DMA sends hierarchy changes and policy status notifications to the DMS. DMA notifications are handled in the following way:

1. The DMA stores notification messages in an update file.
   These are hierarchy change and policy deployment and undeployment notifications.

2. The DMA contacts the DMS:
   - If a DMS is unavailable, the DMA tries to communicate with the DMS periodically, until all messages are successfully sent.
   - If the DMS is available, the DMA sends the stored notifications.
   **Note:** Each DMA communicates directly with a DMS, bypassing the hierarchy and reducing dependency.

3. The DMS stores the information it receives from each DMA for later use.
   Each time you create a report, eTrust AC retrieves the information on the DMS.
How Deviation Notifications Work

The deviation calculator is installed with eTrust AC and runs locally on the end point it calculates deviations for. The deviation calculator performs the following actions and sends these notifications to the DMS:

1. The calculation process is triggered by sending a selang command (start devcalc) to the end point.
   We recommend you do this as a scheduled execution by a customized script.

2. Once the calculation is complete, the deviation calculator stores the result in a data file.
   The file is `<eTrustAC_Dir>\data\devcalc\deviation.dat`

   **Note:** The reporting utility can retrieve the deviation details (using the `-dev` option). Alternatively, you can retrieve the contents of the data file using the `get devcalc` command on the end-point.

3. The deviation calculator then sends the deviation status (was a deviation found or not) to the DMS.
   The deviation itself (the contents of the data file) is not sent with the status notification.

Remove Obsolete Nodes from a Hierarchy

The DMS stores information about your hierarchy. If you remove a computer from the hierarchy when you uninstall eTrust AC from that computer, the DMS still contains a reference to that node. As a routine maintenance procedure, you should clean the DMS from these obsolete nodes.

To remove obsolete nodes from a hierarchy run the dmsmgr utility on the DMS computer to perform a routine clean up:

```
dmsmgr -dms -cleanup <number_of_days>
```

where `<number_of_days>` is the minimum number of days in which the eTrust AC node has been unavailable for.

**Note:** You can also manually delete a specific node by issuing the following selang command on the DMS computer:

```
rr HNODE <HNODE_name>
```
How Advanced Policy-based Management Works

Advanced policy-based management lets you store, deploy, and undeploy policy versions, and later create reports on deployment status, deployment deviation, and deployment hierarchy. Each policy is a pair of selang script files you create. The first script file is called a deployment script and contains a set of selang commands that construct the policy. The second script file is an undeployment script and contains commands that are required for undeploying (removing) the policy from the end-point database.

Each policy is applied in two stages to target databases you specify:

1. You store policy details in the DMS.
   Policy details include the deployment and undeployment scripts, and the policy signature which is created automatically (used to detect variations of the same policy).
   If policy details cannot be stored in the DMS, make sure you:
   - Store the policy from a computer that has TERMINAL rights to the DMS.
   - Have sub-administration permissions to the POLICY and RULESET classes on the DMS.
   - Do not have a deployment or undeployment script that contains a syntax error.

2. The utility stores the policy using automatic version-control.
   Depending on whether the policy already exists on the DMS, the utility does one of the following:
   - If the policy name does not exist on the DMS, it creates the first version of the policy (policy_name#01).
   - If the policy name already exists on the DMS, it creates a new policy version, incrementing the highest found policy version by one.

3. You deploy a stored policy version to target databases.
   If the stored policy cannot be deployed in your target hierarchy, make sure you:
   - Deploy from a computer that has TERMINAL rights to your target Policy Model root.
   - Have sub-administration permissions to the POLICY, RULESET, and HNODE classes on the DMS and each database in the hierarchy where you are deploying the policy.
   - Have sub-administration permission on the target Policy Model root computer.
   - Do not have a version of the same policy already deployed on a host which is part of the deployment hierarchy.
4. Each rule—the selang commands specified in the deployment script—is executed on the target databases.

   If a rule cannot be deployed in a certain database, the policy is considered as deployed with failures (status Failed).

   This happens if you try to deploy a policy on a host and the deployment script contains:
   - A reference to an object that does not exists. For example:
     
     ```
     cr FILE /does_not_exist comment(123)
     ```
     
     For this reason, the policy deployment script must be self-sufficient. That is, the deployment script must be constructed so that it creates all the resources it uses.
   - A command that causes an error.
   - A command you do not have sub-administration permissions to execute.

5. The policy status is recorded.

   Policy status can be Deployed, Undeployed, Transferred, Failed (deployed with failures), Queued, TransferFailed, SigFailed (signature failed), UndeployFailed (undeployed with failures), or Unknown.

   **Note:** If a policy deployed with errors, you need to view the log file on the computer where the policy was deployed with errors.

Once the policy details are stored on the DMS and then deployed on the target databases, if a target database is a PMDB, the policy is propagated throughout the hierarchy using the automatic rule-based policy updates mechanism.

When a new subscriber is added to the hierarchy, all policies are propagated through the hierarchy and the DMS is notified that a node was added to the hierarchy.
Administration Requirements

You can run the policy deployment utility (policydeploy) from any computer, as long as eTrust AC is installed. In order to store policies on the DMS, or deploy and undeploy policies on databases in your hierarchy, you and the computer you are working from need to have appropriate permissions.

To store policies on the DMS:
- The computer you are running the policydeploy utility from, needs to have terminal rights (TERMINAL class) for the DMS.
- You need to have sub-administration rights for the POLICY and RULESET classes on the DMS.

To deploy and undeploy policies throughout your hierarchy:
- The computer you are running the policydeploy utility from, needs to have terminal rights (TERMINAL class) for the computer which is the target Policy Model root.
- You need to have:
  - Read rights for the POLICY and RULESET classes, and sub-administration rights for the HNODE class on the DMS.
  - Sub-administration rights for the POLICY, HNODE, and RULESET classes on each database in the hierarchy where you are deploying the policy.
  - Appropriate sub-administration rights on each of the databases in the hierarchy where you are deploying the policy.

These are the permissions necessary to deploy the selang commands that form the policy on each of these computers.

For example, you'll need sub-administration rights for the FILE class if you are creating a new file resource:

nr FILE /inetpub/* defaccess(none)
How to Deploy Approved Policy Versions

Using advanced policy-based management you can store a draft version of a policy, have it reviewed and modified as required, then deploy the approved version.

To deploy approved policy versions, do the following:

1. Store a policy version on the DMS (see page 155).
   Once you have a stored policy version, the policy can be reviewed and deployed.

2. Review the policy (see page 157).
   Anyone with read permissions to the POLICY and RULESET objects can view the policy and its associated rules.

3. If required, store a new version of the policy with the approved changes.
   Whenever you need to update a policy, you must store a new version of the policy that contains the required modified policy deployment and undeployment rules.

4. Deploy the approved policy version to your hierarchy (see page 158).
Store a Policy Version

Every policy you store on the DMS automatically gets a version number. The first time you store a policy it receives a version number "01". For example, the first time you store policy myPolicy, the policydeploy utility creates a POLICY object named myPolicy#01. Every time you store a policy that already exists on the DMS, the latest stored version of the policy will be incremented by one to create the new policy version. For example, when you store a version of myPolicy for the twenty-eighth time, the policydeploy utility creates a POLICY object named myPolicy#28.

You can then view stored policies and deploy them to your hierarchy as required.

To store a policy version

1. Create a new script file with selang deployment commands.
   
   These are the commands necessary to construct the policy you want to deploy on each computer in a hierarchy.
   
   **Important!** Policy deployment does not support commands that set user passwords. Do not include such commands in your deployment script file. UNIX (native) selang commands are supported but will not show in deviation reports.

2. Create a new script file with selang undeployment commands.
   
   These are the commands necessary for undeploying (removing) the policy from a computer in the hierarchy.
   
   **Note:** These commands are used by default when you undeploy a policy from a targeted hierarchy unless you supply a new policy undeployment script at the time you undeploy a policy.

3. Run the policydeploy utility with the -store option:
   
   ```
   policydeploy -store name -ds file1 -uds file2 [-dms list]
   ```
   
   where name is the name of the policy you want to store, file1 is the full path and name of your deployment script file, file2 is the full path and name of your undeployment script file, and list is an optional comma-separated list of DMS nodes.

   The policydeploy utility prompts you for whether to create a new version of the policy on the DMS.
   
   **Note:** Policy names cannot include the # (hash) character which is reserved for denoting policy version numbers and is added automatically.

4. Enter y to confirm this action.
   
   The policydeploy utility creates a new version of the policy on the DMS.

**Example: Storing an IIS 5 protection policy**
The following example shows you how to store a policy for securing Internet Information Services (IIS) 5 web servers. This is the first time we store this policy on the DMS.

We will deploy the policy IIS5 on a Policy Model hierarchy for which iis5@host.company.com is the root PMDB.

1. Save a file called IIS5.selang with the following IIS script:

   ```
   nu inet_pers owner(nobody)
   nr FILE /inetpub/* defaccess(none) owner(nobody)
   authorize FILE /inetpub/* uid/inet_pers access(all)
   nr FILE /inetpub/scripts defaccess(none) owner(nobody)
   nr FILE *.asp defaccess(none) owner(nobody)
   authorize FILE *.asp uid/inet_pers via(pgm/inetinfo.exe)) access(read, execute)
   ```

   These are the commands necessary to deploy an IIS 5 protection policy.

2. Save a file called IIS5_rm.selang with the following script:

   ```
   ru inet_pers
   rr FILE /inetpub/*
   rr FILE /inetpub/scripts
   rr FILE *.asp
   ```

   These are the commands necessary to undeploy the IIS 5 protection policy we created in step 1.

3. Run the policydeploy utility:

   ```
   policydeploy -store IIS5 -ds IIS5.selang -uds IIS5_rm.selang
   ```

   This stores the first version of the IIS5 policy (IIS5#01) as defined in IIS5.selang and IIS5_rm.selang on the DMS.
View the Rules Associated with a Policy

Once a policy is stored on the DMS, anyone with read permissions to the POLICY and RULESET objects can view the policy and its associated rules. If you do not know which is the latest version of the stored policy, you can find this out first.

To view the rules associated with a policy

1. Connect to the DMS via selang:
   
   hosts dms_name@hostname
   
   You can now query our DMS via selang.

2. If you want to know which is the latest version of the policy, issue the following selang command to find all versions of the policy:
   
   find POLICY policy_name#*
   
   The selang window lists all versions of the policy_name policy.

3. Issue the following selang command to view the policy deployment and undeployment scripts:
   
   sr RULESET policy_name#xx
   
   where xx is the number of the policy you want to view the rules for.

   The selang window displays the policy_name#xx RULESET object, including the deployment and undeployment rules that relate to the xx version of the policy_name policy.
Deploy a Stored Policy Version

You can deploy a stored version of a multiple-rule policy in your hierarchy in a way that lets you undeploy the policy later on, and create reports on deployment status, deployment deviation, and deployment hierarchy.

To deploy a stored policy version

Do one of the following:

- If you want to deploy the latest version of the stored policy, run the policydeploy utility specifying the policy name and target hierarchy:
  
  policydeploy -deploy name -root db1,[db2] [-dms list]

  The utility finds the latest version of the policy on the DMS with the name you supplied, and tries to deploy it on the target databases. Policy commands are then propagated to subscribing databases if any exist.

- If you want to deploy a specific version of the stored policy, run the policydeploy utility specifying the policy name, its version, and a target hierarchy:
  
  policydeploy -deploy name#xx -root db1,[db2] [-dms list]

  The utility tries to deploy the specified version of the policy on the target databases. Policy commands are then propagated to subscribing databases (if any exist).

Note: For more information about the policydeploy utility, see the Utilities Guide for UNIX or the Reference Guide for Windows.

Important! You cannot deploy a policy if a version of the same policy is already deployed on any of the hosts within the deployment hierarchy.

Example: Deploying an IIS 5 protection policy

The following example shows you how to deploy a policy for securing Internet Information Services (IIS) 5 web servers. We will review the fourth version of policy IIS5 and then deploy it on a Policy Model hierarchy for which iis5@host1.company.com is the root PMDB. Policy IIS5 is stored on the crDMS@cr_host.company.com DMS node.

1. Connect to the DMS via selang:
   
   hosts crDMS@cr_host.company.com

   You can now query our DMS via selang.

2. If you’re not sure what is the latest available version of the policy, issue the following selang command to find all versions of the policy:
   
   find POLICY IIS5#*

   The selang window lists all versions of the IIS5 policy.
3. Issue the following selang command to view the policy deployment and undeployment scripts:
   
   ```
   sr RULESET IIS5#04
   ```

   The selang window displays the IIS5#04 RULESET object, including the deployment and undeployment rules that relate to the fourth version of the IIS5 policy.

4. In a command line window, run the policydeploy utility:
   
   ```
   policydeploy -deploy IIS5#04 -root iis5@host1.company.com
   ```

   This deploys the fourth version of the IIS5 policy on the PMD hierarchy under iis5@host.company.com
Undeploy a Policy

You can undeploy multiple-rule policies from a targeted hierarchy if you no longer want to have the policy deployed on those computers. You also need to undeploy a policy if you want to modify it (create an updated version of the policy).

To undeploy policy

1. (Optional) Create a new script file with selang undeployment commands. These are the commands necessary to undeploy (remove) the policy from a computer in the hierarchy.

   If you do not create and specify a new undeployment script, the undeploy command uses the script assigned to this policy when it was deployed.

   **Important!** Even if you specify a policy undeployment script, the DMS still records the original rules that were provided when you stored the policy and not the new script which is used to undeploy the policy.

2. Do one of the following:

   - If you want to undeploy the latest version of the policy, run the policydeploy utility specifying the policy name and target hierarchy:

     ```
policydeploy -undeploy name -root db1[ ,db2] [-dms list] [-uds file2]
     ```

     The utility finds the latest version of the policy on the DMS with the name you supplied and tries to undeploy it from the target databases. Policy undeployment commands are then propagated to subscribing databases if any exist.

     **Important!** If the policy version on any end-point in your hierarchy contains other versions than the latest version found on the DMS, you will have to undeploy each of these specific versions explicitly.

   - If you want to undeploy a specific version of the policy, run the policydeploy utility specifying the policy name, its version, and a target hierarchy:

     ```
policydeploy -undeploy name#xx -root db1[ ,db2] [-dms list] [-uds file2]
     ```

     The utility tries to undeploy the specified version (xx) of the policy from the target databases. Policy undeployment commands are then propagated to subscribing databases (if any exist).

     **Note:** For more information on the policydeploy utility, see the *Utilities Guide* for UNIX or the *Reference Guide* for Windows.

**Note:** When you undeploy a policy, the DMS reports that the status of the policy is *Undeployed*. The POLICY and RULESET objects remain on all of the hosts the policy version was deployed on (including the DMS) so that it can be redeployed or queried at a later time.
Modify a Deployed Policy

To modify a deployed policy, you need to first undeploy the deployed policy version, store a new version of the policy with the modified deployment and undeployment scripts, and then redeploy the policy using the new version.

To modify a deployed policy
1. Store a new policy version (see page 155).
   A new version of the policy is stored on the DMS.
2. Undeploy the policy (see page 160).
   The policy is undeployed from the target hierarchy.
3. Deploy the new version of the policy (see page 158).
   The policy is redeployed to the target hierarchy with the modified policy.

How Advanced Policy Reporting Works

Advanced policy reporting lets you create reports on deployment status, deployment deviation, and deployment hierarchy for a configured hierarchy and for policies that were created using the advanced policy-based management method. The report generation utility (policyreport) generates point-in-time (static) HTML reports based on DMS contents.

The policyreport utility creates hierarchy and policy reports by performing the following actions:
1. The utility queries the DMS for the requested information.
   The information retrieved depends on the type of report generated.
2. The utility queries end points for policy deviation results, if a deviation calculation is requested.
   The deviation status exists on the DMS, but the actual deviation needs to be retrieved from each end-point.
3. The utility generates a set of XML documents.
   This is an XML report.
4. The utility formats the XML report into HTML.
   The report is now ready for viewing in a browser.

Note: The policyreport utility stores the report in a subdirectory you specify using the -name option under the directory you specify using the -targetpath option.
Report Types

The report generation utility lets you view the policies deployed across your hierarchy in two modes:

- **By host**
  
  Host reports present computer-centric information. Use this mode when you want to view your environment by host. In this mode you can see how your computers are configured, what is the status of each computer in your hierarchy, which computers have which policies and what their status is, and how the actual rules deployed deviate from the rules that should be deployed on each computer.
By policy

Policy reports present the policy-centric information. Use this mode when you want to view the status of one or more policies throughout your environment.

In addition to the type of report, you can also affect the output by:

- Choosing the part of the hierarchy for which you want the report generated.
- Generating the report for a single computer.
- Filtering by host name, status, or status update time, or by policy name or status (wildcards are supported).
- Including or excluding deviation calculation results.
- Selecting a tree-like format.
- Hiding report columns.
Create a Host Report

A host report lets you see how your computers are configured in the hierarchy, what is the status of each computer in your hierarchy, or which computers have which policies and what their status is.

To create a host report, run the policyreport utility in the h mode:

```
policyreport -name <name> -mode h -dms <dms_name> -root <pmd1>[,<pmd2>] -tree \ -targetpath <path>
```

**Note:** You can also fine-tune the report by using additional optional flags. For more information about the policyreport utility, see the *Utilities Guide*.

**Example: Create a report for computers whose host name matches a specified mask**

The following example shows how you can use the policyreport utility to perform the following task:

- Generate a host report in the following directory:
  
  `$HOME/eac_data/reports/production_March2006`

- Retrieve the information is from the DMS:
  
  `mainDMS` on the `mainhost.domain.com` computer.

- Include only computers that are hierarchically beneath the following PMDB:
  
  `rootPMD@root.domain.com`

- Include only computers whose host name begins with:
  
  `prod`

```
policyreport -name production_March2006 -mode h \ 
-dms mainDMS@mainhost.domain.com -root rootPMD@root.domain.com -hn pro* \ 
-targetpath $HOME/eac_data/reports
```

The policyreport utility stores the report in a subdirectory we specified using the (-name production_March2006) option under the directory we specified using the -targetpath option ($HOME/eac_data/reports). We can update this report at a later time by adding the -f option that creates the report even if one already exists in the output directory.

**Note:** If you also specify the -tree flag, the report will show a graphical representation of the hierarchy. This includes parents of all computers in the report, even if the parents’ host name does not match the specified mask.

**Example: Create a report for computers whose status last changed within a certain date range**
The following example shows how you can use the policyreport utility to perform the following task:

- Generate a host report in the following directory:
  
  `$HOME/eac_data/reports/Feb06-Mar06`

- Retrieve the information is from the DMS:
  
  `mainDMS` on the `mainhost.domain.com` computer.

- Include only computers that are hierarchically beneath the following PMDB:
  
  `rootPMD@root.domain.com`

- Include only computers whose host status was last updated in February 2006.

```
policyreport -name Feb06-Mar06 -mode h \
  -dms mainDMS@mainhost.domain.com -root rootPMD@root.domain.com \
  -sd 01-02-2006 -ed 28-02-2006 -targetpath $HOME/eac_data/reports
```

**Example: Create a report that recalculates policy deviation results and store it in the current working directory**

The following example shows how you can use the policyreport utility to perform the following task:

- Generate a host report in the following directory:
  
  `<working_directory>/hierarchy_20March06`

- Retrieve the information is from the DMS:
  
  `mainDMS` on the `mainhost.domain.com` computer.

- Include only computers that are hierarchically beneath the following PMDB:
  
  `rootPMD@root.domain.com`

- Include deviation calculation results.
- Graphically represent the hierarchy using indentation.

```
policyreport -name hierarchy_20March06 -mode h -dms mainDMS@mainhost.domain.com \
  -root rootPMD@root.domain.com -targetpath . -dev -tree
```
Create a Policy Report

A policy report lets you see which policies are deployed on which computers.

To create a policy report, run the policyreport utility in the p mode:

```
policyreport -name <name> -mode p -dms <dms_name> -root <pmd1>[,<pmd2>] \ 
-targetpath <path>
```

**Note:** You can also fine-tune the report by using additional optional flags. For more information about the policyreport utility, see the *Utilities Guide*.

**Example: Create a report for all versions of a specified policy**

The following example shows how you can use the policyreport utility to perform the following task:

- Generate a policy report in the following directory:
  ```
  $HOME/eac_data/reports/iis5Policies_March2006
  ```
- Retrieve the information from the following DMS:
  ```
  mainDMS on the mainhost.domain.com computer.
  ```
- Include only computers (subscribers) that are hierarchically beneath the following PMDB:
  ```
  rootPMD@root.domain.com
  ```
- Include only versions of the following policy:
  ```
  iis5
  ```

```
policyreport -name iis5Policies_March2006 -mode p \ 
-dms mainDMS@mainhost.domain.com -root rootPMD@root.domain.com -pn iis5#* \ 
-targetpath $HOME/eac_data/reports
```

The policyreport utility stores the report in a subdirectory we specified using the -name option (iis5Policies_March2006) under the directory we specified using the -targetpath option ($HOME/eac_data/reports). We can update this report at a later time by adding the -f option that creates the report even if one already exists in the output directory.

**Example: Create a report for policies that were deployed with errors**

The following example shows how you can use the policyreport utility to perform the following task:

- Generate a policy report in the following directory:
  ```
  $HOME/eac_data/reports/policyErrors
  ```
- Retrieve the information from the DMS:
  ```
  mainDMS on the mainhost.domain.com computer.
  ```

```
policyreport -name policyErrors -mode p \ 
-dms mainDMS@mainhost.domain.com -root rootPMD@root.domain.com -pn policyErrors \ 
-targetpath $HOME/eac_data/reports/policyErrors
```
- Include only computers (subscribers) that are hierarchically beneath the following PMDB:

  `rootPMD@root.domain.com`

- Include only policies that were deployed with errors (Failed)

  `policyreport -name policiesErrors -mode p \`
  `-dms mainDMS@mainhost.domain.com -root rootPMD@root.domain.com \`
  `-pstat "Failed" -targetpath $HOME/eac_data/reports`

**View a Policy or Host Report**

Once you generate the report, you need to navigate to the folder where the report is stored and open it for viewing in a browser.

**To view a policy or host report**

1. Navigate to the folder where the report is stored.
   This is `<target_path>/<report_name>/html`

   where `<target_path>` is the directory you specified using the `-targetpath` flag and `<report_name>` is the report name you specified using the `-name` flag when generating the report.

2. Open the `index.html` file for viewing in your browser.
   The report's main page displays in your browser.
How Policy Deviation Calculations Work

Advanced policy management and reporting let you see the difference between the policy rules that are meant to be deployed on the end-point, and the actual policy rules that are applied on the end-point. This lets you deal with problems associated with the deployment of your policies.

The policy deviation calculation is run on each end-point and performs the following actions:

1. Retrieves from the local host the list of rules that should be deployed on the end-point.

   These are the rules that are specified for each of the deployed policies as specified in the local RULESET object that is associated with the POLICY object for each deployed policy.

2. Checks that each of these rules is applied to the end-point.

   Important! The deviation calculation does not check whether UNIX (native) rules are applied. It also ignores rules that remove objects (user or object attributes, user or resource authorization, or actual users or resources) from the database. For example, the calculation cannot verify whether the following rule is applied:

   ```
   rr FILE /etc/passwd
   ```

3. (Optional). Compare between the policies associated with the local HNODE object and the one on the first available DMS.

   Normally, the deviation calculator checks for deviations only on the local host. If you specify the `-strict` option, this option, the deviation calculator also compares the local policies to the policies on the first available DMS in the list. It compares the:

   a. List of policies associated with the HNODE object representing the local host.

   b. Policy state of each POLICY object associated with the HNODE object.

   c. Policy signature of each POLICY object associated with the HNODE object.

4. Outputs the following two files:

   - `<eTrustACDir>/data/devcalc/deviation.log`
     Log and error messages collected during the last deviation calculation.

   - `<eTrustACDir>/data/devcalc/deviation.dat`
     List of policies and their deviations. You can get the contents of this file using the selang command `get devcalc` on the end point.

Note: eTrust AC also sends audit events which can be viewed using `seaudit -a`. For more information about the `seaudit` utility, see the *Utilities Guide*. 

5. Notifies one or more DMSs of any deviations found.

The DMSs to notify are either specified manually (-dms option), or if no DMS is specified, the deviation calculator uses the DMS list specified for the local eTrust AC database.

Specify the DMS Databases to Send Policy Deviation Status Notifications To

When you install or upgrade eTrust AC using a the -advreport option, the installation process configures the policy deviation calculator to send policy deviation status to specified DMS databases. You can also do this post installation, or add additional DMS databases to notify in this manner.

To specify the DMS databases to send policy deviation status notifications to, in a selang command window, enter the following command:

```
so dms+ (<DMS1> [, <DMS2>])
```

where <DMSx> is the name of the DMS you want the deviation calculation to send policy deviation status notifications to. Each DMS has to be specified in the following format: DMS_name@hostname.

Policy Deviation Log and Error File

The policy deviation calculation writes a new log during each deviation calculation. This log also contains error messages and is stored in `<eTrustACDir>/data/devcalc/deviation.log`

Use this log when the deviations you see in your reports (that are retrieved from the DMS) are not gathered from the last time a deviation calculation should have run. It can help you diagnose why the deviation calculation results were not sent to the DMS.

**Important!** If the deviation log contains the error "ERROR: failed to initialize DB library, database is open", you need to recreate the database's index files. To do this, exit selang and run the following command from the `<eTrustACDir>/data/devcalc/init_ac_db` directory, then rerun the deviation calculation (see page 172):

```
 selang -I -d .
```

Example: Deviation log and error file

The following is an example deviation log and error file:

```
start time: Mon Jan 23 13:04:48 2006
WARNING,"failed to retrieve DMS host name, deviation will be stored locally"
found deviation(s) for policy 'iis8#02'
end time: Mon Jan 23 13:05:04 2006
```
Policy Deviation Data File

The policy deviation calculation writes a data file that contains a list of policies and their deviations. This data file is stored in `<eTrustACDir>/data/devcalc/deviation.dat`

**Note:** The list of policies included in the data file depends on the policies that a deviation is calculated for (by default, all the policies and all policy versions on the end-point).

**Important!** The deviation calculation does not check whether UNIX (native) rules are applied. It also ignores rules that remove objects (user or object attributes, user or resource authorization, or actual users or resources) from the database. For example, the calculation cannot verify whether the following rule is applied:

```
rr FILE /etc/passwd
```

The deviation status is sent (whether a deviation exists or not) to the DMS but the actual deviation is stored locally. When a report is created, the actual deviation results can be retrieved from this file and added to the report.

The following lines can appear in the policy deviation data file:

**Date**
Date of deviation calculation.

**Format:** DATE, DDD MMM DD hh:mm:ss YYYY

**Strict**
Specifies that the deviation calculation was run with the -strict option.

**Format:** STRICT, DMS@hostname, policy_name#xx, {1|0}

where {1|0} signifies whether a deviation was found (1) between the policies associated with the local HNODE object and the ones associated with the HNODE object on DMS@hostname (the first available DMS), or not (0).

**Policy Start**
Starts a policy block which defines the deviation for this policy version.

**Format:** POLICYSTART, policy_name#xx

**Difference**
Describes a deviation that was found for a policy. The name of the policy for which the deviation applies to is the nearest **policy line** above this line.

There are four types of deviations which are described in the following table:
### Deviation Type Format

<table>
<thead>
<tr>
<th>Deviation Type</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class not found</td>
<td>DIFF, (&lt;class_name&gt;), (<em>)&amp;,</em> (<em>), (</em>)</td>
</tr>
<tr>
<td>Object not found</td>
<td>DIFF, (&lt;class_name&gt;), (&lt;object_name&gt;), (<em>), (</em>)</td>
</tr>
<tr>
<td>Property not found</td>
<td>DIFF, (&lt;class_name&gt;), (&lt;object_name&gt;), (&lt;property_name&gt;), (*)</td>
</tr>
<tr>
<td>Property value mismatch</td>
<td>DIFF, (&lt;class_name&gt;), (&lt;object_name&gt;), (&lt;property_name&gt;), (&lt;expected_value&gt;)</td>
</tr>
</tbody>
</table>

**Policy End**

Ends a policy block which defines the deviation for this policy.

**Format:** POLICYEND, policy_name#xx, {1|0}

where {1|0} signifies whether a deviation was found (1) or not (0).

**Warning**

Describes a warning.

**Format:** WARNING, "warning_text"

**Example: Deviation data file**

Date, Sun Mar 19 08:30:00 2006
WARNING, "failed to retrieve DMS host name, deviation will be stored locally"
POLICYSTART, iis8#02
DIFF, (USER), (am), (*), (*)
POLICYEND, iis8#02, 1
Perform a Deviation Calculation

A deviation calculation should be performed regularly so that the DMS contains recent information about policy deviation status. We recommend that you schedule a policy deviation calculation to occur in an interval that supports your reporting requirements.

To perform a deviation calculation on an end point, from a selang window, enter the following command:

```
start DEVCALC
```

**Example: Schedule a routine deviation calculation**

The following example shows how you can create on Solaris a deviation calculation task that:
- Runs daily at midnight.
- Sends the deviation status to the DMS:
  
  ```
  mainDMS on the mainhost.domain.com computer.
  ```

To do this, add the following line to your Crontab file:

```
0 0 * * * selang -c "start DEVCALC \
params('dms mainDMS@mainhost.domain.com')"
```

Mainframe Password Synchronization

eTrust AC supports password synchronization among mainframes (see page 261) running eTrust CA-Top Secret Security, eTrust CA-ACF2 Security, or RACF security products (and CA Common Services CAICCI package) and Windows or UNIX computers running eTrust AC. Synchronization is accomplished using the standard eTrust AC password Policy Model method.

Any password change a mainframe user makes is propagated to all the machines in the password Policy Model hierarchy.
Chapter 10: General Security Features

This section contains the following topics:

- Locking Idle Stations (see page 173)
- Protecting Resources Using APIs (see page 174)
- Protecting Against Stack Overflow: STOP (see page 175)
- Defining Day and Time Access Rules for Resources (see page 176)
- B1 Security Level Certification (see page 176)

Locking Idle Stations

Information is extremely vulnerable when terminals are left open and active. An intruder who happens upon such a terminal (for example, during a lunch break) need not try to break passwords or have complicated equipment to sniff the network lines, since all terminals at the site are already logged in and ready for work. Although screen savers that prompt for the password before restoring the desktop are useful, the security administrator cannot make sure that all users are using secured screen savers.

eTrust AC provides selock, a screen-locking utility that guards all terminals and stations by locking them whenever they are idle for more then a specified period of time. When returning to work, the user is prompted to specify the password. If the correct password is not specified within one minute, the terminal remains locked.

**Note:** For more information about the screen lock utility, see the *Utilities Guide*.

Selock offers three modes of operation:

- **Monitor**-If selock detects no keyboard or mouse activity during a time-out period, and the transparent parameter is off, selock automatically switches to the screen saver mode. No password entry is required for the transition from the monitor mode to the saver mode.

- **Saver**-Selock blanks the entire screen and displays a shifting system icon. As soon as selock detects any keyboard or mouse activity, it returns to the monitor mode. As long as no keyboard or mouse activity occurs, selock remains in the saver mode. If selock remains in the saver mode for the period specified by the lock-timeout parameter, selock automatically switches to the lock mode, without any visual indication of the transition.
Lock - By default, selock continues to display a moving eTrust logo on a black background. When selock detects any keyboard or mouse activity, a dialog box that prompts for the user's password appears. If the user enters the correct password, selock switches back to the monitor mode. Otherwise, the password-entry dialog closes and selock remains in the lock mode.

To protect unattended stations and X terminals, place the selock command in the user's login script (the .login file). Alternatively, you can place the selock command in the /etc/login or /etc/cshrc file.

Note: For the selock command to work, you must set the DISPLAY environment variable. You can also specify the target display directly, instead of specifying $DISPLAY.

The following is a typical startup command, suitable to be placed in X startup files:

```
selock -display $DISPLAY -timeout 5
```

This command activates selock after five minutes of terminal inactivity.

We recommend that you place the following line in the global xstartup script. The xstartup script usually resides in the directory /usr/lib/X11/xdm/Xstartup.

```
selock -display $DISPLAY -user $USER -timeout 3 &
```

This statement enforces use of the terminal locking program for all users who are using X terminals.

Protecting Resources Using APIs

If you have defined resources that are not part of eTrust AC (that is, in-house resources), you can protect them by using eTrust AC APIs. Each API has two layers:

**The function library**

- Enables programmers to use the eTrust AC authorization engine.

**The user exits**

- Enable the system administrator to tailor eTrust AC behavior to the requirements of the site.

Note: For more information about eTrust AC APIs, see the **SDK Guide**.
Protecting Against Stack Overflow: STOP

Stack overflow enables hackers to execute arbitrary commands on remote or local systems, many times as the root user (the superuser). They do this by exploiting bugs in the operating system or other programs. These bugs allow users to overwrite the program stack, changing the next command to be executed.

Stack overflow is not simply a bug; it is possible to create a block that overwrites the return address with a meaningful address, resulting in transferred control to unauthorized code (usually in the same block).

Stack Overflow Protection (STOP) is a feature that prevents hackers from creating and exploiting stack overflow to break into systems.

Starting and Stopping STOP

When STOP is first installed, stack overflow protection is activated by default. To deactivate it, you must change a token in the [seos_syscall] section of the seos.ini file and restart eTrust AC. To do this, use the seini command as follows:

```
seini -s SEOS_syscall.STOP_enabled 0
```

You could manually change the seos.ini file instead.

To re-enable STOP, change the value of the token to 1 and restart eTrust AC.

**Note:** When STOP is active on Sun Solaris 7 systems, the dbx program cannot work properly. If you need to use dbx on a system that is protected by STOP, you must first disable STOP.
Defining Day and Time Access Rules for Resources

You can use eTrust AC to specify day-of-week and time-of-day restrictions for resource access. This feature can be exploited for TERMINAL access, SURROGATE requests, and user-defined resources. For example, the following rule completely disables the terminal ws3 on weekends and outside the 08:00-19:00 time period every day:

```
eTrustAC> chres TERMINAL ws3 restrictions(days(weekdays) time(0800:1900))
```

No login request from that station is accepted outside these periods.

You can use eTrust AC to protect against substitution requests to highly authorized users outside work hours. Suppose user AcctMgr is the Accounting Manager, who is allowed to perform financial transactions, and you have restricted AcctMgr login to work hours and weekdays only. Intruders or unauthorized personnel may try to access the account of AcctMgr by invoking the command `su AcctMgr`. Use the following command to make it impossible to substitute the user name to AcctMgr outside the specified period:

```
eTrustAC> chres SURROGATE USER.AcctMgr restrictions(days(weekdays) time(0800:1900))
```

The same technique can be implemented for any protected resource, including user-defined abstract classes that are used for implementing in-house applications.

B1 Security Level Certification

eTrust AC includes the following B1 "Orange Book" features:

- Security categories
- Security labels
- Security levels

Security Levels

When security level checking is enabled, eTrust AC performs security level checking in addition to its other authorization checking. A security level is a positive integer between 1 and 255 that can be assigned to users and resources. When a user requests access to a resource that has a security level assigned to it, eTrust AC compares the security level of the resource with the security level of the user. If the user's security level is equal to or greater than the security level of the resource, eTrust AC continues with other authorization checking; otherwise, the user is denied access to the resource.
If the SECLABEL class is active, eTrust AC uses the security level associated with the security labels of the resource and user; the security level that is explicitly set in the resource and user records is ignored.

To protect a resource with security level checking, assign a security level to the resource's record. The level parameter of the newres or chres command assigns a security level to a resource.

To allow a user access to resources protected by security level checking, assign a security level to the user's record. The level parameter of the newusr or chusr command assigns a security level to a user.

**Enabling Security Level Checking**

The following setoptions command enables security level checking:

```
eTrustAC> setoptions class+ (SECLEVEL)
```

**Disabling Security Level Checking**

The following setoptions command disables security level checking:

```
eTrustAC> setoptions class- (SECLEVEL)
```

**Security Categories**

When security category checking is enabled, eTrust AC performs security category checking in addition to other authorization checks. When a user requests access to a resource that has one or more security categories assigned to it, eTrust AC compares the list of security categories in the resource record with the category list in the user record. If every category assigned to the resource appears in the user's category list, eTrust AC continues with other authorization checking; otherwise, the user is denied access to the resource.

If the SECLABEL class is active, eTrust AC uses the list of security categories associated with the security labels of the resource and user; the lists of categories in the user and resource records are ignored.

To protect a resource by security category checking, assign one or more security categories to the resource's record. The category parameter of the newres or chres command assigns security categories to a resource.

To allow a user access to resources protected by security category checking, assign one or more security categories to the user's record. The category parameter of the newusr or chusr command assigns security categories to a user.
Enabling Security Category Checking

The following setoptions command enables security category checking:

```
eTrustAC> setoptions class+ (CATEGORY)
```

Disabling Security Category Checking

The following setoptions command disables security category checking:

```
eTrustAC> setoptions class-(CATEGORY)
```

Defining a Security Category

Define a security category by defining a resource in the CATEGORY class. The following newres command defines a security category:

```
eTrustAC> newres CATEGORY name
```

where `name` is the name of the security category.

To define the security category “Sales,” enter the following command:

```
eTrustAC> newres CATEGORY Sales
```

To define the security categories “Sales” and “Accounts,” enter the following command:

```
eTrustAC> newres CATEGORY (Sales,Accounts)
```

Listing Security Categories

To display a list of all the security categories that are defined in the database, use the show command as follows:

```
eTrustAC> find CATEGORY
```

The list of security categories displays on the screen.

Deleting a Security Category

Delete a security category by removing its record from the CATEGORY class. The following rmres command removes a security category:

```
eTrustAC> rmres CATEGORY name
```

where `name` is the name of the security category.

To remove the security category “Sales,” enter the following command:

```
eTrustAC> rmres CATEGORY Sales
```
Security Labels

A security label represents an association between a particular security level and zero or more security categories.

When security label checking is enabled, eTrust AC performs security label checking in addition to other authorization checks. When a user requests access to a resource that has a security label assigned to it, eTrust AC compares the list of security categories specified in the resource record's security label with the list of security categories specified in the user record's security label. If every category assigned to the resource's security label appears in the user's security label, eTrust AC continues with the security level check; otherwise, the user is denied access to the resource. eTrust AC compares the security level specified in the resource record's security label with the security level specified in the user record's security label. If the security level assigned in the user's security label is equal to or greater than the security level assigned in the resource's security label, eTrust AC continues with other authorization checking; otherwise, the user is denied access to the resource.

When security label checking is enabled, the security categories and security level specified in the user and resource records are ignored; only the security level and categories specified in the security label definitions are used.

To protect a resource by security label checking, assign a security label to the resource's record. The label parameter of the newres or chres command assigns a security label to a resource.

To allow a user access to resources protected by security label checking, assign a security label to the user's record. The label parameter of the newusr or chusr command assigns security labels to a user.

Enabling Security Label Checking

The following setoptions command enables security label checking:

eTrustAC> setoptions class+(SECLABEL)

Disabling Security Label Checking

The following setoptions command disables security label checking:

eTrustAC> setoptions class-(SECLABEL)
Defining a Security Label

Define a security label by defining a resource in the SECLABEL class. The following newres command defines a security label:

```bash
eTrustAC> newres SECLABEL name category(securityCategories) level(securityLevel)
```

where:

- **name**
  Specifies the name of the security label.

- **securityCategories**
  Specifies the list of security categories. To specify more than one, separate the security category names with a space or a comma.

- **securityLevel**
  Specifies the security level. Use an integer between 1 and 255.

To define the security label Managers to contain the security categories Sales and Accounts and a security level of 95, enter the following command:

```bash
newres SECLABEL Manager category(Sales,Accounts) level(95)
```

Listing the Security Labels

To display a list of all the security labels that are defined in the database, use the show command as follows:

```bash
eTrustAC> find SECLABEL
```

The list of security labels appears on the screen.

Deleting a Security Label

A security label is deleted by removing its record from the SECLABEL class. The following rmres command removes a security label:

```bash
eTrustAC> rmres SECLABEL name
```

where *name* is the name of the security label.

To remove the security category "Manager" enter the following command:

```bash
eTrustAC> rmres SECLABEL Manager
```
Chapter 11: Auditing Events

This section contains the following topics:
Setting Audit Rules (see page 181)
Audit Logs (see page 183)

Setting Audit Rules

eTrust AC keeps audit records for events of access denial and access grants according to the audit rules defined in the database. The decision whether to log a certain event is based on the following rules:

- Every accessor and resource has an AUDIT property that can be set to one or more of the following values:
  
  **FAIL**
  Logs access failures to protected resources.

  **SUCCESS**
  Logs successful accesses to protected resources.

  **LOGINFAIL**
  Logs every login failure of the user.

  **LOGINSUCCESS**
  Logs every successful login of the user.

  **ALL**
  Logs the same as FAIL, SUCCESS, LOGINFAIL, and LOGINSUCCESS together.

  **TRACE**
  Logs every event from the trace to the audit log. In addition to what ALL includes, TRACE includes fork events, exec events, network connections, and more.

  **NONE**
  Logs nothing.

<table>
<thead>
<tr>
<th>Value</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>Users and resources</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>Users and resources</td>
</tr>
</tbody>
</table>
### Setting Audit Rules

<table>
<thead>
<tr>
<th>Value</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINFAIL</td>
<td>Users</td>
</tr>
<tr>
<td>LOGINSUCCESS</td>
<td>Users</td>
</tr>
<tr>
<td>ALL</td>
<td>Users and resources</td>
</tr>
<tr>
<td>TRACE</td>
<td>Users</td>
</tr>
<tr>
<td>NONE</td>
<td>Users and resources</td>
</tr>
</tbody>
</table>

- If either the resource or the accessor (user) has AUDIT(ALL) set, all events concerning protected resources are logged, regardless of whether access failed or succeeded.
- If the access to a protected resource is successful and the user or the resource has AUDIT(SUCCESS) set, the event is logged.
- If the access to a protected resource fails and the user or the resource has AUDIT(FAIL) set, the event is logged.

### Using the Warning Mode

If a particular resource is set to warning mode, a violation of access rules for that resource results in an audit record mentioning that the violation was permitted because warning mode is in effect.

**Note:** In Warning Mode, eTrust AC does not create warning messages for resource groups.
**Audit Logs**

**Note:** The seauditx graphical utility for viewing audit logs is part of the Security Administrator.

The audit records are stored in a file called the audit log. The location for the audit log is specified in the seos.ini file. The seaudit and seauditx utilities can be used to list recorded events in the audit log, filter events by time restrictions or event type, and so on.

**Note:** For more information about seaudit, see the *Utilities Guide*. For more information about seauditx, see the *User Guide*.

The audit logs are stored locally, but you can use eTrust AC to distribute the auditing information by using the log routing facility. Consider archiving old audit logs to tape, to allow you to scan the events later.

By default, the authorization daemon seosd creates the audit logs with root ownership, since the seosd program is executed by the user root. For the same reason, the audit logs are created with read/write permissions granted only to root.

To enable other users to read the audit logs without having to su (substitute user) to root, eTrust AC includes two entries in the seos.ini file that specify which group ownership is assigned to the log files.

- One entry is for the audit log.

  Suppose the auditors at your site are all members of a group named auditforce. You want these users to be able to browse through the local audit log files. Edit the seos.ini file so that the audit_group token in the [logmgr] section is set to auditforce. eTrust AC then gives the auditforce group read permission to your local audit logs. From this point, any local audit logs created at your station have the auditforce group as their owner.

  The log routing daemons consult the same token to see who should have access rights to the audit logs that the daemons produce and collect. Note that the audit logs are subject to access control like any other files, and eTrust AC rules can keep users from accessing them.

- The other entry is for the error log, and it is used in the same way to specify group ownership for that file.
The System Auditor

A system auditor is a user to whom the AUDITOR attribute is assigned. Users defined as system auditors are permitted to perform auditing tasks such as changing the auditing attribute that is assigned to users and resources.

Auditing tasks can be carried out from central locations. To collect auditing information from the various stations on the network in a single host, the auditor can use the log routing facility.

Setting Up the Log Routing Facility

To set up the log routing facility, complete these steps:

1. Create a log routing configuration file.
   
   Unless you specify otherwise with the RouteFile token in the seos.ini file, eTrust AC expects your log routing configuration file to be named `eTrustACDir/log/selogrd.cfg`
   
   where `eTrustACDir` is the installation directory for eTrust AC, by default `/opt/CA/eTrustAccessControl`.
   
   You can find sample log routing configuration files in the directory `eTrustACDir/samples/selogrd.init`. Alternatively, as a very simple log routing configuration file, you can create a file consisting of the following three lines:
   
   ```
   Rule
   host destination
   .
   ```
   
   For `destination`, enter the name of the host that should receive the audit records. All classes, resources, accessors, and results are logged.
   
   **Note:** For more information about the syntax of the configuration file, see the `seologrd` utility in the *Utilities Guide*.

2. Start the emitter daemon on all hosts that are to route auditing information, and execute the collector daemon on all hosts that are to collect auditing information.
   
   **Note:** For more information about using these daemons, see the *Utilities Guide*.
File Notifications

Besides compiling the log, the log routing facility can also send notifications to the host's display screen, to an email address, or to other destinations. You can base notifications on information from your station's own audit log or from logs that the collector daemon has brought to your station.

To set up such notifications, you need to use the log routing configuration file and a selang command. For example, suppose you want to notify the user John whenever a setuid request to user root is successfully made.

1. Issue the following selang command:

   eTrustAC> chres SURROGATE USER.root notify(John)

   This chres command specifies that each time someone surrogates user to root, a special audit log record is created, and the seosd daemon is to notify the user named John. The daemon also creates a special kind of audit record called a notification record.

2. Once you have specified notification for one or more resources, you can add the following three lines to the log routing configuration file.

   Rule2
   notify default
   .

   This line causes the log routing emitter to create a mail message for the notification audit record.

   **Note:** For more information about the configuration file format and setting up the log routing daemons, see the Utilities Guide.
Chapter 12: Remote Status View

This section contains the following topics:

Requirements (see page 187)
Installing and Starting RSV (see page 187)
Status Categories (see page 190)
Using RSV (see page 194)
RSV Security (see page 195)

Requirements

Remote Status View (RSV) is a web-based status information utility. Use a browser to connect to RSV, which displays status information about eTrust AC for different hosts.

Before using RSV, you must have a web browser such as Microsoft Internet Explorer (IE) or Netscape Navigator, version 4.0 or later. You must also install and start the RSV daemon, as described in the following section.

Use the following tips when installing and using RSV:

- Install eTrust AC version 5.1 or higher before installing RSV.
- Install RSV in the eTrustACDir/rsv directory, where eTrustACDir is the eTrust AC home directory.
- Read the README file in the eTrustACDir/rsv/doc directory for RSV usage instructions.
- Uninstall RSV by removing the eTrustACDir/rsv subdirectory.

Installing and Starting RSV

RSV is normally installed with eTrust AC as part of the server package. However, you can opt to install it separately.
**Installing**

To install RSV manually, complete the following steps:

1. Log in as root.
2. Change to the directory that contains the RSV tar file.
3. Run the install script with the following command:
   ```bash
   # install_rsv
   ```

   The install script installs the components and directories for RSV. When installation is complete, the script displays instructions for starting the RSV daemon.

**Starting**

To start the RSV daemon, enter the following command:

```bash
# eTrustACDir/rsv/adm/startrsv [portno]
```

where `eTrustACDir` is the installation directory for eTrust AC (by default `/opt/CA/eTrustAccessControl`) and `portno` is the http port number (by default 8080).

**Note:** You can change the default port number by editing the `rsv_port` token. You can also designate a UNIX user to which the RSV daemon switches (SETUID) after being started by root, by editing the `rsv_user` token. Both tokens are located in the `[rsv]` section of the `seos.ini` file.
Displaying

To display RSV in your browser, open your browser and enter the following URL in the Address or Location window:

http://hostname:portno

Where *hostname* specifies the name of the host running RSV and *portno* specifies the http port number (default is 8080).

The RSV main screen appears.

The RSV screen is divided into three frames:

- The top, or title, frame displays the current host name beneath the title. The right side displays the Access eTrust AC version, the date and time the RSV host was loaded, and the date and time when audit collection was started display.

  **Note:** The information in this frame appears only when you load a host; it is updated only when you reload the host.

- The left, or category, frame contains the function and category links. A red category link indicates a possible security leak: at least one of the status items in that category has a value of at least one.
The contents of the right frame vary depending on the function selected in the left frame. Initially, and whenever you click the Load link, this frame contains the following elements:

**Host**

Use this field to enter the name of the host for which you want to display status information.

**NIS**

Use this list box to select a host from the NIS database. When you click a host name, it appears in the Host field.

**DNS**

Use this list box to select a host from the DNS database. When you click a host name, it appears in the Host field.

**Apply**

Click to load the host and display its status information.

When you click a category link in the left frame, the right frame displays the status items for that category. The following section describes these categories.

**Status Categories**

The left frame contains six categories, each of which displays several related status items.

**Security Status Summary**

The Security status summary screen displays the following status items:

**Login events**

Number of logins (except from selang).

**Failed login events**

Number of failed logins. A value of at least one indicates a possible security leak; in that case, the value and the category name appear in red.

**Administrative transactions**

Number of updates to the database.

**eTrust AC restarts**

Number of times eTrust AC has been restarted.

**eTrust AC denials**

Number of requests denied by eTrust AC.
Warning messages
Number of resources in warning mode.

Attempts to kill eTrust AC
Number of attempts to kill the seosd daemon. A value of at least one indicates a possible security leak; in that case, the value and the category name appear in red.

File Protection

The File protection screen displays the following status items:

File rules
Number of FILE objects in database.

Monitored files
Number of SECFILE objects in database.

Denied access to files
Number of failed attempts to access protected files.
Program Protection

The Program protection screen displays the following status items:

- For SUID/SGID program protection:
  - **SUID/SGID protection**
    Whether the PROGRAM class is active.
  - **Protected programs**
    Number of PROGRAM objects in database.
  - **Untrusted programs**
    Number of programs marked as untrusted. A value of at least one indicates a possible security leak. In that case the value and the category name appear in red; the item also becomes a link to a list of these programs.

- For Protection from kill attempts:
  - **Protected processes**
    Number of PROCESS objects in the database.
  - **Violations**
    Number of attempts to kill a protected process. A value of at least one indicates a possible security leak. In that case the value and the category name appear in red; the item also becomes a link to a list of relevant audit records.
Account Protection

The Account protection screen displays the following status items:

- For SU protection (SURROGATE class):
  - **su enabled**
    Whether the SURROGATE class is active.
  - **root protection**
    Whether a surrogate USER.root record is defined.
  - **Failed su attempts**
    Number of attempts to break su protection. A value of at least one indicates a possible security leak; in that case, the value and the category name display in red.
  - **Defined SUDO jobs**
    Number of SUDO jobs.
    If more than 0, links to a list of these jobs.

- For account information:
  - **List of revoked users**
    Link to a list of users with login limitations.
  - **Inactive users list**
    Link to a list of accounts that were inactive for at least seven* days.

Password Control

The Password control screen displays the following status items:

- **Password control active**
  Whether the PASSWORD token in the SEOS class is active.

- **Password change report**
  Link to a list of users that must change passwords within seven days. The PASSWORD token must be active to receive this report.

**Note:** You can modify the number of days by editing the sereport.cfg file located in the *eTrustACDir/rsv/sereport* directory.
Optional Services

The Optional services screen displays the following status items:

**User revoking service (serevu)**
- Whether the Revoke User daemon is running.

**Log emitting service (selogrd)**
- Whether the audit log emitter daemon is running.

**Log collection service (selogrcd)**
- Whether the audit log collector daemon is running.

Using RSV

Control RSV operations from the left frame. The top of the frame contains two functions, and the bottom of the frame contains category links that display various groups of status information in the right frame. If a category link appears in red, it indicates a possible security leak in one or more status items in that category; the relevant status item also appears in red.

To view status information:
1. Enter the name of the host in the Host field, or click on the host name in the NIS or DNS list box.
   - Click Apply. The right panel displays the Security status summary panel.
2. Click the category links in the left panel to view other status pages.
3. The Account protection and Password control pages contain additional links; click a link to open another window with more detailed information.

To view the status of another host, complete the following steps:
1. Click Load Host. The right frame redisplays the host panel.
2. Enter the name of the host in the Host field, or click on the host name in the NIS or DNS list box.
3. Click Apply to display status information for the new host.

RSV presents a snapshot of the host's status at the time you load the host. To view the most recent status information, you must reload the host.

To update the status display, click Reload Host in the left panel. The right frame clears briefly, while the most recent data loads. Note that the Load Time in the top frame is updated to the current time.
The installation process creates a special, logical eTrust AC user called RSV. This user has access authorization to and from all eTrust AC resources—in both local and remote hosts—that RSV requires. Note that the RSV user is not a UNIX user.

Additionally, a SPECIALPGM record is created that causes the RSV daemon to run under the security context of the RSV user. From the point of view of the software, the RSV user runs the RSV daemon.

Consequently, all users have access to the RSV daemon, and can view the status information using their browsers. If necessary, you can limit a host’s RSV access by applying HOST or TCP rules (see page 107) in the RSV server host.
Chapter 13: Scope of Administration Authority

This section contains the following topics:

- Global Authorization Attributes (see page 197)
- Group Authorization (see page 199)
- Ownership (see page 203)
- Authorization Examples (see page 204)
- The ADMIN Class (see page 206)
- Environmental Considerations (see page 207)

Global Authorization Attributes

Global authorization attributes are set in the user record. Each global authorization attribute permits the user to perform certain types of functions. This section describes the functions and the limits of each global authorization attribute.

ADMIN Attribute

The ADMIN attribute allows a user to execute almost all commands in eTrust AC. This is the most powerful attribute in eTrust AC, but it does have limitations. These limitations include the following:

- If only one user in the database has the ADMIN attribute, that user cannot be deleted, and the ADMIN attribute cannot be removed from the record.
- Users with the ADMIN attribute but without the AUDITOR attribute cannot update the audit mode. If you have the ADMIN attribute and need to make changes to the audit mode, assign yourself the AUDITOR attribute.
- Users with the ADMIN attribute cannot delete root, but they can set root to be a non-ADMIN user.
AUDITOR Attribute

Users with the AUDITOR attribute set can monitor the use of the system. The explicit privileges of a user with the AUDITOR attribute are included in the following table:

<table>
<thead>
<tr>
<th>Access</th>
<th>Description</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>List information in the database.</td>
<td>showusr, showgrp, showres, showfile, find</td>
</tr>
<tr>
<td>Modify</td>
<td>Set the audit mode for existing records.</td>
<td>chusr, chgrp, chres, chfile</td>
</tr>
</tbody>
</table>

OPERATOR Attribute

Users with the OPERATOR attribute have READ access to all files. With this access, they can list everything in the database, and they can run backup jobs. To list database records, operators use the showusr, showgrp, showres, showfile, and find commands. The OPERATOR attribute also allows the user to use the secons utility (described in the Utilities Guide).

PWMANAGER Attribute

The PWMANAGER attribute gives an ordinary user (that is, a user without the ADMIN attribute) the authority to use the chusr or sepass command to change the passwords of other users (excluding users who have the ADMIN attribute).

The PWMANAGER can change the ADMIN user's password if the cng_adminpwd parameter is set in the setoptions command.

To set the option, issue the following command from selang:

eTrustAC> so cng_adminpwd

To remove this option, issue the following command:

eTrustAC> so cng_adminpwd-

The PWMANAGER attribute does not include authority to change the number of grace logins, the password interval of another user, or general password rules.

The PWMANAGER's authority also includes use of the showusr and find commands.

Note: If a user has the nochngpass property set to yes, a PWMANAGER cannot change the password for that user.
eTrust AC, like many other security models, does not allow a regular user to ask: "Can user A please access resource X?" The only question a regular user can ask is: "Can I please access resource X?" However, it should permit a process that supplies services to many users, such as a database server daemon or an in-house application, to ask for authorization on behalf of other users.

The SERVER attribute allows a process to ask for authorization for users. Users with the SERVER attribute set can issue the SEOSROUTE_VerifyCreate API.

**Note:** For more information about the server attribute and eTrust AC APIs, see the *SDK Guide*.

IGN_HOL Attribute

The IGN_HOL attribute allows users to log in during any period defined in a holiday record. Each record in the HOLIDAY class defines one or more periods when users need extra permission to log in. With the IGN_HOL attribute, users can log in at any time, regardless of the periods defined in holiday records.

**Note:** For more information about the HOLIDAY class, see the *Reference Guide*.

Group Authorization

It is necessary to understand the concept of parentage before discussing group authorization attributes.
Parentage

The concept of subordinate and superior groups, also known as parentage, is important when discussing group administration privileges. One group can be the parent-superior-of one or more groups. A child or subordinate group can have only one parent. Assigning a parent to a group is optional. Consider the following diagram:

![Group Authorization Diagram]

Group 1 is the parent of the three Groups 20, 30, and 40. Group 30 is also the parent of three groups-500, 600, and 700. Group 600 has only one parent-Group 30. Group 1 has no parent.

Group Authorization Attributes

All records, including resource records and accessor records alike, have owners. Owning a record means having authorization to view, edit, and remove it, as described in Ownership in this chapter.

A group can own its own records. However, within a group that owns records, only certain privileged users can manage the records. These special users have a group authorization attribute set in their own user records. The group authorization attributes are the following:

- GROUP-ADMIN
- GROUP-AUDITOR
- GROUP-OPERATOR
- GROUP-PWMANAGER

The join command—which only a properly authorized user can issue—sets these attributes. The join command serves the purpose of both putting a user into a group, and specifying the user's group authorization attribute (if any).

The privileged members of the group may or may not be authorized to manage the user records that define the members of the group, depending on who owns those records.
**GROUP-ADMIN Attribute**

Users with a group administration authorization attribute can create a certain set of records. In order to create a record, the group administrator has to specify the owner of the record.

The owner of the records must be the group in which the user has a group authorization attribute. If that group is the parent of other groups, the owner can also be from one of the sub groups. The whole set of records is called the group scope. The authorization examples (see page 204) provided illustrate the concept of group scope.

Users with the GROUP-ADMIN attribute have the following access authority for the records within their group scope:

<table>
<thead>
<tr>
<th>Access</th>
<th>Description</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Show the properties of the record.</td>
<td>showusr, showgrp, showres, showfile</td>
</tr>
<tr>
<td>Create</td>
<td>Create new records in the database. You must specify the owner.</td>
<td>newusr, newgrp, newres, newfile</td>
</tr>
<tr>
<td>Modify</td>
<td>Change the properties of the record.</td>
<td>chusr, chgrp, chres, chfile</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove records from the database.</td>
<td>rmusr, rmgrp, rmres, rmfile</td>
</tr>
<tr>
<td>Connect</td>
<td>Join a user to a group or separate a user from a group.</td>
<td>join, join-</td>
</tr>
</tbody>
</table>
The GROUP-ADMIN attribute also has limits:

- GROUP-ADMIN users cannot make resources inaccessible to themselves, so:
  - GROUP-ADMIN users cannot assign a security level that is higher than their own security level.
  - GROUP-ADMIN users cannot assign a security category or security label that they do not have.
- GROUP-ADMIN users cannot delete the user root from the database.
- Several limitations concern the global authorization attributes described in Global Authorization Attributes in this chapter:
  - A GROUP-ADMIN user cannot delete the only ADMIN user record in the database.
  - A GROUP-ADMIN user cannot remove the ADMIN attribute from the record of the last ADMIN user in the database.
  - GROUP-ADMIN users without the AUDITOR attribute cannot update the audit mode. Only a GROUP-ADMIN user with the AUDITOR attribute can update the audit mode.
  - GROUP-ADMIN users cannot set the global authorization attributes-ADMIN, AUDITOR, OPERATOR, PWMANAGER, and SERVER-for any user.

**GROUP-AUDITOR Attribute**

A user with the GROUP-AUDITOR attribute can list the properties of any record within the group scope. The group auditor can also set the audit mode for any record within the group scope.

**GROUP-OPERATOR Attribute**

A user with the GROUP-OPERATOR attribute can list the properties of any record within the group scope.

**GROUP-PWMANAGER Attribute**

A user with the GROUP-PWMANAGER attribute can change the password of any user whose record is within the group scope.
Ownership

Every record in the database—including both accessor records and resource records—has an owner. When you add a record to the database, you can either explicitly assign its owner by using the owner parameter or allow eTrust AC to assign the user who defines the record as the owner of the record.

An owner can be a user or a group. When a group owns a record, only users who have joined the group with the GROUP-ADMIN property can use the privileges of ownership. If you remove a user or group that owns records from the database, the records no longer have an owner.

Users who own records have the following access authority for the records they own:

<table>
<thead>
<tr>
<th>Access</th>
<th>Description</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Show the properties of the record.</td>
<td>showusr, showgrp, showres, showfile</td>
</tr>
<tr>
<td>Modify</td>
<td>Change the properties of the record.</td>
<td>chusr, chgrp, chres, chfile</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove the record from the database.</td>
<td>rmusr, rmgp, rmres, rmfile</td>
</tr>
<tr>
<td>Connect</td>
<td>Join a user to a group or separate a user from a group.</td>
<td>join, join-</td>
</tr>
</tbody>
</table>

If you do not want a user or group to have ownership authority over a particular record, assign the owner nobody to the record.

The limits of the ownership privileges are as follows:

- The owner of the last ADMIN user in the database cannot delete that user record.
- Owners who do not have the AUDITOR attribute cannot update the audit mode. Only an owner with the AUDITOR attribute can update the audit mode.
- The owner of root cannot delete root from the database.
- Owners cannot set the global authorization attributes-ADMIN, AUDITOR, OPERATOR, and PWMANAGER—for the users they own.
- Owners cannot make resources inaccessible to themselves, so:
  - Owners cannot assign a security level that is higher than their own security level.
  - Owners cannot assign a security category or security label that they do not have.
Authorization Examples

File Ownership

When a user creates a file, UNIX assigns the user as the owner of the file. eTrust AC allows the owner of a file to protect the file by defining a record in the FILE class. The owner of the file has full authority over the record of that file, so the owner can use the newfile, chfile, showfile, authorize, and authorize- commands with all parameters for the record that protect the file.

eTrust AC allows UNIX file owners to define FILE records, unless this feature is explicitly disabled. If you do not want file owners to define FILE records, make sure that the use_unix_file_owner token in the [seos] section of the seos.ini file to no. (This is the default setting.)

Authorization Examples

Following are diagrams that illustrate the concepts of group authorization attributes, parentage, ownership, membership, and group scope. These diagrams only contain users and groups, but the concept of ownership also applies to resource and file records.

Single Group Authorization

In the following diagram, four users are members of Group 1: MU1, MU2, MU3, and MU4. Group 1 also owns three users-OU5, OU6, and OU7. The member MU4 has the GROUP-ADMIN attribute.

The ellipse indicates the group scope of the commands executed by user MU4. It includes all the users owned by Group 1-OU5, OU6, and OU7.
Parent and Child Groups

In the following diagram, four users are members of Group 1: MU1, MU2, MU3, and MU4. Group 1 also owns three users-OU5, OU6, and OU7. The member MU4 has the GROUP-ADMIN attribute set in its record.

Group 1 is also the parent of three groups-20, 30, and 40. Each of these subordinate groups has two users who are members of the group and two users who are owned by the group.

The four ellipses indicate the group scope of the commands executed by user MU4. It includes all the users owned by Group 1, as well as the users owned by the groups subordinate to Group 1. The users in the group scope of MU4 are OU5, OU6, OU7, OU23, OU24, OU33, OU34, OU43, and OU44.

If there were groups subordinate to Groups 20, 30, or 40 that owned users, groups, or resources, the records owned by these groups would also be in the group scope of commands executed by user MU4.
The ADMIN Class

Users listed in the access control list (ACL) of records in the class ADMIN have privileges similar to users with the ADMIN attribute. However, the privileges of users in the ACL for records in the class ADMIN are limited to the particular class represented by the record. For example, the SURROGATE record in the ADMIN class determines which users can administer records of the SURROGATE class.

**Note:** For more information about eTrust AC classes, see the *Reference Guide*.

A user in the ACL for a particular record in class ADMIN can execute the following commands:

<table>
<thead>
<tr>
<th>Access</th>
<th>Description</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Show the properties of the record in the class.</td>
<td>showusr, showgrp, showres, showfile, find</td>
</tr>
<tr>
<td>Create</td>
<td>Create new database records in the class.</td>
<td>newusr, newgrp, newres, newfile</td>
</tr>
<tr>
<td>Modify</td>
<td>Change properties in the class.</td>
<td>chusr, chgrp, chres, chfile</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove existing class records from the database.</td>
<td>rmusr, rmgrp, rmres, rmfile</td>
</tr>
<tr>
<td>Connect</td>
<td>Add users to and remove users from groups. This access is valid only in the ACL of the GROUP record.</td>
<td>join, join-</td>
</tr>
<tr>
<td>Password</td>
<td>Control the password of all users within the database, and their password attributes. This access grants the same authority as the access permitted a user with the PWMANAGER attribute. This is valid only in the ACL for record USER.</td>
<td>chusr</td>
</tr>
</tbody>
</table>
Users with ADMIN class privileges have the following limitations:

- Users defined in the ACL of the USER record in class ADMIN cannot delete the last ADMIN user in the database.
- ADMIN class users cannot set the global authorization attributes-ADMIN, AUDITOR, OPERATOR, and PWMANAGER-for the users they own.
- ADMIN class users cannot necessarily update the audit mode. Only an ADMIN class user with the AUDITOR attribute can update the audit mode.
- ADMIN class users cannot delete root, but they can set root to be NOADMIN.
- ADMIN class users cannot make resources inaccessible to themselves, so:
  - ADMIN class users cannot assign a security level to a resource that is higher than their own security level.
  - ADMIN class users cannot assign a security category or security label that they do not have.

These limitations are part of the B1 security level certification (see page 176).

**Environmental Considerations**

One of the factors governing whether you can update information in your database is the position you occupy in the environment.

**Remote Administration Restrictions**

You may access a remote station over a network and update the database on the remote station. To update the database on the remote station, both you and your terminal need permission.

- You must be explicitly defined as a user in the database of the remote station. For whatever commands you want to execute, the appropriate attribute must be set in your user record in the database of the remote station.
- You must explicitly mention your local terminal’s needs in a rule granting it WRITE permission for accessing the remote station; otherwise, you cannot perform eTrust AC administration there.

With WRITE permission through a default access field (_default), or through the UACC class, you can enter the selang command shell at the remote station. However, you cannot execute any selang commands or otherwise access to the remote database. With READ permission, you can log in to the remote station but you cannot perform eTrust AC administration there.
Here is an example of this distinction between WRITE and READ permission:

1. To specify a new terminal with READ as default access, where administrators can log in from the terminal but cannot manipulate the database from it, issue the following command:

   ```bash
eTrustAC> newres TERMINAL tty13 defacc(read)
   ```

2. To grant user ADMIN1 permission to manipulate the database from the new terminal (that is, grant WRITE permission as well as READ permission), issue the following command:

   ```bash
eTrustAC> authorize TERMINAL tty13 uid(ADMIN1) access(r,w)
   ```

**UNIX Environment**

For managing users and groups in UNIX, users in eTrust AC with global or group authorization attributes have the same privileges and limits for UNIX as they do for eTrust AC.

If you use selang while the seosd daemon is not running (for example, at installation time), you must follow these rules:

- You must include the -l option in the selang command.
- The user of selang must be root. (This exclusive root privilege complies with regular UNIX restrictions.)
Chapter 14: Improving Performance

This section contains the following topics:

- Using Global Access Check (see page 210)
- Using the Resource Cache (see page 214)
- Using the Network Cache (see page 215)
- Using the Real Path Cache (see page 216)
- Using Fork Synchronization (see page 216)
- Using High Priority (see page 216)
- Bypassing the Process File System (see page 217)
- Bypassing Real Paths (see page 217)
- Bypassing Trusted Process Authorization (see page 218)
- Bypassing Ports for Network Activity (see page 218)
- Reducing Audit and Trace Loads (see page 218)
- Reducing Database Loads (see page 219)
- Improving PMDB Updates (see page 219)
- Improving Watchdog Performance (see page 220)
- Improving Class Parameters (see page 220)
- Resolving Names (see page 221)
**Using Global Access Check**

The Global Access Check feature (GAC) lets you access protected, frequently opened files-whose access rules are unlikely to change-much faster than otherwise possible.

GAC allows an eTrust AC administrator to cache rules for read, write, chown, chmod, rename, unlink, utimes, chattr, link, chdir, create, and all, so that appropriate access to files is granted without passing control to seosd. The default is all. Execute requests, however, are not eligible for GAC because they could pose a security loophole.

Without GAC, eTrust AC runs thorough security checks whenever a user or program attempts to access protected files. Frequently accessed files need repeated in-depth checks to confirm access permissions.

GAC allows an administrator for eTrust AC to take for granted that certain frequently accessed protected files require shorter security checks. An administrator for eTrust AC can select files suitable for a shorter check. Before eTrust AC allows a shorter security check, the file must first undergo a full security check based on the set rule. The rule itself consists of a generic file name and a list of accesses. Rules are cached according to users.

Selecting certain files for a shorter check is reliable because, with the GAC feature in place, if a change is actually made to rules regarding the protected files, the shorter security check table is flushed, and an initial full security check is instituted.

**Note:** GAC restrictions (see page 213) mean that this feature works for every user except root.
Using Global Access Check

How Does GAC Work?

eTrust AC monitors access to specified files and builds a table of permitted accesses during execution time. These are the files you specify in advance in order to set up GAC rules (see page 212).

Whenever eTrust AC concludes that a user should be granted a certain level of access to a certain file, it checks whether the following two additional conditions are met:

- The granted access is unconditional (that is, not dependent on time, day, program from which executed, or other like conditions).
- The file matches one of its preselected sets of file masks.

Note: File rules define permissions for access to files.

If these conditions are met, eTrust AC generates a UID-file rule-access triplet and stores it in a table composed of such triplets. This table is examined before any database access rule interpretation takes place. Whenever a user attempts to access a file, this table is consulted as a filtering mechanism.

The table is best described as a do-not-call-me table because it contains a list of file masks that, once recognized, no longer need to undergo access permission checks. It is also described as an always-grant table because access is always granted to files specified within its list of file masks.

Whenever a user attempts to access a file, the table is consulted. If the file matches one of the triplets found in the table, the appropriate access is granted without passing control to seosd. This bypasses the access rules analysis. Subsequently, all access to files that match this pattern is granted, based on the triplet stored in the table, without consulting the access rule database.

Whenever a new access rule is added to the database, the entire table is flushed, and the learning process starts from the beginning.

Implementing GAC

To set up GAC, you must choose masks for sets of files that are accessed often, set up a GAC file containing these file masks, and then start the caching process.
Setting Up GAC Rules

**Note:** File rules in the database are created using the class FILE parameter and file masks. Rules apply to all files matching the file masks. FILE access types include: all, chdir, control, create, delete, execute, none, read, rename, sec, update, utime, write.

From the file rules defined in the database, choose the file masks that you want to cache. Enter a list of file masks into the `eTrustACDir/etc/GAC.init` file (where `eTrustACDir` is the installation directory for eTrust AC, by default `/opt/CA/eTrustAccessControl`), in exactly the same form as they appear in the database.

Each such mask should be specified on a separate line. For example, if the database contains a file mask for `/tmp/mydir/*` and you want it to be cached, add the following line to the `eTrustACDir/etc/GAC.init` file:

```
/tmp/mydir/*
```

**Note:** Specific file names cannot be specified in the GAC.init file. Only file masks are used.

Starting GAC

To turn your current version of eTrust AC into a GAC compatible version, prepare the file `eTrustACDir/etc/GAC.init` with the file masks that are eligible for caching. Only file masks can be used.

An example is a file named GAC.init in `eTrustACDir/etc/` with only one line:

```
/IBBS/REL63/*
```
GAC Restrictions

GAC implementation has proved to be very efficient, especially in cases where there are hundreds of file accesses in a second, but it has the following restrictions:

- By default, GAC rules are not applicable for the root user (usually ADMIN). To make the rules applicable to root, set the following token in the [SEOS_syscall] section of the seos.ini file:

  GAC_root=1

  The default value of the token is 0. To restore the default, set the token to 0, or remove the token.

- You must not include a file rule that is protected conditionally (for example with day or time restrictions, program pathing, and so on) in the table. If you do specify such a file rule in the GAC.init file, the day or time restrictions and other restrictions no longer apply.

- A file rule that has audit(ALL) or audit(success) attributes must not be included in the GAC.init file. If such file rule is specified in the GAC.init file, audits of successful accesses are not recorded.

- The filtering process uses the real (current) UID (that is, the UID that is associated with the process at the time of execution). This provides a loophole to the eTrust AC tracking of the original UID (the one with which the user has originally logged in) and not the current UID. (eTrust AC implements tracking of UID usage to provide the security of more accountability.)

  Let us examine an example of how someone might try to take advantage of this loophole. User Tony is not authorized to access the file Accounts/tmp. So Tony surrogates (through /bin/su) to user Sandra, who is authorized to access Accounts/tmp. If Sandra has already accessed the Accounts/tmp file, the file appears in the do-not-call-me table with her UID. Tony, using Sandra’s UID, is then permitted to access the file. This is because the kernel code does not maintain the history of UIDs.

  However, if Sandra has not previously accessed the file, the access permissions are checked in the regular manner using seosd, and Tony is denied access to the file. To close this loophole, the ADMIN user must protect the SURROGATE objects in the database. For this example, the ADMIN could add the following rule to the database:

  eTrustAC> newres SURROGATE USER.Sandra default(N) owner(nobody)

  This command ensures that Tony cannot use the su command to gain Sandra’s access privileges.

- The caching system does not have any impact if the accessor is root. The reason is that no access is granted to root without consulting the database.
Troubleshooting GAC

You can test GAC as follows to see if it is working:

1. Enable the trace (secons -t+).
2. Access a file that corresponds to one of the file masks specified in GAC.init. The first access should be reported in the trace.
3. Try to access the file again. The second file access should not be recorded in the trace.

If it is, GAC is not working. Check the GAC.init to see that it contains the correct format.

Using the Resource Cache

Another performance improvement tool that eTrust AC offers is resource caching (file cache).

The cache "remembers" the previous answer to an authorization request (permit or deny) for resources in the FILE class. The result is saved with the file name, user name, and authorization response (access mode, program name, and result). When an identical authorization is requested, the request is answered with the last response that was stored in the cache memory tables. This saves time because eTrust AC does not have to reevaluate the request; eTrust AC can return the answer immediately. When rules are changed, the cache is automatically and immediately synchronized.

The cache is a runtime table. An administrator can configure it in two ways:

- Set initialization parameters in the seos.ini file.
- Switch caching to ON or OFF and change parameters at runtime.

The security administrator can define table size, intervals between cleaning tables, and other internal table parameters with tokens in the seos.ini file.

A user with administrative privileges can switch cache tables ON or OFF, change cache parameters, and write cache tables to standard output.

**Note:** For more information about the secons utility, see the *Utilities Guide* or the [seosd] section of the seos.ini initialization file in the *Reference Guide*. 
### Using the Network Cache

The network or IP caching feature stores accepted, incoming TCP requests, so they are not sent to the database; instead, they are permitted automatically with the syscall function. This feature improves performance for hosts, which launch many incoming TCP connections.

To activate the IP caching feature, change the following tokens in the [seosd] section of the seos.ini file and restart eTrust AC:

- **network_cache_timeout**
  - Defines how often to clean the cache table. This token is important if you want to set time limits for the accept requests.

- **UseNetworkCache**
  - Set this token to yes to activate IP caching.

When caching is enabled, all accepted TCP connections are saved in the kernel table. The records consist of a peer IP address, peer port, and local port. Every new connection is searched in this cache. If a matching set of data for IP address, IP port, and local port is located, the connection is immediately permitted. The time to establish connection is reduced.

---

### Tuning Recommendations

Use these recommendations to improve performance even more:

- If one of the three tables (pools) has the maximum number of records and another table does not, expand the size of the full table.
  - **Note:** The three tables are: file, user, and authorization.
  - If a pool has low settings, increase them to expand the pool.

- Do not set the maximum size tokens unless you must. Larger tables take more time when scanning for records.
Using the Real Path Cache

File name resolution is a long process because eTrust AC uses information from file system. The kernel of eTrust AC translates node numbers to full file names when it intercepts appropriate events. Real path caching saves file names within an internal table.

To enable this feature, set the token cache_enabled to 1 in the [SEOS_syscall] section of the seos.ini file. File names are cached in the table with a data pair: inode number and device number.

Note: For more information about the seos.ini initialization file, see the Reference Guide.

Using Fork Synchronization

The fork synchronization token (synchronize_fork) in the [SEOS_syscall] section of the seos.ini file manages fork event behavior when new processes are created. Lowering the value of this token improves performance because fork events are frequent.

Note: For more information about seos.ini initialization file, see the Reference Guide.

Using High Priority

eTrust AC contains an option to set a real-time priority for the seosd daemon on some platforms. To activate this feature, set the rt_priority token in the [seosd] section of the seos.ini file to yes. Running in real time improves system performance.

Note: For more information about the seos.ini initialization file, see the Reference Guide.
Bypassing the Process File System

To reduce system load, you can specify whether eTrust AC should check file access when the file belongs to a process file system (/proc).

To activate this feature, use the proc_bypass token in the [SEOS_syscall] section of the seos.ini file. The token stores access information to be bypassed whenever eTrust AC must access the process file system.

**Note:** For more information about seos.ini file tokens, see the *Reference Guide*.

You can set this token as a sum of accesses. Access values are as follows:
- 1-read
- 2-write
- 4-chown
- 8-chmod
- 16-rename
- 32-unlink
- 64-utimes
- 128-chattr
- 256-link
- 512-chdir
- 1024-create

For example, `proc_bypass=513` specifies that all read(1) and chdir (512) access attempts should not be verified (1+512=513).

### Bypassing Real Paths

Searching for files with absolute file paths (instead of relative paths) creates heavier system loads; bypassing this search accelerates file events.

To activate this bypass, set the bypass_realpath token to 1 in the [SEOS_syscall] section of the seos.ini file. If you enable this token, eTrust AC does not obtain real file names, which, for example, could be a symbolic link.

**Note:** For more information about seos.ini file tokens, see the *Reference Guide*.

**Important!** This feature should be used with extreme care because it impacts security-generic rules do not work when files are accessed with a relative path.
Bypassing Trusted Process Authorization

eTrust AC allows you to define programs as trusted. eTrust AC stores the trusted programs and their children programs in a table. All events (inbound and outbound) related to trusted processes (and their corresponding ports) are permitted without authorization as part of a full network bypass.

To specify these programs, use the SPECIALPGM class:
- To bypass file and network events for the specified program, use the property SPECIALPGMCTYPE with values pbf and pbn.
- To bypass setuid and setgid events for a specified program, use the property SPECIALPGMCTYPE with the value surrogate.

Bypassing Ports for Network Activity

eTrust AC allows you to specify TCP/IP ports for exemption from seosd events. These ports are bypassed during eTrust AC network events.

To use this bypass, you must define the following token in the [seosd] section of the seos.ini file:
- bypass_TCPIP

For example, if you set the bypass_TCPIP token to 23 (the telnet port), the seos trace file no longer logs INET when you telnet to that workstation.

Reducing Audit and Trace Loads

eTrust AC uses a file system to keep audit data and trace data. Most processes in the system could be blocked while eTrust AC writes to this file system. To reduce access time to the file system, do the following:
- Set the audit mode only for resources and accesses you need.
- Open the trace only when you need to.
- Store audit file, trace file, and eTrust AC database files on the fastest available file system.
- Store the lookaside database directory on a fast file system.
Reducing Database Loads

How you define rules to the database affects system performance:

- Generic rules for commonly used directories produce many verifications, resulting in a greater system load.
  
  For example, protecting `/usr/lib/*` causes every action in the system to be checked by eTrust AC. To improve performance, avoid using generic rules for frequently used files.

- Deep hierarchies of users and resources require system loads to obtain and check all dependencies. To improve performance, avoid deep hierarchies in the database.

Improving PMDB Updates

Policy Models send commands to their subscribers one by one in a loop. To control the maximum number of commands that the Policy Models sends to each subscriber during each loop, use the `updates_in_chunk` token, which is described in the `[pmd]` section of the appendix “The pmd.ini File.”

If you increase the value of this token, the Policy Model uses fewer cycles to send commands. After each loop, the Policy Model checks for new requests. If the token is set higher, the Policy Model does not check for new requests as often.

For example, when you add a new subscriber to the Policy Model (using the `sepmdb -n` option), increase the token value because other subscribers have already received the commands that the Policy Model is sending. The Policy Model spends less time sending commands to the other subscribers and spends more time sending commands to the new subscriber, shortening the time it takes to add the subscriber.

**Note:** Do not set this token value to more than 100.
Improving Watchdog Performance

To reduce system load, set the Watchdog daemon (seoswd) to periodically scan secured files instead of constantly scanning. You can specify the Watchdog to scan at times when the system is less loaded.

To activate this feature, use the IgnoreScanInterval token in the [seoswd] section of the seos.ini file, and set additional tokens for intervals and start times.

**Note:** For more information about these tokens, see the seos.ini initialization file in the *Reference Guide*.

Improving Class Parameters

Use the class activation and class authorization features for eTrust AC to improve performance further.

**Class Activation**

eTrust AC stores information about whether a CLASS is active or inactive in the database. When eTrust AC starts, it passes a list of active classes to SEOS_syscall, so eTrust AC does not have to constantly intercept these classes. The only time eTrust AC intercepts a class is when a user changes the activity status of a class. If a class is inactive, access to the resource is not intercepted.

You can use the inactive class bypass with the following classes: FILE, HOST, TCP, CONNECT, and PROCESS.

**Class Authorization**

The resource class SEOS controls the behavior of the eTrust AC authorization system. The SEOS class has modifiable properties that specify whether a class is active. You can disable unused classes (using the setoptions command) to reduce authorization time.
Resolving Names

Several tokens in the [seosd] section of the seos.ini file (including GroupidResolution, HostResolution, ServiceResolution, and UseridResolution) control how eTrust AC performs name resolution. Setting these tokens appropriately improves performance.

Alternatively, you can create a lookaside database (instead of using system name resolution). To improve performance, select the lookaside database option. Tokens for this feature include the lookaside_path and use_lookaside.

**Note:** For more information about these tokens, see the seos.ini initialization file in the *Reference Guide*. 
Chapter 15: Using UNIX Exits

This section contains the following topics:

- UNIX Exits (see page 223)
- User or Group Record Update Exits (see page 224)
- eTrust AC Kernel Loader Exits (see page 229)

UNIX Exits

A UNIX exit is a specified program—a shell script or an executable—that runs automatically as a result of another defined eTrust AC activity taking place. eTrust AC supports UNIX exits when loading or unloading the eTrust AC kernel module, or when issuing specific selang commands. For example, you can run an initialization process for each new user that you add.

A UNIX exit can run on one or more of the following occasions:

- As a pre-update exit, before each selang command that updates a user or group record
- As a post-update exit, after each selang command that updates a user or group record
- As a pre-load exit, before SEOS_load loads the eTrust AC kernel
- As a post-load exit, after SEOS_load loads the eTrust AC kernel
- As a pre-unload exit, before SEOS_load -u unloads the eTrust AC kernel
- As a post-unload exit, after SEOS_load -u unloads the eTrust AC kernel
User or Group Record Update Exits

UNIX exits are called whenever a selang command that updates user or group records is executed in the UNIX environment, regardless of whether the tool is a command-line interface (selang) or GUI (such as Security Administrator).

The term update refers to creating, modifying, or deleting a user or group record. Querying a user or a group does not cause any UNIX exit to run. These are the commands that can cause a UNIX exit to run:

- newusr
- newgrp
- chusr
- chgrp
- editusr
- editgrp
- rmusr
- rmgrp

From the UNIX point of view, each exit processes runs as a root process, but from the eTrust AC point of view, it runs under the agent identity _seagent.

How the Provided selang Exit Script Works

eTrust AC provides a script that you can use as a master script to call other programs according to the nature and status of the current selang command. The exit script that is supplied as part of eTrust AC is eTrustACDir/exits/lang_exit.sh (where eTrustACDir is the eTrust AC installation directory.) Here is how it works:

1. eTrust AC automatically gives values to three parameters of the script.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS</td>
<td>USER</td>
</tr>
<tr>
<td>ACTION</td>
<td>CREATE</td>
</tr>
<tr>
<td>STAGE</td>
<td>PRE</td>
</tr>
</tbody>
</table>
The parameters indicate whether eTrust AC is dealing with a user or a group; whether the user or group is being created, deleted, or modified; and whether the selang command is about to be executed (PRE) or has just been executed (POST).

The script can pass the parameter values to programs that it calls.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXEC_RV</td>
<td>Receives the return value of a UNIX command that you use to determine whether the exit command succeeded or failed. For PRE commands, the value is always zero. For POST commands, you can use the value to decide whether to run or skip an exit. For an example of how to use this parameter, locate eTrustACDir/samples/exits_src.</td>
</tr>
</tbody>
</table>

2. Using the CLASS and STAGE parameters, eTrust AC looks for programs in the appropriate directory:

   eTrustACDir/exits/USER_PRE/
   eTrustACDir/exits/USER_POST/
   eTrustACDir/exits/GROUP_PRE/
   eTrustACDir/exits/GROUP_POST/

3. In the appropriate directory, eTrust AC selects all the programs that have file names that begin with a capital S, refer to the appropriate action, and have the following format:

   Snnaction_string

   Where nn is a two-digit decimal number defining the order of the program in the execution sequence, action is one of CREATE, MODIFY, or DELETE, and string is a descriptive string.

4. eTrust AC runs all the appropriate programs according to the numerical order of the second and third characters of their names.

**Example: UNIX Exit Script**

You are going to delete a user, and the directory eTrustACDir/exits/USER_PRE/ includes the following files:

- S10CREATE_precustom.sh
- S10DELETE_precustom.sh
- S99DELETE_prermusrdir.sh

When you issue the command to delete the user, the first program is not run because you are deleting and not creating a user. The second and then the third programs are run in that order based on the two digits after the initial S.
Arguments You Can Pass to selang Exits

When writing exits you can take advantage of the three parameters mentioned previously (CLASS, ACTION, and STAGE), and all the standard eTrust AC data such as names and permissions. You can also designate extra user or group data especially for use by the exit scripts. To store such additional data for a user or group, define it within single quotes as the value of the user's or group's UNIX APPL property in a newusr, chusr, newgrp, or chgrp command. For example:

```plaintext
eTrustAC> chusr JONESY unix APPL('HIRED=MAY93,CLEARANCE=2')
```

Your exit program must be able to handle whatever is between the single quotes.

**Note:** If you are using Security Administrator, then you have some additional capabilities and graphical tools. For more information about these options, see the *User Guide*. 
Specify selang Exit Programs to Run

To tell eTrust AC which exit programs to run, modify the [lang] section of the seos.ini file. eTrust AC provides the lang_exit.sh script for pre-user, post-user, pre-group, and post-group exits. You can also specify no exit or create your own exit.

To specify your own selang exits
- Set any or all of the following seos.ini tokens as required:

  **pre_group_exit**
  Specifies the path of the exit program to call *before* a group update command is executed in the UNIX environment.

  **pre_user_exit**
  Specifies the path of the exit program to call *before* a user update command is executed in the UNIX environment.

  **post_group_exit**
  Specifies the path of the exit program to call *after* a group update command is executed in the UNIX environment.

  **post_user_exit**
  Specifies the path of the exit program to call *after* a user update command is executed in the UNIX environment.

**Note:** An exit is called only if its full *path name* appears as the value of an exit token.

**Example: Specify selang exists**

In the following example, the seos.ini file tokens are set so that the program groupcheck runs before group operations, the program flag_exceptions runs after group operations, the program lang_exit.sh runs after user operations, and no exit program runs before user operations. The seos.ini file tokens are set as follows:

```
[lang]
pre_group_exit = /opt/CA/eTrustAccessControl/exits/groupcheck
post_group_exit = /opt/CA/eTrustAccessControl/exits/flag_exceptions
post_user_exit = /opt/CA/eTrustAccessControl/exits/lang_exit.sh
```
Time Out and Other Failures

Exit execution times out after 15 seconds, unless the exit_timeout variable in the seos.ini file specifies otherwise. A nonzero return value indicates failure.

- If a pre-update exit times out or returns a return code of greater than or equal to 16, then eTrust AC kills the exit process, displays an error message, and aborts execution of the eTrust AC update command. Any other positive return code does not abort the execution of the command.
- If a post-update exit times out or returns a nonzero value, then eTrust AC kills the exit process and displays an error message. Having already been executed, the eTrust AC update command remains in force.

selang Exit Samples

By examining the scripts in the following directories, you can familiarize yourself with recommended scriptwriting techniques.

- eTrustACDir/samples/exits-src
- eTrustACDir/samples/sample_exits

eTrust AC Kernel Loader Exits

UNIX exits are called whenever the eTrust AC kernel is being loaded or unloaded (SEOS_load). This lets you define how you want to handle operating system and third-party programs when loading or unloading the eTrust AC kernel. For example, you can use kernel-unloading UNIX exits to automatically stop, and later restart, processes that prevent eTrust AC from unloading when running SEOS_load -u.

For some operating systems, eTrust AC comes with some kernel load exits, kernel unload exits, or both out of the box.

Note: For more information about identifying processes that prevent eTrust AC kernel from unloading, see the secons utility in the Utilities Guide.
**How the Kernel Loading Exits Work**

To let you control operating system and third-party processes, eTrust AC lets you automatically make calls to UNIX exits when loading the eTrust AC kernel extension.

When you run `SEOS_load`, eTrust AC performs the following actions:

1. Looks for programs in the following directory:
   
   ```
   eTrustACDir/ exits/LOAD
   ```

2. Selects all the programs that have file names of the following format:

   ```
   SEOS_load_string.always
   ```

   Where `string` can be any descriptive strings.

3. Executes, in lexicographical order, each file it found in the directory `eTrustACDir/ exits/LOAD`:

   ```
   SEOS_load_string.always -pre
   ```

   Each file is executed with the `-pre` parameter so that you can write your exits to detect the parameter and perform the actions required before the kernel is loaded.

   **Note:** If the exit returns a nonzero value, eTrust AC kills the exit process, displays an error message, and aborts the kernel loading.

4. Loads the kernel (SEOS_syscall).

5. Executes, in lexicographical order, each file it found in the directory `eTrustACDir/ exits/LOAD`:

   ```
   SEOS_load_string.always -post
   ```

   Each file is executed with the `-post` parameter so that you can write your exits to detect the parameter and perform the actions required after the kernel is loaded.

   **Note:** If the exit returns a nonzero value, eTrust AC kills the exit process and displays an error message. Having already been loaded, the eTrust AC kernel remains loaded.
How the Kernel Unloading Exits Work

To let you control operating system and third-party processes, eTrust AC lets you automatically make calls to UNIX exits when unloading the eTrust AC kernel extension.

When you run `SEOS_load -u`, eTrust AC performs the following actions:

1. Looks for programs in the following directory:
   ```
etTrustACDir/exits/LOAD
```
2. Selects all the programs that have file names of the following format:
   ```
SEOS_unload_string.always
```
   Where `string` can be any descriptive strings.
3. Executes, in lexicographical order, each file it found in the directory `eTrustACDir/exits/LOAD`:
   ```
SEOS_load_string.always -pre
```
   Each file is executed with the `-pre` parameter so that you can write your exits to detect the parameter and perform the actions required before the kernel is unloaded.

   **Note:** If the exit returns a nonzero value, eTrust AC kills the exit process, displays an error message, and aborts the kernel unloading.

4. Tries to unload the kernel.

   If the kernel *does not* unload:
   a. Selects all the programs that have file names of the following format:
      ```
SEOS_unload_string.opt
```
   b. Executes, in lexicographical order, each file it found in the directory `eTrustACDir/exits/LOAD`:
      ```
SEOS_unload_string.opt -pre
```
      Each file is executed with the `-pre` parameter so that you can write your conditional exits to detect the parameter and perform the additional optional actions required before the kernel is unloaded.

      **Note:** If the exit returns a nonzero value, eTrust AC kills the exit process, displays an error message, and aborts the kernel unloading.

   c. Unloads the kernel.
   d. Executes, in lexicographical order, each file it found in the directory `eTrustACDir/exits/LOAD`:
      ```
SEOS_unload_string.opt -post
```
      Each file is executed with the `-post` parameter so that you can write your conditional exits to detect the parameter and perform the additional optional actions required before the kernel is unloaded.
Note: If the exit returns a nonzero value, eTrust AC kills the exit process and displays an error message. Having already been unloaded, the eTrust AC kernel remains unloaded.

5. Executes, in lexicographical order, each file it found in the directory eTrustACDir/exits/LOAD:

SE05_unload_string.always -post

Each file is executed with the -post parameter so that you can write your exits to detect the parameter and perform the actions required after the kernel is loaded.

Note: If the exit returns a nonzero value, eTrust AC kills the exit process and displays an error message. Having already been unloaded, the eTrust AC kernel remains not loaded.
Chapter 16: Interacting with LDAP

This section contains the following topics:

- Transferring User Names (see page 233)
- ldap2seos (see page 234)
- seos2ldap (see page 236)
- S50CREATE_u_LdapE (see page 238)

Transferring User Names

If you are using both eTrust AC and LDAP, you can transfer user names between them using scripts of your own design; three sample scripts are provided.

Two of the provided scripts-ldap2seos and seos2ldap-export whole sets of users from eTrust AC to an LDAP server and imports them from an LDAP server to eTrust AC.

A third sample script, S50CREATE_u_LdapE.sh, automatically transfers new UNIX user names from eTrust AC to LDAP as they are created.

The sample scripts require access to a TCL shell environment; they use the Language Client API (LCA) library extension, tcllca.so.

**Note:** For more information about LCA and the TCL extension, see the Language Client API and the appendix the LCA Extension respectively in the SDK Guide.

If you do not have TCL, consult the FAQ posted monthly to comp.lang.t_c_l by Larry Virden, which is available on the MIT web site and the Terafirm website.

You can also refer to the Sun web site for TCL news, documentation, and resources.
ldap2seos extracts users from an LDAP database located at the server host and adds them to the eTrust AC database.

The ldap2seos utility extracts information from an LDAP server about the defined users. The extracted information is automatically used to execute selang commands to add the users to the database. The generated commands are also printed to the standard output and saved automatically to the file named /tmp/ldap2seos.tcl.log.

This utility requires access to a TCL shell environment. The ldap2seos script assumes that the TCL shell path is /usr/local/bin/tclsh. If the TCL shell is placed elsewhere, just change the first line in the script.

For the utility to work correctly, eTrust AC must be running. The utility updates the database, so it must be run by an ADMIN. This user must also be authorized in the LDAP database settings to make the search query.

This script has the following format:

ldap2seos [options]

-accfld account-field
  Defines the LDAP field name containing the user ID for eTrust AC.
  If the UNIX user ID is in the LDAP userid field, this option is unnecessary.
  If the UNIX user ID is assigned to an LDAP field other than the userid field, specify the LDAP field as account-field and the LDAP userid field will be ignored.
  **Note:** If the script cannot find the userid, users are not uploaded to the eTrust AC database.

-b base-entry
  Defines the base entry, in the LDAP database, from which the users are taken. The entry must be valid inside the LDAP database. If the base entry is omitted, LDAP uses the default base entry to provide the users.

-d dn
  Defines an entry name to be used with the -w switch to authenticate to LDAP as another user; mostly needed to log into LDAP as admin user.

-f filename
  Defines a file to which data retrieved from the LDAP server may be temporarily stored.

-h
  Specifies a request for a help screen. The screen contains a listing and explanation of ldap2seos usage and options.
-h ldap-host
Defines the name of the host where the LDAP database is located. The default is the local host.

-l ldap-dir
Defines the directory containing the line command utilities that are assumed to be in the bin subdirectory. The default is /usr/local/ldap.

-p port
Defines the port LDAP uses for connections. The default is port 389.

-u
Identical to -h, requests a help screen. The screen contains a listing and explanation of ldap2seos usage and options.

-w bindpasswd
Defines the user password. To be used with the -d option where authentication is needed to access the LDAP database.

**Example: Extract User Information**

The following command extracts information about users from the LDAP database at host myhost.mysite.com and tries to add them to the eTrust AC database.

eTrustAc> ldap2seos -h myhost.mysite.com
seos2ldap exports eTrust AC users from the database to an LDAP database located at a server host. It extracts appropriate information about users from the eTrust AC database. It then transmits the information to the selected server's LDAP database. The extracted information is used to generate an LDIF file. Specified users are added to the LDAP database. The responses are saved automatically to the file named /tmp/seos2ldap.tcl.log.

As mentioned in the introduction to this chapter, this utility requires access to a TCL shell environment. ldap2seos assumes that the TCL shell path is /usr/local/bin/tclsh. If the TCL shell is placed elsewhere, just change the first line in the script.

For the utility to work correctly, eTrust AC must be up and running. The utility reads from the database, so it must be run by an ADMIN. This user must also be authorized in the LDAP database settings to make changes.

The entry schema, if you elect to use one, for the LDAP database should look like the schema for the Netscape server. If you have changed the Netscape schema, or are using another type of LDAP server, you may need to edit the seos2ldap sample script accordingly.

If an eTrust AC database user already appears in the LDAP database, the user is not added. An error message is produced but the export process continues.

This script has the following format:

```
seos2ldap [options]
```

- **-b base entry**
  
  Defines the base entry, in the LDAP database, that stores user information. The entry must be valid inside the LDAP database. If the base entry is omitted, LDAP prompts the user to provide it.

- **-d dn**
  
  Defines an entry name to be used with the -w switch to authenticate to LDAP as another user. This option is needed to log into LDAP as an admin user.

- **-f filename**
  
  Defines a file to which data retrieved from the LDAP server may be temporarily stored.

- **-h**
  
  Specifies a request for a help screen. The screen contains a listing and explanation of ldap2seos usage and options.

- **-h ldap-host**
Defines the name of the host where the LDAP database is located. The default is the local host.

-\texttt{ldap-dir}

Defines the directory containing the line command utilities that are assumed to be in the bin subdirectory. The default is /usr/local/ldap.

-\texttt{noprompt}

Cancels base entry prompt. If you did not use the \texttt{-b base-entry} flag to specify the base LDAP entry, by default seos2ldap prompts for a base entry. This flag suppresses the prompt.

-\texttt{port}

Defines the port LDAP uses for connections. The default is port 389.

-\texttt{u}

Identical to \texttt{-h}, requests a help screen. The screen contains a listing and explanation of ldap2seos usage and options.

-\texttt{w bindpasswd}

Defines the user password. Use this with the \texttt{-d} option where authentication is needed to access the LDAP database.

**Example: Export User Information**

The following command extracts information about users from the eTrust AC database and creates an LDIF file named SeOS_user_dump. The command adds records to the LDAP database at host myhost.mysite.com. You can edit the LDIF file later and update LDAP manually.

```
eTrustAc> seos2ldap -h myhost.mysite.com
```
S50CREATE_u_LdapE

S50CREATE_u_LdapE.sh uploads new UNIX users to LDAP as they are created.

eTrust AC supplies a sample shell script to import new UNIX users automatically to an LDAP server. The script you need can vary from the sample.

To employ the sample shell script, assuming that you are already using the provided exit script (see page 224), do the following:

1. Copy the S50CREATE_u_LdapE.sh file to the directory eTrustACDir/exits/USER_POST (where eTrustACDir is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl). In this directory, the script becomes a post-user exit.

2. In the seos.ini file in the [ldap], set the base_entry token to the LDAP base entry.
   For example, for an organization named ServerWorld, located in Canada, the base entry might be: o=ServerWorld, c=CA.

3. In the same section, set the host name to the host name of the LDAP server. Set the path to the LDAP base directory. (The sample script looks for the line command utilities in the bin directory under that directory.)

Common Names (cn) are derived from the user's full name. If the eTrust AC database contains, for example, only the user name and surname, these will comprise the Common Name. You are essentially locked into the Common Name, so we recommend that you do not base it on a user name.

Each user subsequently added to UNIX with selang is automatically uploaded to the LDAP server. If the user already exists in LDAP, an error message results.

When you add users with this script, the relevant LDAP replies and warnings, if any exist, are collected in the /tmp/add_User2Ldap.tcl.log file. You can examine this file, using vi or any other standard UNIX editor, to check for errors. The file is overwritten with the new set of replies and warnings each time you add new users.
Chapter 17: Unicenter Security Migration and Integration

This section contains the following topics:
- Installing Unicenter Security Integration Tools (see page 239)
- Unicenter Security Integration Features (see page 239)
- Unicenter Security Data Migration Features (see page 244)
- Unicenter TNG Calendar (see page 249)
- Certification with Unicenter TNG and Unicenter NSM (see page 251)
- Audit Events Integration (see page 251)

Installing Unicenter Security Integration Tools

eTrust AC is fully integrated into the Unicenter Enterprise Management environment.

**Important!** To have Unicenter TNG integration with eTrust AC, you must have Unicenter TNG installed on the same machine as eTrust AC.

**Note:** For complete installation instructions, see the *Implementation Guide*.

Unicenter Security Integration Features

The following sections describe how eTrust AC integrates with Unicenter TNG.
SSF/EMSec API Support

The security APIs on UNIX all channel into a message queue. A new utility now processes the API requests sent through the message queue and routes these reformatted and rerouted requests to eTrust AC. The utility then translates the return codes of eTrust AC to Unicenter TNG equivalents. This approach protects the integrity of existing applications that are currently using the EMSec API.

In UNIX the utility is called `sessfgate`. The gateway is active after the Unicenter Integration setup procedure completes. In fact, if Unicenter NSM Event Notification Facility (ENF) is running, the gateway is automatically started or stopped whenever the eTrust AC services are started (using seload) or stopped. If ENF is not running, the eTrust AC seload program will not start the sessfgate daemon. The Unicenter Integration setup installs the sessfgate program in the `eTrustACDir/tng/bin` directory. After Unicenter Security is shut down and eTrust AC is started, sessfgate can accept API requests instead of SSF.

Run sessfgate as follows:

```
# sessfgate [-i|-s|-l] -t
-1
     Specifies to start the gateway.
-s
     Specifies to stop the gateway.
-l
     Specifies the status.
-t
     Toggles the tracing file (log file = `eTrustACDir/log/sessftrace.log`).
```

**Note:** If you run seload before running Unicenter TNG, you must start sessfgate manually with the following command:

```
eTrustACDir/tng/bin/sessfgate -I
```

where `eTrustACDir` is the directory in which you installed eTrust AC.
Unicenter Security Integration Features

**eTrust AC to Unicenter Security Synchronization Utility**

Unicenter Security and eTrust AC manage the administration of your enterprise IT environment before total migration occurs. To reduce the complexity of using different product tools to perform administrative tasks, we are providing a synchronization daemon.

The new daemon is called seostngd. eTrust AC sends Policy Model Database (PMDB) updates through CA Common Communication Interface (CAICCI) to seostngd. The daemon listens for updates on CAICCI and then translates the messages into equivalent cautil commands to update the Unicenter Security database with this global data.

Current Unicenter TNG processing can still update other Unicenter TNG client installations. You must run seostngd on the same machine as the Unicenter Security database (normally referred to as a Unicenter TNG master machine.) eTrust AC should also be running on the same machine.

**Note:** The daemon should only be used during the migration process. Once the migration process has been completed, the daemon should not be used. This daemon is not a long-term solution for your IT environment.

The following figure demonstrates the function of seostngd:

![Diagram of seostngd function](image)
The major task of seostngd is to take changes you make on eTrust AC (using either a graphical interface or selang commands) and apply them to the Unicenter Security database. If the changes can be applied to the Unicenter Security database successfully, you should see the same behaviors as you did on eTrust AC.

For example, when you create a USER object with eTrust AC, you should see the same USER object being created in Unicenter TNG (as long as the required fields are present).

**Starting and Stopping the Subscription Daemon**

To start the subscriber daemon, enter:

```
eTrustAc> seostngd
```

To stop the daemon, enter one of the following:

```
eTrustAc> seostngd -shut
```

or

```
eTrustAc> seostngd -stop
```

**Important!** Do not use selang -c during migration if you are listing more than one command. Instead, use selang -f input_file_name.

**Notes:**

- In order to start the daemon, you must be user root in the Unicenter TNG SSF_AUTH user list.
- The subscriber daemon should be manipulated manually. For example, if you reboot the machine, restart this utility with the seostngd command because this is not controlled by the eTrust AC startup program.
- You cannot use this before performing the Unicenter TNG integration installation procedure. During the setup procedure, a configuration file, `eTrustACDir/data/tng/assettypes.txt`, is generated. This file is required to run this utility.
- Be sure that `$CAIGLBL0000/bin` is in the PATH environment, so you can run the Unicenter command line utility, cautil. To do this, run the script file (from ksh or sh #) `$CAIGLBL0000/scripts/envset`.
- The Unicenter Security daemon (sdm) must be running, otherwise, the Unicenter TNG subscriber daemon cannot apply changes to the Unicenter Security database.
SEOSTNGD Limitations

Different maximum field lengths between eTrust AC and Unicenter Security can cause truncation of data values. The following table lists the significant differences in supported field lengths.

<table>
<thead>
<tr>
<th></th>
<th>Unicenter TNG</th>
<th>eTrust AC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User ID</strong></td>
<td>20 characters</td>
<td>256 characters</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>8 characters</td>
<td>14 characters</td>
</tr>
<tr>
<td><strong>User group ID</strong></td>
<td>8 characters</td>
<td>254 characters</td>
</tr>
<tr>
<td><strong>Asset group ID</strong></td>
<td>8 characters</td>
<td>255 characters</td>
</tr>
</tbody>
</table>

- Unicenter TNG does not support renaming a user or asset object, so the seostngd daemon ignores eTrust AC `rename` commands for users and assets.
- Unicenter TNG user groups and asset groups must have at least one member. If no member is specified in the eTrust AC command for creating user groups or asset groups, the corresponding Unicenter TNG user group or asset group is not created until at least one member is added.
- If the last member is removed from an eTrust AC User group or asset group, that user group or asset group is removed from Unicenter TNG.
- Unicenter TNG assets require at least one defined accessor, so a new Unicenter TNG asset cannot be created until at least one eTrust AC "authorize" command is executed for the asset.
- eTrust AC removes any associated rules for an object when it is deleted. However, Unicenter Security does not.
- eTrust AC users have an ADMIN attribute that has a similar meaning as when a Unicenter TNG user is a member of the SSF_AUTH user list in the Security Options. However, Unicenter TNG does not provide any automatic way for manipulating remote Security Options, so manual modifications to SSF_AUTH user list are required.
- In order to force the creation of a new Unicenter TNG user, the new eTrust AC user must be created in the Native environment with a value supplied for the eTrust AC password field. Unicenter TNG has default password restrictions that require a minimum length of six characters-two alphabetic and one numeric.
- The eTrust AC `authorize` command supports the asterisk (*) as an accessor ID, but this is not supported in Unicenter TNG. The seostngd daemon ignores eTrust AC `authorize` commands like this.
Unicenter Security Data Migration Features

- The eTrust AC `authorize` command supports conditional access rules using the `via` parameter, but this is not supported in Unicenter TNG. The seostngd daemon ignores eTrust AC `authorize` commands like this.
- If the `access` parameter is not specified on an eTrust AC `authorize` command, the seostngd daemon grants READ permission for any UNIX-FILE or Unicenter TNG asset group, and grants all permissions for any other Unicenter TNG predefined asset type.

Unicenter Security Data Migration Features

The following sections describe how to migrate Unicenter Security data to eTrust AC.

Unicenter Security Options Migration

eTrust AC features a program called migopts that extracts selected Unicenter Security options and customizes the targeted eTrust AC database according to these options. To activate this feature, you must run the Unicenter Integration with Unicenter Security Data Migration setup procedure. This setup procedure automatically runs migopts.

*Important!* This feature is part of Unicenter Security Data Migration and is intended for users who use Unicenter Security only for its integration with the cautil command processor, Event Management, and Workload Management. Because of differences between Unicenter Security and eTrust AC architecture, Unicenter Security data migration is not intended for people who use Unicenter Security to protect their file systems.

*Note:* The following Unicenter Security options can be migrated completely into the eTrust AC environment.
- CREDAUTHEXIT
- DEFSESID
- PASSWORD_ALPHA
- PSWDVALEXIT
- PWDQUEUESIZE
- SSF_MAXPWDVIO
- SSF_MINPWDLEN
- SSF_NUMSUBP
- SSF_SECPWEXCL
- SSO_APPLNAME
Unicenter Security Data Migration Features

- USER_PWDCHANGE
- USER_PWDCHGMAXDAYS
- USER_PWDCHGMINDAYS

Additionally, exporttngdb migrates Unicenter TNG users who are members of the SSF_AUTH Unicenter Security Option into the eTrust AC environment by setting the Users' "admin" attribute before adding them to eTrust AC.

**Note:** The migopts utility is run by the migration scripts eTrustACDir/tng/bin/uni_migrate_master.sh and eTrustACDir/tng/bin/uni_migrate_node.sh. For more information, see the Utilities Guide.

Unicenter Security Database Migration

eTrust AC features a program called exporttngdb that extracts data from the Unicenter Security database and translates it into eTrust AC commands to populate the eTrust AC database. exporttngdb migrates the following:

- Unicenter Security Users
- Unicenter Security User Groups
- Unicenter Security Rules

**Notes:**

1. We do not recommend running Unicenter TNG login intercepts after running the Unicenter Integration and Migration Installation. Once the Unicenter Integration and Migration Installation is successfully completed, you should verify that Unicenter TNG login intercepts are disabled.

2. Unicenter TNG Data Scoping and Keyword Scoping rules (rules that target Unicenter TNG asset types with a -DT or -KW suffix) are not supported by the eTrust AC Migration process. Rules of this type are ignored during the migration process.

3. Unicenter Security rules that have been implemented against any of the following Unicenter Security asset types are obsolete because Unicenter Security is no longer used: CA-USER, CA-ACCESS, CA-USERGROUP, CA-ASSETGROUP, CA-ASSETTYPE, and CA-UPSNODE. Rules that target any of these asset types, or any of their derivatives, are ignored during the migration process.

4. The exporttngdb utility is run by the migration scripts eTrustACDir/tng/bin/uni_migrate_master.sh and eTrustACDir/tng/bin/uni_migrate_node.sh.

**Note:** For more information about the exporttngdb utility, see the Utilities Guide.
In order to activate exportntgdb, you must run the Unicenter Integration with Unicenter Security Data Migration setup procedure. This setup procedure automatically performs the Unicenter Security Data Migration process.

**Important!** This feature is part of Unicenter Security Data Migration and is intended for users who use Unicenter Security only for its integration with the cautil command processor, Event Management, and Workload Management. Due to the dramatic differences between Unicenter Security and eTrust AC architecture, Unicenter Security Data Migration is not intended for people who use Unicenter Security to protect their file systems.

**Note:** Creation and modification statistics of all Unicenter TNG objects are lost in the migration process.

Due to Unicenter TNG and eTrust AC product differences, the following attributes of Unicenter Security users cannot be migrated to eTrust AC:

**Statistics**

The following User statistics are not supported by eTrust AC:

- Last login statistics (date and time, node of last login)
- Password change statistics (date and time, node, user who changed last password, and expiration date of the password)
- Password violation statistics (date and time, node of last unsuccessful login, and number of unsuccessful logins since last successful login)
- Access violation statistics (date and time, node of last access violation, and number of access violations)
- Suspension statistics (date and time of suspension)

**PWDCHANGE VALUE (RANDOM)**

Random password generation

**UPSSTATGROUP**

UPS station group. It is not supported by eTrust AC.

**VIOLMODE**

Violation mode (FAIL, MONITOR, WARN, QUIET). eTrust AC supports FAIL mode only.

**VIOLACTION**

Violation action (CANUSER, CANU&LOG, CANU&LOG&SUS). eTrust AC supports CANUSER action only.

Due to Unicenter TNG and eTrust AC product differences, the following attributes of Unicenter Security rules cannot be migrated to eTrust AC:

**EXPIRES**

Rule expiration date is not supported by eTrust AC.
Unicenter TNG User Exit Support

To help current Unicenter TNG users who are migrating to eTrust AC, eTrust AC lets you run existing Unicenter Security user exits unchanged in the eTrust AC environment. You do not have to rewrite all user exits as part of the migration.

Using only the existing user exit interfaces in Unicenter Security and eTrust AC, each installed component is registered as a standard eTrust AC user exit, which then brings up the corresponding Unicenter Security exit.

In order to initiate this feature, you must run the Unicenter Integration with Unicenter Security Data Migration setup procedure. Once the setup procedure is completed, this functionality is active.

**Important!** This feature is part of Unicenter Security Data Migration and is intended for users who use Unicenter Security only for its integration with the cautil command processor, Event Management, and Workload Management. Due to the dramatic differences between Unicenter Security and eTrust AC architecture, Unicenter Security Data Migration is not intended for people who use Unicenter Security to protect their file systems.

**Note:** Because Unicenter TNG and eTrust AC use different architectures, only the exit points and data items that are comparable between Unicenter Security and eTrust AC are supported. The following Unicenter Security exit points are supported:

**EmSec_CredExit()**

The input to the Unicenter Credential Authentication exit, EmSec_CredExit(), is mapped by EMSECSIGNON. With eTrust AC, only the user and node members within this structure have meaningful data. The user member is set to the user name being authenticated, and the node member is set to the current local node name. All other members of the EMSECSIGNON structure are set to binary zeros. The other parameters, the detailed return code, and the message passed back from the Unicenter Resource Check Exit are ignored.

**EmSec_PwExit()**

The Password Validation exit, EmSec_PwExit(), is fully supported.

**Note:** The exits are contained in libemsec2.xx which is located at: /usr/local/Calib/ or eTrustACDir/Calib/ (where eTrustACDir is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl).

Once the Unicenter Integration setup completes, this functionality is active.
**Use a PMDB with Unicenter Security Objects**

eTrust AC PMDBs can be used with Unicenter TNG objects to create rules that secure Unicenter TNG objects from being manipulated by the various Unicenter TNG components (such as the cautil command processor, Event Management, and Workload Management). You must perform the integration manually.

**To use a PMDB for Unicenter TNG objects:**

1. Create the PMDB.

2. Migrate Unicenter Security options into the PMDB with the following command:
   
   \[ \text{migopts -d pmdb-name} \]
   
   where \( \text{pmdb-name} \) is the name of your PMDB.

   **Important!** This step is required only if you used Unicenter Security and ran the Unicenter Integration installation script for Security Data Migration (uni_migrate_master.sh and uni_migrate_node.sh). If you did not use Unicenter Security, then you never established any security options and there is nothing to migrate into your PMDB.

   **Note:** Migration (see page 244) and integration (see page 239) are two separate procedures.

3. Create classes for any user-defined Unicenter TNG asset types with the following command:
   
   \[ \text{defclass.sh pmdb-name} \]
   
   where \( \text{pmdb-name} \) is the name of your PMDB

   **Important!** This step is only required if you used Unicenter Security and created user-defined asset types. Unicenter TNG asset types are automatically defined in every new PMDB if you selected Unicenter Integration during the eTrust AC installation.
Unicenter TNG Calendar

Unicenter TNG provides a calendar facility, with which you can set time restrictions for users, groups, and resources. The calendar contains time intervals of 15 minutes that you can set to ON or OFF. A calendar time interval set to OFF prevents access to resource; a calendar time interval set to ON continues resource authorization.

In UNIX, an administrator can set calendar usage before security startup only.

To use Unicenter TNG calendars in eTrust AC, complete the following steps:

**Note:** Unicenter TNG must be installed on the local machine. eTrust AC uses local Unicenter TNG services to retrieve calendar settings.

1. Stop eTrust AC security. Enter:
   ```
   # secons -s
   ```
2. Set the TNG_calendars token in the [seauxd] section of seos.ini to yes.
   ```
   TNG_calendars=yes
   ```
3. Start eTrust AC security. Enter:
   ```
   # seosd
   ```
4. Check that the auxiliary daemon seauxd is running. Enter:
   ```
   eTrustAc> issec
   ```

You can also modify the following tokens in the seos.ini file:

**TNG_refresh_interval**

Specifies the time interval in minutes to refresh eTrust AC calendars. The default is 10 minutes.

**TNG_lib_path**

Specifies the full path to Unicenter TNG libraries. The default is /usr/local/Calib.

**TNG_cal_lib**

Specifies the name of the Unicenter TNG calendar library. The default is libcalendar.

To link an eTrust AC resource with the calendar, you must issue the following database commands:

**Note:** The issued calendar name must be identical to the case-sensitive Unicenter TNG calendar name.

```bash
eTrustAc> nr CALENDAR calendar_name
eTrustAc> nr file /tmp/test calendar (calendar_name) defaccess (a)
```
The Unicenter TNG calendar Access Control List (ACL) is an additional security constraint feature. The regular Unicenter TNG calendar property restricts the current resource according to the appropriate Unicenter TNG calendar status. The Unicenter TNG calendar ACL property restricts access for (or gives access to) specific users and groups for the current resource according to the Unicenter TNG calendar status.

Two types of ACL Unicenter TNG calendar properties are regular and restrictive:

- The regular calendar ACL property permits user or group access to the resource accordingly to ACL access.
- The restrictive (denied) calendar ACL property denies user or group access to the resource accordingly to ACL.

To add a user or group to the regular calendar ACL (CALACL), enter the following command in selang:

```
eTrustAc> auth resource_class_name object_name \ uid_or_gid_name calendar(calendar_name) access(access_value)
```

For example:

```
eTrustAc> auth file file1 uid(george) calendar(basecalendar) access(rw)
```

To add a user or group to the denied calendar ACL, enter the following command in selang:

```
eTrustAc> auth resource_class_name object_name uid_or_gid_name \ calendar(Unicenter_calendar_name) deniedaccess(access_value)
```

For example:

```
eTrustAc> auth file file2 uid(george) calendar(holidays) access(rw)
```

You can use both regular and restrictive properties for the same resource (such as calendar and uid). The following command adds a user named George with read access to the denied calendar ACL for file1.

```
eTrustAc> auth file file1 uid(george) calendar(holidays) deniedaccess(r)
```

To remove a user or group from a Unicenter TNG calendar ACL property, use auth-:

```
eTrustAc> auth- file file2 uid(george) calendar(holidays)
```

Use the Show Resource (sr) command to see all Unicenter TNG calendar ACLs assigned to a specific resource:

```
eTrustAc> sr file file1
```
Certification with Unicenter TNG and Unicenter NSM

The following features comply with Unicenter TNG 2.2 SP1, Unicenter TNG 2.4, and Unicenter NSM 3.0:

- Sending “events”
- Synchronizing mainframe passwords
- Using the Unicenter TNG calendar

Audit Events Integration

Integration with Unicenter TNG is set up at installation.

You can choose to send audit data to Unicenter TNG. Audit events that are passed to Unicenter TNG appear in the Console logs in the Unicenter Enterprise Management\Enterprise Managers\Windows NT\Event window.

<table>
<thead>
<tr>
<th>Audit Event</th>
<th>Display Color</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Blue</td>
<td>S</td>
</tr>
<tr>
<td>Denied</td>
<td>Orange</td>
<td>F</td>
</tr>
<tr>
<td>Fail</td>
<td>Orange</td>
<td>F</td>
</tr>
<tr>
<td>Warning</td>
<td>Blue</td>
<td>W</td>
</tr>
<tr>
<td>eTrust AC stopped (audit down)</td>
<td>Blue</td>
<td>I</td>
</tr>
<tr>
<td>eTrust AC started (audit start)</td>
<td>Blue</td>
<td>I</td>
</tr>
</tbody>
</table>
The second option permits launching eTrust AC from the Unicenter WorldView menu by pointing to the icon representing the TCP/IP Network in the Managed Objects window and selecting eTrust AC from the right-click menu.

eTrust AC also sends following information about events:
- Product name (eTrust Access Control + version number)
- User name
- Terminal name
- Class name
- Resource name
- Process name
- Event’s time
- Full audit message in the format of eTrust AC auditing

The fields User name, Terminal name, Class name, Resource name, and Process name are not always sent, depending on event type.
Appendix A: NIS Configuration

This section contains the following topics:

- Installation Notes (see page 253)
- Name Resolution (see page 254)
- Avoiding Deadlocks: The Lookaside Database (see page 256)
- Configuration Tokens: The seos.ini File (see page 259)

Installation Notes

**Note:** This section supplements material covered by the installation script. This appendix assumes you are familiar with Network Information Systems (NIS), Domain Name Services (DNS), and UNIX name resolution concepts.

During installations of eTrust AC, you can use one of two options to resolve user ID to user name, group ID to group name, host IP address to host name, and service port to service name:

- Use the system functions, which define a bypass for the net caching daemon on your system.
  - If you use Digital DEC UNIX and it is *not* an NIS server, the default uses the system functions for name resolution.
  - If you use Digital DEC UNIX and it *is* an NIS server, the installation prompts you to choose one of two options: use a lookaside database or use system functions, which define a bypass for the net caching daemon.

- Use a lookaside database, which is created by the sebuildla utility.
  - If you are using eTrust AC configured to run on an NIS server, use the lookaside database.
  - The installation default uses the lookaside database on the following platforms: HP-UX 11.0 and higher, Sun Solaris 2.6 and higher, IBM AIX 5.1L and higher, and all supported Linux platforms.

**Note:** On IBM AIX platforms, you must use the lookaside database; there is no option to use the system functions.
eTrust AC intercepts requests to access system resources and decides whether to permit or deny these requests. The decision is based on access rules and policies that are defined in the database. The interception of requests to access system resources takes place at the kernel level.

To control hosts, groups, users, and services, the kernel and the relevant system calls use codes or numbers (that is, IP addresses, group IDs, user IDs, and service numbers) instead of names. eTrust AC defines access rules based on names. eTrust AC translates names into codes recognizable by the kernel. This process is called name resolution.

On stand-alone stations, except for stations running Sun Solaris 2.5 or higher, name resolution is completed directly through the local user, group, and host files (/etc/passwd, /etc/group, and /etc/hosts). When eTrust AC needs to resolve a name, it simply calls a system function that in turn reads the relevant file.

On larger networks, however, this information is seldom stored locally. When you use NIS, DNS, or both, there are no local files that you can consult during name resolution. The information is requested and received from a server over the network.

**Name Resolution on an NIS/DNS Client**

eTrust AC performs name resolution on a client-only NIS or DNS station (which is not its own server) as follows:

1. eTrust AC generates a network request to connect to the relevant server.
2. The eTrust AC kernel extension intercepts the request.
3. The eTrust AC kernel extension permits the request because it knows that the request was made internally by the eTrust AC process.
4. A connection to the NIS or the DNS server is established and the information necessary for name resolution is retrieved.
5. Once the name is resolved, eTrust AC continues the process of deciding whether to permit or deny the original access request.

A standard eTrust AC configuration is sufficient for eTrust AC to easily handle name resolution on a client server.
**Name Resolution on a Server: Deadlock**

eTrust AC performs name resolution on a server that includes itself as a client as follows:

1. eTrust AC generates a network request to connect to the relevant server.
2. The kernel extension intercepts this request.
3. The kernel extension permits the request because it knows that the request was made internally by the eTrust AC process.
4. The NIS or DNS server (which is located on the same station) generates a request to accept the network connection.
5. The kernel extension intercepts this request.
6. The kernel extension knows that an eTrust AC process did not make this request. It places this request on the queue of requests awaiting seosd decision.
7. The seosd daemon is now caught in a deadlock. It is waiting for the reply necessary to complete name resolution, but the process that should provide this reply cannot proceed until seosd gives it permission to accept the network connection. The first request generates the second, and creates a deadlock.

**Name Resolution on Sun Solaris: Deadlock**

Name resolution on Sun Solaris entails accessing the *nscd* cache. The *nscd* is a process that provides a cache for the most common name service requests. *nscd* furnishes caching for the *passwd*, *group*, and *hosts* databases.

The cache is not permanent. It becomes invalid as changes are made to the *passwd*, *group*, and *hosts* databases, or as the time-to-live stamp expires.

The Sun Solaris setup can create a deadlock like the one described in the previous section. Here, the interaction between eTrust AC and the *nscd* process causes the deadlock.

1. During name resolution, eTrust AC accesses the *nscd* cache.
2. The *nscd* process can decide that the cache is too old. In this case, it attempts to refresh the information by accessing the *passwd*, *group*, and *hosts* databases (locally or on a server).
3. The request to access these databases is intercepted by the kernel extension. Since an eTrust AC process is not making the request, it is placed on a queue awaiting seosd decision. But no such decision is possible because seosd is still engaged in the previous request. The first request generates the second, and creates a deadlock.
Avoiding Deadlocks: The Lookaside Database

The setting of the under_NIS_server token in the seos.ini configuration file has a default setting of yes to avoid deadlocks. The token tells eTrust AC to use its own internal name resolution tables instead of NIS, DNS, or the nscl cache. Unless otherwise specified, these tables reside in memory.

eTrust AC internal name resolution is much faster than NIS name resolution and even faster than usin files; using eTrust AC internal name resolution improves performance even in an environment where there is no danger of deadlocks.

Storing Resolution Tables on Disk

eTrust AC name resolution tables are generated while eTrust AC is starting up. The tables should be maintained on disk, not in memory because storage in memory can lead to memory overload. Also, when the information is read into memory, it is static. Because of this, eTrust AC would not know of any changes made to user, group, or host information. The only way to update the tables in memory is to restart eTrust AC.

To keep data current, eTrust AC provides a lookaside database that makes sure internal name resolution tables are stored on disk. To implement the lookaside database you need to use seos.ini configuration tokens (see page 259).

Setting Up the Lookaside Database

The four tables in the lookaside database are userdb.la, groupdb.la, hostdb.la, and servdb.la. These four tables handle user, group, host, and service name resolution requests. The tables are located in the directory specified by the lookaside_path token in the seos.ini file, which by default is /opt/CA/eTrustAccessControl/ladb.
Lookaside Database with Four Tables

To set up the lookaside database with the four tables, do one of the following:

- If you are installing eTrust AC, answer yes when asked if you want to create the lookaside database.
- If you already installed eTrust AC:
  a. In the [seosd] section of seos.ini change the following tokens to yes:
     - under_NIS_server
     - use_lookaside
  b. Run sebuildla -a to create all four tables.

Lookaside Database with Less Than Four Tables

You can also create one, two, or three tables. For example, if you want to use the lookaside database to resolve hosts only, complete the following steps:

1. After you install eTrust AC, change the following tokens in the [seosd] section of the seos.ini file:
   - Set under_NIS_server to blank.
   - Set HostResolution to ladb.
2. Run sebuildla -h to create a table of all hosts, including local and DNS hosts.
   or
   Run sebuildla -e to create a table of local hosts only (defined in /etc/hosts).

To create a lookaside database with other tables, use the appropriate tokens in the seos.ini file and then run the appropriate option with sebuildla.

Note: For descriptions of these tokens, see the seos.ini initialization file in the Reference Guide. For more information about sebuildla, see the Utilities Guide.

Important! Run sebuildla whenever you add a host.
How the Lookaside Database Works

The four tables in the lookaside database (groupdb.la, hostdb.la, servdb.la, and userdb.la) contain resolution information for groups, hosts, services, and host names. The tables are located in the directory specified by the lookaside_path token in the seos.ini file, which by default is /opt/CA/eTrustAccessControl/ ladb.

eTrust AC internal name resolution is much faster than NIS name resolution and even faster than looking up th files.

Implementing the Lookaside Database

**Note:** The problems and solutions outlined here are for informational purposes only. Actual settings are correct upon installation and most users need not take any action.

Here is a broad overview of how eTrust AC implements the lookaside database:

- The relevant tokens in the seos.ini file are set.
- The relevant symbolic links in the /opt/CA/eTrustAccessControl/exits directory are defined.
- The command /opt/CA/eTrustAccessControl/bin/sebuildla -a was issued to build the lookaside database.

The sebuildla utility taps into the native resolution mechanisms such as th files and NIS to build the lookaside database.

No security-sensitive information (such as password, location of the home directory, or gecos) is kept in the lookaside tables. The lookaside database tables contain only a numeric ID number and a name.

Once the lookaside database is created, update it using the sebuildla utility. You do **not** need to restart eTrust AC.

Updating the Hosts Lookaside Table

You must update the hosts lookaside table. To do so, execute sebuildla -h at regular intervals (site-specific). Use cron jobs to do this.

Every time you change the UNIX user or group databases utilizing selang, you must run the sebuildla utility. eTrust AC provides exit scripts for this purpose, which runs sebuildla with the appropriate parameters.
Configuration Tokens: The seos.ini File

Following are the seos.ini tokens used in the eTrust AC initialization process.

<table>
<thead>
<tr>
<th>Token</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>exits_dir</td>
<td>lang</td>
</tr>
<tr>
<td>GroupidResolution</td>
<td>seosd</td>
</tr>
<tr>
<td>HostResolution</td>
<td>seosd</td>
</tr>
<tr>
<td>lookaside_path</td>
<td>seosd</td>
</tr>
<tr>
<td>nis_env</td>
<td>passwd</td>
</tr>
<tr>
<td>parent_pmd</td>
<td>seos</td>
</tr>
<tr>
<td>passwd_pmd</td>
<td>seos</td>
</tr>
<tr>
<td>pre_user_exit</td>
<td>lang</td>
</tr>
<tr>
<td>pre_group_exit</td>
<td>lang</td>
</tr>
<tr>
<td>post_group_exit</td>
<td>lang</td>
</tr>
<tr>
<td>post_user_exit</td>
<td>lang</td>
</tr>
<tr>
<td>ServiceResolution</td>
<td>seosd</td>
</tr>
<tr>
<td>under_NIS_server</td>
<td>seosd</td>
</tr>
<tr>
<td>AllowedGidRange</td>
<td>passwd</td>
</tr>
<tr>
<td>AllowedUidRange</td>
<td>passwd</td>
</tr>
<tr>
<td>UseridResolution</td>
<td>seosd</td>
</tr>
<tr>
<td>use_lookaside</td>
<td>seosd</td>
</tr>
<tr>
<td>YpGrpCmd</td>
<td>passwd</td>
</tr>
<tr>
<td>YpMakeDir</td>
<td>passwd</td>
</tr>
<tr>
<td>YpPassCmd</td>
<td>passwd</td>
</tr>
<tr>
<td>YpServerGroup</td>
<td>passwd</td>
</tr>
<tr>
<td>YpServerPasswd</td>
<td>passwd</td>
</tr>
<tr>
<td>YpServerSecure</td>
<td>passwd</td>
</tr>
</tbody>
</table>
Appendix B: Password Synchronization with Mainframes

This section contains the following topics:

- Password Synchronization Support (see page 261)
- Password Policy Model Methods (see page 261)
- Installing Password Synchronization (see page 262)
- Starting Mainframe Synchronization (see page 270)
- The CAICCI Configuration File (see page 273)

Password Synchronization Support

eTrust AC supports password synchronization among mainframes running eTrust CA-Top Secret Security, eTrust CA-ACF2 Security, or RACF security products (and CA Common Services CAICCI package) and Windows or UNIX machines running eTrust AC. Synchronization is accomplished using the standard eTrust AC password Policy Model method.

Password Policy Model Methods

To implement password synchronization with a mainframe in your network, choose a UNIX machine running eTrust AC to serve as a parent to the mainframe and make sure the Mainframe Password Synchronization option is installed on it. Define the mainframe to eTrust AC and subscribe the mainframe to the password Policy Model from that UNIX machine. When you have done this, any password change a mainframe user makes is propagated to all the machines in the password Policy Model hierarchy.

When you give mainframe administrators access control authorization to make password changes, any user password change, suspend, resume user action they take on the mainframe is propagated from the mainframe through the password Policy Model hierarchy. Likewise, administrative password changes and suspend or resume user actions made anywhere in the password Policy Model hierarchy are propagated to the mainframe.

The Mainframe Synchronization daemon (mfsd) uses a translation file: 
`eTrustACDir/data/trans_mfsd.txt`, which contains a pattern and pairs of mainframe commands and their translation to selang commands. (`eTrustACDir` is the directory where eTrust AC installed.)
Installing Password Synchronization

Installation Requirements on the Mainframe

On each mainframe that you want to add to the password Policy Model hierarchy, you must install CA Common Services. You can find instructions for this installation and for configuring the mainframe for password synchronization in the following locations:

- For eTrust CA-ACF2 Security, in the eTrust CA-ACF2 Security Administrator Guide
- For eTrust CA-Top-Secret Security, in the eTrust CA-Top-Secret Security Administrator Guide
- For RACF, on the eTrust AC installation CD

Note: We recommend that the CAICCI connection be driven from the UNIX side. To do this, add a connection to the mainframe in the CCIRMTD file, and do not add NODE or CONNECT statements to the mainframe CAICCI parameters.
Installation Requirements on UNIX

On each UNIX machine that you want to use as a parent to a mainframe for password synchronization, you must install eTrust AC with the Mainframe Password Synchronization option.

**Note:** If you have eTrust AC already installed, run the installation script again and choose the Mainframe Password Synchronization option. This does not alter your current eTrust AC database or settings.

Before you begin the installation, you might want to obtain the host name, SYSID, and administrator name for each mainframe that you want to subscribe to the Policy Model from this machine. However, if you do not have access to this information at installation time, you can skip that part of the installation and subscribe the mainframes later.

Install the MFSD package using "install_base -mfsd." The SERVER package must be installed before installing the MFSD package. Another package called CAICCI installs automatically with the MFSD package. This is a stand-alone version of the CA Common Communication Interface.

If you already have Unicenter TNG, the mfsd daemon uses the CAICCI communication package. In this case, the installation script does not install CAICCI. It instead proposes shutting down Unicenter TNG (if it is running) to upgrade its security option file `UniDir/secopts` (where `UniDir` is the directory in which Unicenter TNG is installed.) Remember to restart Unicenter TNG after eTrust AC installs. If you do not have Unicenter TNG, the CAICCI package is installed in the eTrust AC directory `*/uni/cci`.

The installation allows you to subscribe hosts to the Policy Model. If you have the host names and SYSIDs for the mainframes, you can subscribe them now. Otherwise, you can skip this step and subscribe these hosts later. If you choose to enter host names later, you must update `eTrustACDir/uni/cci/config/hostname/ccirmtd.prf` with the following line:

```
"REMOTE = mainframeName mainframeSysid 1024 startup port=1721"
```
Checking the Installation

1. The mfsd and unixcpfd files should appear in the eTrust AC directory */sbin.
   If Unicenter TNG is not installed, the CAICCI package should appear in the eTrust AC directory */uni/cci and the shared libraries set should be added to the directory /usr/local/CAlib.

2. The mfsctrl script in the eTrust AC directory */bin starts and stops the service.

3. If Unicenter TNG is not installed, the profile files are updated to set the CAICGLBL0000 environment variable to the eTrust AC directory /seos/uni. If you need to define the variable manually, enter the command:
   ```
   setenv CAICGLBL0000 eTrustACDir/uni
   ```
   where eTrustACDir is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl.

Configuring the mfsd

To make the mfsd work, you must configure a few files:

1. Configure the CAICCI configuration file.
2. Configure the eTrust AC database on the UNIX machine that mfsd runs on.
3. Configure the Policy Models that distribute the Mainframe updates.

A script can help you configure mfsd. The script is located at eTrustACDir/bin/mfsdconf.sh (where eTrustACDir is the directory where you installed eTrust AC). You can also configure everything manually as described in the follow sections.

Prior to running this script you should have:

- A Policy Model that will use the mainframe updates for propagation.
- The mainframe SYSID and mainframe names that the Policy Model receives updates from, as well as the administrator’s users.
Configuring the Translation File

To understand the translation file trans_mfsd.txt in the eTrustACDir/data directory, use the following notes:

- The first line is the pattern with a letter sub (for recognizing words that are inserted in the selang command) that mfsd uses. This pattern describes a copy (as is) from the Mainframe command to the selang command with the same letter.

  You can change this pattern, but you must change it in all the places it appears in the file. Find an example in the file.

- Note the pairs of mainframe commands and their translation to selang commands.

- In the RACF section, if you want to translate commands that use brackets and commands to those that do not, you must write the translations that use brackets () around a word before the lines that do not. The translation file is case-sensitive.

- Spaces are significant in the command.

- You can use the following pseudo-regular expressions in the commands:

<table>
<thead>
<tr>
<th>Character</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Matches to zero or more characters</td>
</tr>
<tr>
<td>?</td>
<td>Matches one character</td>
</tr>
<tr>
<td>[ ]</td>
<td>Delimit a range or value list</td>
</tr>
</tbody>
</table>

  A range can be:

  - Complete     [a-z]
  - Negated Complete     [^a-z]
  - Prefix-Incomplete     [-z]
  - Negated Prefix-Incomplete     [^-z]
  - Suffix-Incomplete     [a-]
  - Negated Suffix-Incomplete     [^a-]

  A value list can be:

  - Plain     [abf]
  - Negated     [^abf]

- You can add new translations to the file in the same format:

  - MF is the Mainframe command.
  - PM is the selang command to translate to.
Translation File Configuration Example 1

If you set your trans_mfsd.txt like this:

 reciprocity TSS REP*(sub(x)) PAS*(sub(y))*
 reciprocity PM->env seos; cu sub(x) password(sub(y))

It causes MFSD to translate like the following. The following mainframe command is translated to the selang command following it.

TSS REPLACE(username) PASSWORD(123)
env seos; cu username password(123)

Translation File Configuration Example 2

If you set your trans_mfsd.txt like this:

sub
 reciprocity MF->ALT* (sub(x)) PASSWORD(sub(y))*
 reciprocity PM->env seos; cu sub(x) password(sub(y))
 reciprocity MF->ALT* sub(x) PASSWORD(sub(y))*
 reciprocity PM->env seos; cu sub(x) password(sub(y))

MFSD translates both RACF commands ALT username PASSWORD(123) and ALT (username) PASSWORD(123) to the following selang command:

env seos; cu username password(123)

You must restart the mfsd daemon after editing the trans_mfsd.txt file.

Defining Exit Functions

If you want to edit the selang command after it is translated from the mainframe command but before it is executed (the mfsd output), you must define an exit function.

Within the exit function, you can edit all parts of the selang command except the password field (for security reasons).

You can see an example of this by installing the API package. The sample looks for commands with the string "isrdv" in the username field and changes the string to "ISRDV" instead. For example, the first command that follows would be changed to the second command after the exit:

cu isrdv01 password(*)
cu ISRDV01 password(*)
Completing the Policy Model Configuration

Once you have installed the appropriate software on both your mainframes and the parent UNIX system, you must perform the following procedures on the UNIX system to complete the configuration required for password synchronization:

1. Make the following changes on the local eTrust AC database where mfsd is running:
   a. Because the user executing mfsd is root, you must define user root as an eTrust AC administrator and have read and write access to the local terminal.

   
   eTrustAc> edituser root admin
   (localhost)

   Successfully updated USER root

   eTrustAc> auth terminal name.companyname.com uid(root) acc(r w)
   (localhost)

   Successfully added root to name.companyname.com's ACL

   eTrustAc> |

   b. If you do not want root to be the administrator, you can use the SPECIALPGM class.

   SPECIALPGM enables the functionality of a program running under a UNIX user or an eTrust AC user in eTrust AC. Assign the program a UNIX user and a logical user (the eTrust AC user). When the user calls the program, eTrust AC uses the authorization of the logical user.

   eTrustAc> edituser mfs_adm admin
   (localhost)

   Successfully created USER mfs_adm

   eTrustAc> auth terminal name.companyname.com uid(mfs_adm) acc(r w)
   (localhost)

   Successfully added mfs_adm to name.companyname.com's ACL

   eTrustAc> newres specialpgm opt/CA/eTrustAccessControl/lbin/mfsd
   owner(nobody) unix(root) seosuid(mfs_adm)
   (localhost)

   Successfully created SPECIALPGM opt/CA/eTrustAccessControl/lbin/mfsd

   eTrustAc> |

   c. Create a PMDB, set the password_pmd token in seos.ini to point to this database, and connect to the Policy Model (host can be a local host or a remote host):

   eTrustAc> host pmd@
   (pmd@localhost)

   Successfully connected

   INFO: Target host's version is 8.0 (8.0)

   Unix OS info:hostname SunOS 5.8 Feb 2004 12:05:32 IST

   eTrustAc> |
2. Define user root or mfs_adm as an administrator in the Policy Model.

   eTrustAc> edituser mfs_adm admin
   (pmd@localhost)
   Successfully created USER mfs_adm
   eTrustAc> ▌

   a. Create a TERMINAL record in the Policy Model. Give mfs_adm read and write authorization on the terminal (localhost is the name of the host where the mfs daemon is running):

   eTrustAc> newres terminal name.companyname.com owner(nobody)
   (pmd@localhost)
   Successfully created TERMINAL name.companyname.com
   eTrustAc> auth terminal name.companyname.com uid(mfs_adm) acc(r w)
   (pmd@localhost)
   Successfully added mfs_adm to name.companyname.com's ACL
   eTrustAc> ▌

   b. Create an MFTERMINAL record in the PMDB for each mainframe host that you plan to subscribe to the Policy Model (mfSYSID is the SYSID of the mainframe):

   eTrustAc> newres mfterminal mfSYSID defaccess(none) owner(nobody)
   (pmd@localhost)
   Successfully created MFTERMINAL mfSYSID
   eTrustAc> ▌

When you create a record in the PMDB instead of the local eTrust AC database, this record gets propagated to all the hosts in the Policy Model hierarchy.

1. Create a USER record in the PMDB for each mainframe administrator who will be issuing password changes, giving these users Administrator authority so that eTrust AC recognizes their right to make password changes.

   eTrustAc> newusr mfAdmin admin
   (pmd@localhost)
   Successfully created USER mfAdmin
   eTrustAc> ▌

2. Again in the PMDB, give these mainframe administrators read access to the MFTERMINAL records for any mainframe from which they can issue a password change (mfSYSID is the SYSID of the mainframe and mfAdmin is the mainframe administrator user):

   eTrustAc> authorize MFTERMINAL mfSYSID uid(mfAdmin) access(read)
   (pmd@localhost)
   Successfully added mfAdmin to mfSYSID's ACL
   eTrustAc> ▌

3. Subscribe the mainframes to the Policy Model (if you did not do this during installation).

   [131] % sepmd -sm pmdb unixhost.ca.com TSS sys1 user1
Note: These items do not need to be performed in any particular order (except, of course, that you cannot give mainframe administrators access permissions to the MFTERMINAL records until you have created both the administrator user record and the mainframe MFTERMINAL record).
Starting Mainframe Synchronization

To provide the synchronization service, you need to start three components:

- CAICCI
- unixcpfd
- mfsd daemons

You must also start Unicenter TNG if you are using it. Check the process status to see that three CAICCI processes (caiccid, ccirmtd, and cciclnd) are running.

If Unicenter TNG is installed, run the utility `UniDir/cci/bin/cciito` test whether CAICCI is working properly. If the ccirmtd.prf file is properly configured, the names of the mainframe hosts display on the monitor.

If Unicenter TNG is not installed, the eTrust AC install_base adds the stand-alone CAICCI package to its installation, and you need to start the CAICCI daemons from the eTrust AC directory. Use the `mfscntrl` utility to start CAICCI.

```
unixhost:bin> mfscntrl start cci
Starting cci daemon
unixhost:bin> |
```

To be sure that communication was established, run the utility:

```
unixhost:bin> /opt/CA/eTrustAccessControl/unl/cci/bin/ccii
```
Starting Mainframe Synchronization

If the eTrust AC directory /uni/cc/config/hostname/ccirmtd.prf was not updated in the eTrust AC installation process, you must configure it before starting the CAICCI daemons.

Use the following command to start the Command Propagation Facility daemons (unixcpfd). The mfscntrl command executes the unixcpfd program in the eTrust AC directory /lbin/. You should see four daemons with the name unixcpfd in the process status reporting. The first is a parent process, the second is waiting for updates from eTrust CA-Top Secret Security mainframe systems, the third is waiting for updates from RACF mainframes, and the fourth is waiting for updates from eTrust CA-ACF2 Security mainframes:

```
unixhost:bin> mfscntrl start unixcpfd
```
Starting unixcpfd daemon.

`unixhost:bin> |`

The daemon creates the file `/opt/CA/eTrustAccessControl/log/unixcpfd.log` when `unixcpfd` receives messages from the CAICCI service. You can run `unixcpfd` in the trace mode by setting the environment variable `CA_CAIDEBUG` to yes and running `unixcpfd` from this shell prompt.

Use the following command to start the mainframe synchronization daemon. This creates the trace file `/opt/CA/eTrustAccessControl/log/mfsd.trace` if you set the environment variable `CA_CAIDEBUG` to yes before running `mfsd`.

```
unixhost:bin> mfsctrl start mfsd
unixhost:bin> Starting mfsd. PID = 5983
unixhost:bin> |  
```
The CAICCI Configuration File

During installation, and anytime you subscribe a mainframe to an eTrust AC Policy Model, eTrust AC automatically updates the CAICCI configuration file.

If, for some reason, you want to perform these updates manually, use the following procedure:

1. Open the CAICCI configuration file.
   - If you are using Unicenter TNG, enter the following command:
     
     ```
     UniDir/cci/config/hostname/ccirmtd.prf
     ```
     where `UniDir` is the Unicenter TNG installation directory
   - If you are using eTrust stand-alone CAICCI, enter the following command:
     
     ```
     eTrustACDir/uni/cci/config/hostname/ccirmtd.prf
     ```
     where `eTrustACDir` is the installation directory for eTrust AC, by default `/opt/CA/eTrustAccessControl`

2. Add the following line:
   
   ```
   REMOTE = mainframeName mfSysid 1024 startup port=1721
   ```
   where `mfSYSID` is the mainframe SYSID.

3. Save the file.

4. Stop remote CAICCI services.
   - If you are using Unicenter TNG, shut down Unicenter TNG services with the following command:
     
     ```
     unicntrl stop all
     ```
   - If you are using eTrust stand-alone CAICCI, use the following command:
     
     ```
     eTrustACDir/uni/cci/bin/ccicntrl shutdown
     ```

5. Restart remote CAICCI services:
   - If you are using Unicenter TNG, start Unicenter TNG services with the following command:
     
     ```
     unicntrl start all
     ```
   - If you are using eTrust stand-alone CAICCI, use the following command:
     
     ```
     eTrustACDir/bin/mfscntrl start cci
     ```
This section contains the following topics:

- eTrust Audit Integration (see page 275)
- eTrust AC Logs (see page 276)
- Configuring eTrust AC (see page 276)

**eTrust Audit Integration**

This appendix provides an overview of the integration between eTrust AC and eTrust Audit 1.5 and describes how eTrust AC can be configured to send its logs to eTrust Audit.

Because eTrust AC is host-based, its audit logs are stored locally on each host. These logs can be centrally collected using native eTrust AC tools on UNIX, but this capability does not extend to eTrust AC on Windows. In addition, to get meaningful reports or perform event correlation, native UNIX tools (like awk and sed) must be used on the centrally collected log file, which can prove to be cumbersome and time consuming.

eTrust Audit collects audit logs from various sources such as Windows Event Logs or UNIX syslogs and stores them in one ODBC database. eTrust Audit includes tools for reporting and event correlation, and because the data is in a database, not in a text file, it is much easier to manipulate.

You can configure eTrust AC to send logs to eTrust Audit to make use of all the benefits of storing information in a database.
eTrust AC Logs

eTrust AC for UNIX stores its audit logs in the file $eTrustACDir/log/seos.audit$. eTrust AC for Windows stores its audit logs in the file $eTrustACDir\log\seos.audit$. This file is located on each server that runs eTrust AC and cannot be removed by any user (including root on UNIX or an Administrator on Windows).

Audit logs are written to by the eTrust AC daemons and services in real time depending on the audit settings for resources in the eTrust AC database. By default, all eTrust AC database rule changes and failures are audited; any other event that needs to be audited has to be explicitly defined in the resource ACL.

These local audit log files of eTrust AC are automatically overwritten depending on a configuration setting in the seos.ini file. The token audit_size defines the maximum size of the seos.audit file. When the audit log reaches the specified size, eTrust AC automatically saves the current log file into a file called $eTrustACDir/log/seos.audit.bak$ and re-creates the seos.audit file. When the threshold is reached again, the process is repeated; this may result in loss of audit data.

Configuring eTrust AC

To collect the eTrust AC logs into eTrust Audit, use the following steps.

For UNIX

It is assumed that eTrust AC is already installed and running on a UNIX server, you are logged in as root, and root is defined as the eTrust AC Security Administrator.

1. Shut down eTrust AC.
   
   ```
   # eTrustACDir/bin/secons -s
   ```

2. Enter the following command to automatically start the eTrust AC log-routing daemon selogrdr whenever eTrust AC starts up.
   
   ```
   # eTrustACDir/bin/seini -s daemons.selogrdr yes
   ```

3. Create a new file called selogrdr.cfg
   
   ```
   # vi eTrustACDir/log/selogrdr.cfg
   ```
4. Enter three lines in this file as follows

    Hostrule
    hostname_or_IP_address_of_eTrust_Audit_Router

**Note:** The last line of selogrd.cfg file **must** end with a period.

The simple configuration you have created will send all audit records from eTrust AC to eTrust Audit.

Follow these steps to ensure that you have correctly identified the eTrust Audit router server:

- If the system uses DNS, then enter the fully qualified DNS name of the server. For example:
  
  name.companyname.com

- If the system is using the local /etc/hosts file for lookup, then enter the name of the server as it appears first in the /etc/hosts file. For example, if you see the following in the /etc/hosts file, then enter name01:

  141.202.243.10 name01 name01.companyname.com

  If you are not sure, enter the IP address of the server.

  To find out which hostname resolution method the system is using, check the /etc/nsswitch.conf file and look for the token `hosts`, which specifies the order of resolution.

5. Invoke the eTrust AC daemons:

    `# eTrustACDir/bin/seload`

    Four eTrust AC daemons start up. The last one is the selogrd daemon.

---

**eTrust Audit Configuration**

In order to receive eTrust AC logs into the eTrust Audit database, you must determine where the eTrust Audit router is running. This can vary depending on the implementation of eTrust Audit. Listed below are some of the common cases.
Collecting eTrust AC for UNIX Logs into Audit

If your intent is to collect only eTrust AC for UNIX logs into Audit (not the native UNIX syslog or sulog), then you can set up the eTrust Audit router on the same server that has the eTrust Audit collector.

In this scenario, for an implementation of 100 UNIX servers running eTrust AC running, there is no need to install an additional 100 eTrust Audit agents on any UNIX server.

The eTrust Audit collector and the eTrust Audit router can be installed on the same Windows server, and all UNIX servers that run eTrust AC can reference this server in the selogrd.cfg file mentioned in the above section.

See the eTrust Audit documentation to:
- Create an audit node in eTrust Audit.
- Create a policy for eTrust AC in the eTrust Audit Policy Manager.
- Activate and distribute the policy.

Collecting eTrust AC for UNIX logs, syslogs, and sulogs into eTrust Audit

If the intent is to collect eTrust AC for UNIX logs as well as native UNIX syslogs and sulogs, then you must install the eTrust Audit client on the UNIX server running eTrust AC.

In this scenario, the eTrust Audit router runs on the same UNIX server that runs eTrust AC. The entry in the selogrd.cfg file should be set to localhost.

See the eTrust Audit documentation to:
- Create two audit nodes in eTrust Audit.
- Create two policies for eTrust AC in the eTrust Audit Policy Manager.
- Activate and distribute the policy.
Collecting eTrust AC for Windows Logs into eTrust Audit

If you have eTrust AC running Windows servers and want to centralize logs into eTrust Audit, you must install the eTrust Audit client on all servers that are running eTrust AC for Windows.

See the eTrust Audit documentation to:

- Create an audit node in eTrust Audit.
- Create a policy for eTrust AC in the eTrust Audit Policy Manager.
- Activate and distribute the policy.
GHOST class • 54, 107
Global Access Check • 210
global authorization attributes • 197
grace logins • 72, 102
  specifying • 74
group
  authorization attributes • 199
definition as accessor • 46
  modifying group records • 50
  permissions • 46
  record • 46
  scope • 201
GROUP class • 54
GROUP-ADMIN attribute • 201
GROUP-AUDITOR attribute • 202
GROUP-OPERATOR attribute • 202
GROUP-PWMANAGER attribute • 202
groups • 46
  _restricted • 50
  member • 50
  nested • 50
  super • 50
GSUDO class • 54
GTERMINAL class • 54
GUI • 48, 50

H
hackers, preventing entry • 64, 175
HOLIDAY class • 54
HOST class • 54, 107
host name • 100
  pattern • 107
HOSTNET class • 54, 107
HOSTNP class • 54, 107
hosts
  lookaside table • 258
  managed by NIS • 107

I
idle work stations, locking • 173
IGN_HOL attribute • 199
inactive
  accounts • 66
  class bypass • 220
  login • 66
inode • 16, 90
installation
  password synchronization • 263
  RSV • 188
integrating with eTrust Audit • 275
IP address • 97
IP cache • 215

J
join command • 50, 200
  example • 51
join-command • 50

K
keyword • 245
kill command • 17, 93

L
Language Client API • 233
LCA • 233
LDAP • 233
ldap2seos • 234
local host terminals, restricting • 100
locking idle workstations • 173
log routing • 184
  configuration file • 184
login
  day and time restrictions • 103
  protecting login commands • 92
  restrictions • 17
LOGINAPPL class • 54
lookaside database • 256
  setting up • 256
loopback terminals, restricting • 100

M
mainframe synchronization, starting • 270
mask values • 107
match values • 107
member groups • 50
mfsd • 261
  configuring • 264
MFTERMINAL class • 54
migopts • 244
migrating
  Unicenter Security database • 245
  Unicenter Security options • 244
monitoring sensitive files • 89
N
NACL • 24, 85
name patterns • 107
name resolution • 254
  on NIS/DNS clients • 254
  on servers • 255
  on Sun Solaris systems • 253
native UNIX security • 86
negative access control list • 24, 85
nested groups • 50
network
  access rules • 107
  cache • 215
network bypass for ports • 218
newgrp • 50
  example • 51
newusr • 48
  example • 49
NIS • 101
nobody • 203
notation conventions • 13
notification record • 185
O
OPERATOR attribute • 198
Orange Book features • 176
overview • 16
owner, nobody • 83
ownership • 203
P
PACL • 23
  generic • 23
parent
  group • 200
passwd
  maps • 101
password
  attack • 17
  checking • 72, 102
  expiration • 72
  history • 70
password policy model methods • 261
  policies • 17, 70, 71, 102
  preventing password attacks • 64
  synchronization • 261
password synchronization
  installing • 263
  passwords
    applying policy with setoptions • 72
    changing • 71
    setting interval for profile groups • 73
    specifying interval • 73
    performing superuser tasks • 62
permissions • 46, 77
  permission bits • 90
pluggable authentication module • 65
PMDB
  about • 117
  architecture • 120
  create and configure • 124, 126, 128
  filter updates • 133
  hard disk location • 118
  management • 118, 119, 122
  policy-based management • 144, 148
  rule-based policy updates • 122, 123, 130
PMDB update file
  cleaning up • 131
  encrypting • 131
Policy Model
  configuration • 267
  error log • 136
  port numbers, providing service to • 107
  predefined classes • 54
  preventing surrogate bypasses • 59
  priority • 216
PROCESS class • 54
process file system bypass • 217
profile group • 46
program access control list • 23
PROGRAM class • 54, 90
program pathing • 17, 23, 84
programs, protecting regular • 92
properties, restrictive • 249
protecting
  accounts, overview • 59
  binary files • 93
  regular programs • 92
  the network • 107
  user-defined programs • 17
PWMANAGER attribute • 198
PWPOLICY class • 54
R
RACF • 261
real path cache • 216
reducing
audit loads • 218
database loads • 219
regular programs, protecting • 92
remote administration • 207
remote status view • 187
displaying • 189
displaying statuses • 194
installing • 188
security • 195
starting • 188
status categories • 190
resource
abstract object • 52
cache • 214
defined in-house • 174
in database • 20
overview • 52
physical object • 52
record • 53
RESOURCE_DESC class • 54
RESPONSE_TAB class • 54
restricting
access to files and directories • 64
access to resources • 176
generic login applications • 96
login applications • 95
TCP/IP services • 107
terminals • 97
terminals, citing ownership • 97
user substitution • 59
user surrogates • 59
restrictive properties • 249
rlogin • 97
rmgrp • 50
eample • 51
rmusr • 48
eample • 49
root • 59
root restrictions • 17, 77
RSV • 187
host • 189
rwx permissions • 86
S
S50CREATE_u_LdapE.sh • 238
SAF • 16
screen lock program • 173
screen savers • 173
seaudit • 183
seauditx • 183
sebuildl utility • 253
SECFILE class • 54
SECLABEL class • 54, 176, 180
secons utility • 103
security
synchronization • 86
Security Administrator • 48, 50
security category • 177
defining • 178
deleting • 178
disable checking • 178
enable checking • 178
listing • 178
overview • 25
Security Enabling Services (SES) • 16
security features
API • 174
B1 security level certification • 176
locking idle stations • 176
STOP • 175
security label • 25
defining • 180
deleting • 180
disable checking • 179
enable checking • 179
listing • 180
overview • 179
security level • 25, 176
disable checking • 177
enable checking • 177
segrace
utility • 102
selock
utility • 173
sensitive files, monitoring • 89
SEOS class • 54
seos.ini tokens
Dictionary • 70
pam_seos • 65
SyncUnixFilePerms • 86
seos2ldap • 236
seosd
daemon • 89
seostngd • 241
seoswd • 89
seospass utility • 71
serevu
utility • 64
environment • 208
exit, description • 223
native security • 86
untrusted programs • 90
user
  _restricted group • 50
defined as accessor • 45
exits • 174
groups • 46
name • 45
permissions • 46
record • 45
records, modifying • 48
restricting substitute users • 59
restricting surrogate users • 59
USER class • 54
USER_ATTR class • 54
USER_DIR class • 54
user-defined classes • 57

W
warning mode • 182
watchdog • 89
  performance • 220
web-based viewer • 187
wildcards, using to protect files • 81
work hours • 17

X
X terminals • 173
XDM • 97
xstartup script • 173
X-terminal • 97