*e*Trust™ Access Control for UNIX

Administrator Guide

5.3

Computer Associates®

G00151-1E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is 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 this documentation for their own internal use, 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 confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the 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 return to CA the reproduced copies or to certify to CA that same have been destroyed.

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 this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2003 Computer Associates International, Inc.

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

Contents

Chapter 3: Users, Groups, and Resources

Chapter 4: Protecting Accounts

Chapter 5: Password Control

Chapter 6: Protecting Files and Programs

Chapter 7: Controlling Login Commands

Chapter 8: Protecting TCP/IP Services

Chapter 9: Managing the Policy Model Database

Chapter 10: General Security Features

Chapter 11: Auditing Events

Chapter 12: Remote Status View

Chapter 13: Scope of Administration Authority

Chapter 14: Improving Performance

Chapter 15: UNIX Exits

Chapter 16: Interacting with LDAP

Chapter 17: Unicenter Security Migration and Integration

Appendix A: Installing and Customizing

Appendix B: NIS Configuration

Appendix C: The seos.ini Initialization File

Index

Chapter 1

Introduction

Welcome to eTrust™ Access Control (eTrust AC)! 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, and it concludes with an implementation guide.

Who Should Use this Guide

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

Notation Conventions

The eTrust AC documentation uses a few special conventions.

- Programming code is shown in this special font. For example:

  ```
  eTrust> nu
  ```

- The username root refers to the account with user ID number 0 (zero). The login name may be root, superuser, or any other name mapped to the same account.
- Sometimes, to make clear the difference between what you must copy character by character from the instructions and what you must replace with your own data, **bold** and *italic* type are used.
  - Characters that are **bold** should be copied literally from the instructions.
  - Characters that are *italic* serve as placeholders for items that you must supply yourself, such as parameters.

  For example, when the guide says to enter **newusr** ***userName***, you could type the following command for a new user named Lebois:

  ```
  eTrust> newusr lebois
  ```

  Or enter the following for a user named Wald:

  ```
  eTrust> newusr wald
  ```

- Bold text is also used for simple emphasis. For example:

  You should **never** tape your password to the monitor.

- A technique shown in boldface is not necessarily the only effective technique, but the guide does not list all alternatives at every opportunity. For example, in describing a sequence of commands, the guide may mention **newusr wald** and not mention that **nu wald** works equally well.

- Sometimes a command does not fit on a single line in your window, or in this guide. In both cases, the same technique is used: a backslash (\) at the end of a line indicates that the command continues on the following line.

  Note: The use of backslashes in the guide does not necessarily indicate that you need backslashes in the same places.

- Square brackets mean that an item is optional. For example the following example means that you must type ./install_base, then a file name, then one or more options or no options;

  ```
  ./install_base filename [options]
  ```

- A pipe separates mutually exclusive items. Often 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}
  ```

Chapter

2 Basic Concepts

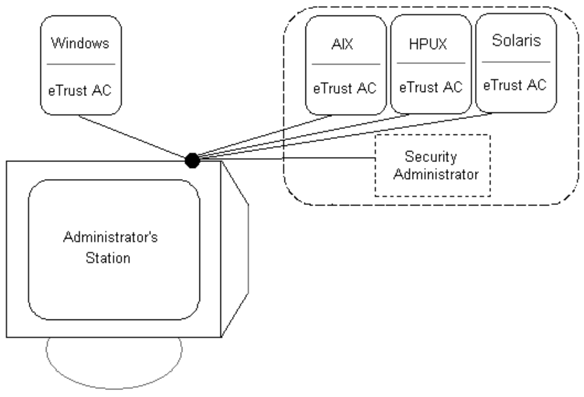
eTrust AC is a software product that is an active, comprehensive security software solution for Open Systems, tied dynamically to the operating system. Each time a user requests a security-sensitive operation—such as opening a file, substituting a user ID, or obtaining a network service—eTrust AC for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

This chapter introduces the components of eTrust AC, and discusses various concepts needed to understand and implement eTrust AC security.

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 with regard to 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 end 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.

- **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.

- **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 allowed 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 superuser authority**—eTrust AC provides the means to both delegate superuser authorities to operators and restrict root itself.

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

- **Kill attempts**—eTrust AC can be used to protect critical servers and daemons 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 a maximum and minimum life time for passwords, restrict certain words, disallow repetitive characters, and enforce other restrictions. Passwords are not allowed 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 both 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). *T*he 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:

```
eTrust> 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.

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:

```
eTrust> {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:

  ```
  eTrust> 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:

  ```
  eTrust> 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.

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

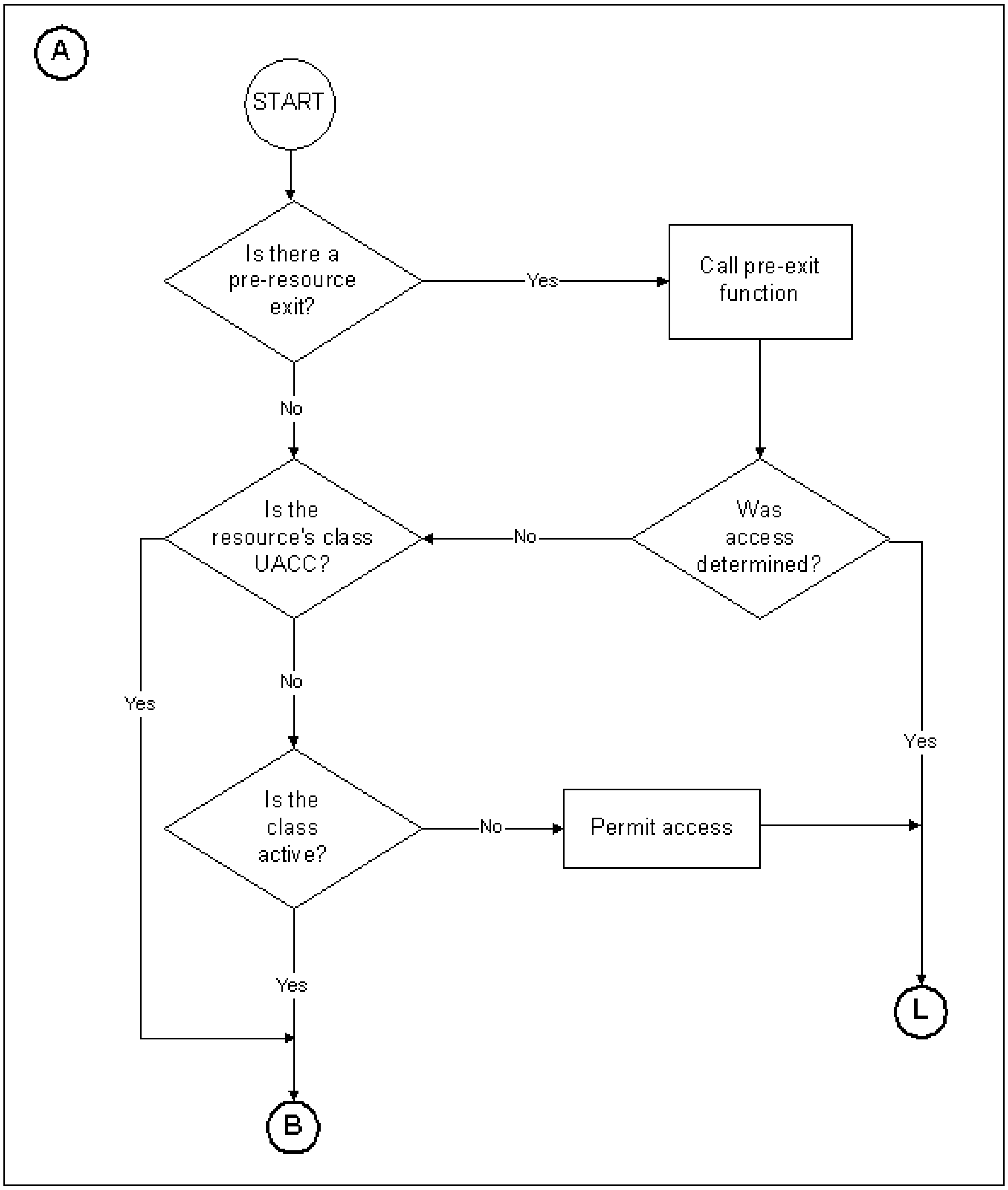
For example, a record in the FILE class (that protects files and directories) with the key /usr/lpp/bin/* protects all files, sub-directories, sub-sub-directories, and so forth, under the path /usr/lpp/bin. This allows you to protect many resources using a compact definition consisting of one access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

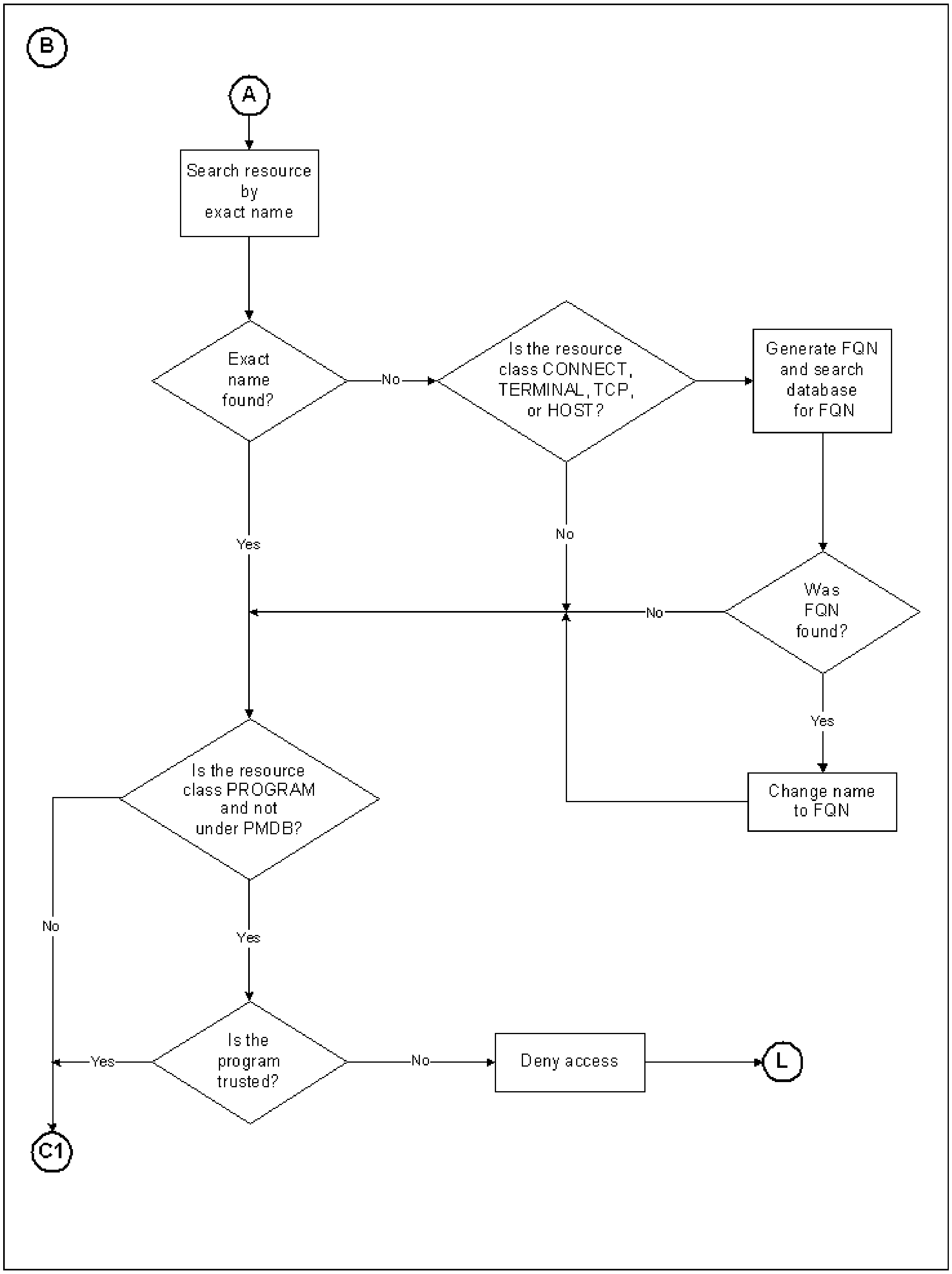
eTrust AC does **not** accept the following patterns in generic access rules:

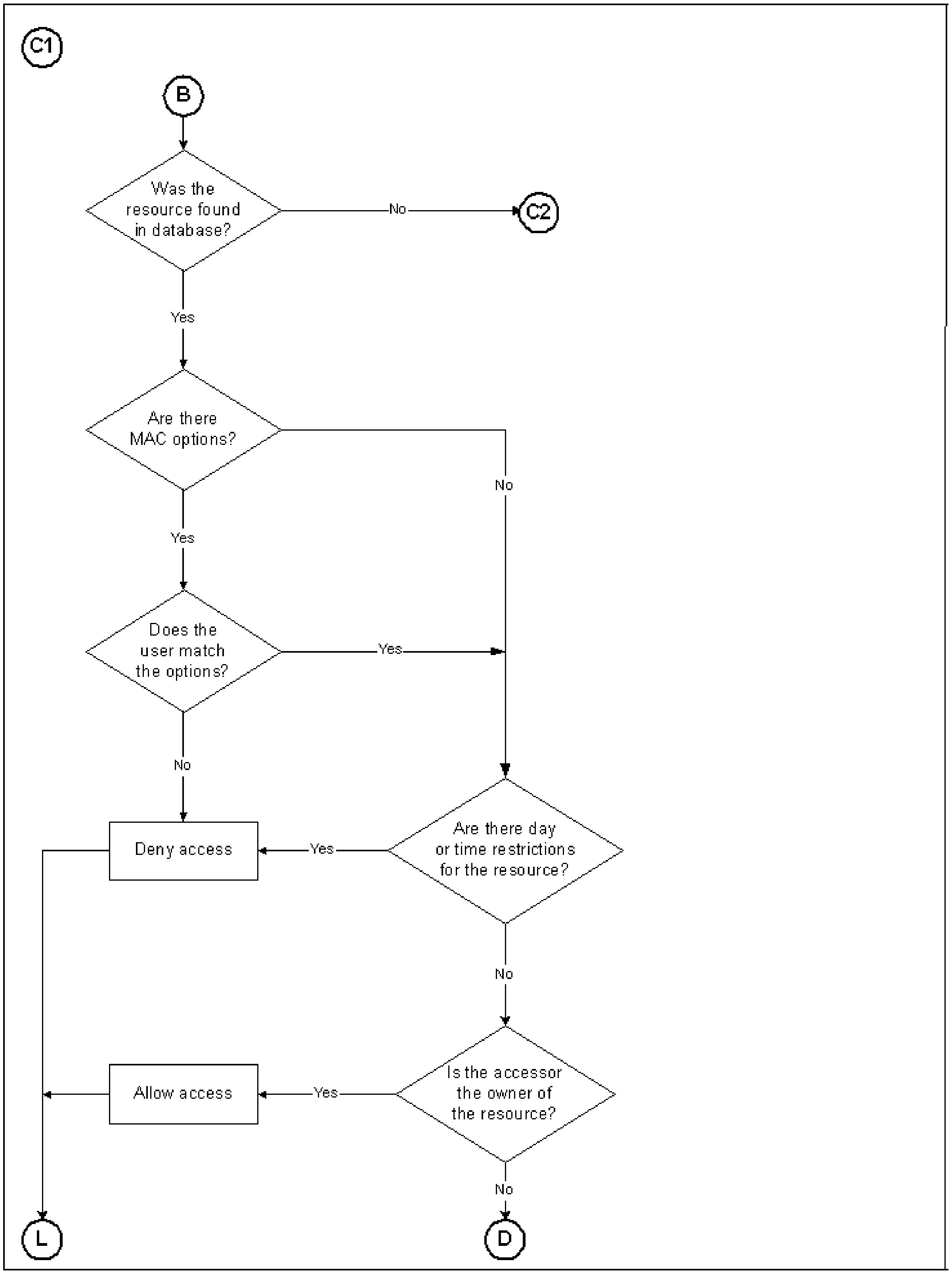
- /*
- /tmp/*
- /etc/*

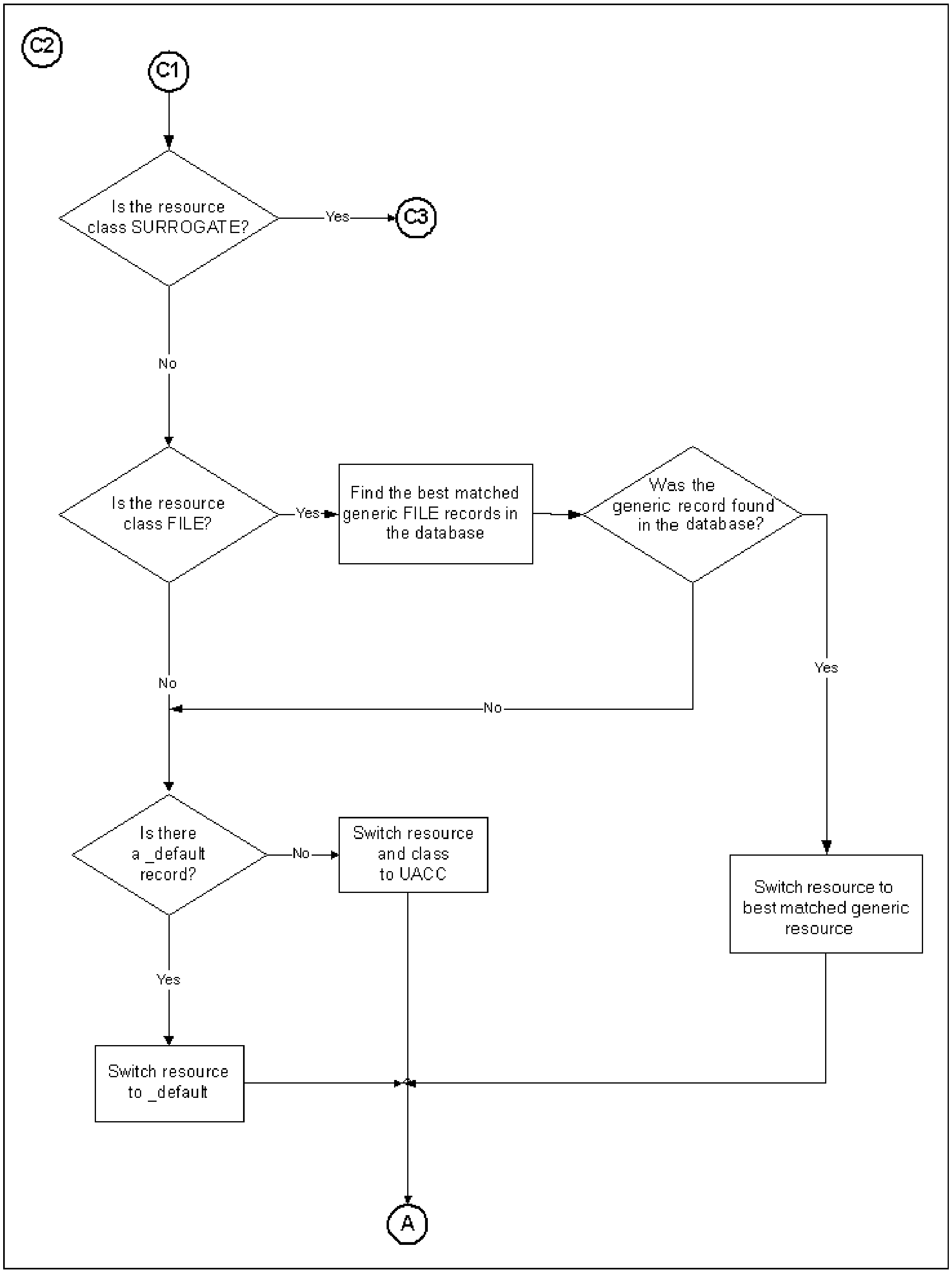
Graphical Access Summary

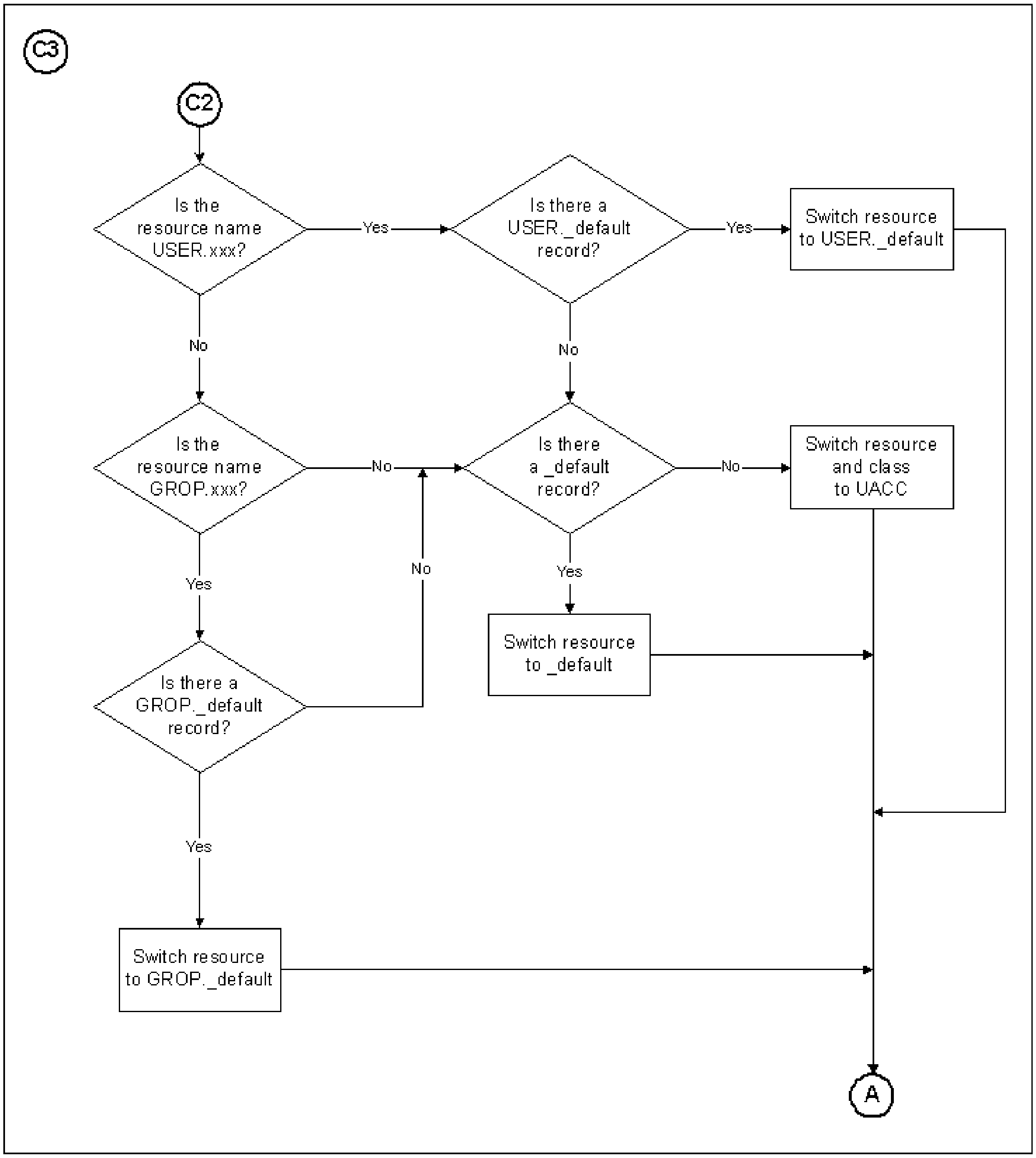
The following set of charts summarizes how eTrust AC allows or stops a particular user's attempt to access a particular resource.

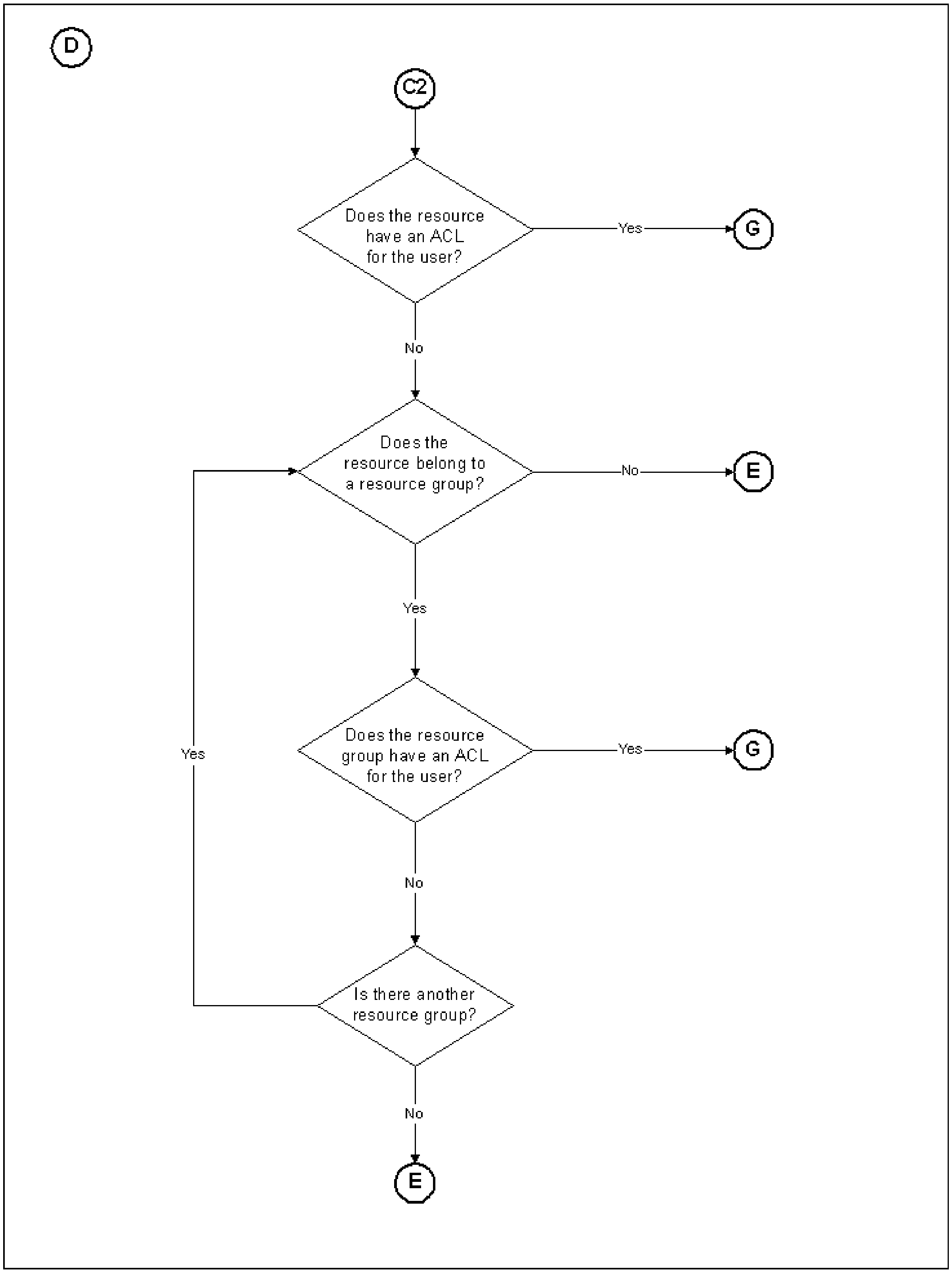


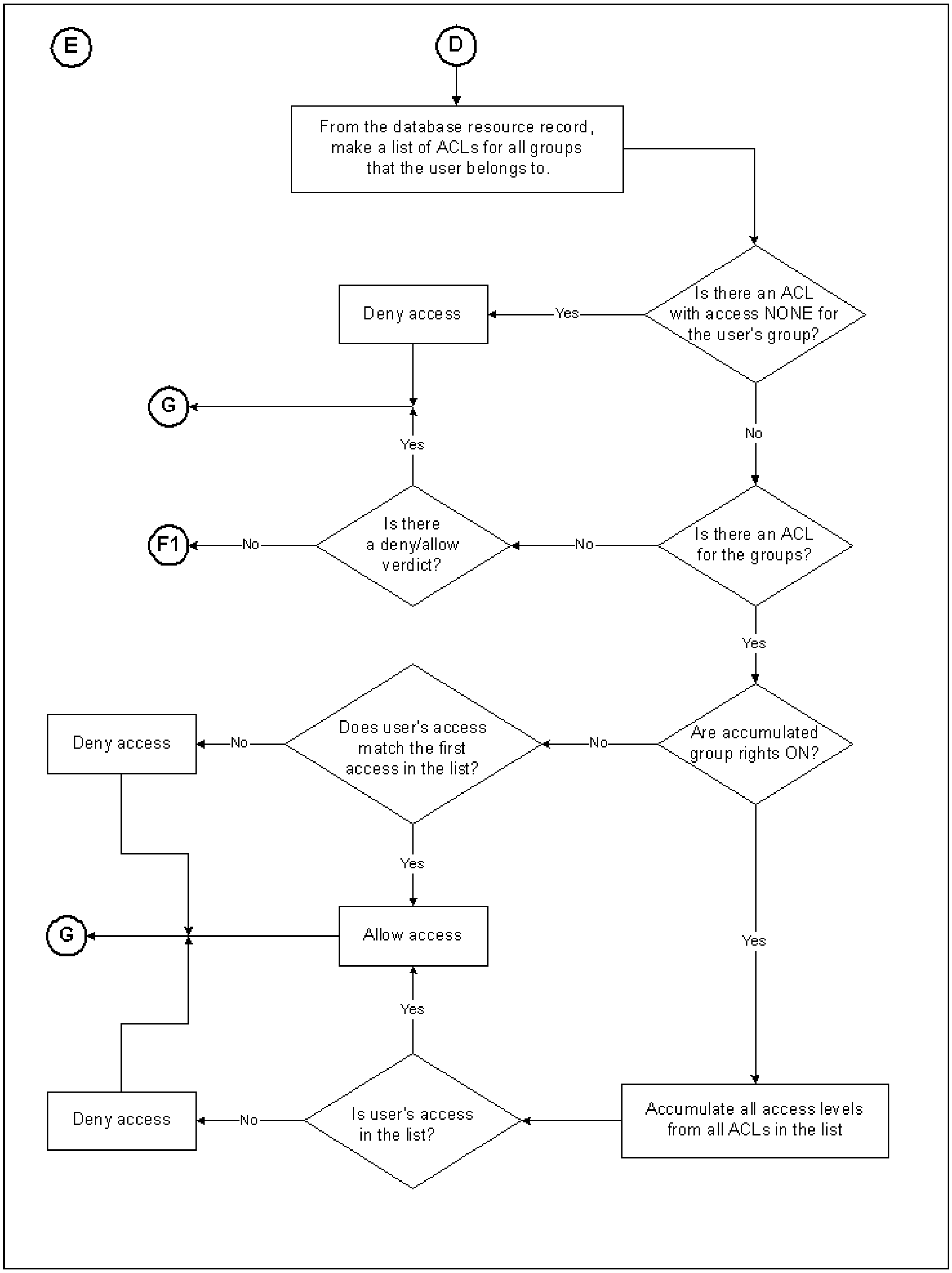

B
A
Search resource by exact name
Exact name found?
No
Is the resource class CONNECT, TERMINAL, TCP, or HOST?
Generate FQN and search database for FQN
Yes
No
Was FQN found?
No
Yes
Change name to FQN
Is the resource class PROGRAM and not under PMDB?
No
Yes
Is the program trusted?
Yes
No
Deny access
L
C1

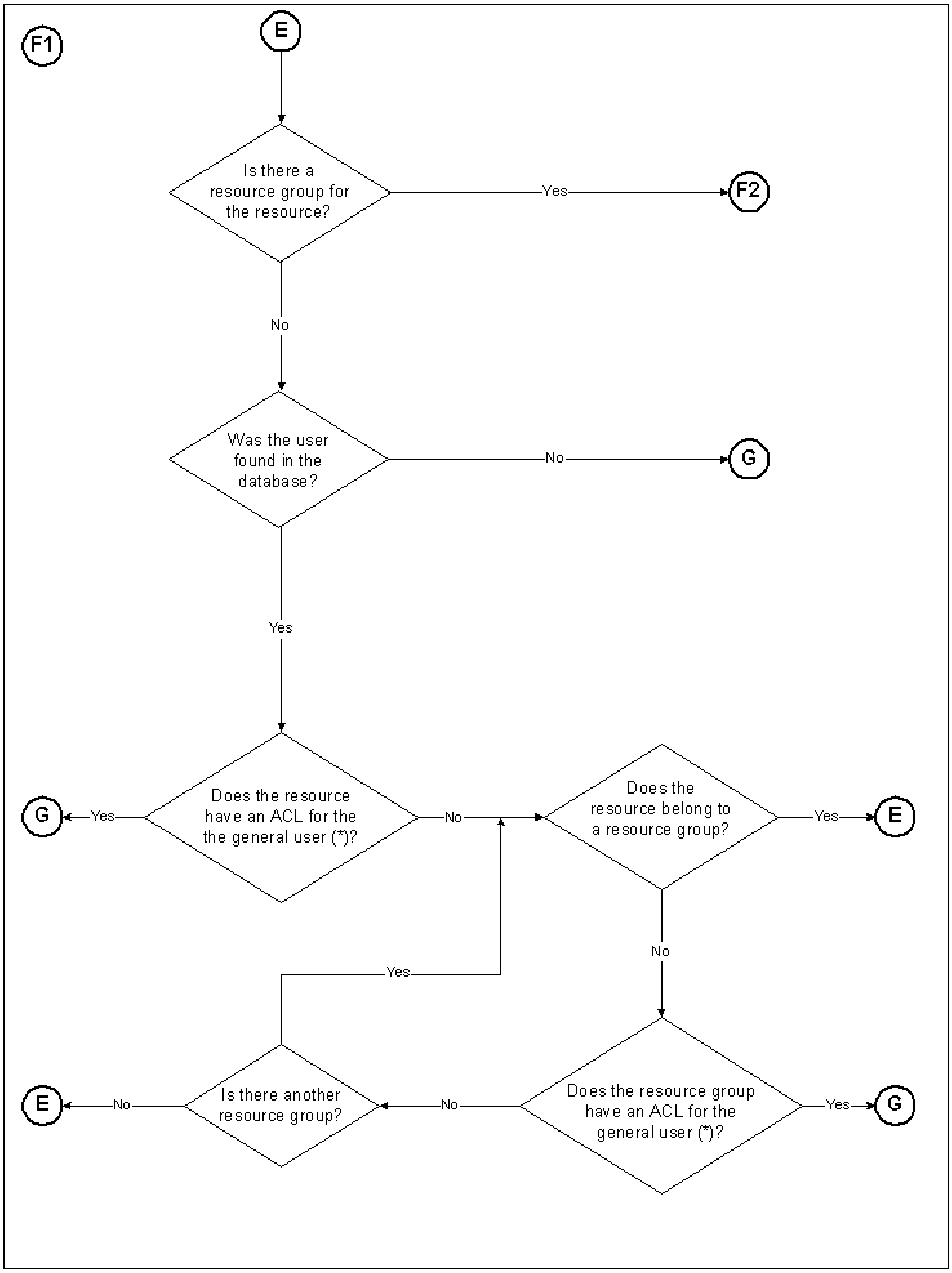

C1
B
Was the resource found in database?
No
C2
Yes
Are there MAC options?
No
Yes
Does the user match the options?
Yes
No
Deny access
Yes
Are there day or time restrictions for the resource?
No
Allow access
Yes
Is the accessor the owner of the resource?
No
L
D

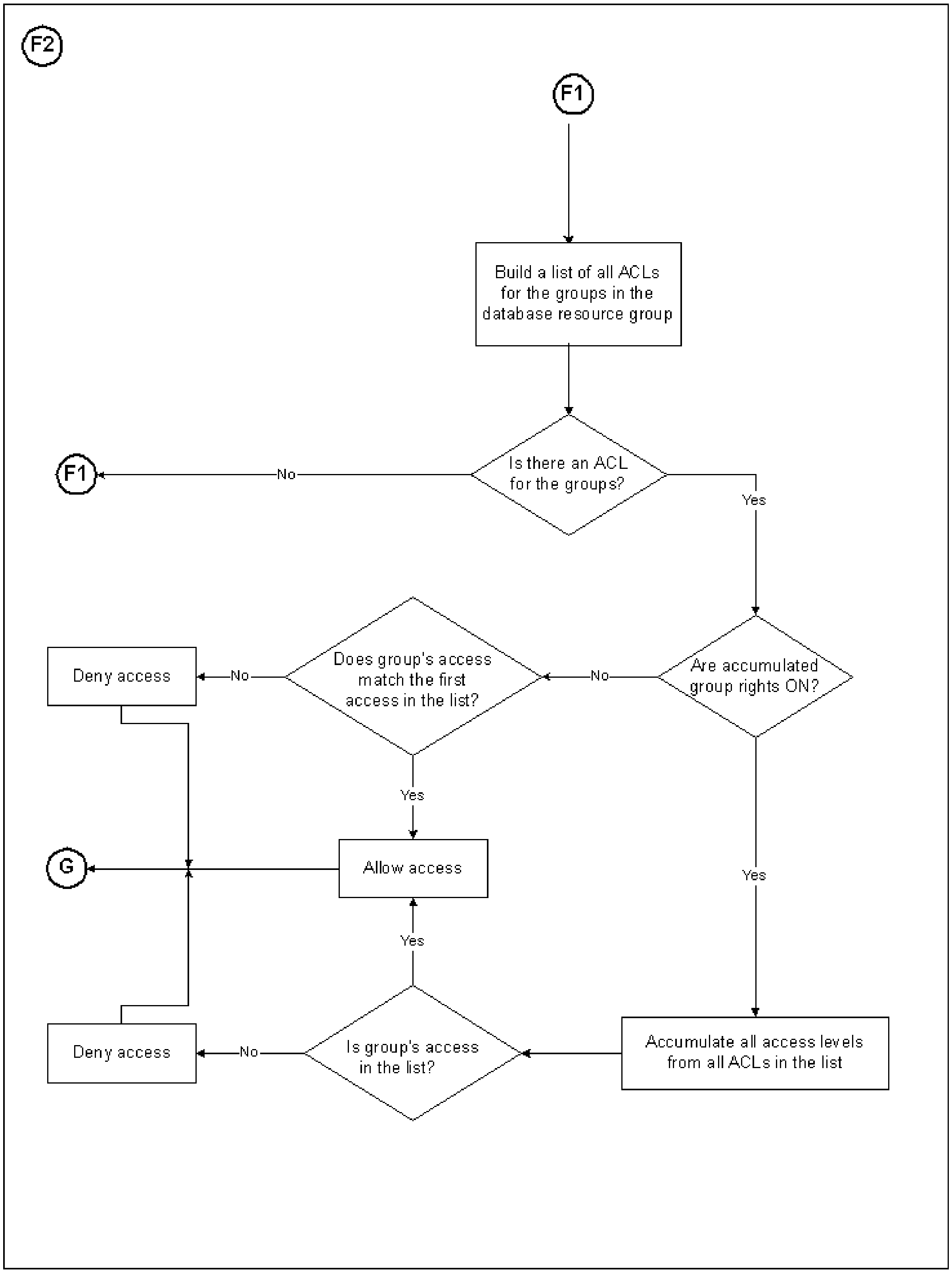

C2
C1
Is the resource class SURROGATE?
Yes
C3
No
Is the resource class FILE?
Yes
Find the best matched generic FILE records in the database
Was the generic record found in the database?
No
Yes
No
Is there a _default record?
No
Switch resource and class to UACC
Switch resource to best matched generic resource
Yes
Switch resource to _default
A

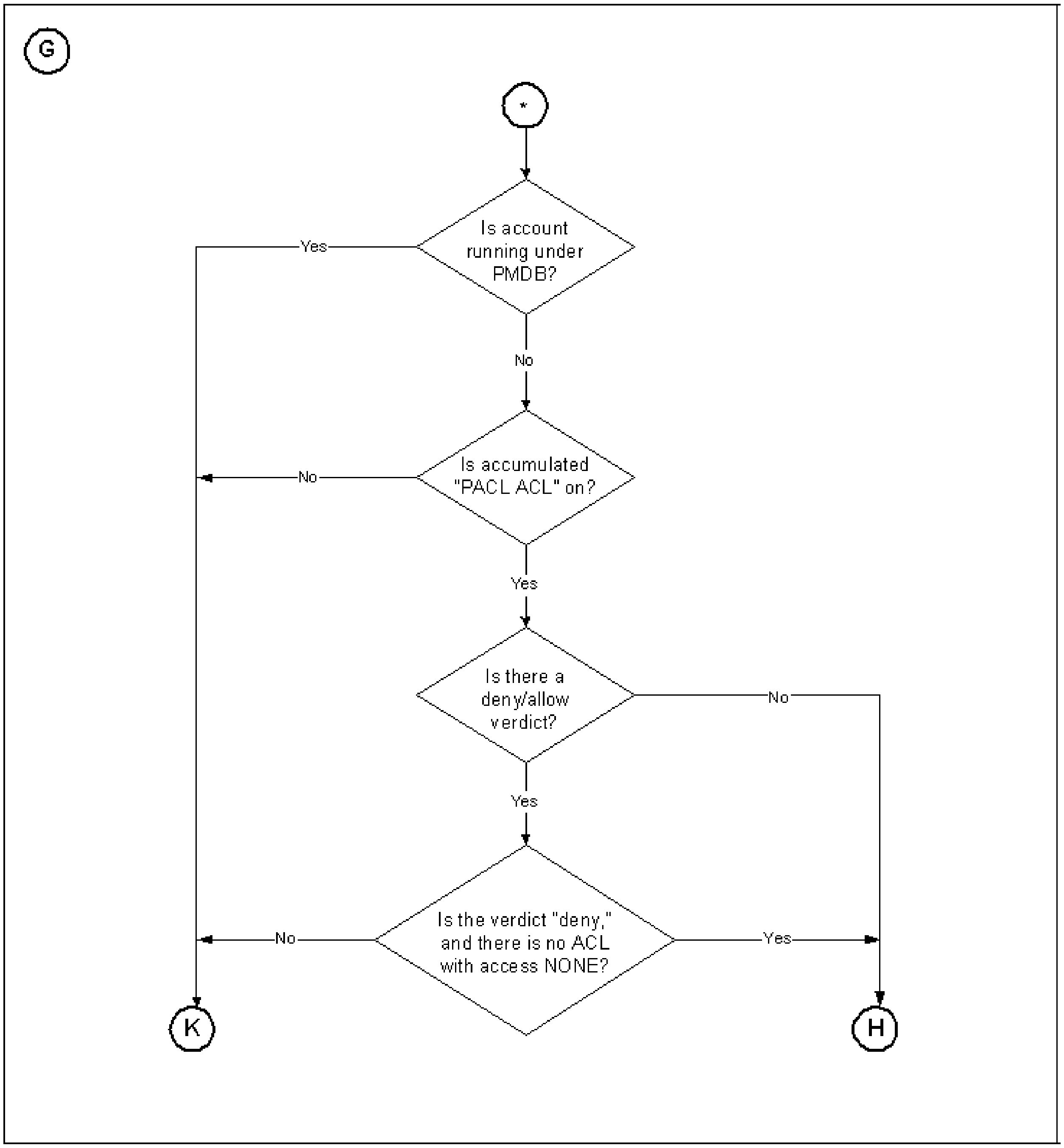

C3
C2
Is the resource name USER.xxx?
Yes
Is there a USER._default record?
Yes
Switch resource to USER._default
No
No
Is the resource name GROP.xxx?
No
Is there a _default record?
No
Switch resource and class to UACC
Yes
No
Yes
Switch resource to _default
Is there a GROP._default record?
Yes
Switch resource to GROP._default
A

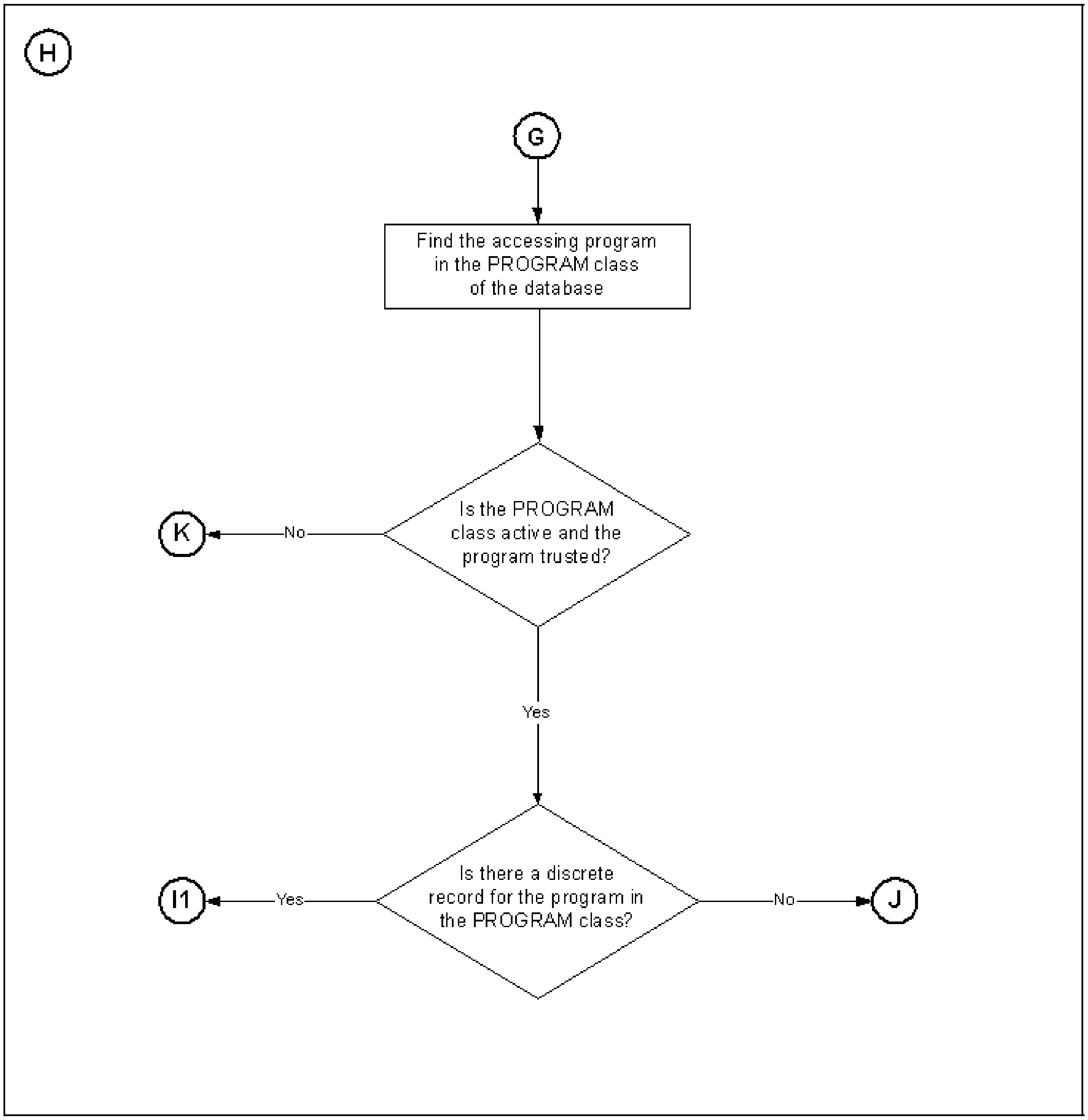

D
C2
Does the resource have an ACL for the user?
Yes
G
No
Does the resource belong to a resource group?
No
E
Yes
Does the resource group have an ACL for the user?
Yes
G
No
Is there another resource group?
Yes
No
E

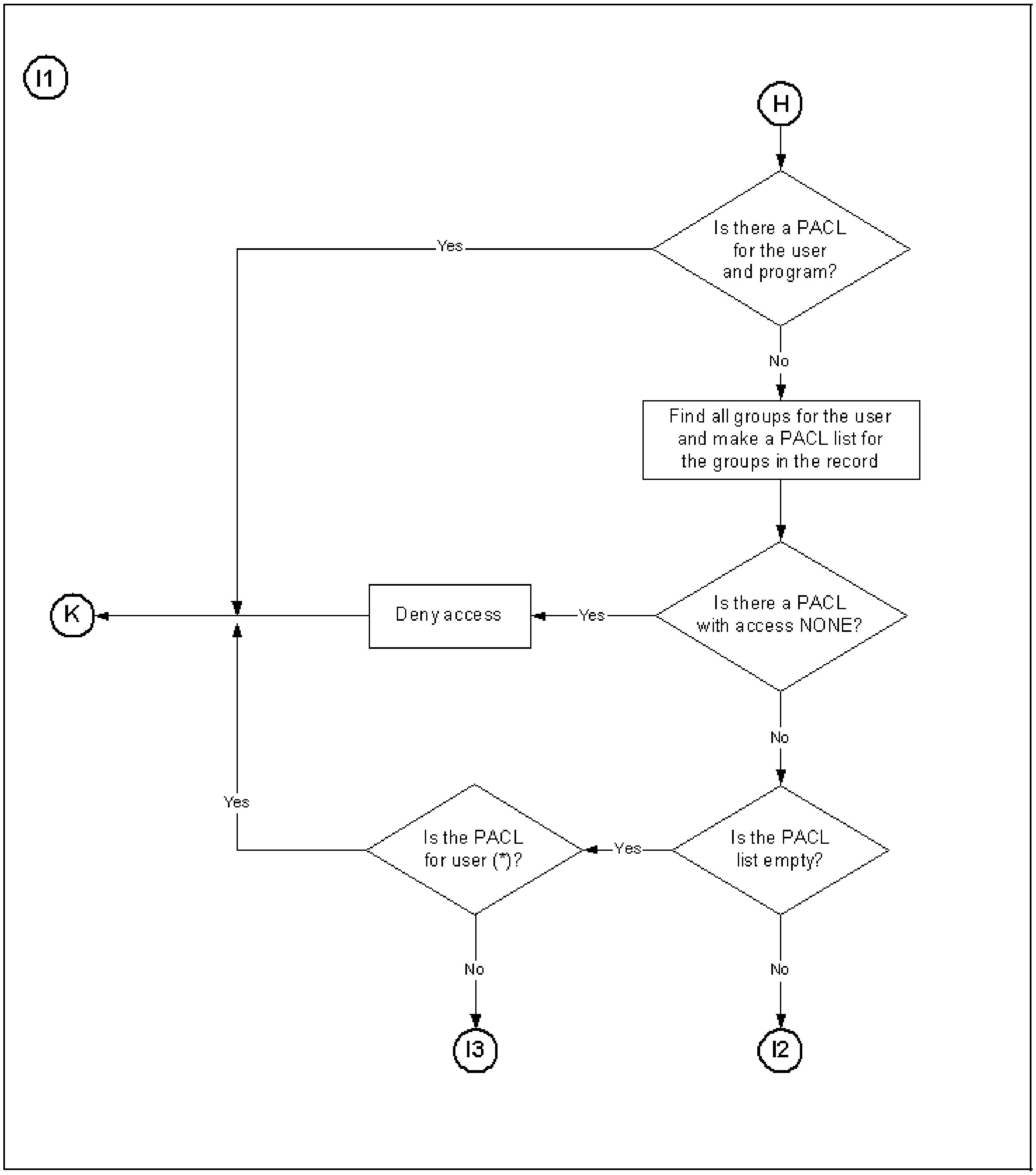

E
D
From the database resource record, make a list of ACLs for all groups that the user belongs to.
Is there an ACL with access NONE for the user's group?
Yes
Deny access
No
G
Yes
Is there a deny/allow verdict?
No
F1
Is there an ACL for the groups?
No
Yes
Are accumulated group rights ON?
No
Does user's access match the first access in the list?
No
Deny access
Yes
Allow access
G
Yes
Accumulate all access levels from all ACLs in the list
Is user's access in the list?
Yes
No
Deny access

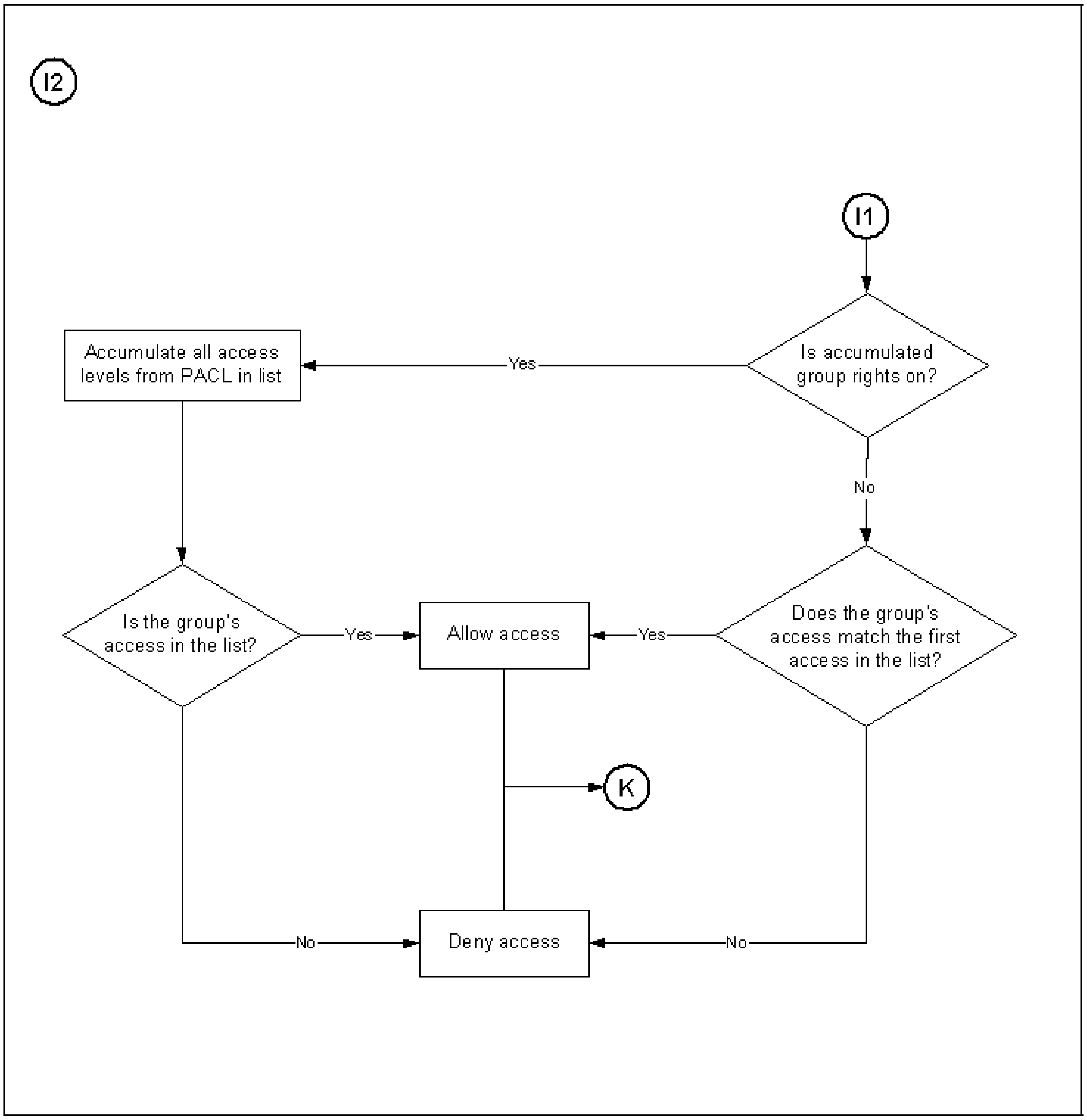

F1
E
Is there a resource group for the resource?
Yes
F2
No
Was the user found in the database?
No
G
Yes
Does the resource have an ACL for the the general user (*)?
Yes
G
No
Does the resource belong to a resource group?
Yes
E
No
Does the resource group have an ACL for the general user (*)?
Yes
G
No
Is there another resource group?
Yes
No
E

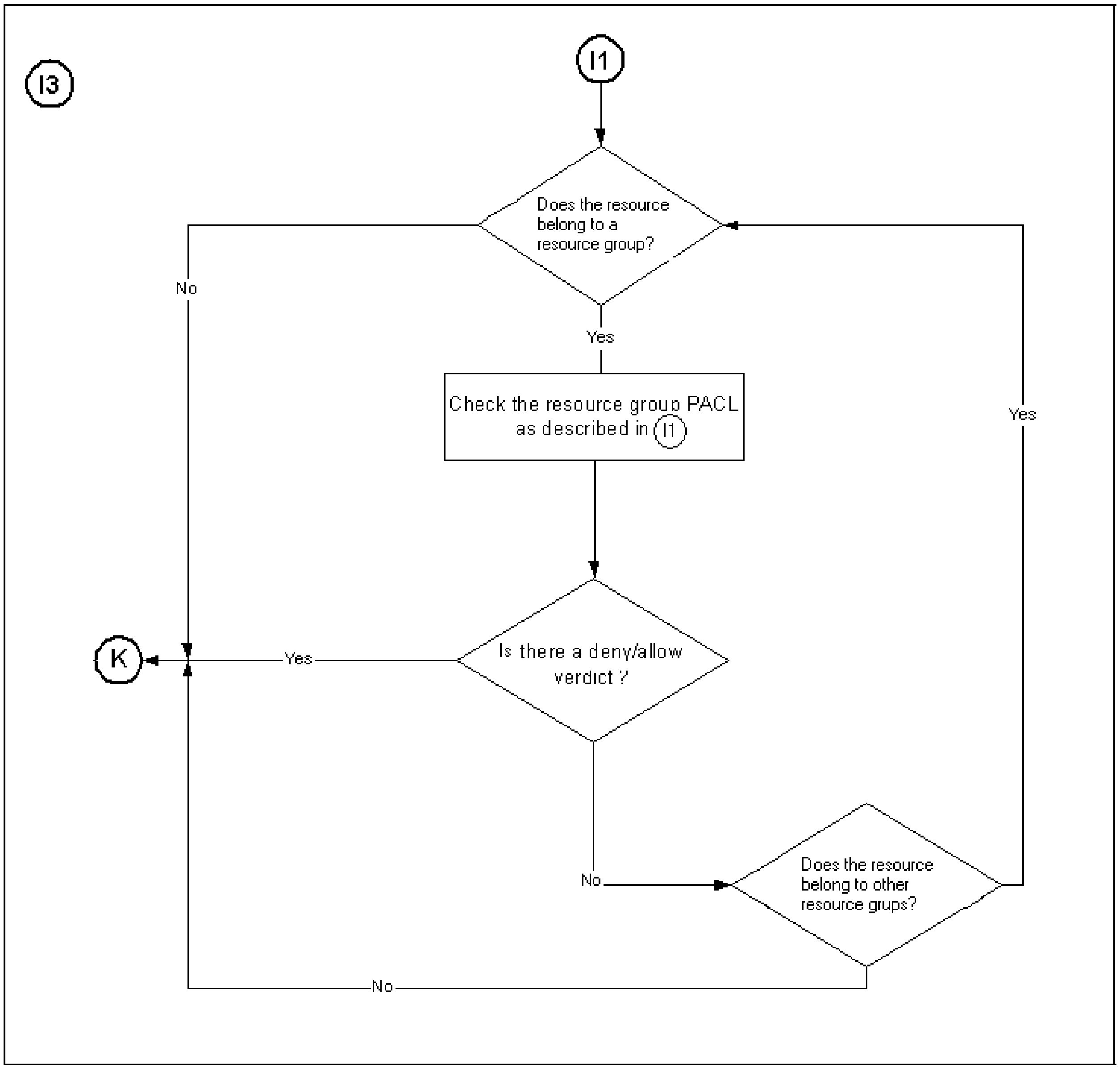

F2
F1
Build a list of all ACLs for the groups in the database resource group
Is there an ACL for the groups?
No
F1
Yes
Are accumulated group rights ON?
No
Does group's access match the first access in the list?
No
Deny access
Yes
Allow access
G
Yes
Accumulate all access levels from all ACLs in the list
Is group's access in the list?
Yes
No
Deny access

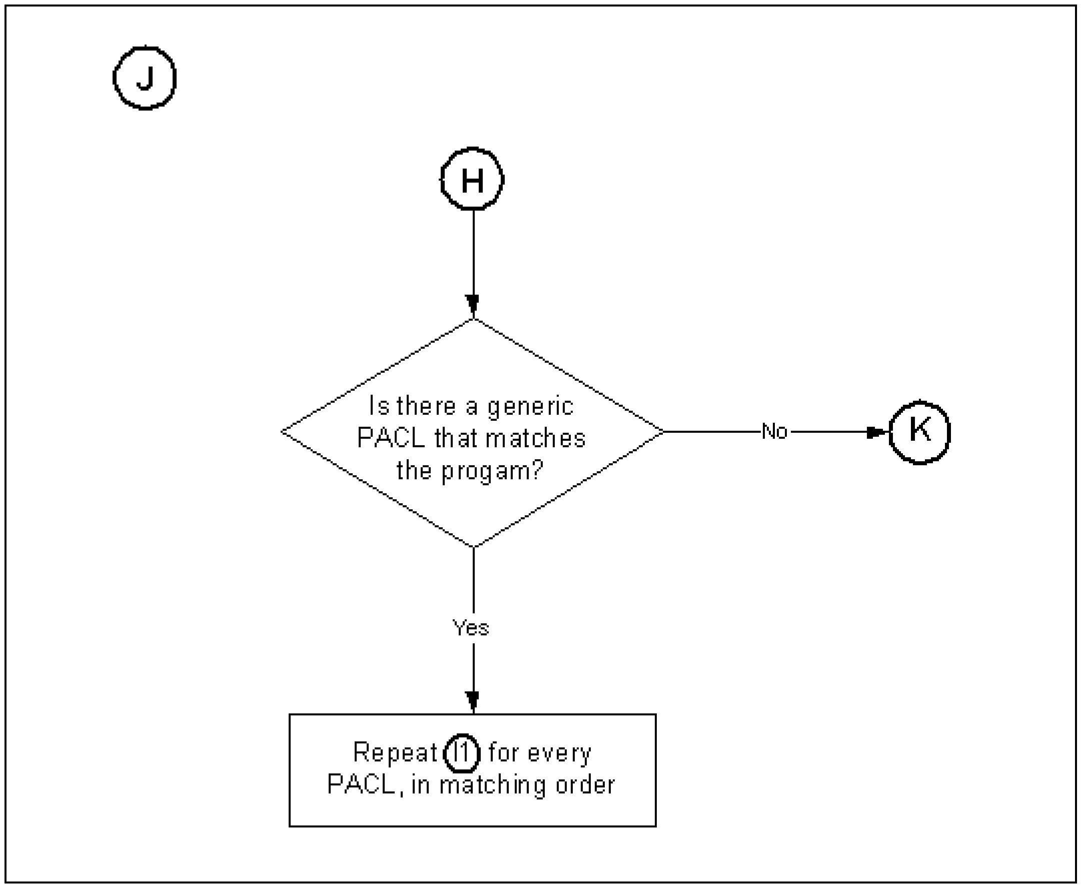

G
*
Is account running under PMDB?
Yes
No
Is accumulated "PACL ACL" on?
No
Yes
Is there a deny/allow verdict?
No
Yes
Is the verdict "deny," and there is no ACL with access NONE?
No
Yes
K
H

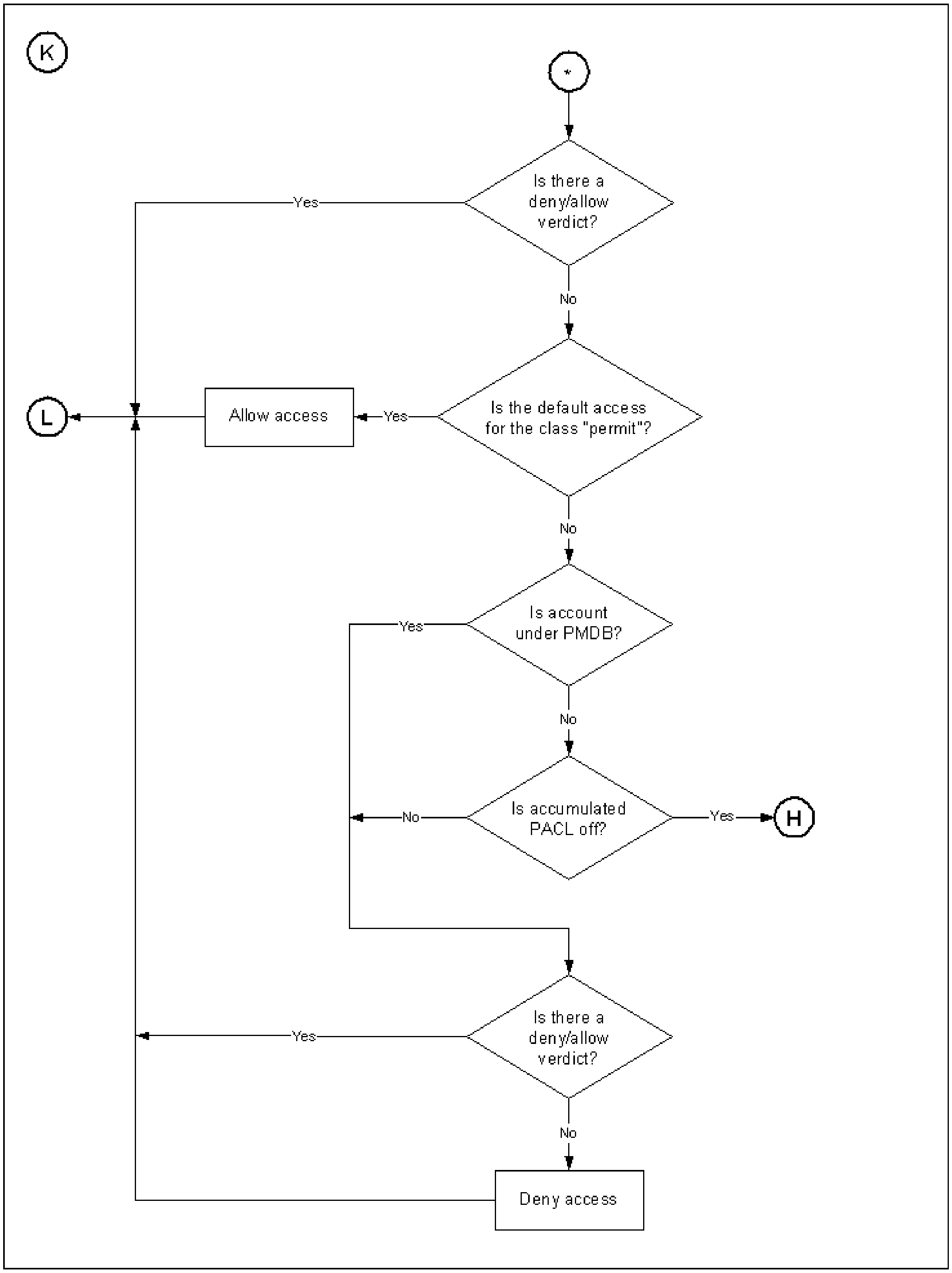

H
G
Find the accessing program in the PROGRAM class of the database
Is the PROGRAM class active and the program trusted?
No
K
Yes
Is there a discrete record for the program in the PROGRAM class?
Yes
I1
No
J

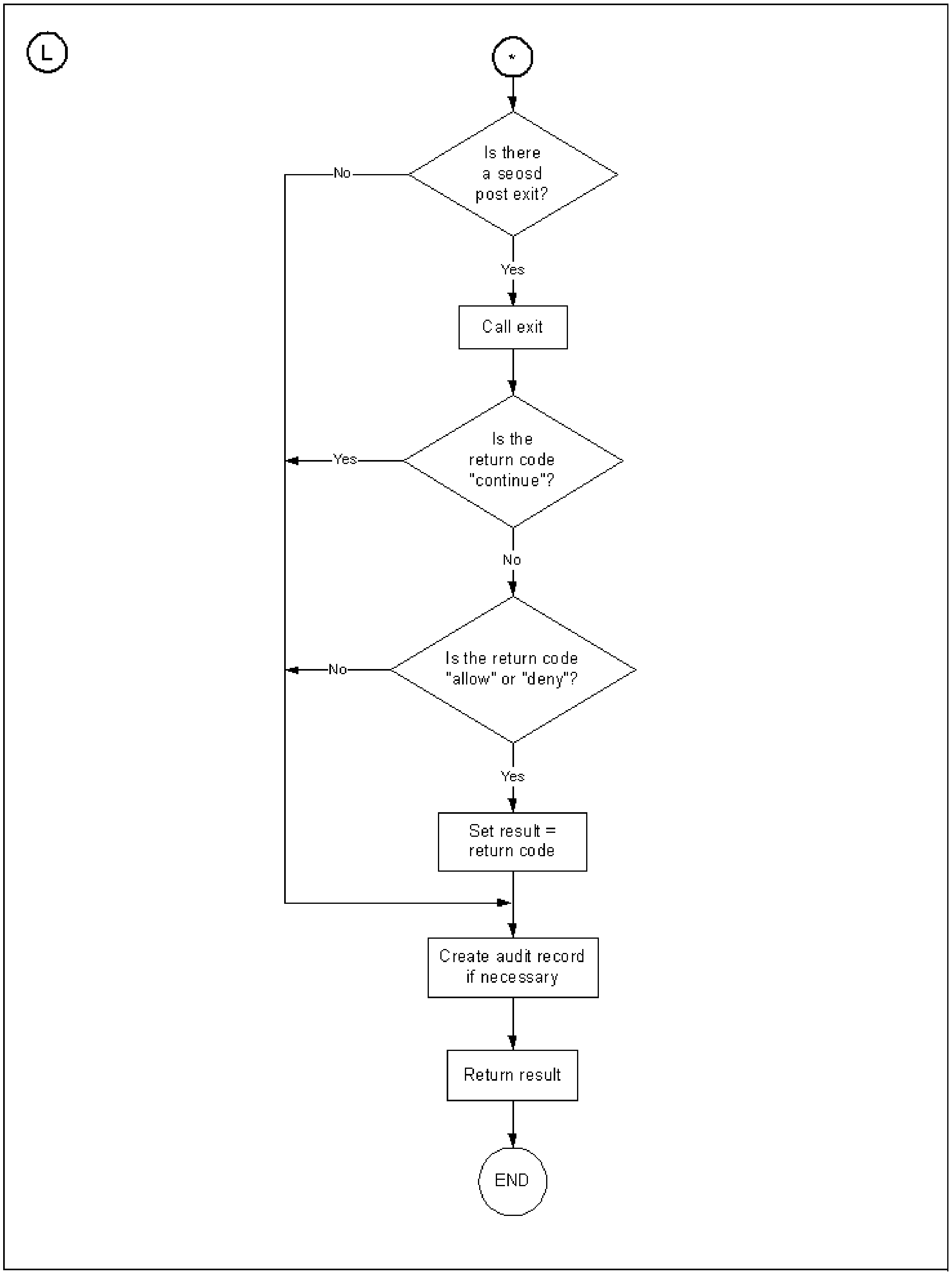

I1
H
Is there a PACL for the user and program?
Yes
No
Find all groups for the user and make a PACL list for the groups in the record
K
Deny access
Yes
Is there a PACL with access NONE?
No
Yes
Is the PACL for user (*)?
Yes
Is the PACL list empty?
No
No
I3
I2


I2
I1
Is accumulated group rights on?
Yes
Accumulate all access levels from PACL in list
No
Is the group's access in the list?
Yes
Allow access
Yes
Does the group's access match the first access in the list?
K
No
Deny access
No


I3
I1
Does the resource belong to a resource group?
No
Yes
Check the resource group PACL as described in I1
Yes
K
Yes
Is there a deny/allow verdict ?
No
Does the resource belong to other resource grups?
No


J
H
Is there a generic PACL that matches the progam?
No
K
Yes
Repeat I1 for every PACL, in matching order


K
*
Is there a deny/allow verdict?
Yes
No
Is the default access for the class "permit"?
Yes
Allow access
L
No
Is account under PMDB?
Yes
No
Is accumulated PACL off?
No
Yes
H
Is there a deny/allow verdict?
Yes
No
Deny access


L
*
Is there a seosd post exit?
No
Yes
Call exit
Is the return code "continue"?
Yes
No
Is the return code "allow" or "deny"?
No
Yes
Set result = return code
Create audit record if necessary
Return result
END

Chapter 3 Users, Groups, and Resources

This chapter introduces you to accessors and resources and discusses the concepts needed to understand and work with them. The two types of accessors—users and groups—are discussed in detail.

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. For 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:

```
eTrust> 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 (see the chapter "The selang Command Language" in the *Reference Guide*). 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.
- Security Administrator, the UNIX-based interface for eTrust AC (see the *User Guide*.)
- Policy Manager, the Windows-based interface (see the *User Guide*).

Examples

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:

   ```
   eTrust> 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:

   ```
   eTrust> 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:

   ```
   eTrust> rmusr Terry
   ```

Special Predefined Users

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

_seagent
A logical user associated with the seagent daemon that allows access to ladb files through selang exits. 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, runs; therefore, you cannot assign another logical user to sepmdd.

_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
The eTrust AC command language (see the chapter “The selang Command Language” in 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 (see the *User Guide.*)

Policy Manager
The Windows-based interface for eTrust AC (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

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:

   ```
   eTrust> 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:

   ```
   eTrust> chgrp sales name('West Coast Sales Dept')
   ```

3. To add members to a group, use the **join** command. For example, to make users Terry and Jack members of group "sales," enter the following command:

   ```
   eTrust> join (Terry Jack) group(sales)
   ```

4. To give Terry group auditor privileges in the group "sales," enter:

   ```
   eTrust> join (Terry) group(sales) auditor
   ```

5. To remove a user from a group, use the **join-** command. For example, to remove Jack from group "sales," enter the following command:

   ```
   eTrust> join- (Jack) group(sales)
   ```

6. To remove a group from the database, use the **rmgrp** command. For example, to delete group "sales," enter the following command:

   ```
   eTrust> rmgrp (sales)
   ```

The following section introduces you to resources and discusses the concepts needed to understand and work with resources.

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. See the *SDK Developer Guide* for a description of eTrust AC APIs.

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).

Setting ACLs

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

Examples

- 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:

  ```
  eTrust> 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:

  ```
  eTrust> 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:

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

For more information, see the chapter "The selang Command Language" in the *Reference Guide*.

Classes

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.

Class Type	Purpose
Accessor	Defines objects that access resources.
Definition	Defines objects that define security entities, such as security labels and categories.
Installation	Defines objects that control the behavior of eTrust AC.
Resource	Defines objects that are protected by access rules.

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

Class Name	Class Type	Description
ADMIN	Definition	Use this class to delegate administrative responsibilities to users who do not have the ADMIN attribute. (For more about administrative responsibilities, see Global Authorization Attributes in the chapter "Scope of Administration Authority.")
AGENT	Resource	Each record in this class defines an object that is used as an agent by eTrust SSO or eTrust™ Web Access Control (eTrust Web AC).
AGENT_TYPE	Resource	Each record in the AGENT_TYPE class defines an agent type used by eTrust SSO or eTrust WebAC.
APPL	Resource	Each record in the APPL class defines an application used by eTrust SSO or eTrust WebAC.
AUTHHOST	Accessor	Each record in the AUTHHOST class defines an authentication host in eTrust Web AC or SSO.
CALENDAR	Resource	Each record in the CALENDAR class defines a Unicenter® TNG calendar object for user, group, and resource enforced time restrictions in eTrust AC.
CATEGORY	Definition	Each record in this class defines a security category.
CONNECT	Resource	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. For more information, see Using the TCP Class in the chapter "Protecting TCP/IP Services."

Class Name	Class Type	Description
CONTAINER	Resource	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.
FILE	Resource	Each record in this class defines a file, a directory, or a file name mask.
GAPPL	Resource	Each record in the GAPPL class defines a group of applications used by eTrust Web AC or SSO.
GAUTHHOST	Definition	Each record in the GAUTHHOST class defines a group of authentication hosts used by eTrust Web AC or SSO.
GFILE	Resource	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.
GHOST	Resource	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.
GROUP	Accessor	Each record in this class defines a group of users.
GSUDO	Resource	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.
GTERMINAL	Resource	Each record in this class defines a group of terminals.
HOLIDAY	Definition	Each record in this class defines one or more periods when users need extra permission to log in.

Class Name	Class Type	Description
HOST	Resource	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. For more information, see Using the TCP Class in the chapter "Protecting TCP/IP Services."
HOSTNET	Resource	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.
HOSTNP	Resource	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.
LOGINAPPL	Definition	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.
MFTERMINAL	Definition	Each record in the MFTERMINAL class defines a Mainframe computer that is used to administer eTrust AC.
PROCESS	Resource	Each record in this class defines an executable file.
PROGRAM	Resource	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.

Class Name	Class Type	Description
PWPOLICY	Definition	Each record in the PWPOLICY class defines a password policy.
RESOURCE_DESC	Definition	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.
RESPONSE_TAB	Definition	Each record in the RESPONSE_TAB class defines an eTrust Web AC response table to different authorization decisions.
SECFILE	Definition	Each record in this class defines a file that must not be altered.
SECLABEL	Definition	Each record in this class defines a security label.
SEOS	Installation	The one record in this class specifies your active classes and password rules.
SPECIALPGM	Installation	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.
SUDO	Resource	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.
SURROGATE	Resource	Each record in this class protects a user or group from surrogate requests issued by other users.
TCP	Resource	Each record in this class defines a TCP/IP service, such as mail or http or ftp.
TERMINAL	Resource	Each record in this class defines a terminal—a device from which a user can log in.
UACC	Resource	Defines default access rules for each resource class.

Class Name	Class Type	Description
USER	Accessor	Each record in this class defines a user to eTrust AC.
USER_ATTR	Definition	Each record in the USER_ATTR class defines the valid user attributes of an eTrust Web AC user directory.
USER_DIR	Resource	Each record in the USER_DIR class defines an eTrust Web AC user directory.

eTrust AC classes, USER and GROUP classes, are described in the *Reference Guide*. To find out how to set access control lists, see Setting ACLs in this chapter.

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.
4. 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 chapter shows you how to protect user and root accounts from unauthorized access.

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. If in the initial stage you wish to grant access to any su request, use the following command:

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> 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.”

Preventing Bypasses

See Recommended Restrictions in the chapter “Controlling Login Commands” for a discussion on preventing loopback and local host bypasses of the surrogate access rules.

See Protecting the Login Command in the chapter “Protecting Files and Programs” for a discussion on preventing exec login bypasses of the surrogate access rules.

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:

```
eTrust> 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

WARNING! *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:

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> sesudo MountCd
```

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.

For more information, see the sesudo utility in 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: serevu and pam_seos.

Another method of protecting against password attacks is by setting password policy rules; for more details, see the chapter "Password Control."

serevu

The serevu 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 (serevu never revokes root, so the system is never locked out).

To prevent this, the serevu 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.

See the *Utilities Guide* for a complete discussion of the serevu daemon.

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

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 and NCR, 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:

```
eTrust> 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:

```
eTrust> {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:

```
eTrust> {chusr | editusr | newusr} userName inactive (numdays)
```

To reactivate a suspended user account, use the following command:

```
eTrust> {chusr | editusr} userName resume
```

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

```
eTrust> {chgrp | editgrp} userName resume
```

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

```
eTrust> setoptions inactive-
```

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

```
eTrust> {chgrp | editgrp} groupName inactive-
```

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

```
eTrust> {chusr | editusr} userName inactive-
```

For information about setting inactivity in the Security Administrator, see the chapter "Account Administration" in the *User Guide*.

Chapter 5

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.

This chapter discusses password protection policies and shows you how to set, change, and control passwords and their rules.

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:

   ```
   eTrust> setoptions class+ (PASSWORD)
   ```

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

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

   The *rule* parameter specifies one or more rules. For the syntax of the rule settings, see the chapter “The selang Command Language” in the *Reference Guide*.

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

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

For information about changing passwords with the Security Administrator, see the chapter "Account Administration" in 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

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:

```
eTrust> 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:

```
eTrust> chgrp MailRoom interval(30)
```

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

```
eTrust> setoptions password(norules)
```

To learn more 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 is reached (see Specifying Grace Logins in this chapter). 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:

```
eTrust> 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:

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

To set or cancel the password interval for a group:

```
eTrust> {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.

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.

For more information on 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:

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

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

```
eTrust> chusr userName {grace(nLogins) | grace-}
```

To set or cancel grace logins for a profile group, enter the following command:

```
eTrust> 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.

Chapter 6

Protecting Files and Programs

This chapter shows you how to restrict access to files, directories, programs, and processes, and how to monitor sensitive files for unauthorized changes.

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)

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 trough 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, all remaining files can also be protected. (See Special Predefined Groups in the chapter "Users, Groups, and Resources.")

- 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. For details, see the FILE class in the *Reference Guide*.

eTrust AC supports the following access types for files. For explanations, see the ADMIN class in the *Reference Guide*.

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

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. (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:

```
eTrust> newres FILE filename
```

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

```
eTrust> 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):

```
eTrust> 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. For more details, see the chapter "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:

```
eTrust> 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.*

Tip: 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. For more information, see "Resource Administration" in 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:

```
eTrust> 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:

```
eTrust> 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. For details, 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:

```
eTrust> 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. For details, see the chapter “Resource Administration” in the *User Guide*.

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.

Tip: 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.

```
eTrust> newres FILE /etc/passwd owner(nobody) defaccess(R)
eTrust> 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:

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

eTrust> 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 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:

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

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

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

For more details about the Unicenter TNG calendar, see the chapter "Unicenter Security Migration and Integration."

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)]
```

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, HP-UX, and Tru64. For platform-specific limitations, see the HP-UX Limitations and Sun Solaris Limitations sections in this chapter.

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.

Synchronization Example

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:

   ```
   eTrust> 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. For more information about seoswd, see the chapter “Utilities in Detail” in 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. For a description of the seos.ini file, see the appendix “The seos.ini Initialization File.” 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:

```
eTrust> 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. For more information, see the chapter "Resource Administration" in 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:

```
eTrust> 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):

```
eTrust> newres PROGRAM /bin/su defaccess(NONE)
eTrust> 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:

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

This description is also true for setgid executables.

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. For a description of 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 Using Conditional Access Control Lists in this chapter.

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.)

```
eTrust> 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:

```
eTrust> 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.

For more information, see the *User Guide*.

Protecting Regular Programs

eTrust AC can also protect regular programs in the same way it protects setuid and setgid programs. To do this, set the blockrun property in the PROGRAM class to the value you choose. For possible options, see the Reference Guide.

For more details, see Protecting setuid and setgid Programs.

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.

Protecting Binary Files from the kill Command

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:

   ```
   eTrust> 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:

      ```
      eTrust>  /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:

```
eTrust> 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:

```
eTrust> 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 chapter shows you how to restrict a user's ability to log on to a system protected by eTrust AC.

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. For complete details and examples, see the TERMINAL class in 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.

LOGINAPPL Examples

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:

```
eTrust> cr LOGINAPPL FTP defaccess(NONE) owner(nobody)
```

2. Permit the user anonymous to use ftp with the following command:

```
eTrust> 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:

```
eTrust> auth LOGINAPPL(RLOGIN RSH) gid(account) access(N)
```

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

```
eTrust> 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.

For details 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:

```
eTrust> 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:

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

To permit a user to log in from a particular terminal, enter the following command:

```
eTrust> 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:

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

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

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

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

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

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

```
eTrust> 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.

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:

```
eTrust> newres TERMINAL loopback defaccess(N) owner(nobody)
eTrust> newres TERMINAL localhost defaccess(N) owner(nobody)
eTrust> 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. To learn more 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. For more information about the segrace command, see the chapter "Utilities in Detail" in 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:

```
eTrust> 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:

```
eTrust> 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. (For information about the Activity window, see the chapter "Security Administrator Basics" in the *User Guide*.)

Limiting Concurrent Logins Individually

In selang, enter the following command:

```
eTrust> chusr username maxlogins(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. (For information about the Activity window, see the chapter “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. (For information about the Activity window, see the chapter “Security Administrator Basics” in the *User Guide*.)

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.

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

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:

```
eTrust> 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:

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

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

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> newres GHOST gh1 mem(ws2, ws3, ws5)
eTrust> 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:

```
eTrust> newres HOSTNET hn1 mask(255.255.0.0) match(192.168.0.0)
eTrust> authorize HOSTNET hn1 service(*) access(NONE)
eTrust> 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. For information on how eTrust AC performs string matching, see the appendix "String Matching" in 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:

```
eTrust> newres HOSTNP lin*.org.com
eTrust> 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. (See the TCP class in 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.

```
eTrust> newres TCP ftp defaccess(READ)
eTrust> 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:

```
eTrust> newres HOST hermes
eTrust> newres TCP ftp owner(nobody) defaccess(read)
eTrust> newres TCP telnet owner(nobody) defaccess(read)
eTrust> authorize TCP ftp uid(*) host(hermes) access(write)
eTrust> 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 HP-UX and 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
   ```

To load eTrust AC streams module on NCR, SINIX, or UnixWare, complete the following steps:

1. Stop eTrust AC. Enter the following command:

   ```
   secons -s
   ```

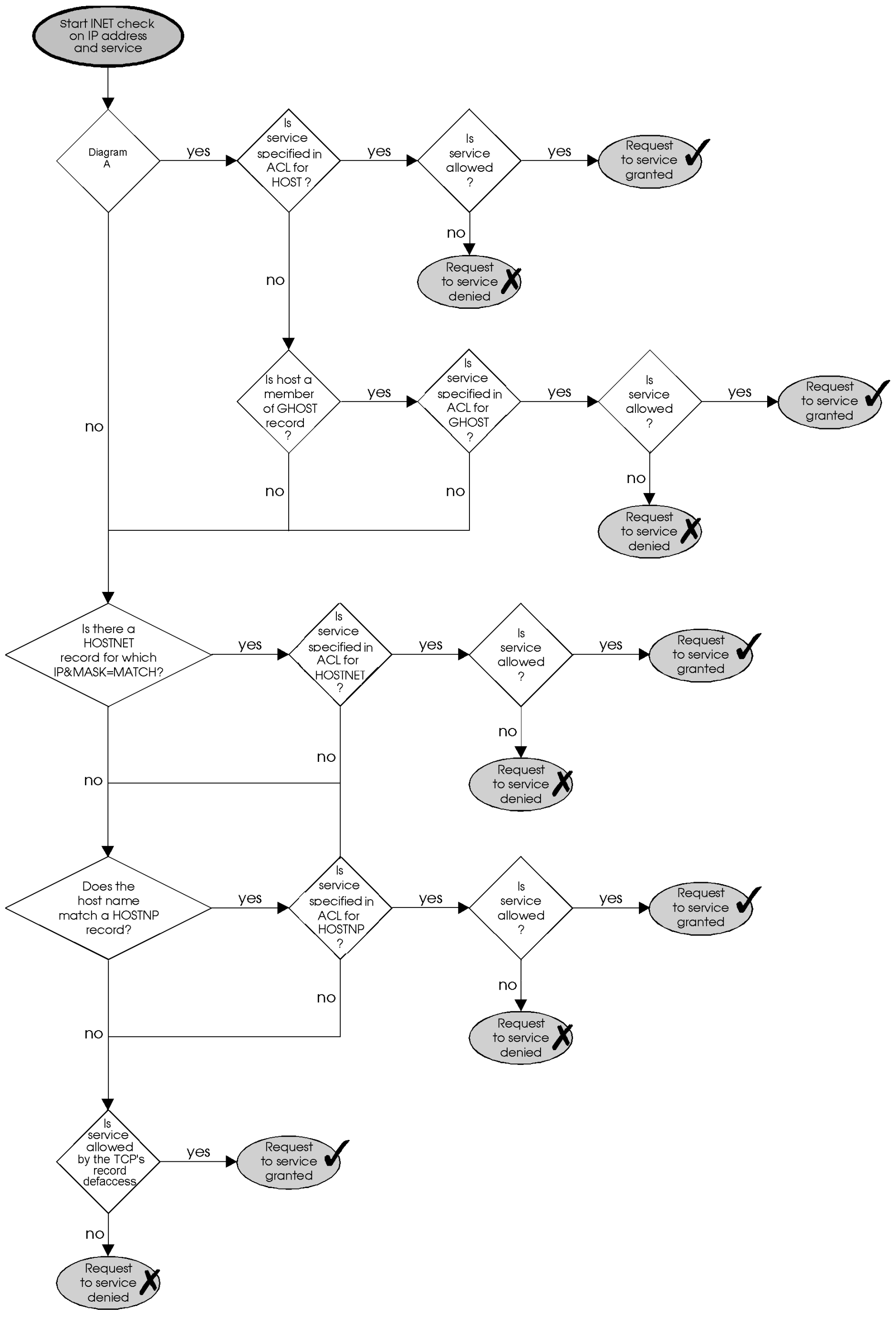
2. Set the token SESO_use_streams to yes in the seos.ini.
3. Reboot the machine.
4. Start eTrust AC. Enter the following command:

   ```
   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:


Start INET check on IP address and service
Diagram A
yes
Is service specified in ACL for HOST ?
yes
Is service allowed ?
yes
Request to service granted
no
Request to service denied
no
no
Is host a member of GHOST record ?
yes
Is service specified in ACL for GHOST ?
yes
Is service allowed ?
yes
Request to service granted
no
no
no
Request to service denied
Is there a HOSTNET record for which IP&MASK=MATCH?
yes
Is service specified in ACL for HOSTNET ?
yes
Is service allowed ?
yes
Request to service granted
no
Request to service denied
no
no
Does the host name match a HOSTNP record?
yes
Is service specified in ACL for HOSTNP ?
yes
Is service allowed ?
yes
Request to service granted
no
Request to service denied
no
no
Is service allowed by the TCP's record defaccess
yes
Request to service granted
no
Request to service denied

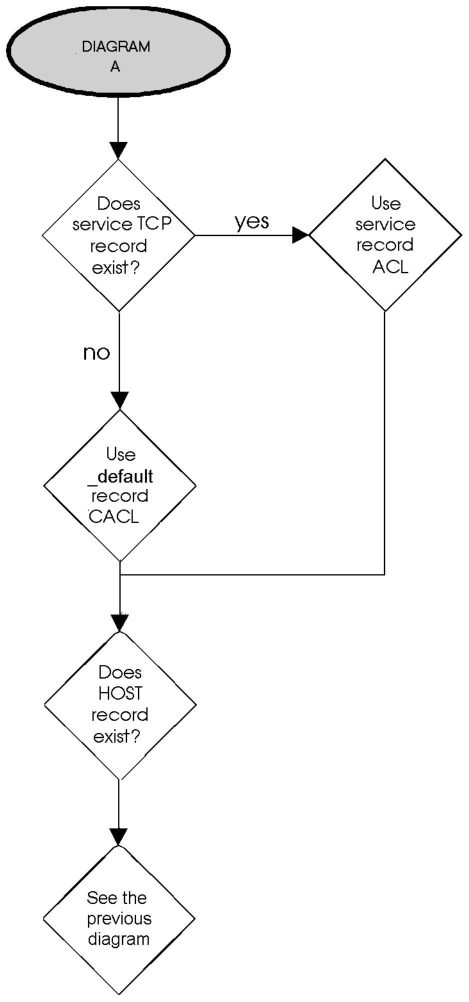
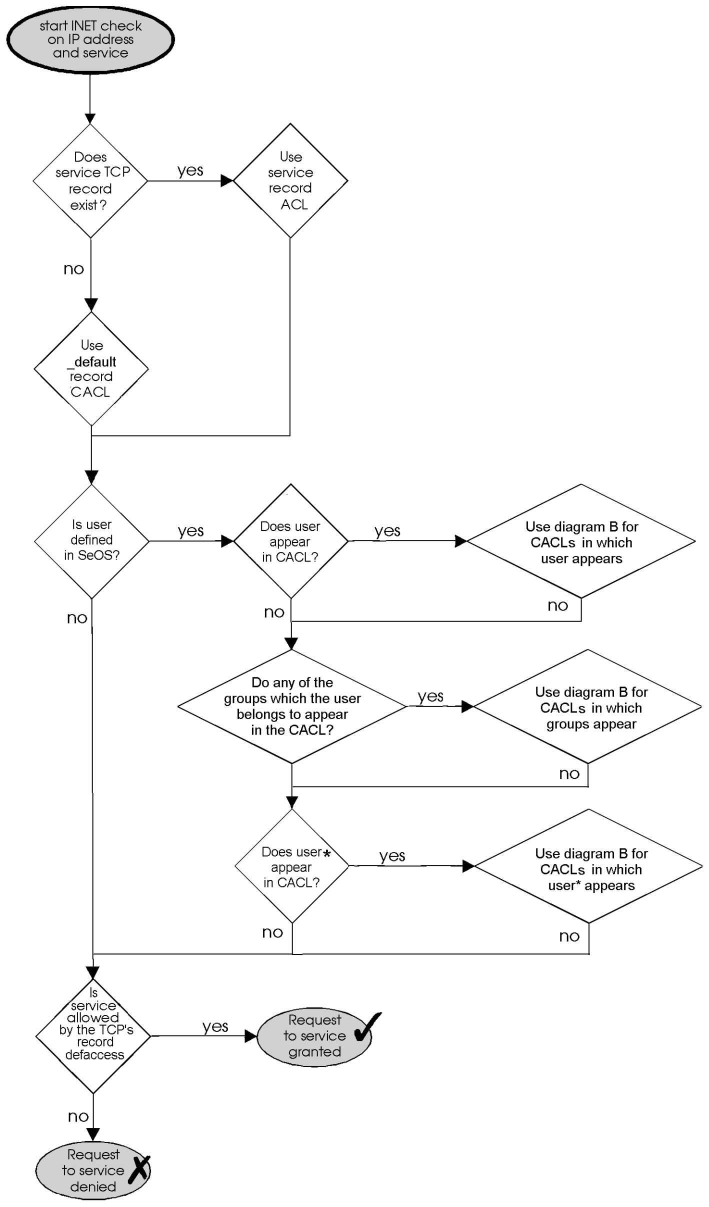


DIAGRAM A
Does service TCP record exist?
yes
Use service record ACL
no
Use _default record CACL
Does HOST record exist?
See the previous diagram

The algorithm for outgoing authorizations is:


start INET check on IP address and service
Does service TCP record exist?
yes
Use service record ACL
no
Use _default record CACL
Is user defined in SeOS?
yes
Does user appear in CACL?
yes
Use diagram B for CACLs in which user appears
no
no
no
Do any of the groups which the user belongs to appear in the CACL?
yes
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
no
Is service allowed by the TCP's record defaccess
yes
Request to service granted ✓
no
Request to service denied ✗

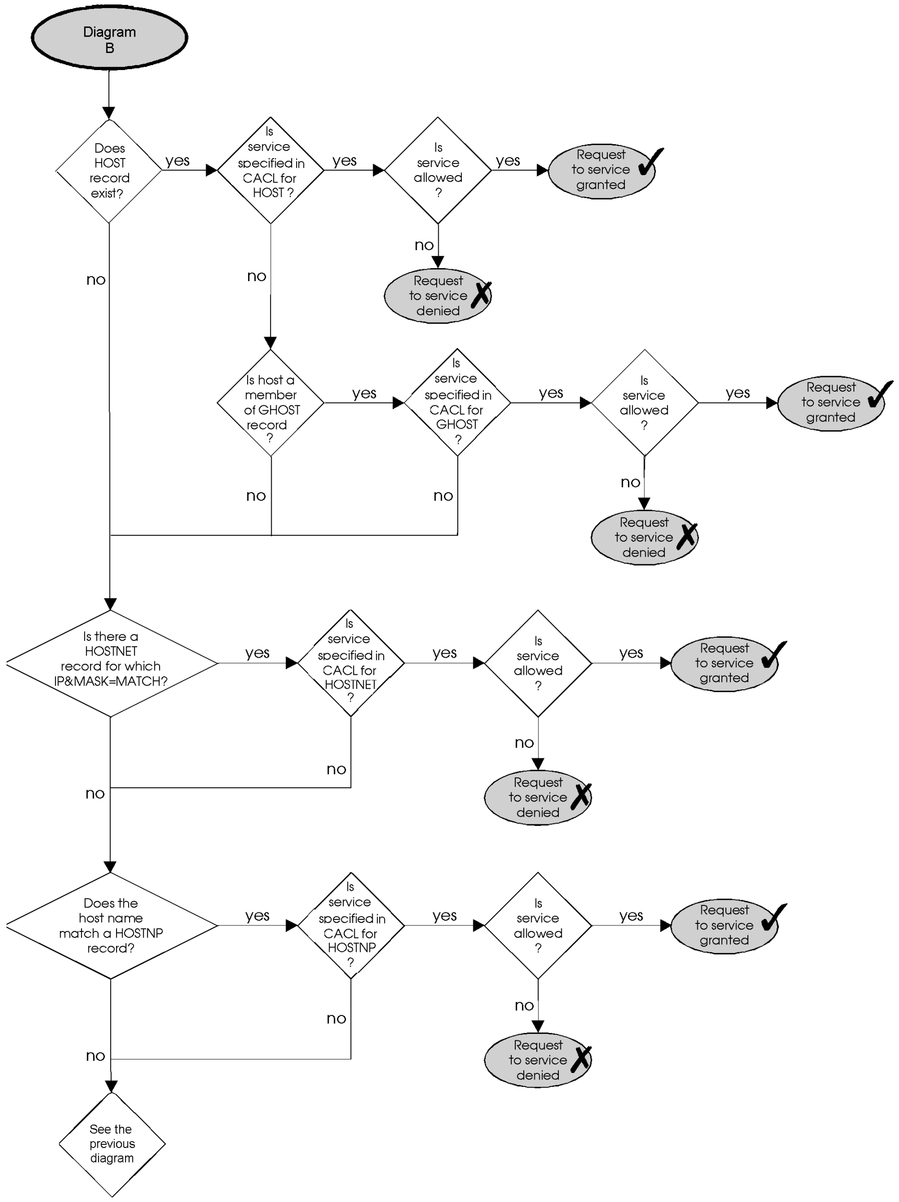


Diagram B
Does HOST record exist?
Is service specified in CACL for HOST ?
Is service allowed ?
Request to service granted
Request to service denied
Is host a member of GHOST record ?
Is service specified in CACL for GHOST ?
Is there a HOSTNET record for which IP&MASK=MATCH?
Is service specified in CACL for HOSTNET ?
Does the host name match a HOSTNP record?
Is service specified in CACL for HOSTNP ?
See the previous diagram
yes
no

Chapter 9

Managing the Policy Model Database

eTrust AC allows the management of several databases from one single central database called a Policy Model Database (PMDB). The rules you define in the central database are propagated to the other databases, called subscribers.

This chapter describes the PMDB and shows you how to set it up, and how to create and maintain subscriber databases.

About the Policy Model Database

A PMDB that updates a subscriber is called the subscriber's *parent*. The security administrator need only update the central database to change all the other databases. The PMDB is a useful tool for managing many stations that are similar or identical in terms of authority restrictions and access rules.

The PMDB is a regular eTrust AC database with the addition of a list of subscribers. Each subscriber is an eTrust AC database that resides on a separate computer. Whenever a change is made to the PMDB, the subscriber databases are automatically updated.

For example, if a user is deleted from the PMDB using the rmusr command, the same rmusr command is sent to all the subscriber databases. In this way, a single rmusr command removes a user from many databases on a variety of computers.

If a subscriber database does not respond, the PMDB sends the command every 30 minutes until the subscriber database has been updated. Or you can update subscriber databases as soon as they become available, by setting the pull_option token to yes in the [pmd] section of the subscriber database's seos.ini file. (See the appendix "The seos.ini Initialization File.")

If a subscriber database is responding, but refuses to apply the command, the PMDB places the command in an error log. (See The Policy Model Error Log in this chapter).

A subscriber database can also be a PMDB for other subscribers. This allows for a hierarchical configuration of PMDBs.

All PMDBs reside in a common directory (one per workstation). 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 Policy Model occupies a subdirectory in the common directory. The name of the subdirectory is the name of the Policy Model. The subdirectory contains all the data required to define the Policy Model, including the pmd.ini file. For more information on the pmd.ini file, see the appendix "The pmd.ini File."

A PMDB can have only one parent Policy Model. The name of the parent Policy Model, if one exists, is specified in the parent_pmd token in the [seos] section of the pmd.ini file. If this token is blank, the PMDB does not accept updates from any other PMDB.

Setting Up a PMDB

Setting up a PMDB involves the following procedures:

- Creating the PMDB
- Defining host databases as subscribers of the PMDB

The following sections show how to set up a PMDB. For a comprehensive discussion of the Policy Model utilities, see the *Utilities Guide*.

Creating a PMDB Locally

To create a PMDB on a local host, you can use the sepmdadm command. The following example shows the interactive (dialog-based) form of the command. An alternative form, described in the *Utilities Guide*, uses command-line parameters for all input.

This example creates a PMDB named policy1, registers the workstations workstat1 and workstat2 as its subscribers, authorizes two users (adm1 and adm2) to administer the PMDB, registers one auditor, and specifies that the PMDB must be administered from the terminal called pmdbhost.com.

1. Issue the sepmdadm command with the -i parameter from the shell prompt:

   ```
   /opt/CA/eTrustAccessControl/bin/sepmdadm -i
   ```

2. eTrust AC displays a menu of options. Select the first option: Create a master PMDB and define its subscribers.
3. eTrust AC asks for the name of the Policy Model. Enter policy1

4. eTrust AC asks you to define a subscriber. Enter workstat1.
5. eTrust AC asks for the next subscriber. Enter workstat2.
6. eTrust AC asks for the next subscriber. Press Enter to indicate no more subscribers at this time.

   You still must perform a complementary procedure at the subscriber stations. (The procedure is described in Pointing Subscriber Workstations to the PMDB in this chapter.)
7. If you are running NIS, NIS+, or DNS, eTrust AC asks whether you want to update the NIS/DNS tables. Updates are made to users and groups in the PMDB. The tables provide information on users and their characteristics. Answer with y if you want the tables updated. eTrust AC now asks for the full path of the NIS password file and the full path of the NIS group file. If you provide this, a UNIX user or UNIX group updated through the Policy Model is also updated in the NIS passwd and group files. If you do not want the tables updated, either enter **n**, or just press Enter.
8. eTrust AC asks whether you want to define users with special attributes for the PMDB.

   - The administrators of a PMDB are users authorized to change the properties of the PMDB. At least one administrator must be defined in a PMDB.

     eTrust AC asks you to enter the name of an administrator. Enter adm1.

     eTrust AC asks for the next administrator. Enter adm2.

     eTrust AC asks for the next administrator. Press Enter to indicate that you have no more administrators to register at this time.
   - The auditors of a PMDB are users authorized to view the PMDB's audit log files.

     eTrust AC asks you to enter the name of an auditor. Enter policy1_auditor.

     eTrust AC asks for the next auditor. Press Enter to indicate that you have no further auditor to register at this time.
   - The password managers of a PMDB are users authorized to change passwords in the PMDB.

     Press Enter only, to indicate that you have no password manager to register at this time.
9. eTrust AC asks you to enter the name of an administration terminal. Enter pmdb_host.com.
10. eTrust AC asks for the next administration terminal. Enter workstat1.com.
11. eTrust AC asks for the next administration terminal. Press Enter to indicate that you have no administration terminals to register at this time.

12. eTrust AC reports the selections you have made and asks you to confirm them. Check the report before you answer. If you confirm your selections, the PMDB is built using the answers you supplied.

Creating and Deleting a PMDB Remotely

In addition to managing PMDB remotely, you can now create and delete PMDBs on a remote station to which you are connected. Two new commands perform these operations: createpmd and deletepmd.

Before you can create or delete a PMDB remotely, you must:

- Have permission to use the selang command language
- Have permission to connect to remote hosts
- Have at least PMDB administrator privileges
- Connect to the remote station with the hosts command
- Switch to the pmd environment with the env pmd command

To create a PMDB remotely, enter the following command:

```
eTrust> createpmd pmdname [options]
```

To delete a PMDB remotely, enter the following command:

```
eTrust> deletepmd pmdname
```

where *pmdname* is the name of the PMDB you are creating or deleting. For a description of these commands, see the createpmd and deletepmd commands in the *Reference Guide*.

Pointing Subscriber Workstations to the PMDB

To establish a workstation as a subscriber to a PMDB, you must do more than register the subscriber's name at the PMDB's station. You must complete a procedure at the subscriber station. (You can accomplish this extra procedure in multiple ways.)

- At the subscriber station, you can issue the following command and use the instructions that then appear on your screen.

  ```
  sepmdadm -i
  ```

- Or, use the sepmd command at the subscriber station. Suppose, for example, that you have defined the PMDB pmdb2 on the station stat3 and that the PMDB policy1 is defined on the local host. You can issue the following command to define pmdb2 as a subscriber of policy1:

  ```
  sepmd -s policy1 pmdb2@STAT3
  ```

By subscribing several stations to the same PMDB, and by subscribing one PMDB to another, you can create a hierarchy of PMDBs.

UID/GID Synchronization

As an administrator, you may receive messages that refer to users by UID and to groups by GID. 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.

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

Synchronizing Your Files

To allow the lists of users and groups in your various databases to 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 ensure that the lists are identical before you start working with your PMDB and subscribers, complete the following steps:

1. Copy your /etc/passwd file and /etc/group file to your Policy Model directory. This is a one-time procedure. It destroys any previous passwd and group files in your Policy Model directory.

   Note: If you are using a shadow file and want to synchronize passwords, we recommend using the secrepsw utility. (See the Utilities Guide for information about this utility.)

   The Policy Model directory is *policies***/***pmdbname*, where *policies* is the value of the _pmd_directory_ token in the pmd section of the seos.ini file and *pmdbname* is the name of the particular PMDB.

   For example, if _pmd_directory_=/opt/CA/eTrustAccessControl/policies and your PMDB is named mfg_group, then you should enter:

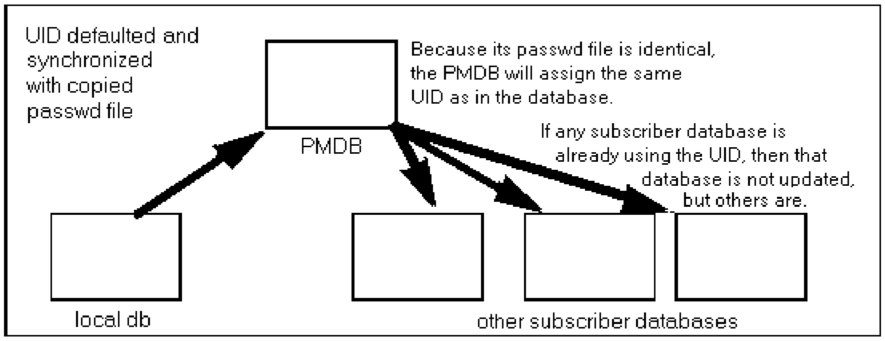
   ```
   # cp /etc/passwd /opt/CA/eTrustAccessControl/policies/mfg_group
   # cp /etc/group /opt/CA/eTrustAccessControl/policies/mfg_group
   ```

2. Make sure that the /etc/passwd file and /etc/group file on each subscriber station are also identical to the ones on your own station.

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

Then, assuming that your local database is a subscriber to your PMDB, and that the PMDB is the only source of new users and new groups for your subscriber stations, you can depend on compatibility between the UIDs and between the GIDs of your local database, your PMDB, and your PMDB subscribers.

If you happen to create a new user with a UID that is already in use in the PMDB or in some other subscriber station, that user was added in some other way. This causes the subscriber's individual update to fail; but at other subscriber stations where no such conflict exists, the update succeeds.


UID defaulted and synchronized with copied passwd file
PMDB
Because its passwd file is identical, the PMDB will assign the same UID as in the database.
If any subscriber database is already using the UID, then that database is not updated, but others are.
local db
other subscriber databases

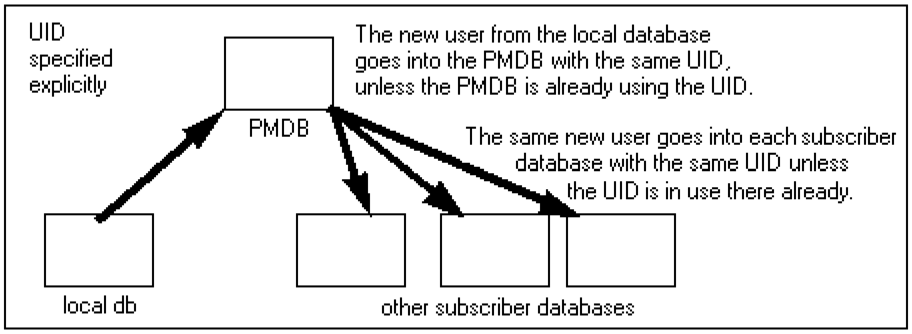
By default, the value of the token synch_uid is yes, but if you ever want a subscriber database to have independent default UIDs and default GIDs (that is, not necessarily attempting to match those of the PMDB), you can set synch_uid to no.

Specifying the UID Explicitly

Another way to send an identical UID or GID to the PMDB and to all its subscribers is to use the userid or groupid parameter with each newusr command.

For example, 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:

```
eTrust> newusr terry_jones unix (userid(1234))
```


UID specified explicitly
PMDB
The new user from the local database goes into the PMDB with the same UID, unless the PMDB is already using the UID.
The same new user goes into each subscriber database with the same UID unless the UID is in use there already.
local db
other subscriber databases

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.

Updating a Policy Model Database

Working at the station 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, you can use the hosts command in the selang command shell. For example, to set the target database to policy1 on pmd_host, issue the following command:

```
eTrust> hosts policy1@pmd_host
```

All selang commands now update the policy1 database. The commands then automatically propagate to the active databases of station workstat1 and the other subscriber stations. For example, if you enter the newusr command, the new user is added to the policy1 database as well as the active databases on the workstat1 station and the other subscriber stations.

Updating Subscribers

The Policy Model tries to fully qualify subscriber names as they are added or deleted from the Policy Model.

The PMDB daemon, sepmdd, attempts to update a subscriber database for the amount of time defined by the token _QD_timeout_. 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. 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.

The Policy Model Error Log

Whenever a PMDB encounters an error while propagating updates to subscribers, the sepmdd daemon creates an entry in the Policy Model error log file. This file, ERROR_LOG, is located in the PMDB directory.

Error Log Format

The error log in eTrust AC versions 5.1 and higher has a format that is not compatible with the format of earlier versions. sepmd cannot handle error logs from previous 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.

Although the old error log was a standard ASCII file that you could view with any editor, the new error log is in binary format; you can view it only by entering the following command:

```
eTrust> sepmd -e pmdname
```

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

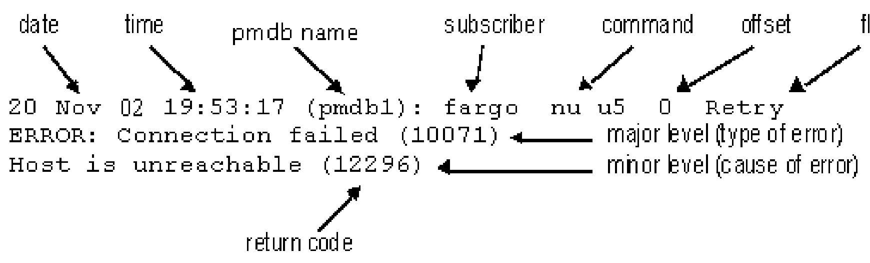
```
eTrust> sepmd -c pmdname
```

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

Error Text	Definition
`20 Nov 02 11:56:07 (pmdb1): fargo  nu u5  0  Retry`	Configuration Errors
`ERROR: Login procedure failed (10068)`	
`ERROR: Cannot accept update from a non-parent PMDB`	
`(pmdb1@oxygen.memco.co.il) (10104)`	
`20 Nov 02 19:53:17 (pmdb1): fargo  nu u5  0  Retry`	Connection Errors
`ERROR: Connection failed (10071)`	
`Host is unreachable (12296)`	
`20 Nov 02 11:57:06 (pmdb1): fargo  nu u5  560  Cont`	Database Update Errors
`ERROR: Failed to create USER u5 (10028)`	
`Already exists (-9)`	
`20 Nov 02 11:57:06 (pmdb1): fargo  nu u5  1120  Cont`	
`ERROR: Failed to create USER u5 (10028)`	
`Already exists (-9)`	

Fri Dec 29 10:30:43 2000 CIMV_PROD:Release failed. Return code = 9241

The following example shows a typical error message:



```
20 Nov 02 19:53:17 (pmdb1): fargo  nu u5  0  Retry
ERROR: Connection failed (10071)
Host is unreachable (12296)
```

- 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.

Note: 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.

For example, the following error has only one message level:

```
Fri Dec 29 10:30:43 2002 CIMV_PROD:Release failed. Return code = 9241
```

This message occurs when sepmdpull attempts to release a subscriber that is already available.

Cleaning Up the Update File

The sepmd utility automatically writes each update it receives in the updates.dat file. In order to prevent the file from growing too large, we recommend that you delete processed updates from the file periodically. Do this by entering the following sepmd command:

```
eTrust> 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.

Encrypting 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 do this, set the UseEncryption token in the [pmd] section of the pmd.ini file to yes.

To decrypt the updates.dat file, invoke the sepmd utility with the -de switch. For more information about sepmd, see the sepmd utility in the *Utilities Guide*.

Excluding Subscribers

You can skip subscribers so that they do not receive updates. You can exclude the local host by setting 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*).

When you specify excluded subscribers, these subscribers will not receive updates from the PDDB; if you want a subscriber to receive an update, you must remove the subscriber from the excluded list.

Propagating Passwords

When a user changes a password using the sepass utility, the new password is normally sent to the station'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 do this, use the pmdb parameter with the newusr, chusr, or editusr command.

For example, to specify that the user Tony's new passwords created with sepass should be sent to and propagated by a separate PMDB, enter the following command:

```
eTrust> editusr tony pmdb(pw_pmdb@name1.yourorg.com)
```

Removing a Subscriber

Suppose that the active database on station STAT1 is currently a subscriber of the PMDB policy1 but you no longer want it to be a subscriber.

1. Remove the station STAT1 from the PMDB policy1 by using the following sepmd command:

   ```
   sepmd -u policy1 STAT1
   ```

2. Shut down seosd on the station workstat1 by entering the following command:

   ```
   # secons -s
   ```

3. Edit the seos.ini file on *workstat1* and delete the value of the parent_pmd token in the [seos] section.
4. Restart seosd on the station workstat1 with the command:

   ```
   # seosd
   ```

The active database of station STAT1 is no longer a subscriber of policy1.

Note: You can restart seosd at a later time. Starting seosd clears the token, once the station is unsubscrbed from the PMDB, the PMDB no longer sends commands.

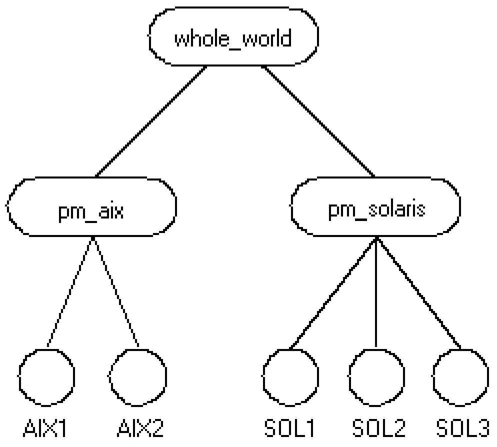
Architecture Dependency

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 PMDB for architecture independent rules, and give it subscriber PMDBs that define architecture dependent rules.

To set up a multiple-architecture PMDB, use the procedure that the following example demonstrates. 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. Set up the 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_solaris, to contain all the Sun Solaris specific rules.

The PMDBs pm_aix and pm_solaris are subscribers of the PMDB whole_world. All IBM AIX stations at the site are subscribers of pm_aix. All Sun Solaris computers at the site are subscribers of pm_solaris. The concept is illustrated in the following chart.



4. Enter platform independent commands, such as adding a user or setting a SURROGATE rule, in whole_world so that all stations at the site are automatically updated.
5. Add a trusted program to pm_aix to update all IBM AIX computers, without affecting the Sun Solaris systems.

Filtering Updates

If you want your PMDB to update different subsets of data at different subscriber stations:

1. Configure PMDBs to serve as parents to subsets of subscribers.
2. Define which records are sent to subscriber stations.

   To do this, point the filter token in the pmd.ini file of the parent PMDB to a filter file you set up on the same machine.

   Updates to the subscriber stations are then limited to the records that pass the filter.

A filter file consists of lines with six fields per line. 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 station or not: PASS or NOPASS

You can use an asterisk to mean “all possible values” in any field. If more than one line covers the same records, the *first* applicable line is used.

In each line of the filter file, spaces separate the fields. In fields with more than one value, semicolons separate the values. Any line beginning with “#” is considered a comment line. Empty lines are not allowed.

The following example describes a line from a filter file:

CREATE	**eTrust**	**USER**	*****	**FULL_NAME;OBJ_TYPE**	**NOPASS**
↑	↑	↑	↑	↑	↑
form of access	environment	class	record name (* =all)	properties	treatment

If, for example, the file with this line is named TTY1_FILTER and the pmd.ini file for PMDB TTY1 contains the line 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.

Using 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:

   ```
   migopts -d pmdb-name
   ```

   where *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 that migration and integration are two separate procedures. For more information see the chapter "Unicenter Security Migration and Integration."

3. Create classes for any user-defined Unicenter TNG asset types with the following command:

   ```
   defclass.sh.pmdb-name
   ```

   where *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.*

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, and authorizing the transaction for execution.

- In the first stage, 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.
- In the second stage 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.

  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 processing by a different checker. For more information about the sepmd utility, see the sepmd utility in the *Utilities Guide*.

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. For more information about the sepmdd daemon, see the sepmdd utility in the *Utilities Guide*.

Updating the PMDB in Dual Control

Dual Control requires two users to execute commands in the PMDB—a maker and a checker. This section explains how makers and checkers update the PMDB.

Before Creating Transactions

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

```
is_maker_checker=yes
```

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

For more information about the pmd.ini file, see the appendix “The pmd.ini File.” For more information about the seos.ini file, see the appendix “The seso.ini Initialization File.”

Before you begin to enter a new transaction or edit an unprocessed transaction, make sure the following is true:

- You 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.)

Creating Transactions

To create a transaction, complete the following steps:

1. In the first stage of updating the PMDB, you (as a maker) must use selang. If you want to create a transaction, enter the following command:

   ```
   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.

2. Enter the following command:

   ```
   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.

3. Once you have entered the hosts and start_transaction commands, you are ready to enter your transaction. When you are finished, enter the end_transaction command. For example:

   ```
   newusr mary owner(bob) audit(failure,loginfailure)
   chres TERMINAL tty30 defaccess(read) \
   restrictions(days(weekdays)time(0800:1800))
   end_transaction
   ```

4. 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. The following examples illustrate how to update existing commands.

   - 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
     ```

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

     ```
     chusr mary category (FINANCIAL)
     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).

Command with Parameter	Description
sepmd -m l	Lists the unprocessed transactions of the user who invoked the parameter.
sepmd -m la	Lists all the transactions of all the makers that are waiting to be processed.
sepmd -m lo	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. (See Step 1 in this section.)

- 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
  ```

Before Checking Transactions

Before checking transactions, follow the same procedures as you did before creating transactions (in the previous section):

To activate the Dual Control feature, the value of the is_maker_checker token in the pmd.ini file **and** the [pmd] section of the seos.ini file must be set to yes:

```
is_maker_checker=yes
```

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

For more information about the pmd.ini file, see the appendix “The pmd.ini File.” For more information about the seos.ini file, see the appendix “The seos.ini Initialization File.”

Before you begin processing any transactions, make sure the following is true:

- You have ADMIN authority.
- You locked the transaction before you began to process it.
- Another Checker does not lock the transaction.
- None of the commands pertain to you. (You cannot process commands that involve yourself.)

Checking and Processing Transactions

1. As a checker, you must be in the *eTrustACDir*/bin path (where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl). To view all the transactions that are waiting to be processed before execution, use the **la** parameter. Or to view all the transactions except the transactions that you yourself created, use the **lo** parameter. Each transaction includes the name of the maker, the ID number of the transaction, and the name or description of the transaction.

   ```
   sepmd -m la
   sepmd -m lo
   ```

2. When you have viewed the transactions that are waiting to be processed, you must lock the transactions before processing them. A locked transaction cannot be changed.

   ```
   sepmd -m r transactionId
   ```

3. After a transaction is locked, then you can process it. Three codes can process a transaction:

   - 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.

     ```
     sepmd -m p transactionId code
     ```

   When you have entered one of these codes, the procedure is complete. A message appears stating which commands were successful and which failed.

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 stations 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 allow 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.

WARNING! *Do not change the UseSystemFiles token yourself. The sepmdd or seagent daemons change it automatically.*

For more information about the seagent or sepmdd daemons, see the seagent and sepmdd utilities in the *Utilities Guide*.

Using a PMDB for Mainframe Synchronization

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.

Any password change a mainframe user makes is propagated to all the machines in the password Policy Model hierarchy. For more information about mainframe synchronization, see the appendix "Password Synchronization with Mainframes."

Chapter 10

General Security Features

This chapter describes several eTrust AC security features including:

- Idle station locking
- In-house resource protection
- Stack overflow protection
- B1 security level certification

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. For more information on how to set up the screen lock utility, see the chapter "Utilities in Detail" in 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.

For more information, see the *SDK Developer 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.

STOP Support

STOP is supported on the following operating systems:

- HP-UX
- Sun Solaris
- Linux

Starting and Stopping STOP

When the STOP is first installed (see the appendix “Installing and Customizing”), 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:

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> setoptions class+ (SECLEVEL)
```

Disabling Security Level Checking

The following setoptions command disables security level checking:

```
eTrust> 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:

```
eTrust> setoptions class+ (CATEGORY)
```

Disabling Security Category Checking

The following setoptions command disables security category checking:

```
eTrust> 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:

```
eTrust> newres CATEGORY name
```

where *name* is the name of the security category.

To define the security category "*Sales*," enter the following command:

```
eTrust> newres CATEGORY Sales
```

To define the security categories "*Sales*" and "*Accounts*," enter the following command:

```
eTrust> 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:

```
eTrust> 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:

```
eTrust> rmres CATEGORY name
```

where *name* is the name of the security category.

To remove the security category "*Sales*," enter the following command:

```
eTrust> 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:

```
eTrust> setoptions class+(SECLABEL)
```

Disabling Security Label Checking

The following setoptions command disables security label checking:

```
eTrust> 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:

```
eTrust> 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:

```
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:

```
eTrust> 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:

```
eTrust> rmres SECLABEL name
```

where *name* is the name of the security label.

To remove the security category *"Manager"* enter the following command:

```
eTrust> rmres SECLABEL Manager
```

Chapter 11

Auditing Events

This chapter explains how to set up log auditing and routing procedures.

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:

Value	Description	Application
FAIL	Logs access failures to protected resources.	Users and resources
SUCCESS	Logs successful accesses to protected resources.	Users and resources
LOGINFAIL	Logs every login failure of the user.	Users
LOGINSUCCESS	Logs every successful login of the user.	Users
ALL	Logs the same as FAIL, SUCCESS, LOGINFAIL, and LOGINSUCCESS together.	Users and resources
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.	Users
NONE	Logs nothing.	Users and resources

- 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.

About Audit Logs

Tip: 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. For more information on seaudit, see the *Utilities Guide*. For 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.

About 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 (where *eTrustACDir* is the installation directory for eTrust AC). For complete information on the syntax of the configuration file, see the selogrd utility in the *Utilities Guide*. 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.

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. For more information on 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:

   ```
   eTrust> 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. For more information on the configuration file format and on setting up the log routing daemons, see the *Utilities Guide*.

Chapter 12 Remote Status View

This chapter introduces you to Remote Status View (RSV), a web-based status information utility. Use a browser to connect to RSV, which displays status information about eTrust AC for different hosts.

Requirements

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/docdirectory 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 service package. However, you can opt to install it separately.

Installation

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:

```
# 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:

```
# 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.

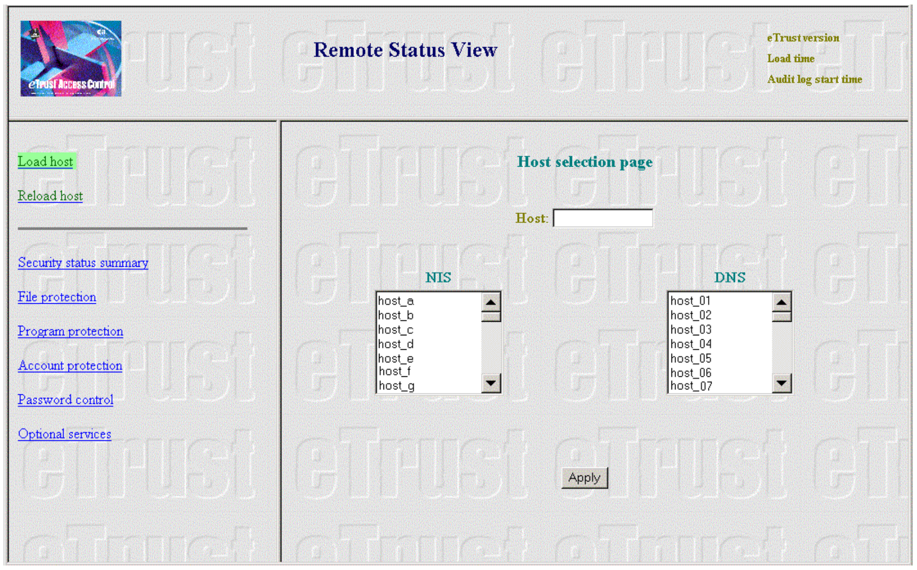
Displaying

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. The following table details these categories.

Category	Status Item	Indicates
Security status summary	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	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	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.
	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.

Category	Status Item	Indicates
Account protection	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.
	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	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	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.

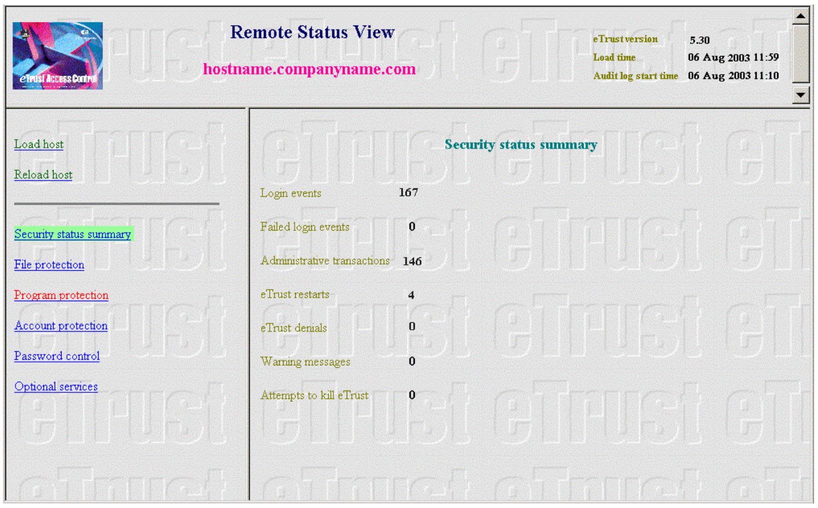
Using RSV

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.

2. Click Apply. The right panel displays the Security status summary panel.



Remote Status View

hostname.companyname.com

eTrust version	5.30
Load time	06 Aug 2003 11:59
Audit log start time	06 Aug 2003 11:10

Load host
Reload host

Security status summary
File protection
Program protection
Account protection
Password control
Optional services

Security status summary

Login events	167
Failed login events	0
Administrative transactions	146
eTrust restarts	4
eTrust denials	0
Warning messages	0
Attempts to kill eTrust	0

3. Click the category links in the left panel to view other status pages.
4. 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.

RSV Security

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 in the RSV server host. See the chapter "Protecting TCP/IP Services" for details.

Chapter

13 Scope of Administration Authority

This chapter discusses the scope of the security administration privileges that you can assign to a user. It describes the security privileges that exist, and what each privilege allows its owner to do.

eTrust AC offers several different types of security administration privileges. Privileges are granted by:

- Global authorization attributes
- Group authorization attributes
- Ownership
- Entries in the ADMIN class

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:

Access	Description	Commands
List	List information in the database.	showusr, showgrp, showres, showfile, find
Modify	Set the audit mode for existing records.	chusr, chgrp, chres, chfile

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:

```
selang> so cng_adminpwd
```

To remove this option, issue the following command:

```
selang> 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.

SERVER Attribute

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. For more information about the server attribute and eTrust AC APIs, see the *SDK Programmer 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.

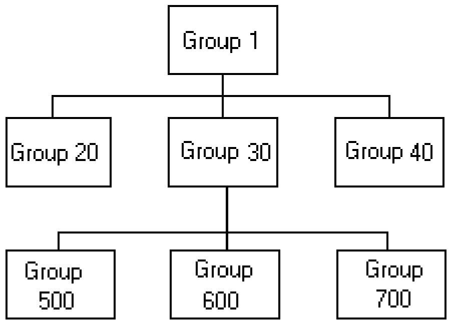
For more information on the HOLIDAY class, see the *Reference Guide*.

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 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 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. For examples of group scope, see Authorization Examples in this chapter.

Users with the GROUP-ADMIN attribute have the following access authority for the records within their group scope:

Access	Description	Commands
Read	Show the properties of the record.	showusr, showgrp, showres, showfile
Create	Create new records in the database. You must specify the owner.	newusr, newgrp, newres, newfile
Modify	Change the properties of the record.	chusr, chgrp, chres, chfile
Delete	Remove records from the database.	rmusr, rmgrp, rmres, rmfile
Connect	Join a user to a group or separate a user from a group.	join, join-

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:

Access	Description	Commands
Read	Show the properties of the record.	showusr, showgrp, showres, showfile
Modify	Change the properties of the record.	chusr, chgrp, chres, chfile
Delete	Remove the record from the database.	rmusr, rmgrp, rmres, rmfile
Connect	Join a user to a group or separate a user from a group.	join, join-

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.

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.

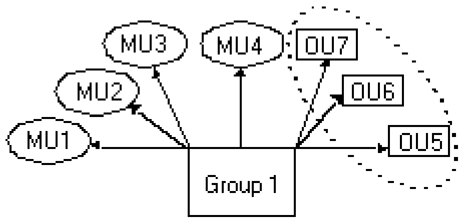
eTrust AC allows UNIX file owners to define FILE records, unless this feature is explicitly disabled. If you do not want file owners to define FILE records, make sure that the use_unix_file_owner token in the [seos] section of the seos.ini file to no. (This is the default setting.)

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

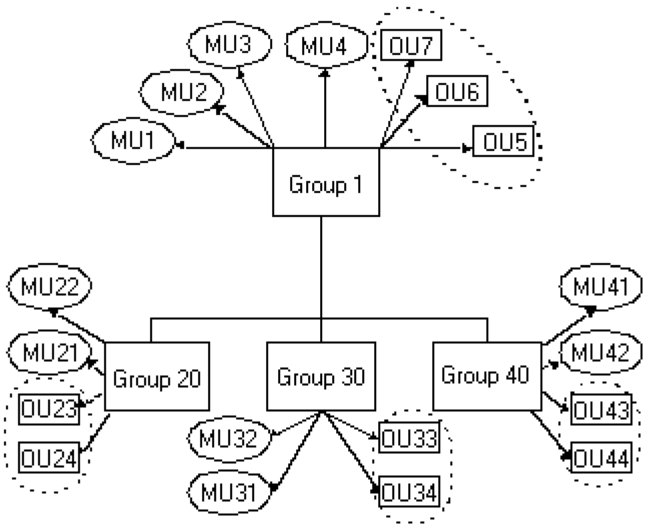
In the following diagram, four users are members of Group 1: MU1, MU2, MU3, and MU4. Group 1 also owns three users—OU5, OU6, and OU7. The member MU4 has the GROUP-ADMIN attribute.



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. 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:

Access	Description	Commands
Read	Show the properties of the record in the class.	showusr, showgrp, showres, showfile, find
Create	Create new database records in the class.	newusr, newgrp, newres, newfile
Modify	Change properties in the class.	chusr, chgrp, chres, chfile
Delete	Remove existing class records from the database.	rmusr, rmgrp, rmres, rmfile
Connect	Add users to and remove users from groups. This access is valid only in the ACL of the GROUP record.	join, join-
Password	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.	chusr

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.

For more information, see B1 Security Level Certification in the chapter “General Security Features.”

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:

     ```
     eTrust> 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:

     ```
     eTrust> 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

Using eTrust AC can reduce system performance due to multiple access checks for accessed entities (files, registry keys, and the like).

This chapter provides several options for improving system performance with eTrust AC. It includes information about the Global Access Check feature, the resource cache, and the IP caching feature. Other ways to improve system performance are included as well: real path caching, name resolution, class parameters, and real path bypasses.

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. This feature works for every user except root. (See GAC Restrictions in this chapter.)

How Does GAC Work?

GAC is applied to a set of files that you specify in advance. (See Setting Up GAC Rules in this chapter.) eTrust AC monitors access to the specified files and builds a table of permitted accesses during execution time.

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.

Tip: 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

Tip: 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:

  ```
  eTrust> 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.

For more information, see the secons utility in the *Utilities Guide* or the [seosd] section in the appendix "The seos.ini Initialization File."

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 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.

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. (See the appendix "The seos.ini Initialization File" for more information about this token.) File names are cached in the table with a data pair: inode number and device number.

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.

For more information about this token, see the appendix "The seos.ini Initialization File."

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.

For more information about this token, see the appendix "The seos.ini Initialization File."

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. (For more information about this token, see the appendix "The seos.ini Initialization File.") The token stores access information to be bypassed whenever eTrust AC must access the process file system.

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. (For more information about this token, see the appendix "The seos.ini Initialization File.") If you enable this feature, eTrust AC does not obtain real file names, which, for example, could be a symbolic link.

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 SPECIALPGMTYPE with values pbf and pbn.
- To bypass setuid and setgid events for a specified program, use the property SPECIALPGMTYPE 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 sepmd -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. (See the appendix “The seos.ini Initialization File” for more information about these tokens.)

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.

For more information about these tokens, see the appendix “The seos.ini Initialization File.”

Chapter 15 UNIX Exits

You can start programs of your own by issuing specific selang commands. For example, you can perform an initialization process for each new user that you add; or you can perform extra logging or screening of commands before they are executed. This chapter explains how.

What Is a UNIX Exit?

A UNIX exit is a specified program—a shell script or an executable—that runs automatically on one or more of the following occasions:

- As a *pre*-update exit
  - Before each selang command that updates a *user* Record
  - Before each selang command that updates a *group* record
- As a *post*-update exit
  - After each selang command that updates a *user* record
  - After each selang command that updates a *group* record

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 creation, modification, or deletion. Querying a user or group does not cause any UNIX exit to run. Specifically, these are the commands that can cause a UNIX exit to run:

```
newusr
newgrp
chusr
chgrp
editusr
editgrp
rmusr
rmgrp
```

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 Provided Exit Script

The exit script that is supplied as part of eTrust AC is /*eTrustACDir*/exits/lang_exit.sh. (where *eTrustACDir* is the installation directory for eTrust AC—this may be a symbolic link.) Here is how it works:

1. eTrust AC automatically gives values to three parameters of the script.

Parameter	**Possible Values**
CLASS	USER | GROUP
ACTION	CREATE | MODIFY | DELETE
STAGE	PRE | POST

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).

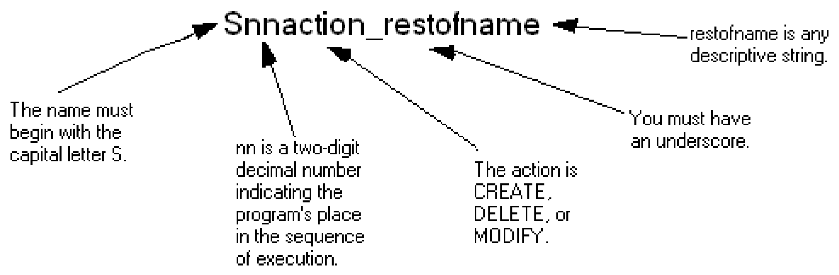
The script can pass the parameter values to programs that it calls.

Parameter	**Possible Values**
EXEC_RV	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, see *eTrustACDir*/samples/exis_src.

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 in the following format and that refer to the appropriate action.


Snnaction_restofname
restofname is any descriptive string.
The name must begin with the capital letter S.
nn is a two-digit decimal number indicating the program's place in the sequence of execution.
The action is CREATE, DELETE, or MODIFY.
You must have an underscore.

   eTrust AC runs all the appropriate programs according to the numerical order of the second and third characters of their names.

For example, suppose that someone is about to delete a user, and the directory *eTrustACDir*/exits/USER_PRE/ (where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl) includes the following files:

```
S10CREATE_precustom.sh
S10DELETE_precustom.sh
S99DELETE_prermusrdir.sh
```

In this case, the first program is not run because the operation we want is delete, not create. The second program is run, and the third program is run after the second because the number after the initial S is lower.

Passing Arguments to UNIX Exits

In writing exits, not only can you take advantage of the three parameters mentioned previously—and all the standard eTrust AC data such as names and permissions—but you can also designate extra user or group data especially for use at exit time.

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:

```
eTrust> 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 the Security Administrator, then you have some additional capabilities and graphical tools. See the chapter “seam.ini and UNIX Exits” in the *User Guide*.

Permissions

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.

Time Out and Other Failures

Exit execution times out after 15 seconds, unless the exit_timeout variable in the seos.ini file specifies otherwise. As usual, 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.

Specifying Exit Programs to Run

To tell eTrust AC which exit programs to run, use the [lang] section of the seos.ini file. You can specify up to four tokens, with the *full path name* of one exit program per token.

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, set any or all of the following tokens to your own preferences:

Token	**Meaning**
pre_group_exit	The specified exit is called before each group update is executed.
pre_user_exit	The specified exit is called before each user update is executed.
post_group_exit	The specified exit is called after each group update is executed.
post_user_exit	The specified exit is called after each user update is executed.

An exit is called only if its full *path name* appears as the value of an exit token.

In the following example, 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:

```
[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
```

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_exit
```

Chapter 16 Interacting with LDAP

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. For more information about LCA and the TCL extension, see the chapter "Language Client API" and the appendix "tcllca: The LCA Extension," respectively, in the *SDK Developer Guide*.

If you do not have TCL, consult the FAQ posted monthly to comp.lang.t_c_l by Larry Virden, which is also available on the Internet on the MIT web site and the Terafirm website.

See also the Sun web site for TCL news, documentation, and resources.

ldap2seos

ldap2seos extracts users from an LDAP database located at the server host and adds them to the eTrust AC database.

Syntax

ldap2seos [*options*]

Options

-accfld *account-field*	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 is ignored. **Note:** If the script cannot find the userid, users are not uploaded to the eTrust AC database.
-b *base-entry*	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*	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*	A file to which data retrieved from the LDAP server may be temporarily stored.
-h	A request for a help screen. The screen contains a listing and explanation of ldap2seos usage and options.
-h *ldap-host*	The name of the host where the LDAP database is located. The default is the local host.
-l *ldap-dir*	The directory containing the line command utilities that are assumed to be in the bin subdirectory. The default is /usr/local/ldap.
-p *port*	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.

Description

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.

As mentioned in the introduction to this chapter, 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.

Example

The following command extracts information about users from the LDAP database at host myhost.mysite.co.na and tries to add them to the eTrust AC database.

```
eTrust> ldap2seos -h myhost.mysite.co.na
```

seos2ldap

seos2ldap exports eTrust AC users from the database to an LDAP database located at a server host.

Syntax

seos2ldap [options]

Options

-b *base entry*	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*	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*	A file to which data retrieved from the LDAP server may be temporarily stored.
-h	A request for a help screen. The screen contains a listing and explanation of ldap2seos usage and options.
-h *ldap-host*	The name of the host where the LDAP database is located. The default is the local host.
-l *ldap-dir*	The directory containing the line command utilities that are assumed to be in the bin subdirectory. The default is /usr/local/ldap.
-noprompt	Cancel base entry prompt. If you did not use the -b *base-entry* flag to specify the base LDAP entry, by default seos2ldap prompts for a base entry. This flag suppresses the prompt.

-p *port*	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. Use this with the -d option where authentication is needed to access the LDAP database.

Description

The seos2ldap utility 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.

Example

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.co.ca. You can edit the LDIF file later and update LDAP manually.

```
eTrust> seos2ldap -h myhost.mysite.co.na
```

S50CREATE_u_LdapE

S50CREATE_u_LdapE.sh uploads new UNIX users to LDAP as they are created.

Description

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 The Provided Exit Script in the chapter "UNIX Exits"), 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

eTrust AC is fully integrated into the Unicenter Enterprise Management environment. The following sections describe how eTrust AC handles the integration.

Important! *To have Unicenter TNG integration with eTrust AC, you must have Unicenter TNG installed on the same machine as eTrust AC.*

Installing Unicenter Security Integration Tools

For complete installation instructions, see the appendix "Installing and Customizing."

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 ENF is running, the gateway is automatically started or stopped whenever the eTrust AC services are started (using seload) or stopped. If Unicenter 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
```

-I	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 sessfagte manually with the following command:

```
eTrustACDir/tng/bin/sessfagte -I
```

where *eTrustACDir* is the directory in which you installed eTrust AC.

eTrust AC to Unicenter Security Synchronization Utility

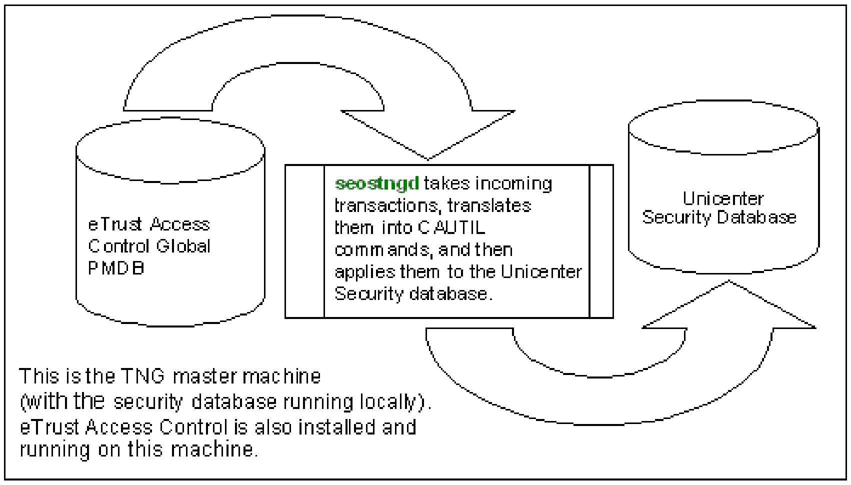
Unicenter Security and eTrust AC manage the administration of your enterprise IT environment before total migration occurs. To reduce the complexity of using different product tools to perform administrative tasks, we are providing a synchronization daemon.

The new daemon is called seostngd. eTrust AC sends Policy Model Database (PMDB) updates through Computer Associates 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:


eTrust Access Control Global PMDB
seostngd takes incoming transactions, translates them into CAUTIL commands, and then applies them to the Unicenter Security database.
Unicenter Security Database
This is the TNG master machine (with the security database running locally). eTrust Access Control is also installed and running on this machine.

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:

```
eTrust> seostngd
```

To stop the daemon, enter one of the following:

```
eTrust> seostngd -shut
```

or

```
eTrust> 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.

	Unicenter TNG	**eTrust AC**
User ID	20 characters	256 characters
Password	8 characters	14 characters
User group ID	8 characters	254 characters
Asset group ID	8 characters	255 characters

- 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.
- 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
- 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. See the *Utilities Guide* for more information.

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. See the *Utilities Guide* for more information.

In order to activate exporttngdb, 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:

Attribute Name	Description
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)

Attribute Name	Description
PWDCHANGE VALUE (RANDOM)	Random password generation
UPSSTATGROUP	UPS station group ■ 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:

Attribute Name	Description
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.

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:

   ```
   eTrust> 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.

```
eTrust> nr CALENDAR calendar_name
```

```
eTrust> 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:

```
eTrust> auth resource_class_name object_name \
uid_or_gid_name calendar(calendar_name) access(access_value)
```

For example:

```
eTrust> 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:

```
eTrust> auth resource_class_name object_name uid_or_gid_name \
calendar(Unicenter_calendar_name) deniedaccess(access_value)
```

For example:

```
eTrust> 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.

```
eTrust> auth file file1 uid(george) calendar(holidays) deniedaccess(r)
```

To remove a user or group from a Unicenter TNG calendar ACL property, use auth- :

```
eTrust> 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:

```
eTrust> 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

Appendix

A

Installing and Customizing

This appendix guides you step-by-step through the eTrust AC for UNIX installation process. When you have finished installing eTrust AC following the instructions in this appendix, your system should contain a copy of the eTrust AC software and an elementary eTrust AC database. The appendix then explains how to start eTrust AC and how to use its commands. Later, by editing the database, you can define access rules to protect your system.

Preliminary Steps

Before you can install eTrust AC, you must make sure sufficient disk space is available and assemble several items of information, described in this section.

Memory Space and Disk Space

Make sure you have enough memory and enough space on the file system where you are installing eTrust AC. In the partition, you need space both for the program itself and for your eTrust AC database. (Use the UNIX df command to check available space.)

For all platforms, eTrust AC requires a minimum of 64 MB RAM (memory), and 60 MB of hard disk space for minimal installations. For general installations, be sure to have 100 MB hard disk space.

In addition, allow disk space for your eTrust AC database, which is the repository of records describing your users and user groups, your protected files and other resources, and the authorizations that permit controlled access to the resources.

A database for one thousand users, one thousand files, and five hundred access rules, for example, could occupy approximately 2 MB.

Note: In order to ensure the proper functioning of eTrust AC, the installation script does not allow installation unless the destination directory has at least 100 MB of free space (regardless of installed platform).

Necessary Users and Groups

You will need a number of users and groups with which to practice. Make sure you have permission to modify the definitions of the users, including their UNIX group memberships. You can work with preexisting UNIX groups, but we recommend creating new UNIX groups especially for eTrust AC.

If you want, you can specify one or more of your real-life users now and add more later; or you can specify fictitious users to practice with now, and later specify real users instead. However, you must specify at least one user to fill each of the following roles:

- A security administrator who is in general command of eTrust AC at your site. You may find it convenient to put all such administrators into a UNIX group of security administrators. (A recommended name for the group is secadm.)
- An eTrust AC auditor who can view and manage system security reports. The auditor can be the same user as the security administrator, though often it is a good idea to maximize accountability by keeping the jobs separate. You may find it convenient to put all such auditors into a UNIX group of security auditors (a recommended name for the group is sysaudit). Alternatively, if the auditors and administrators are the same people, you might put them into the security administrators' group.
- At least one everyday user for you to practice on, who has neither eTrust AC administrative capabilities nor eTrust AC auditing capabilities.
- A UNIX system administrator who can become root at any time, unrestricted by UNIX. You may find it convenient to put all such UNIX administrators into a UNIX group of system administrators.

Before You Begin

If you are upgrading from a previous version of eTrust AC, note the following:

- Read the *eTrustACDir*/README.TXT file (where *eTrustACDir* is the installation directory).
- If your network uses PMDBs, you should update it *bottom-up* (subscribers first), so that at any given moment no version 5.3 PMDB has any earlier version subscribers. Earlier PMDB versions are permitted to have later versions of subscribers.
- You can upgrade the existing seos.ini and pmd.ini files, or create new ones. Regardless of your decision, the installation script saves a copy of the old seos.ini file in seos_ini.back. It saves copies of pmd.ini files as pmd_ini.back, in its Policy Model directory.

- If you are upgrading from eTrust AC 5.0 or eTrust AC 5.1 on Solaris, AIX, HP-UX, or Linux platforms, you must reboot the machine during installation. However, if this is a first-time original installation of eTrust AC, you do not have to reboot during the installation process.

You will use three files from the eTrust AC media (that is, from the eTrust AC CD-ROM or tape):

- A script that installs eTrust AC from the tar file. Its name is install_base.
- A compressed tar file containing all the eTrust AC files. Its name is *_opSystemVersion_eTrustACVersion*.tar.Z. For example, if you are installing eTrust AC version 5.3 on IBM AIX version 4.3 then your tar file is _AIX43_530.tar.Z.
- A compressed tar file named pre.tar containing messages for installation as well as the license agreement.

  After you have read the license agreement file, you can continue the installation by entering the command found at the end of that file. If you are running a silent install (using -autocfg), you can use the -command flag with the command that can be found at the bottom of the license agreement file. If you are using a response file (-autocfg file_name), you do not need to use the -command flag. To get the license file name and location, you must run install_base -h. You can also get the file name and location if you enter the wrong command.

Note: To install the standalone Policy Manager, use the installation instructions provided in the *User Guide*. The installation files for the Policy Manager are available on the eTrust AC for UNIX CD.

Installation Procedure

To install eTrust AC, complete the following steps:

1. If you already have eTrust AC installed and it is running, shut it down by logging in as an administrator and entering the following command:

eTrust AC Version	Command
5.1 or lower	`# eTrustACDir/bin/secons -s`

5.3 and higher	`# eTrustACDir/bin/secons -sk`
	`# eTrustACDir/bin/SEOS_load -u`

2. Log in as root.

3. This step of the installation procedure depends on the type of media you are installing from.

   - If you are installing on **HP** from a **CD-ROM**, you need to ensure the proper reading of file names from the CD-ROM. To prevent the file names from being forced into a shortened and all-uppercase format, type the following commands:

     ```
     # pfs_mountd &
     # pfsd &
     ```

     These commands should invoke four daemons: pfs_mountd, pfsd.rpc, pfs_mountd.rpc, and pfsd. To verify that those four daemons are all running, type the following command:

     ```
     # ps -ef | grep -v grep | grep pfs
     ```

     The output should list all four daemons; for example:

     ```
     root  6016  5948  0 10:34:38 ttypf  0:00 pfs_mountd
     root  6065  6064  0 10:39:18 ttypf  0:00 pfsd.rpc
     root  6017  6016  0 10:34:38 ttypf  0:00 pfs_mountd.rpc
     root  6064  5948  0 10:39:18 ttypf  0:00 pfsd
     ```

     Mount the CD-ROM by using the following command:

     ```
     # pfs_mount /dev/dsk/c0t4d0 /cdrom
     ```

     For further explanations, see the man pages of the particular pfs* daemons and commands.

   - If you are installing from a CD-ROM but **not** on HP, mount the CD-ROM as usual.

4. Run the install_base script from the CD, by using the install_base command. The installation script finds the appropriate compressed tar file, so typing the name of your tar file is optional.

   Note: Running the install_base script involves approving the Software Evaluation License Agreement. After you have read the license agreement, you can continue the installation by entering the command found at the end of that file. If you are running a silent install (using -autocfg), you can use the -command flag with the command that can be found at the bottom of the license agreement. If you are using a response file (-autocfg file_name), you do not need to use the -command flag. To get the license file name and location, you must run install_base -h. You can also get the file name and location if you enter the wrong command.

   If you are installing from a CD-ROM, use the following command:

   ```
   # /cdrom/Unix/Access-Control/install_base \
   [/cdrom/ Unix/Access-Control/_opSystemVersion_eTrustVersion.tar.Z] \
   [options]
   ```

   where *cdrom* is the name of the directory that represents your own CD-ROM drive.

The following section describes the options of the install_base script.

Important! *You must run install_base with the –client parameter before you can run it with the –server parameter. You can, however, use both parameters at the same time.*

Installation Script Options

-admin | -all | -api | -client |-mfsd | -server | -stop |-tng |-uni

install_base installs one or more installation packages:

-admin
The administration tools package (the Motif GUI).

-all
The eTrust AC server package, the client package, the API package, the UNI package, the MFSD package, the STOP feature, and the Administrator tools package.

-api
The API libraries and sample programs.

-client
All parts of eTrust AC other than the API samples and the server.

Note: The client package has everything needed for a standalone machine. The server package adds more daemons (like sepmdd or selogrcd), so you can use the machine as a PMDB or log routing daemon.

-mfsd
The mfsd package.

-server
The entire eTrust AC package.

Note: The client package has everything needed for a standalone machine. The server package adds more daemons (like sepmdd or selogrcd), so you can use the machine as a PMDB or log routing daemon.

-stop
The STOP feature.

-tng
The Unicenter Security migration package.

-uni
The Unicenter Security integration and migration package.

Notes:

If you do not use any of these options, install_base assumes that both client and server packages should be installed.

	If you are upgrading your version of eTrust AC, and you do not specify the appropriate flags to upgrade a package that you installed with an earlier version, a warning appears.
-autocfg [*filename*]	Uses the preferences stored in *filename* to automatically respond to the interactive installation process. If you do not specify *filename*, the option defaults to a preset configuration. You must add the *filename* of the response file or the -command flag to the -autocgf flag. **Notes:** ■ You cannot change the encryption key when using the -autocfg option. ■ To install the Administrator package or the STOP feature using autocfg, you must also include the -admin or -stop parameters, respectively.
-command [*command*]	Uses the command specified in *command* with the -autocfg flag to accept the license agreement during silent installations. The command is found at the end of the license agreement file (located at /tmp/pre_install/ eACLicenseAgreementUNIX_english_cp1252.txt).
-d *target_dir*	Specifies the name of the installation directory. The default value is /opt/CA/eTrustAccessControl. After installation, set the environment variable in the configuration file setenv _ _eTrustAccessControl_ _ *target_dir*.
-*dns* | -nodns	Creates a lookaside database with or without DNS hosts. The -nodns option specifies that eTrust AC should not perform an nslookup on any hosts in the DNS during installation.
-force_encrypt	Forces the installation to accept the default encryption key without warnings. The installation will not warn you if your key is not the default. Be aware that after the upgrade, your encryption key is set to the default. eTrust AC also provides DES and 3DES encryption options that you can choose.
-force_install	Forces the installation of the new version over the version already installed.
-g *groupname*	Specifies the name of the group owner of the eTrust AC files. The default value is 0.
-h	Displays help.
-ignore_dep	Directs the installation procedure **not** to check for dependency with other products.

-key *encryption_key*	Restores your encryption key during an upgrade. Note that during an upgrade you must use the same encryption key that you were using before the upgrade.
-lang	Specifies the language in which to install eTrust AC.
-log	Stores all transactions associated with the installation process. The file is located in *eTrustACDir*/eTrust_install.log (where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl).
-post *programname*	Runs the specified program after installation.
-pre *programname*	Runs the specified program before installation.
-savecfg *filename*	Stores your responses to the interactive setup for later use by the -autocfg option.
-system_resolve	Specifies to use system functions, which define a bypass for network caching on your system. You cannot run this flag on IBM AIX platforms.

Continuing the Installation

The install_base script performs the following steps:

- It extracts the data from the tar.Z file into *target_dir*.

  Different platforms cause different actions:

  - For Sun Solaris, the install_base script adds the eTrust AC *syscall* script to the file /etc/name_to_sysnum. The original file is saved as /etc/name_to_sysnum.bak. It then creates the file /etc/rc2.d/S68SEOS that forms part of the boot sequence.
  - For Silicon Graphics IRIX the file /etc/rc2.d/S29SEOS is created and becomes part of the boot sequence.
  - For NCR, Reliant, and UnixWare, the file /etc/rc2.d/S58SEOS is created and incorporated into the boot sequence.
    - For NCR, it calls the script *eTrustACDir*/bin/SEOS_NCR_build to relink the kernel, if necessary.
    - For Reliant it, calls the script *eTrustACDir*/bin/SEOS_SINIX_build to relink the kernel, if needed.
    - For UnixWare, it calls the script *eTrustACDir*/bin/SEOS_UNIXWARE_build to relink the kernel, if needed.
  - For IBM AIX, it loads the SEOS_syscall script.

  - For Sun Solaris 2.5x, 2.6, and 2.7 (32-bit only), as well as HP-UX 11.x (32-bit only), the install_base script looks for the STOP tar file in the /cdrom/Unix/Access-Control/ directory. It prompts you to select whether or not to install STOP.

    If you choose to install, the script then replaces the regular SEOS_syscall file with one that includes both the regular interception code and the STOP code.

    If you include the -stop or -all options in the command line of the install_base script, the script installs STOP without prompting you.

- The script then allocates, initializes, and formats the eTrust AC database and builds the seos.ini file (detailed in the appendix "The seos.ini Initialization File"). The database files are placed in the *eTrustACDir*/seosdb directory (where *eTrustACDir* is the eTrust AC installation directory.).

Note: Under Sun Solaris 2.6 and later, IBM AIX 4.3 and later, HP-UX 11.0 and later, Linux 7.0 and later, and Silicon Graphics IRIX 6.5, the install_base script determines if the machine is running under NIS or DNS (using caching). If it is, the script automatically creates a lookaside database and sets two tokens in the [seosd] section of the seos.ini file to yes: under_NIS_server and use_lookaside.

The script then determines if the machine is NIS+. If it is, it sets the nis_env token in the [passwd] section to **nisplus**; otherwise, if the machine is NIS, it sets the token to **nis**. In addition, if rpc.nisd is running, the script sets the NisPlus_server token in the [passwd] section to yes.

5. The installation prompts whether you want to install a lookaside database or use system defaults (to bypass the IP caching daemon).

6. The installation prompts you for the target installation directory. The default is /opt/CA/eTrustAccessControl. These installation instructions assume you choose the default.

   You can install the eTrust AC files in a different directory; however you must then include the install_base command's -d option.

   WARNING! *You cannot put the eTrust AC database in a mounted network file system (NFS).*

7. The install_base script calls another script, which prompts you for the following information: (You can modify these settings any time after installation.)

   - The name for the group of auditors that can read the audit file.
   - Whether you want to add all your UNIX users, user groups, and hosts to the eTrust AC database now.
   - Whether you want your database to be subscribed to a PMDB; and if so, to which one.

A PMDB is a database from which information is automatically copied into other (subscriber) databases. Your answer does not actually subscribe your database to a PMDB. The most it does is allow the host on which the PMDB resides to create the subscription later.

Two safe responses to this question include:

If you want to:	Respond with:
Allow your database to be subscribed to a specific PMDB	The name of the PMDB.
Prevent your database from being subscribed to any PMDB (at least until you specify otherwise)	The Enter key.

A third response, _NO_MASTER_ , allows your database to be subscribed to any PMDB. However, this can be a dangerous response, because it removes the selection of the PMDB from your control. If you are not sure which PMDB you want to subscribe to, it is generally preferable to press Enter for the time being and then later, when you do know the PMDB name, to specify that name instead.

- The password Policy Model name.
- What users will be security administrators for eTrust AC.

Tip: See sechkey in the *Utilities Guide* for information about encryption.

- Whether you want to replace the default encryption method.

  Note: When your network includes more than one server running eTrust AC, communication between the eTrust AC and Policy Manager programs on the different servers is encrypted. By default, eTrust AC uses a fast and efficient scrambling algorithm for encryption. Installations of eTrust AC and Policy Manager use the same encryption key, enabling encrypted communication as soon as the installation is completed.

  eTrust AC also provides DES and 3DES encryption options that you can choose.
- Whether you want to set a new encryption key.
- Whether you want to install the Baseline Security rules.
- Whether you want to be able to start eTrust AC from a remote host.

Baseline Security rules offer administrators an opportunity to install a package containing two sets of rules to better protect your system, password and log files. One set of rules applies to all platforms to protect eTrust AC files. The other set protects UNIX files and is specific to the Sun Solaris, HP-UX, IBM AIX, Silicon Graphics IRIX, and Digital DEC Unix platforms. You cannot install one set of rules without the other. Baseline Security rules install in Warning mode providing you with information but not actual protection. That is why we recommend that you remove the Warning mode as soon as you become familiar with the rules.

8. Append the *eTrustACDir*/bin directory to your path (where *eTrustACDir* is the installation directory).
9. Check the other seos.ini file tokens to make sure that the settings meet your requirements. If necessary, modify the settings.

   In particular, this is a good time to change the trace_file_type token value from text to binary.

   If you want trace messages to be generated customarily and saved during normal operations, you must change the trace_to token value from file, stop to file.

   For more information on the seos.ini file in general, see the section seos.ini in this appendix.
10. Take the action that corresponds to your platform, according to the following table:

Platform	**Action**
HP-UX, IBM AIX, Linux, or Sun Solaris	For first-time installations, no action is needed. For upgrades from version 5.0 or 5.1, reboot the computer by entering: `# reboot`
NCR	Reboot the computer by entering: `# init6`

For other platforms	For first-time installations, load the eTrust AC kernel extension by entering:
	`# SEOS_load`
	
	For upgrades from version 5.0 or 5.1, reboot the computer by entering:
	`# reboot`

Now the eTrust AC installation is complete; however, it is not yet running.

11. To give yourself access to the eTrust AC man pages, add the directory *eTrustACDir*/man to your MANPATH. For example, if you are using csh, for the sake of your current session, enter the command:

    ```
    # setenv MANPATH $MANPATH:eTrustACDir/man
    ```

    where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl.

    For the sake of future sessions, add a similar line to your .login, .profile, or .cshrc file.

Installing from Unicenter Software Delivery

To install eTrust AC with Unicenter Software Delivery, follow these steps:

Note: The eTrust AC installation CD contains a directory named SDPackages. This directory contains several files that are required to install eTrust AC using Unicenter Software Delivery.

1. To register the eTrust AC Unicenter Software Delivery package, insert the eTrust AC installation CD into your drive.

   The Product Explorer appears.

2. Register the Unicenter Software Delivery package by clicking Supporting Products, and then clicking Registering in Unicenter Software Delivery.

3. Click the package to install. (For example, you might choose **eTrustAccessControlHPUX**.)

   The License Agreement window displays.

4. Scroll down to the end of the license and accept the license by clicking **I agree**.

   The Choose Products to Register window displays again with the package you selected marked.

5. Click Next to proceed with the registration process.

6. Enter the server name for the Unicenter Software Delivery server, and click Next.

   The package is registered on the server you selected.

7. To install the package, open the installation procedure in the All Software\Software Library*package* directory on the Unicenter Software Delivery server explorer and drag it into the agent or agent group you want to install in.

UNIX Command Line

From the UNIX command line, you can create a response file for silent installations, register the package, install eTrust AC to agent machines, and uninstall eTrust AC.

Generate a Reponse File

To generate a response file from the UNIX command line, follow these steps:

1. Enter the following commands on the server:

   ```
   mkdir my_tmp_reginfo/HPUX
   cd my_tmp_reginfo/HPUX
   cp cdrom/SDPackages/eTrustAccessControl.HPUX.@pif
   pifask -f ./eTrustAccessControl.HPUX.@pif -r ./eTrustAccessControl.resp
   ```

   Dialogs will open, simulating installation.

2. Complete the dialogs.

   As the process completes, you will receive a message that the response file is created.

   Note: The response file must be in UNIX format. You can FTP from UNIX using the FTP bin option.

3. Copy the response file back to the to the SDPackages directory.

Register the Package

To register the package from the UNIX command line, follow these steps:

1. Log in to the Unicenter Software Delivery server as root, mount the installation CD, and change the directory to /*cdrom*/SDPackages.
2. Run the **sdregister** command with the platform you want to register the package on:

   ```
   ./sdregister -d platform_dir
   ```

   where *platform_dir* is the name of the subdirectory in the SDPackages directory containing files for the platform you want to register on.

   A message appears, confirming registration of the software package.

3. Accept the license agreement by scrolling to the end of the agreement and clicking Yes.
4. Enter the server name, user name, and password to register with.

   The command line registers the package for you.

Installl the Package

To install eTrust AC using Unicenter Software Delivery, follow these steps:

1. On the computer serving as the Unicenter Software Delivery server, enter the sdcmd installation command for your platform. For example, you might enter the following:

   ```
   >sdcmd install item="eTrustAccessControl(HPUX)" version=5300 after=exacttime
   computer=computer_name\ procedure="Install Package with
   eTrustAccessControl.resp"
   ```

2. Enter the following command to show the status of the installation

   ```
   >sdcmd computer_name action=listJobs name=computer_name
   ```

3. After a message appears that states EXECUTION_OK, check that eTrust AC installed as you desired on *computer_name*.

Uninstalll the Package

To uninstall eTrust AC using Unicenter Software Delivery, follow these steps:

1. On the computer serving as the Unicenter Software Delivery server, enter the following command:

   ```
   >sdcmd uninstall item="eTrustAccessControl(HPUX)" version=5300
   procedure="Uninstall Package" installed with="Install Package with
   eTrustAccessControl.resp" after=exacttime computer=computer_name
   ```

2. After a message appears that states: EXECUTION_OK, check that eTrust AC is uninstalled on *computer_name*.

Starting eTrust AC

Assuming you are working in an X Windows environment, invoke eTrust AC, verify that it is correctly installed on your system, and perform the following steps to initiate important protection:

1. If you have not rebooted since installing eTrust AC, you may need to reboot now. See Step 10 in the Installation Procedure for specifics for your platform.
2. Open two windows under root (superuser) authority.
3. In either window, enter the command:

   ```
   # seload
   ```

4. Wait while the seload command starts three eTrust AC daemons: the Database Server, the Agent, and the Watchdog.

Wait for all the following messages to display.

```
#  /opt/CA/eTrustAccessControl/bin/seload
eTrust kernel extension is already loaded.
Starting eTrust daemon.  (/opt/CA/eTrustAccessControl/bin/seosd)
15 Feb 2003 13:32:11> WAKE_UP : Server going up
15 Feb 2003 13:32:11> INFO    : Filter Mask: ‘WATCHDOG*’ is registered
15 Feb 2003 13:32:11> INFO    : Filter Mask: ‘INFO    : Setting PV*’ is
registered
15 Feb 2003 13:32:11> INFO    : Filter Mask: ‘INFO    : DB*’ is registered
15 Feb 2003 13:32:11> INFO    : Filter Mask: ‘*seosd.trace*’ is registered
15 Feb 2003 13:32:11> INFO    : Filter Mask: ‘*FILE*secons*(*/log/*)*’ is
registered
Starting seosd. PID = 24732.
Starting seagent. PID = 24735
# Starting seoswd. PID - 24739
▌
```

5. After you have started the daemons, go to the other window and enter the command:

   ```
   # secons -t+ -tv
   ```

   eTrust AC accumulates a file of messages reporting operating system events. The secons -tv command displays the messages on the screen as well.

6. In the first window, where you gave the seload command, enter the following command:

   ```
   # who
   ```

   Watch the second window, where eTrust AC is writing the trace messages, to see whether eTrust AC intercepts the execution of the who command and reports on it. eTrust AC is correctly installed on your system if it reports interception of the who command.

7. If you want, enter more commands to see how eTrust AC reacts to them.

   The database does not yet contain any rules for blocking access attempts. Nevertheless, eTrust AC monitors the system so that you can see how the system behaves with eTrust AC installed and running, and which events eTrust AC intercepts.

8. Shut down the seosd daemon, by entering the following command:

   ```
   # secons -s
   ```

   The following message displays on the screen:

   ```
   seosd is now DOWN !.
   ```

Customizing eTrust AC

Implementing full-scale security using eTrust AC requires the definition of the security policies you want enforced. The time taken to make these definitions depends on the size of your site and the way you choose to manage security.

For instance, at a university you would probably not define most students to eTrust AC; they would get access based solely on resource _default settings. At a bank, however, you would probably define every user to eTrust AC and set access lists for every resource to allow specific users access to specific resources. Thus, for the same number of users, implementing eTrust AC at the university would take less time than implementing it at a bank.

As security administrator, you must define the objectives of the project. Decisions regarding site policy must be made carefully. eTrust AC includes several files that you can customize to help you implement the security policies of your site. This appendix discusses some of those files.

Registering Trusted Programs

A trusted program is a program that can be executed only as long as it has not been altered. Ordinarily it is a setuid/setgid program. eTrust AC also allows you to specify regular programs as trusted. When you are sure that the program has not been tampered with, register it in the PROGRAM class, where eTrust AC can guard its integrity.

You may want to use trusted programs together with *program pathing*, so users can perform certain tasks only by means of trusted programs. See the *Getting Started* for more information about program pathing.

eTrust AC can help you with a script to register a whole collection of setuid and setgid programs as trusted.

1. To save yourself the effort of remembering all your setuid and setgid programs, use the seuidpgm program that follows. It scans your file system, locates all setuid and setgid programs, and creates a script of selang commands that will register them all in the PROGRAM class.

   Issue this command:

   ```
   # seuidpgm -q -l -f / > /opt/CA/eTrustAccessControl/seuid.txt
   ```

   Run as shown, seuidpgm does the following:

   - Scans the entire file system (starting from /).
   - Remains quiet (the -q option suppresses the “cannot chdir” messages).
   - Ignores any symbolic links (-l).
   - Registers the programs in both the FILE and PROGRAM classes (-f).
   - Outputs the commands to file init2.

   For a complete description of the seuidpgm program, see the *Reference Guide*.

2. Using a text editor, check the seuid.txt file to be sure that it includes all the setgid/setuid programs that you want to have trusted, and no other programs. Edit the file if necessary.

3. Use selang to run the edited file of commands. If the seosd daemon is not running, include the -l switch.

   ```
   # selang [-l] -f /opt/CA/eTrustAccessControl/seuid.txt
   ```

   It may take a few minutes for selang to finish.

4. Restart the seosd daemon if it is not already running. Then check whether your system works as expected and whether setuid programs can be invoked.

5. It is advisable to change the default access of the PROGRAM class to NONE to prevent new untrusted setuid or setgid programs from being added and run without the knowledge of the security administrator.

   Enter the following command to set that default access value:

   ```
   eTrust> chres PROGRAM _default defaccess(none)
   ```

Note: Veteran eTrust AC users will remember the UACC class in this connection. That class still exists and can be used to specify the default access of a resource. However, for ease of use we recommended that for specifying the default access of a class, you use the class's _default record instead. The _default specification overrides any UACC specification for the same class.

The records in the PROGRAM class representing the setuid, setgid, and regular programs that you have registered store the following attributes of the executable files.

- Device-number
- Inode
- Owner
- Group
- Size
- Creation Date
- Creation Time
- Last-Modification Date
- Last-Modification Time
- MD5 Signature
- SHA1 Signature
- Checksum CRC (Cyclical Redundancy Check)

The most important attribute of each program you register is that the program is *trusted*. That is, the program is considered OK to run. Any change in any of the attributes listed previously causes the program to lose its trusted status, and then eTrust AC can prevent the program from running.

Warning Mode

If you are not sure whether you have successfully registered all the appropriate programs in the database, use the following command to watch for unregistered programs:

```
eTrust> chres PROGRAM _default warning
```

The warning property puts the PROGRAM class into Warning mode, meaning that a special audit record appears as a warning each time an unregistered setuid or setgid program is used but the use of such programs *is not prevented*.

Using the Audit Log

You can search for untrusted records manually in the audit log, or you can set special notification instructions to be informed when certain programs become untrusted. The special notification is especially helpful so that users do not have to contact you to use a program that has become untrusted; instead, you can check the file as soon as you receive a notification that it has become untrusted.

To set up special audit notifications, see File Notifications in the chapter "Auditing Events."

Protection

To prevent execution of setuid and setgid commands that are not trusted, issue the following command:

Note: eTrust AC automatically includes the user "nobody" in the database.

```
eTrust> newres PROGRAM _default defaccess(none) \
owner(nobody) audit(all)
```

eTrust AC then protects you against back doors and Trojan horses by requiring approval from you before allowing any new or changed program to run.

Now suppose, for example, that you have received a new, useful program that is a setuid program. You are sure it is not a Trojan horse, and you want all users to be able to execute it. To register the program as trusted, issue the following command:

```
eTrust> newres PROGRAM program-pathname \ defaccess(EXEC)
```

Retrusting Untrusted Programs

If a program has been untrusted by eTrust AC because of a change in its size, its modification date, or any other monitored property, the program will run again only if you *retrust* it, registering a new approval for it in the database. To retrust a program:

```
eTrust> editres PROGRAM progam_name trust
```

Note: You can also retrust a program with the seretrust utility. For more information about this utility and its options, see the *Utilities Guide*.

Initialization Files

This section describes various files that eTrust AC reads at initialization time. By default, eTrust AC places the initialization files in the directory containing the file seos.ini, which is the installation directory for eTrust AC.

seos.ini

The seos.ini file sets global parameters. The structure of the file and supported tokens are documented in the file itself and in the appendix "The seos.ini Initialization File."

The seos.ini file, as installed, is protected and cannot be updated while eTrust AC is running, though all users can always access it on a READ basis. Enter the following command in order to allow an authorized user to update the file while eTrust AC is running:

```
eTrust> newres FILE eTrustACDir/seos.ini owner(authUser) defacc(read)
```

where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl.

This command establishes that the default access for the file is READ; however, only the owner of the file, *authUser*, can update the file.

Note: It is important that the default access for the seos.ini file be READ because many utilities access seos.ini during their processing. If they cannot read the file, they will fail.

trace_file_type Token

The [seosd] section of the file contains a token named trace_file_type. The default value of the token is text, but you can assign it the value binary instead.

In previous versions of eTrust AC, the trace file—not the audit log, which is selective, but the file in which *all* events are recorded—was a text file. Now, by changing the value of the trace_file_type token, you can use a space-saving binary file instead, which still displays as text by the secons -tv utility.

If you do change the value of the token and a trace file already exists, the existing trace file is saved with the file name extension .backup and then a new trace file is started in the format you specified.

Other Tokens

The following additional tokens are of interest.

Section	Token	Description
seosd	dbdir	The directory in which the database is located.
	trace_to	Determines where trace messages are sent. Valid values are: **file**; **none**; and **file,stop** (stop after initialization; this is the default).
	trace_file	If the trace_to token is set to file, the name of the file in which the trace messages are written.
	trace_filter	The name of the file containing the trace filter masks.
	under_NIS_server	Determines whether the files are on an NIS file server. Valid values are yes, no, and DNS.
logmgr	audit_log	The name of the audit log file.
	error_log	The name of the error log file.
	audit_size	The size, in KB, of the audit log file.
	error_size	The size, in KB, of the error log file.
message	filename	The name of the file containing the error messages.
seos	SEOSPATH	The home directory of eTrust AC. The default setting is /opt/CA/eTrustAccessControl.

Trace Filter File

The trace filter file specifies the trace messages that are to be filtered out (that is, those messages that are not to appear in the trace). Each line specifies a mask that identifies a group of messages to be suppressed. For example, the following file suppresses all messages that begin with WATCHDOG or INFO and all messages that end with BYPASS.

```
WATCHDOG*
*BYPASS
INFO*
```

By default, eTrust AC uses a trace filter file named trcfilter.init. Edit the file as required. To add remarks (comment lines) to the file, place a semicolon (;) at the beginning of the line.

For more information, see the seosd utility in the *Utilities Guide*.

Installing Unicenter Security Integration Tools

Use one of two types of Unicenter Security integration installations for UNIX environments.

Full Integration
The full integration installation is useful for eTrust AC installations with Unicenter Security in use. The integration imports data from Unicenter Security to eTrust AC, so eTrust AC becomes the security system used on that host or group of hosts.

Minimal Integration
The minimal integration installation is useful for eTrust AC installations without Unicenter Security or for installations that include Unicenter Security, but it is not in use.

For **full** integration of Unicenter Security and eTrust AC, complete the following steps:

Important! *To run the migration, you must log on as root; you cannot run the su (substitute user) command to change to root after you install eTrust AC.*

1. Install eTrust AC without populating the eTrust AC database.

   To avoid populating the database, accept the default of No when the following prompt appears on the screen:

   ```
   Import users, groups and hosts now? [y/N] :
   ```

2. Run the uni_migrate_master.sh script on the master node.

   Note: The master node is the machine that hosts the Unicenter Security database.

3. Run the uni_migrate_node.sh script on each satellite node (that is, every Unicenter Security-controlled machine).

4. Run the uni_migrate_node.sh script on the master node.

   The master node is the last machine to disable Unicenter Security after all the other nodes have been integrated.

5. Manually edit the $CAIGLBL0000/secopts file to set the value for the SSF_SCOPE_DATA and SSF_SCOPE_KEYWORD keywords to **NO**.

The installation scripts perform the following tasks:

- Execute a shell script, defclass.sh, to define user-defined security asset types as eTrust AC classes in the eTrust AC database.
- Run a program, migopts, to read and translate the current Unicenter Security environment to a similar eTrust AC environment.
- Run a program, exporttngdb, to read and translate current Unicenter Security database objects to eTrust AC database objects.

- Stop and disable the Unicenter Security daemons.

For **minimal** integration of Unicenter Security and eTrust AC, complete the following steps:

Important! *To run the migration, you must log on as root; you cannot run the su (substitute user) command to change to root after you install eTrust AC.*

1. Run the uni_migrate_node.sh script on all nodes.
2. Manually edit the $CAIGLBL0000/secopts file to set the value for the SSF_SCOPE_DATA and SSF_SCOPE_KEYWORD keywords to **NO**.

Installation Notes

- We do not recommend running Unicenter TNG login intercepts after running the Unicenter Integration and Migration Installation. When the Unicenter Integration and Migration Installation has completed successfully, Unicenter TNG login intercepts are disabled.
- 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.
- Unicenter Security rules that have been implemented against any of the following Unicenter Security asset types are obsolete because Unicenter Security is no longer in use: 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.

  The -e (-edit) flag available for uni_migrate_node.sh and uni_migrate_master.sh allows you to see and edit the rule entering the eTrust AC database.
- If you want full or minimal Unicenter TNG integration, then you must install the Unicenter Integration and Migration package with the –uni option to the install_base script. The Unicenter Integration and Migration Installation installs the Unicenter Integration and Migration scripts and binary files in the *eTrustACDir*/tng directory.
- Do not use selang -c during migration if you are listing more than one command. Instead, use selang -f input_file_name.

Starting eTrust AC Automatically

After you have tested eTrust AC and experimented with its features, you are ready to implement eTrust AC protection.

To arrange for the seosd daemon to start automatically upon boot, so that your resources are protected immediately, use the procedure for your operating system as described in the following sections.

IBM AIX 4.3

To start the seosd daemon automatically upon startup of an IBM AIX 4.3 system:

1. Copy the rc.SeOS.base file from the *eTrustACDir*/samples/system.init/aix directory to the *eTrustACDir* directory (where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl).
2. Copy the inittab.line file (a one-line file) from the *eTrustACDir*/samples/system.init/aix directory to the *eTrustACDir*/etc/inittab file, so that it appears as the file's last line.
3. The rc.SeOS.base file contains two sleep commands. If you encounter problems when starting the seosd daemon, increase the sleep value in the first sleep command. Increase it by a few seconds at a time until the problem disappears.

HP-UX 11.x

To start the seosd daemon automatically upon startup of an HP-UX 11.x system:

1. Copy the rc.SeOS.base file from the *eTrustACDir*/samples/system.init/HPUX10 directory to the /sbin/init.d directory or to the *eTrustACDir* directory (where *eTrustACDir* is the installation directory for eTrust AC, by default /opt/CA/eTrustAccessControl). In either case, you also need a symbolic link, as follows.

   To copy to /sbin/init.d, use the following command:

   ```
   # cp \ /opt/CA/eTrustAccessControl/samples/system.init/HPUX10/rc.SeOS.base \
   /sbin/init.d
   ```

   ```
   # ln -s /sbin/init.d/rc.SeOS.base /sbin/rc2.d/S890seos
   ```

   To copy to /opt/CA/eTrustAccessControl, use the following command:

   ```
   # cp \ /opt/CA/eTrustAccessControl/samples/system.init/HPUX10/rc.SeOS.base \
   /opt/CA/eTrustAccessControl/
   ```

   ```
   # ln -s /sbin/rc2.d/S890seos /opt/CA/eTrustAccessControl/rc.SeOS.base
   ```

2. The rc.SeOS.base file ends with a sleep command. Originally, the sleep value is 30 seconds; on most machines, 30 seconds is sufficient. But if you encounter problems when starting the seosd daemon, you should edit the file to increase the sleep value. Increase it by a few seconds at a time until the value is sufficient.

Sun Solaris 2.x

To start the seosd daemon automatically upon startup of Sun Solaris 2.x or higher:

1. Give yourself root privileges, if you do not have them already.
2. Copy the S99SEOS file from the *eTrustACDir*/samples/system.init/Solaris2.x directory to the /etc/rc2.d directory.

The file with reside in the /etc/rc2.d directory together with a file named S68SEOS that was installed when you set up eTrust AC. Both files are necessary.

Appendix

B NIS Configuration

This appendix provides additional information about name resolution and related issues for eTrust AC users applying Network Information Systems (NIS) or Domain Name Services (DNS).

Note: This section supplements material covered by the installation script. This appendix assumes you are familiar with NIS, DNS, and UNIX name resolution concepts.

Installation Notes

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 cashing daemon on your system.
  - If you use NCR, UnixWare, HP-UX 10.20, DEC, or SINIX, and they are **not** NIS servers, the default uses the system functions for name resolution.
  - If you use NCR, UnixWare, HP-UX 10.20, DEC, or SINIX, and they **are** NIS servers, 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 2.5.1 and higher, IBM AIX 4.3 and higher, Silicon Graphics IRIX 6.5 and higher, DYNIX 4.4.4 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.

Name Resolution

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 nscd 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 using /etc 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. See Configuration Tokens: the seos.ini File in this appendix for a description of the relevant tokens used to implement the lookaside database.

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:
  1. In the [seosd] section of seos.ini change the following tokens to **yes**:
     - under_NIS_server
     - use_lookaside
  2. 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. For descriptions of the lookaside database tokens in seos.ini, see the [seosd] section in the appendix “The seos.ini Configuration File.” 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 the /etc files.

Implementing the Lookaside Database

Tip: 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 the /etc 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

The following table lists the seos.ini tokens used in the eTrust AC initialization process.

Token	Section	Meaning
exits_dir	lang	Specifies the target directory for /opt/CA/eTrustAccessControl/lib/install_exits.sh.
GroupidResolution	seosd	Determines how eTrust AC translates gid numbers to group names.
HostResolution	seosd	Determines how eTrust AC translates IP addresses to host names.
lookaside_path	seosd	Specifies the directory in which the lookaside database resides. Create this directory before running the sebuildla utility.
nis_env	passwd	Specifies whether the local host is an NIS or NIS+ client.
parent_pmd	seos	Specifies the PMDB to which this station subscribes.
passwd_pmd	seos	Specifies a target for password replacement on the Policy Model.
pre_user_exit	lang	Specifies the exit to be called before a user command is executed in the UNIX environment.
pre_group_exit	lang	Specifies the exit to be called before a group command is executed in the UNIX environment.
post_group_exit	lang	Specifies the exit to be called after a group command is executed in the UNIX environment.
post_user_exit	lang	Specifies the exit to be called after a user command is executed in the UNIX environment.
ServiceResolution	seosd	Determines how eTrust AC translates TCP port numbers to service names.
under_NIS_server	seosd	Determines whether eTrust AC uses its own internal name resolution. This token remains for purposes of backward compatibility only.

Token	Section	Meaning
AllowedGidRange	passwd	Specifies reserved numbers. The integers below the first number and above the second number are reserved GIDs that eTrust AC cannot update.
AllowedUidRange	passwd	Specifies reserved numbers. The integers below the first number and above the second number are reserved UIDs that eTrust AC cannot update.
UseridResolution	seosd	Determines how eTrust AC translates uid numbers to user names. The valid values are system, cache, and ladb.
use_lookaside	seosd	This token remains for purposes of backward compatibility only.
YpGrpCmd	passwd	Specifies the command for making the NIS group map.
YpMakeDir	passwd	Specifies the name of the makefile directory to be used when creating NIS maps.
YpPassCmd	passwd	Specifies the command to use to generate the NIS password map.
YpServerGroup	passwd	Specifies the group file from which the NIS group map is made.
YpServerPasswd	passwd	Specifies the source of the NIS password map.
YpServerSecure	passwd	Specifies the name of the security file that contains passwords and that is used for building the password map.

Appendix C

The seos.ini Initialization File

This appendix provides a detailed description of the eTrust AC initialization file, seos.ini.

The seos.ini file contains various setup and initialization tokens used by eTrust AC. Each token occupies a line in the file, in the following format:

```
token = value
```

The lines containing the tokens for a particular utility, daemon, or other facility of eTrust AC are grouped together as a section. Each section starts with a header line that gives the section name inside square brackets. For example, the following line starts the section that governs the serevu utility:

```
[serevu]
```

Every token belongs to a section. This appendix describes the tokens, section by section.

Note: The seos.ini file, as installed, is protected by eTrust AC and cannot be updated while eTrust AC is running. Enter the following command to allow an authorized user to update the file while eTrust AC is running:

```
eTrust> newres FILE /opt/CA/eTrustAccessControl/seos.ini owner(authUser)
```

where *authUser* is the name of an authorized user. This command establishes that *authUser* is the owner of the file, and as the owner of the file, *authUser* can always update it.

The seos.ini file, as defined by default in eTrust AC, has READ access because many utilities access this file during their processing. If they cannot read the seos.ini file, they will fail.

Using the -rl option in secons, you can now update or refresh select tokens inside seosd without having to restart the daemon. The parent_pmd token, for instance, can be updated in this manner.

Sections in seos.ini

The following table lists all the sections in the seos.ini file.

Section	Description
daemons	Specifies programs that seload runs automatically.
dependency	Specifies products that use eTrust AC as an embedded component, as defined by users
lang	Specifies the command language program attributes.
ldap	Defines the location of the LDAP server and its variables.
logmgr	Controls the logging facility.
message	Controls the message services.
mfsd	Defines the mainframe synchronization daemon options.
package	Specifies packages that you selected to install.
pam_seos	Controls the PAM programming interface.
passwd	Defines password replacement and other user-related services.
pmd	Specifies the PMDB attributes.
rsv	Specifies the RSV daemon attributes.
seam	Specifies the Motif GUI attributes.
seauxd	Specifies calendar authorization attributes.
secmon	Specifies the attributes of the GUI audit display.
segrace	Specifies the segrace utility attributes.
seini	Specifies the seini intelligent search attributes.
selock	Controls the selock utility.
selogrd	Controls the log routing daemons.
seos	Specifies the global settings.
seosd	Controls the authorization daemon.
seosdb	Controls database checking and rebuilding.
seoswd	Controls the Watchdog daemon.
seos_syscall	Controls the seos_syscall kernel module.
serevu	Specifies the serevu utility attributes.
sesu	Controls the sesu utility.

Section	Description
sesudo	Specifies the sesudo utility attributes.
tng	Controls the integration of eTrust AC into the Unicenter TNG environment.

daemons

In the [daemons] section, each token specifies whether (and if so, how) the seload utility executes a particular program from the eTrust AC installation directory.

Token	Meaning	Default Value

program-name
 	
(For example: selogrcd, selogrd, serevu, or sersvd)
 	
Note: You do not need to specify the seosd daemon. Seload always ensures that the seosd daemon is running.
	Specifies one of two possibilities:
	
	■ The name of a daemon or other program to be matched with:
	– a **yes** value, so seload runs the program with default parameters
	– a **no** value, so seload does not run the program
	– a set of parameters, so seload runs the program with those parameters
	
	For example, enter the following to run serevu from the eTrust AC installation directory with default parameters:
	`serevu=yes`
	Enter the following to refrain from running serevu; this is the same as using no serevu token at all.
	`serevu=no`
	Enter the following to run serevu from the eTrust AC installation directory with the specified parameters:
	`serevu=-f 3 -d 6m -t 1m -s 5m`
	
	■ A dummy string, to be matched with the absolute path name of a daemon or other program, followed optionally by parameters, so seload runs the program accordingly.
	
	For example, enter the following to run the serevu utility that resides in the /opt/CA/eTrustAccessControl/alt directory, with the specified parameters:
	`run_it=/opt/CA/eTrustAccessControl/`
	` alt/serevu \`
	`-f 3 -d 6m -t 1m`
	
	To include specifications for several programs, use the token once for each program.
		no

dependency

In the [dependency] section, each user-defied token specifies a product that uses eTrust AC as an embedded component.

Token	Description	Default Value
product-name	Specifies a product that uses eTrust AC as an embedded component.	No default

lang

In the [lang] section, the tokens specify the attributes used by the selang command language programs: selang, Security Administrator, and seadm.

Token	Description	Default Value
check_password	Determines whether selang will request users to specify their own passwords. Valid values include: **no**—selang does not require any passwords **yes**—Users are prompted to enter their passwords.	no
exit_timeout	Specifies the maximum time, in seconds, that eTrust AC allows the exit program to execute. After this time has passed, eTrust AC kills the exit program.	15
exits_dir	Specifies the target directory where exits are installed by the /opt/CA/ eTrustAccessControl/lib/install_exits.sh shell script.	/opt/CA/ eTrustAccessControl/ exits
exits_source_dir	Specifies the source directory of the exits to be installed by the /opt/CA/ eTrustAccessControl/lib/install_exits.sh shell script.	/opt/CA/ eTrustAccessControl/samples/ exits-src
help_path	Specifies the directory in which lang help files are located.	/opt/CA/ eTrustAccessControl/ data/langhelp
language	Determines which language eTrust AC will use.	English

Token	Description	Default Value
max_groups_buffsize	Specifies the buffer size, in KB, that the security administrator uses when communicating with the database. This token is used when a UNIX update needs to be applied.	128
no_check_password_ users	Specifies users who are not asked to enter their passwords. This token is relevant only if the token check_password is set to **yes**. Valide values include a list of users separated by commas.	none
passwd_copy	Specifies how the machine password file (/etc/passwd) or PMDB password file (/*PMDB_Directory*/policies/pmdb/ passwd) is updated when you copy the temporary file back to the original after changing user information. Valid values include: **fast_copy**—Copy information over the file. **rename**—Change the directory to point to the new file.	fast_copy
post_group_exit	Specifies the path of the exit program to be called after a group command is executed in the UNIX environment.	/opt/CA/eTrustAccessControl/ exits/lang_exit.sh
post_user_exit	Specifies the path of the exit program to be called after a user command is executed in the UNIX environment.	/opt/CA/eTrustAccessControl/ exits/ lang_exit.sh
pre_group_exit	Specifies the path of the exit program to be called before a group command is executed in the UNIX environment.	/opt/CA/eTrustAccessControl/ exits/ lang_exit.sh
pre_user_exit	Specifies the path of the exit program to be called before a user command is executed in the UNIX environment.	/opt/CA/eTrustAccessControl/ exits/ lang_exit.sh
query_size	Specifies the maximum number of records to be listed in a database query.	100

Token	Description	Default Value
RecvTimeOut	Specifies the maximum time, in seconds, that selang will wait to receive information before timing out. If you set the value to 0, there will be no timeout.	60
SendTimeOut	Specifies the maximum time, in seconds, that selang will wait to send information before timing out. If you set the value to 0, there will be no timeout.	60
setBlockRun	Specifies whether to check if a program is trusted and block the execution of untrusted programs. The execution blocking is performed regardless whether the program is a setuid or a regular program. Valid values include the following: **yes**—All programs defined with viapgm authorization rules have the blockrun property set to yes. **no**—All programs defined with viapgm authorization rules have the blockrun property set to no. **suid**—All setuid programs have the blockrun property set to yes, and all other programs have the blockrun property set to no.	yes
timeout	Specifies the maximum time, in seconds, the client waits for seosd daemon to respond. If seosd does not respond within this period, an error message is sent noting that seosd is not responding. The client then stops trying to connect to seosd.	90

Token	Description	Default Value
use_unix_file_owner	Specifies whether a UNIX owner of a file can define the file to eTrust AC. If the value is yes, an owner of a file in UNIX can define it to eTrust AC, using the newres or newfile command. If the file is already defined to eTrust AC, the user cannot change its parameters in the database unless the user is allowed to do so according to the normal eTrust AC authorization rules. Valid values are yes and no.	no

ldap

In the [ldap] section, the tokens specify the attributes used to locate the LDAP server and input data. These parameters are used only by the ldap sample exit located in /opt/CA/eTrustAccessControl/ samples/ldap/exits/ S50CREATE_Ldap_u.sh.

Token	Description	Default Value
base_entry	Specifies the point in the LDAP directory tree to be used as the base entry point. For example, you may use o=*organization_name*, c=*country_name*.	No default
host	Specifies the host name of the LDAP server.	localhost
past	Specifies the LDAP client base directory.	/usr/local/ldap
port	Specifies the LDAP server port (optional).	389

logmgr

In the [logmgr] section, the tokens control the behavior of the logging facility.

Token	Description	Default Value
audit_back	Specifies the name of the audit log backup file. Only eTrust AC can write to this file. Users can have READ access only to this file.	/opt/CA/ eTrustAccessControl/log/ seos.audit.bak

audit_group	Specifies the group that can read the audit logs. If you set this token to **none**, only root can read the audit logs. eTrust AC does not verify the value of this token, so if you enter an invalid group name, eTrust AC does not assign any group permissions to the audit log files.
	
	To change the group ownership of an existing audit log file, complete the following steps:
	
	1. Use the selang command chgrp to set the group ownership of the files.
	2. Change the UNIX permissions by entering the following command:
	
	`chmod 640 \`
	`/opt/CA/eTrustAccessControl/log/ \`
	`\seos.audit`
		none

audit_log	Specifies the name of the audit log file. When this file reaches the size specified in *audit_size*, eTrust AC closes the file, renames it with the name in *audit_back*, and creates a new audit log. Only eTrust AC can write to this file. Users can have READ access only to this file.	/opt/CA/ eTrustAccessControl/ log/seos.audit
audit_size	Specifies the maximum size, in KB, of the audit log file. Minimum value: 50 KB.	1024

Token	Description	Default Value
BackUp_Date	Sets the criterion by which eTrust AC performs the backup. Valid values are: none, yes, daily, weekly, and monthly. If you specify **yes**, eTrust AC performs backups according to the size limit token audit_size and timestamps the file. If you specify **none**, eTrust AC performs the backup according to the audit_size token but does not timestamp the file. If you specify **daily**, **weekly**, or **monthly**, eTrust AC adds a timestamp when it first creates the audit log file. When the current date passes the timestamp, eTrust AC automatically creates a backup file and timestamps it. However, if the size of the audit log file exceeds the value of the audit_size token first, eTrust AC creates a backup file without issuing a timestamp.	none
error_back	Specifies the name of the error log backup file.	/opt/CA/eTrustAccessControl/log/seos.error.bak

error_group	Specifies the group that can read the error log files. If you set this token to **none**, only root can read the error log files. eTrust AC does not verify the value of this token, so if you enter an invalid group name, eTrust AC does not assign any group permissions to the error log files.
	
	To change the group ownership of an existing error log file, complete the following steps:
	
	1. Use the selang command chgrp to set the group ownership of the files.
	2. Change the UNIX permissions by entering the following:
	
	`chmod 640 \`
	`/opt/CA/eTrustAccessControl/log/ \`
	`seos.audit`
		none

Token	Description	Default Value
error_log	Specifies the name of the error log file. When this file reaches the size specified in *error_size*, eTrust AC closes the file, renames it with the name in *error_back*, and creates a new error log. Only eTrust AC can write to this file.	/opt/CA/ eTrustAccessControl/log/ seos.error
error_size	Specifies the maximum size, in KB, of the error log file. Minimum value: 50	50
logconnected	Prevents that TCP-CONNECTED records from being written to the audit log. Set logconnected to No to use this feature.	none

For more information, see:

- The seaudit and seerrlog utilities in the *Utilities Guide*
- The seauditx application in the *User Guide*

message

In the [message] section, the tokens control the behavior of the message utility semsgtool.

Token	Description	Default Value
filename	Specifies the location and name of the file that supplies most of the messages that appear in response to typed selang commands.	/opt/CA/ eTrustAccessControl/data/s eos.msg

mfsd

In the [mfsd] section, the tokens define the mainframe synchronization daemon options.

Token	Description	Default Value
mfsd_trace_file	Specifies the location of the file to which eTrust AC mainframe synchronization daemon mfsd trace messages are written. If this token is set to **no**, the trace file is not created.	/opt/CA/ eTrustAccessControl/log/ mfsd.trace

package

In the [package] section, the tokens specify the packages you selected to install.

Token	Description	Default Value
Admin Api Client Mfsd Server Stop Uni	Indicates whether you selected to install the specified package.	No default

pam_seos

In the [pam_seos] section, the tokens help you to more fully exploit the programming interface PAM (Pluggable Authentication Module).

Note: The [pam_seos] section is available only for HP-UX 11.00 and above and Sun Solaris 7.1 and above.

Token	Description	Default
call_segrace	Specifies whether to automatically call the segrace utility with any login. Valid values are yes and no.	no

Token	Description	Default
debug_mode_for_user	Specifies whether to inform the user of the reason for login denial. Valid values are yes and no.	no
failed_login_file	Specifies the location of the failed login audit file pam_seos.	opt/CA/eTrustAccessControl/ log/ pam_seos_failed_logins.log
serevu_use_pam_seos	Specifies whether serevu should use the pam_seos login failure log file instead of the system file. This feature increases the accuracy of serevu.	no

passwd

In the [passwd] section, the tokens define password replacement and other user-related services.

Token	Description	Default Value
AllowedGidRange	Specifies the range of GIDs that the user can add, update, and delete. Values outside this range represent reserved GIDs that eTrust AC cannot update. If you specify only one integer, all integers from 1 to the specified integer are reserved GIDs.	100,30000
AllowedUidRange	Specifies the range of UIDs that the user can add, update, and delete. Values outside this range represent reserved UIDs that eTrust AC cannot update. If you specify only one integer, all integers from 1 to the specified integer are reserved UIDs. If you set this toke to **-1**, then the PMDB updates the password for root or propagates the root password to all subscribers. If the value is set to **0**, then the root password is not allowed.	100,30000
Check_Adm_Rules	Specifies whether to enforce password rules for ADMIN users.	yes

Token	Description	Default Value
DefaultHome	Specifies the default home directory of the system. The user's home directory is a subdirectory of the specified system home directory. For example, if the system home directory is /home, the new user's home directory is /home/*username*. If specified, the value for this token overrides the value in the client's lang.ini file. If you specify *nohomedir* then a home directory is not automatically set.	/home
DefaultPasswdCmd	Specifies the default password program. If specified, this password program is used when sepass is started and seosd is not running.	/bin/passwd
DefaultPgroup	Specifies the primary group that eTrust AC assigns to a new UNIX user if no value is entered.	other
DefaultShell	Specifies the default shell that eTrust AC assigns to a new UNIX user if no value is entered. If specified, the value for this token overrides the value in the client's lang.ini file.	/bin/sh
Dictionary	Specifies the file containing the words that cannot be used as passwords. If this token is set to **db** then the entire dictionary is in the database and the password is checked against the eTrust AC database.	/usr/dict/words
GeneratePasswd	Specifies whether sepass generates a new password by itself. Valid values are yes and no. If you set this token to **no**, the user is asked to enter a new password.	no
nis_env	Specifies whether the local host is an NIS or NIS+ client. Valid values are no, nis, or nisplus.	no

Token	Description	Default Value
NisPlus_server	Specifies whether this station is an NIS+ server. Valid values are yes and no. If token value is yes, eTrust AC treats password replacements as NIS+ password replacements.	no
only_pmdb	Specifies whether the default setting for sepass includes the -p flag. If token value is yes, it instructs sepass to change the password only on the PMDB at the host specified. If no such database is defined, sepass does nothing.	no
passwd_format	Indicates whether the password changes are propagated to an NT host. Setting this token to **NT** means that one of the hosts you are administering is an NT host.	none
PromptOldPassword	Specifies whether to prompt local users for their old password when sepass is invoked through /opt/CA/eTrustAccessControl/bin/segrace. (You must use the full path). Setting this token to **yes** indicates that the users are prompted for their old passwords.	yes
quiet_mode	Specifies whether sepass displays a copyright notice and a message about propagating passwords to Policy Models.	no
SaveGroupAttrs	Specifies whether the previous group file owner, group, and mode are preserved after an update of a group in the UNIX environment. Valid values are yes and no. If you set this token to **no**, the new values are set to 0, 0, 644 respectively.	no

Token	Description	Default Value
SavePasswdAttrs	Specifies whether the previous password file owner, group, and mode are preserved after an update of a user in the UNIX environment. Valid values are yes and no. If you set this token to **no**, the new values are set to 0, 0, 644 respectively.	no
UIDAlgorithm	Specifies which free UID algorithm to employ when adding new users. Setting it to any other value would select the older process. The **new** algorithm provides for UID numbers over 64 KB and is generally faster.	new
UseDict	Specifies whether to use the dictionary file (set with the Dictionary token) when verifying a password.	no
YpGrpCmd	Specifies the command to use for generating the NIS group map.	make group
YpMakeDir	Specifies the name of the makefile directory to be used when creating NIS maps.	/var/yp
YpPassCmd	Specifies the command to use for generating the NIS password map.	make passwd
YpServerGroup	Specifies the group file from which the NIS group map is made.	/etc/group
YpServerPasswd	Specifies the password file from which the NIS password map is made.	/etc/passwd
YpServerSecure	Specifies the name of the security file containing passwords that is used for building the NIS password map.	/etc/shadow
YpTimeOut	Specifies the time, in seconds, that a new client (selang, Security Administrator, and so forth) can run the ypbind test, which determines whether the local host is connected to a NIS server. At expiration, the client exits and an error message appears. The default value of zero (**0**) means that no ypbind test is conducted.	0

For more information, see the sepass utility in the *Utilities Guide*.

pmd

In the [pmd] section, the tokens determine the PMDB attributes.

Note: In addition to seos.ini, each policy model has a configuration file named pmd.ini.

Token	Description	Default Value
_min_retries_	Specifies the minimum number of attempts made by sepmdd to access an unavailable subscriber before giving up and temporarily shutting itself down. If sepmdd shuts itself down without updating the subscriber, it attempts to update the subscriber when it next starts.	4
_pmd_directory_	Specifies the directory in which the PMDBs reside. The name can contain up to 70 alphanumeric characters. Specify the full path of the directory. Each Policy Model resides in the directory *pmdDirectory/pmdName*.	/opt/CA/eTrustAccess Control/policies
_QD_timeout_	Specifies the maximum time, in seconds, that the sepmdd daemon waits while attempting to update a subscriber database during the first scan of its subscriber list. If the 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. After completing the first scan of the subscriber list, sepmdd then performs a second scan in which it attempts to update the subscribers it did not succeed in updating during the first scan. During the second scan, it tries to update a subscriber until the connect system call times out (approximately 90 seconds).	3

Token	Description	Default Value
_retry_timeout_	Specifies the time, in minutes, between consecutive attempts to access an unavailable subscriber.	30
_shutoff_time	Specifies the time, in minutes, sepmdd waits before shutting itself off. If the token value is zero, sepmdd never shuts itself off.	0
is_maker_checker	Specifies whether to use Dual Control. Valid values are yes and no. If the token value is **yes**, you cannot update the database directly, but only through a PMDB, and two administrators—a Maker and a Checker—must collaborate on the update.	no
pass_auth	Controls password authentication during a remote password change. Valid values are yes and no. If the token is set to **yes**, sepass compares the old password entered by the user with the password stored in the remote database. Regardless of the value of the token, the utility compares the old password with the password stored in the local station.	yes
pull_option	Specifies whether subscriber databases are updated as soon as they become available. Valid values are yes and no. If the token value is **yes**, seagent sends a message to the parent Policy Models of both the local host and any Policy Model on the machine as soon as the subscriber station becomes available. sepmdd then updates the subscriber immediately, instead of waiting for the next half-hourly retry.	yes

Token	Description	Default Value
send_unix_env	Specifies whether the sepmd -n option sends the contents of the policy model password files and group files. Valid values are yes and no. **yes**—The *sepmd -n* option sends the contents of the policy model password files and group files. **no**—The *sepmd -n* option does **not** send the contents of the policy model password files and group files.	yes
updates_in_chunk	Determines the maximum number of commands that the Policy Model sends to each of its subscribers in each cycle of a loop.	10

For more information, see the chapter "Auditing Events" or the sepmd utility in the *Utilities Guide*.

rsv

In the [rsv] section, the tokens determine the attributes of the Remote Status View (RSV) daemon.

Token	Description	Default Value
rsv_port	Specifies the port that the RSV daemon monitors.	8080
rsv_user	Provides an option to switch the executing UNIX user of the RSV daemon to another UNIX user. This option is applicable only if the executing user is root. If you specify a user, and root loaded the RSV daemon, the RSV daemon switches to the specified user.	No default

seam

In the [seam] section, the tokens determine the attributes of the Motif GUI.

Note: This section of the seos.ini is available only when you install the Security Administrator, the UNIX GUI.

Token	Description	Default Value
data	Specifies the location of the seam.ini file.	/opt/CA/eTrustAccessControl/data/seam

seauxd

In the [seauxd] section, the tokens determine the usage and refresh interval of the Unicenter TNG calendar and help manage name resolution.

Token	Description	Default Value
allow_any_platf	Indicates whether to use name resolution on unsupported platforms. Valid values are yes and no.	no
client_request_timeout	Specifies the time interval, in seconds, to keep a request for resolution.	120
file_time_check	Specifies the time interval, in seconds, to check for changes in /etc/passwd. Specifying **0** disables checking.	10
init_delay	Specifies the time, in seconds, to wait for seauxd to start up.	10
log_file_name	Specifies the name of the auxiliary log file. Its location is SEOSPATH/log.	seauxd.log
log_file_size	Specifies the maximum size, in KB, of the auxiliary log file. If size is exceeded, the file is truncated to 0.	100

Token	Description	Default Value
log_level	Specifies the level of logging to be used. Valid values include the following: **0**—Minimum info **1**—ERR **2**—WARN + ERR **3**—NOTIC + WARN + ERR **4**—DEBUG + INFO + WARN + ERR	0
name_resolving	Indicates whether the auxiliary daemon is used for name resolution. Not all platforms support name resolution. Valid values are yes and no.	no
NR_cache_updates	Indicates whether the auxiliary daemon needs to update cached requests. Valid values are yes and no.	no
NR_cache_updates_check_interval	Specifies the time, in seconds, between cache updates.	120
NR_uid_request_timeout NR_gid_request_timeout NR_host_request_timeout NR_service_request_timeout	Specifies the time interval, in seconds, to resolve these requests: user, group, host, and services.	5 (for all types of requests)
req_poll_timeout	Specifies the time interval, in milliseconds, to wait for input requests.	200
respawn_seauxd_delay	Specifies the time out, in seconds, to respawn the auxiliary daemon if auxiliary quits.	60
TNG_cal_lib	Specifies the name of the shared library containing the Unicenter TNG calendar.	libcalendar
TNG_calendars	Specifies whether to use the Unicenter TNG calendar to restrict resources at set time intervals.	NO
TNG_lib_path	Specifies the path for eTrust AC to find the shared library containing the Unicenter TNG calendar.	/opt/CA/Calib
TNG_refresh_interval	Specifies the refresh interval, in minutes, for eTrust AC to retrieve active calendar information from Unicenter TNG.	10

Token	Description	Default Value
trace_cnt	Indicates whether to write counters in trace file. Valid values are yes and no.	no

seam

In the [secmon] section, the tokens determine the attributes of the SecMon auditing GUI.

Note: This section of the seos.ini is available only when you install the Security Administrator, the UNIX GUI.

Token	Description	Default Value
data	Specifies the location of the secmon settings file.	/opt/CA/eTrustAccessControl/data/secmon

segrace

In the [segrace] section, the tokens determine the attributes of the segrace utility.

Token	Description	Default Value
sepass_command	Specifies the location of the eTrust AC password replacement command that is executed when a user has no remaining grace logins.	/opt/CA/eTrustAccessControl/bin/sepass

For more information, see the segrace utility in the *Utilities Guide*.

seini

In the [seini] section, the tokens determine the attributes of the seini intelligent search feature.

Token	Description	Default Value
get_error_warning	Specifies whether the error and warning messages for the intelligent search feature display.	yes
perform_action	Specifies whether seini performs its operations on the token or section found by the intelligent search feature. Valid values are yes and no. If this token is set to **yes**, the section and token, found by the additional intelligent search, are used for the requested seini operation.	no
use_intelligent_search	Specifies whether to perform an intelligent search when you invoke the seini utility.	no

For more information, see the seini utility in the *Utilities Guide*.

selock

In the [selock] section, the tokens control the behavior of the selock utility.

Token	Description	Default Value
unlocking_user	Specifies the name of a user, other than the owner, who can unlock a locked screen.	root

For more information, see the selock utility in the *Utilities Guide*.

selogrd

In the [selogrd] section, the tokens control the behavior of the log routing daemons selogrd and selogrcd.

Token	Description	Default Value
Caudit_size	Specifies the maximum size, in KB, of the audit collection file, before selogrcd creates a backup file and opens a new file. The minimum value is 50 KB.	1024
CBackUp_Date	Sets the criterion by which selogrcd performs the backup. Valid values include: none, yes, daily, weekly, and monthly. If you specify **yes**, eTrust AC performs backups according to the size limit token Caudit_size and timestamps the file. If you specify **none**, eTrust AC performs the backup according to the Caudit_size token but does not timestamp the file. If you specify **daily**, **weekly**, or **monthly**, selogrcd adds a timestamp when it first creates the file. When the current date passes the timestamp, eTrust AC automatically creates a backup file and timestamps it. However, if the size of the file exceeds the value of the Caudit_size token first, eTrust AC creates a backup file without issuing a timestamp.	none
ChangeLogFactor	Specifies the factor applied to the value in the token *interval* before testing whether the log file was changed to a backup file.	5 (3 x 5 = 15 seconds)

Token	Description	Default Value
CipherName	Specifies the name of the file that contains the encryption functions used by selogrd if the UseEncryption token is set to eTrust. This file must be placed in the /opt/CA/eTrustAccessControl/lib/ directory. The CipherName is a symbolic link to a shared object file.	adcipher
CollectFile	Specifies the name of the file in which the audit collector daemon selogrcd stores the collected audit records.	/opt/CA/ eTrustAccessControl/log/ seos.collect. audit
CollectFileBackup	Specifies the name that selogrcd uses when backing up and renaming the file of collected audit records when it receives the USR1 signal.	/opt/CA/ eTrustAccessControl/log/ seos.collect. bak
DataFile	Specifies the name of the file to which the target routing information is written before being delivered to the specified targets.	/opt/CA/ eTrustAccessControl/log/ logroute.dat
Interval	Specifies the time interval, in seconds, between each poll of the log file by the selogrd daemon.	5
KeyFile	Specifies the name of the file that holds the audit encryption key. This key is used when selogrd performs eTrust AC audit encryption. The location of key file is opt/CA/ eTrustAccessControl/lib/directory. The key can be changed by sechkey utility.	adcipher.bin
Mailer	Specifies the name of the program that selogrd uses to send email. **Note:** This option is relevant only if you set the UseSmtpMail token to yes.	/bin/mail

Token	**Description**	**Default Value**
MaxErrorSending	Specifies whether selogrd will send error messages to syslog regarding difficulties sending audit records to selogrcd, only after the number of difficulties surpasses this token value. The default value is 1, which means that every time selogrd has difficulties sending to selogrcd, it sends a message to syslog.	1
MaxSeqNoSleep	Specifies the maximum number of log records scanned by selogrd without sleeping.	50
RouteFile	Specifies the name of the log routing configuration file. The file is used unless overridden by the selogrd utility's -config option.	/opt/CA/ eTrustAccessControl/log/ selogrd.cfg
SavePeriod	Specifies the time interval, in minutes, between saving information about the number of records sent.	2
sendmail_header_format	Determines the user name format in the header of mail that selogrd sends. **Note:** Change this token value only if selogrd cannot send mail. (That is, if you see an error 4634 from selogrd in your syslog.) Valid values include the following: **1**—The user name format is *SmtpMailFrom* For example: eTrust_Admin **2**—The user name format is *SmtpMailFrom@hostnam*e (where *hostname* is the host which selogrd runs on). For example: eTrust_Admin@machine	1

Token	Description	Default Value
ServicePort	Specifies the name or port number that the log routing facility must use. If the token has a value, selogrd and selogrcd use the specified port; otherwise, selogrd and selogrcd dynamically allocate a UDP port using the RPC portmapper. The service name must be a UDP port because the log routing daemon uses UDP for communication. If the token value is a number, daemons bind to the specified port number or service name. If the token value is a service name (string), /etc/services or NIS services maps are used to resolve the port number.	
SmtpMailFrom	Specifies the identity of the sender for UseSmtpMail.	eTrust Admin
SmtpMailServer	Specifies the address of the remote mail server host. Use this if UseSmtpMail is set to yes. If you do not specify this token, the local machine is assumed to be the mail server.	(blank - local server)
SmtpTimeLimit	Specifies the time limit, in seconds, that selogrd waits for the mail server to answer before timing out.	100
tec_conf_file	Specifies the name of the configuration file that is used for the TEC event creation by the selogrd daemon.	/etc/tecad_seos.conf
UseEncryption	Determines the type of encryption. Valid values include the following: **native**—selogrd uses eTrust AC standard encryption. **eTrust**—selogrd uses audit log encryption through adcipher. **no**—selogrd does not use encryption.	no
UseSmtpMail	Determines whether to use the direct mail feature or the previous Mailer.	no

For more information, see the following documentation:

- The seaudit, selogrcd, and selogrd utilities in the *Utilities Guide*
- The seauditx application in the *User Guide*

seos

In the [seos] section, the tokens determine the global settings used by eTrust AC.

Token	Description	Default Value
admin_data	Specifies the directory where the eTrust AC Security Administrator rulers and other configuration files are stored.	/opt/CA/ eTrustAccessControl/data
backup_pmd	Determines the name of the PMDB used as the backup server for the host, in case its parent pmd is down. The format is *pmd_name@hostname*.	no default
fast_create_db	Specifies whether the PMDB uses the fast database copy device. Valid values include the following: **no**—Use the old device. **yes**—Use the fast database copy device.	yes
full_year	Specifies the format for displaying the year using four digits or last two digits. For example, setting the token to yes displays 2000 instead of 00. Valid values include the following: **yes**—four digits **no**—two digits This token influences the output produced by secons -tv, dbmgr -d, and the seaudit utility.	yes (four-digit)

Token	Description	Default Value
locale	Determines the language for the eTrust AC daemons and utilities. eTrust AC can function in several languages. Supported languages include: C, Japanese, Chinese-s, Chinese-t For the complete list of languages, see /etc/ca/localeX/calocmap.txt.	No default
parent_pmd	Specifies the PMDB to which this machine subscribes. The local eTrust database accepts updates from this policy model and rejects updates from any other policy model. If you do not set the token, the station does not accept updates from any PMDB. If you set this token to _NO_MASTER_, then any PMDB can update this station. The format is *pmd_name@hostname*.	No default
passwd_pmd	Specifies the PMDB to which sepass sends password updates. If you do not set this token, it inherits the value of the parent_pmd token. The format is *pmd_name@hostname.* The parent_pmd and passwd_pmd tokens can have the same value. If the values in the parent_pmd and passwd_pmd tokens are not the same, the passwd_pmd database sends its updates to the parent_pmd database for distribution. Therefore, the parent_pmd database must be a child (subscriber) of the passwd_pmd database.	No default
ReverseIpLookup	Controls the way seagent identifies the connecting client. Valid values include the following: **yes**—seagent looks up the IP address of the open client's socket. **no**—seagent uses the host name as received from the client; seagent does not resolve any host names. (The same effect can be achieved by disabling class TERMINAL.)	yes

Token	Description	Default Value
secondary_pmdb	Specifies the PMDB used as the secondary target for password replacement for users who are not defined in the primary target (passwd_pmd). The format is *pmd_name@hostname*.	No default
SEOSPATH	Specifies the directory in which eTrust AC is installed. You can install eTrust AC in any directory, provided that it is not located on an NFS-mounted file system.	/opt/CA/ eTrustAccessControl
SyncUnixFilePerms	Specifies whether eTrust AC should synchronize its ACL permissions with the ACL and other permissions of the native UNIX system, if they exist. Valid values include the following: **no**—Do not synchronize the UNIX file permissions with eTrust AC ACLs. **warn**—Do not synchronize ACL permissions, but issue a warning if the permissions in eTrust AC and UNIX conflict. **traditional**—Change rwx permissions for the group and the owner according to eTrust AC ACLs, issue a warning in all other cases. **acl**—Change native file-system ACLs according to eTrust AC ACLs (on platforms that support ACLs). **force**—Functions the same as traditional or acl (on platforms that support ACLs), but also forces mapping defaccess to "other" permissions. **Note:** On HP-UX and Sun Solaris 2.5 (and above), support is provided for file system ACLs. On other platforms and operating system versions, only traditional permissions mode of a file are supported.	no
TNG_Environment	Specifies whether the database is created with special Unicenter TNG classes and resources. Valid values include the following: **0**—Create the database without the special Unicenter TNG classes. **1**—Create the database with all the special Unicenter TNG classes.	0

Token	Description	Default Value
TNGDir	Specifies the directory where Unicenter TNG is installed. Valid values are the base Unicenter TNG directory (or .uniprodloc).	No default
use_rpc_protocol	Determines whether the RPC portmapper is required. The presence of the RPC portmapper is required if you want to use the old (1.43) eTrust AC protocol. The old protocol is required to support NIS+ password changes. This token replaces the old_protocol token. Valid values include the following: **yes**—Use the RPC portmapper to assign the port. **no**—Use the port specified by the ServicePort token.	no

For more information about how the password tokens are used, see the sepass utility in the *Utilities Guide*.

seosd

In the [seosd] section, the tokens determine the behavior of the authorization daemon and the cache utility for performance improvement.

Token	Description	Default Value
bypass_suid_program	Allows multiple su commands. On some platforms, such as Silicon Graphics IRIX, the system's su binary works in a nonstandard way: When an su command to a non-root user is requested, it executes su to root prior to executing su to the requested user. If eTrust AC surrogate protection is set for the root user, it may prevent the successful execution of an su to non-root users as well. To use the surrogate protection for the root user on such platforms and still to be able to su to non-root users without interruption, set the bypass_suid_program token to contain the real path for the system's su binary.	none

Token	Description	Default Value
bypass_TCPIP	Allows you to add one or more ports separated by commas for which seos_syscall will not pass events to seosd. The syntax is bypass_TCPIP=*port1*[,*port2*,*portx*]	none
cron_program	Improves the check for cron login in seosd. Set the cron_program token to contain the real path for the system's cron binary.	none
dbdir	Specifies the location of the eTrust AC database.	/opt/CA/ eTrustAccessControl/ seosdb
device_file	Specifies whether to scan all devices in /dev. When the value of this token is set to Yes and the tty is not found in the standard list, eTrust AC scans all the devices located in /dev. (qplib resolves the tty name from the standard devices.) **Note:** You can add devices to the list of the tty names.	No

Token	Description	Default Value

domain_names	Specifies a list of domain names that seosd appends to short host names it receives for authorization purposes in order to create a fully qualified name, so that these names can be authorized in the relevant HOST, CONNECT, or TERMINAL classes.
	
	To identify a full name, seosd tries to append domain names in the domain_names list to the short name for authorization purposes.
	
	seosd first looks for a relevant rule in its database, using the short name only. If it does not find a record that matches the short name, it appends each domain name specified in the domain_names token, one by one, until it finds a match.
	
	For example, suppose you assign domain_names the following list:
	
	domain_names= market.com, journey.com, total.com
	
	Here is how seosd handles the matching process when a request from a subscriber called *acme*—which was not defined as a rule in the database—comes in:
	
	`acme (not found in database)`
	`acme.market.com (not found)`
	`acme.journey.com (not found)`
	`acme.total.com (found)`
	
	seosd uses the first record that matches (acme.total.com in this example) for authorization purposes.
		As defined in /etc/resolv.conf

dns_server	Specifies the DNS server name used to change host resolving from the default server to an other server. This token is usually used when the DNS caching option is enabled.	none
enf_register	Determines whether seosd registers to enf. The valid values include the following: **yes**—seosd registers to the enf. **no**—seosd does not register to the enf.	no

Token	Description	Default Value
FileCache_files	If caching is enabled, specifies the number of records in the file pool. The maximum number of file records that can be cached is 200.	20
FileCache_users	If caching is enabled, specifies the number of records in the user pool. The maximum number of user records that can be cached is 500.	50
FileCache_auths	If caching is enabled, specifies the number of records in the authorization pool. The maximum number of authorization records that can be cached is 800.	80
FileCache_CleanInt	Specifies how often to erase the file cache (in minutes).	60
FileCache_PriorInt	If caching is enabled, specifies the frequency of recalculating priorities in the cache table. Each time a new record is saved counts as one.	1
FileCache_InitPrio	Specifies the initial priority value of new records in the cache table.	10
get_login_terminal	Determines whether seosd attempts to find the peer address of the login program in an alternative way. This is useful for connections such as ssh. Valid values include yes and no.	yes
grace_admin	Determines the number of the grace logins that are set when an administrator changes users' passwords.	1

Token	Description	Default Value
GroupidResolution	Determines how eTrust AC resolves GID numbers to group names. Valid values include the following: **system**—eTrust AC uses a system call to translate gid numbers. This value can be used for stand-alone, DNS client, and DNS server stations. (See also the resolve_timeout token in this table.) **cache**—gid numbers and group names are cached in seosd. This is the fastest and easiest way to do translations but the cache cannot be updated during runtime. **ladb**—eTrust AC uses a lookaside database to translate gid numbers. The sebuildla utility must be run to recreate the lookaside database each time an update to the relevant transaction table takes place. For NIS, and NIS+ servers, you can use either cache or ladb. For Sun Solaris 2.5 and above and HP-UX 11.x, you can use either cache or ladb. For all stations, the value ladb is preferred.	system
HostResolution	Determines how eTrust AC resolves IP addresses to host names. Valid values include the following: **system**—eTrust AC uses a system call to translate IP addresses. This value can be used for stand-alone, NIS/NIS+ client, and DNS client stations. (See also the resolve_timeout token in this table.) **cache**—Host names and their IP addresses are cached in seosd. This is the fastest and easiest way to do translations but the cache cannot be updated during runtime. **ladb**—eTrust AC uses a lookaside database to translate IP addresses. The sebuildla utility must be run to recreate the lookaside database each time an update to the relevant transaction table takes place. For NIS, NIS+, and DNS servers, you can use either cache or ladb; the value ladb is preferred.	system

Token	Description	Default Value
IsolatedDaemon	Determines whether seosd closes the file descriptors stdin, stdout, and stderr when they become a daemon. Valid values include the following: **yes**—seosd closes these file descriptors when they become a daemon. **no**—seosd does not close these file descriptors when they become a daemon.	no
kill_ignore	Specifies whether seosd ignores (denies) the "kill -9" command directed toward any one of the three main eTrust AC daemons. Valid values include the following: **yes**—Ignores the kill command. This is the default value. **no**—The kill command terminates seosd.	yes
lookaside_path	Specifies the directory where the lookaside database is located. Create this directory before running the sebuildla utility. **Note:** The lookaside database files are built and updated using the sebuildla utility.	/opt/CA/ eTrustAccessControl/ladb
network_cache_timeout	Specifies the time interval, in minutes, between network cache table cleanings, if network cache is used.	10

Token	Description	Default Value
nfs_devices	Specifies the name and path of the file that contains the NFS major device numbers. Specify the full file path. The file contains the NFS defaults for major device numbers for every platform. This may vary from system to system. To find the numbers for your system, use a small program with the UNIX getmajor() function. Then edit the nfsdevs.init file (or the file you named with this token) to contain the numbers you find. **Note:** Whenever you mount and remount the NFS system, you should update your nfsdevs.init file. You can also use the first four digits of the device only. These numbers remain unchanged, even when you unmount and remount the system.	/opt/CA/ eTrustAccessControl/etc/ nfsdevs.init
protect_bin	Specifies whether seosd protects the eTrust AC binary files. Specify one of the following values: **yes**—seosd protects the eTrust AC binary files unless rules that allow such access are defined. **Note:** Do not specify yes when the _default access for your FILE records is none because then, unless all /opt/CA/eTrustAccessControl/bin files have FILE records, inaccessibility of files could make eTrust AC unusable. **no**—seosd does not protect the eTrust AC binary files.	no
resolve_rebind	Specifies if seosd re-establishes the connection to the NIS server after a time-out failure. We strongly recommend that you do not change the default value.	IBM AIX 4.xx: no NCR: no Sun Solaris 4.1.4: no Other: yes

Token	Description	Default Value
resolve_timeout	Specifies the maximum number of seconds seosd tries to resolve IP to address, user ID to user name, group ID to group name, or service port number to service name. The value takes effect in two cases: ■ When seosd is using system resolution. (See the HostResolution, ServiceResolution, UseridResolution, and GroupidResolution tokens.) ■ When the under_NIS_server token is set to no. If the specified time expires without a resolution, seosd assumes that no resolution exists for the specified IP, ID, or port. If value is set to 0, there is no time out.	5
rt_priority	Determines whether seosd has real-time priority.	yes
ServiceResolution	Determines how eTrust AC translates TCP port numbers to service names. Valid values include the following: **system**—eTrust AC uses a system call to translate TCP port numbers. This value can be used for stand-alone, NIS/NIS+ client, DNS client, and DNS server stations. (See also the resolve_timeout token in this table.) **cache**—Service names and their TCP port numbers are cached in seosd. This is the fastest and easiest way to do translations but the cache cannot be updated during runtime. **ladb**—eTrust AC uses a lookaside database to translate TCP port numbers. The sebuildla utility must be run to recreate the lookaside database each time an update to the relevant transaction table takes place. For NIS, and NIS+ servers, use either cache or ladb.	ladb
trace_file	Specifies the name of the file to which the trace messages are sent, if trace messages are requested.	/opt/CA/eTrustAccessControl/log/seosd.trace

Token	Description	Default Value
trace_file_type	Determines whether the trace file is written in binary or text format. Valid values include the following: **binary**—The trace file should be written in binary format. This option reduces the space occupied by this file. **text**—The trace file should be written in text format. The daemon seosd checks the value of this token and compares it to the contents of the trace file. If the token value does not match the format of the trace file, seosd saves the trace file under its name and adds the extension .backup.	text
trace_filter	Specifies the name and path of the file that contains the filter data that is used to filter the trace messages.	/opt/CA/ eTrustAccessControl/etc/ trcfilter.init
trace_space_saver	Specifies the amount of free space, in KB, to be left in the file system. When the amount of free space is less than this number, eTrust AC disables the trace. **Note:** Trace is never automatically enabled, even if more space becomes available at a later time.	5120
trace_to	Specifies the destination of trace messages. Valid values include the following: **file**—eTrust AC sends the trace messages to the file specified by the trace_file token. To disable tracing, use the *secons -t-* command. For more information, see the trace_file token in this table. **file,stop**—eTrust AC generates trace messages during daemon initialization. Once the daemon is initialized, trace messages generation stops. **none**—eTrust AC does not issue trace messages. This is the normal setting after you install and implement eTrust AC. **Note:** If the token is set to **file** or **file,stop**, the eTrust AC trace can be toggled with the secons command with the -t option.	file, stop

Token	Description	Default Value
Undef_ForPacl	Determines whether seosd checks an undefined user when there is an asterisk (*) in the accessor's name in a PACL. Valid values include the following: **1**—seosd will not include undefined users with an asterisk in their PACL. **0**—seosd will include undefined users with an asterisk in their PACL.	0
under_NIS_server	Determines whether seosd uses internal name resolution instead of system name resolution. Valid values include the following: **yes**—seosd stores in memory or in a lookaside database (see the use_lookaside token) all user, group, host, and port information during startup. This is required for NIS, NIS+, and DNS server machines, and for the following operating systems: Sun Solaris 2.5 and above, HP-UX 11.x, IBM AIX 4.3.x, and IRIX 6.5. **Note:** Turning this token off could hang the machine if it is an NIS server or one of the previously-mentioned operating systems. **no**—seosd uses system name resolution and the resolve_timeout token takes effect. **Note:** This token is automatically assigned a value during installation. This token remains for purposes of backward compatibility only. If you have a new eTrust AC installation or an installation of version 2 or higher, use the tokens HostResolution, ServiceResolution, UseridResolution, and GroupidResolution instead.	No default

Token	Description	Default Value
use_lookaside	Determines whether seosd stores the user, group, host, and port information in a lookaside database or in memory. **Note:** This token is used in conjunction with the under_NIS_server token and has no relevance unless the under_NIS_server token is set to yes. Valid values include the following: **yes**—seosd uses the lookaside database for user, group, host, and service details. The lookaside database is built by the sebuildla utility and can be refreshed by it at any time. For more information, see the sebuildla utility in the *Utilities Guide*. The location of the lookaside database is set by the lookaside_path token. **no**—seosd caches all user, group, host, and service information during startup so that all translations can be done in memory. We recommend that seosd be restarted daily to refresh the cache. This token remains for purposes of backward compatibility only. If you have a new eTrust AC installation or an installation of version 2 or higher, use the tokens HostResolution, ServiceResolution, UseridResolution, and GroupidResolution instead.	no
use_nfs_devices	Determines whether to use NFS devices. Valid values are yes or no.	yes
use_seauxd	Specifies whether to use the auxiliary daemon, seauxd, for name resolution. **Note:** Not all platforms support seauxd. If your platform does not support it, eTrust AC ignores this token.	no

Token	Description	Default Value
use_trusted_script	Specifies whether seosd will use the trusted script mechanism. When the trusted script mechanism is used, programs called from within a shell script retain the name of the shell script in the internal eTrust AC tables. This means that if a script was used in a PACL, these programs will inherit that privilege. This also means that you cannot protect these programs via eTrust AC. A trusted script begins with #! on the first line. When the trusted script mechanism is **not** used, these programs will be registered in the internal eTrust AC tables under their own names.	yes
UseFileCache	Specifies whether to use the cache tool for file records to improve performance.	No
UseridResolution	Specifies how eTrust AC translates UID numbers to user names. Valid values include the following: **system**—eTrust AC uses a system call to translate uid numbers. This value can be used for stand-alone, NIS/NIS+ client, DNS client, and DNS server stations. **cache**—User names and their uid numbers are cached in seosd. This is the fastest and easiest way to do translations but the cache cannot be updated during runtime. **ladb**—eTrust AC uses a lookaside database to translate uid numbers. The sebuildla utility must be run to recreate the lookaside database each time an update to the relevant transaction table takes place. For NIS and NIS+ servers, Sun Solaris 2.5 and above, or HP-UX 11.x operating systems, you must use either cache or ladb.	system
UseNetworkCache	Determines whether eTrust AC uses cache for accept requests. Valid values are yes and no.	no

Token	Description	Default Value
watchdog_refresh	Determines whether seosd refreshes the Watchdog to scan the privileged programs and secured files for each file handle. Valid values include the following: **yes**—seosd refreshes the Watchdog. **no**—seosd does not refresh the Watchdog.	no

seosdb

In the [seosdb] section, the tokens manage database checking and rebuilding.

Token	Description	Default Value
CheckAlways	Determines whether the database should be checked for corruption at eTrust AC initialization. Valid values are yes and no	yes
CheckProgram	Specifies the full path and parameters of an alternative command to be used instead of the internal code for checking the database. The command should return 0 if the database is valid or a nonzero number if it should be corrected.	Do not run any program; use internal code, which is similar to the command: `dbmgr -u -fast`
CreateNewClasses	Specifies whether you can add new classes, created with the seclassadm utility, to a database. Valid values are yes and no	no
RebuildProgram	Specifies the full path and parameters of an alternative command to be used instead of the internal code for correcting the database.	Do not run any program; use internal code, which is similar to the command: `dbmgr -u -build all`

seoswd

In the [seoswd] section, the tokens determine the behavior of the Watchdog.

Token	Description	Default Value
IgnoreScanInterval	Specifies whether to scan programs and files at specific intervals. If the token value is no, the watchdog performs interval scanning; if yes, it does not scan at intervals. **Note:** If you do not specify scan times with the PgmTestTime or SecFileTestTime tokens, and this token is set to yes, the watchdog does not scan trusted programs or secured files, respectively.	no
PgmRes	Specifies the rest period, in seconds, between checking programs. The program rests to prevent system overload.	10
PgmTestInterval	Specifies the time interval, in seconds, between rescanning of trusted programs.	18000 (5 hours)
PgmTestStartTime	Specifies the start time, in *hh:mm* format, of the first trusted program scan. If you do not set this token, the Watchdog performs the first scan shortly after startup.	No default
PgmTestTime	Specifies fixed scan times, in *hh:mm* format, for trusted programs. You can specify more than one scan time by separating them with spaces. **Note:** If you do not specify scan times, and you set the IgnoreScanInterval token to yes, the Watchdog does not scan trusted programs.	No default
RefreshParams	Specifies the time interval, in seconds, between successive reads by the Watchdog of the seos.ini tokens.	86400 (1 day)

Token	Description	Default Value
SecFileRest	Specifies the rest period, in seconds, between checking secured files. The Watchdog rests in order to prevent system overload. **Note:** If you do not specify scan times, and you set the IgnoreScanInterval token to yes, seoswd does not scan secured files.	10
SecFileTestInterval	Specifies the time interval, in seconds, between rescanning of secured files.	36000 (10 hours)
SecFileTestStartTime	Specifies the start time, in *hh:mm* format, of the first scan of secured files. If no value is given, the Watchdog performs the first scan a short time after eTrust AC daemons start.	No default
SecFileTestTime	Specifies fixed scan times, in *hh:mm* format, for secured files. You can specify more than one scan time by separating them with spaces.	No default
SeosAYT	Specifies the time interval, in seconds, between Watchdog checks of the daemon seosd. ***Warning!*** *Do not modify this token except as directed by Computer Associates technical support. An incorrect value can cause major problems in eTrust AC operation.*	60*
SignalMinInterval	Specifies the interval, in seconds, between scans after a HUP signal triggers a one-time scan on demand, to protect the system against overload. **Note:** Scan on demand is performed both on trusted programs and secured files.	60

Token	Description	Default Value
UnTrustMissing	Determines whether the Watchdog should attempt to untrust a program or file, even though it cannot find it (for example, if the file was deleted or the relevant NFS partition is not mounted). Valid values include the following: **yes**—Attempt to untrust the missing file. **no**—Do not attempt to untrust the missing file.	yes

seos_syscall

In the [seos_syscall] section, the tokens are used by the SEOS_syscall kernel module.

Token	Description	Default Value
bypass_NFS	Determines whether to bypass NFS files from SEOS events. Valid values include the following: **0**—Do not bypass NFS files. **1**—Bypass NFS files.	0
bypass_realpath	Determines whether to check file access when a generic rule in the eTrust database includes the file, default access to the generic rule is none, and the path given to the file is a relative path. Valid values include the following: **0**—Check file access. **1**—Do not check file access.	0
cache_enabled	Determines whether to use caching for full path resolution to determine access permissions for files. Valid values include the following: **0**—No caching. **1**—Use caching.	0

Token	Description	Default Value
cache_rate	Determines the cache rate that used when cache is enabled for full path resolution. Bigger values mean better caching.	1000
debug_protect	Determines whether to allow debugging of any program while eTrust AC is running. Valid values include the following: **0**—Debugging allowed. **1**—Debugging not allowed.	1
DESCENDANT_dependent	Determines whether a descendent of a SEOS daemon can register a SEOS service. Valid values include the following: **0**—Anyone can register a SEOS service. **1**—Only a descendent can register a SEOS service.	0
file_bypass	Indicates whether eTrust AC checks file access for files that are not defined in the database. By default eTrust AC does not check files that are not defined in the eTrust database. Valid values include the following: **-1**—Do not check all files. **0**—Check all files.	-1
GAC_root	Determines whether to use GAC caching for files when the user is root. By default GAC is not used when the user is root. Valid values include the following: **0**—No caching for root user. **1**—Use caching for root.	0
HPUX11_SeOS_Syscall_number	Determines the default syscall number to communicate with SEOS_syscall on HP-UX. Valid vlaues include any unused syscall entry number in sysent.	254

Token	Description	Default Value
kill_signal_mask	Determines which signals to protect. Valid values include a mask that ORs (includes) all the signals that we want SEOS events for.	SIGKILL | SIGSTOP | SIGTERM 0x804100 for HP-UX 0x404100 for IRIX, NCR, Solaris, and UnixWare 0x14100 for AIX and DEC Unix 0x44100 for Linux
link_protect	Determines whether a symbolic link will be protected. Valid values include the following: **0**—Links are not protected. **1**—Links are protected.	0
max_generic_file_rules	Defines the maximum number of generic file rules allowed in the database. **Note:** Using a very large number may cause strange behavior on different platforms. Valid values include any number greater than (<) 511. **Note:** This token is supported only on AIX, HP, Linux, and Solaris.	512
max_regular_file_rules	Defines the maximum number of file rules allowed in the database. **Note:** Using a very large number may cause strange behavior on different platforms. Valid values include any number greater than (<) 4095. **Note:** This token is supported only on AIX, HP, Linux, and Solaris.	4096
mount_protect	Determines whether to allow mount and umount of directories used by eTrust AC. Valid values include the following: **0**—Allow mounting. **1**—Do not allow mounting.	1

Token	Description	Default Value
proc_bypass	Determines whether to check file access when a file belongs to a process file system (/proc). Valid values include the following: **0**—token is ignored OR any sum of accesses: **1**—read **2**—write **4**—chown **8**—chmod **16**—rename **32**—unlink **64**—utimes **128**—chattr **256**—link **512**—chdir **1024**—create	0
s68_reboot_string	Controls the reboot command for S68SEOS to use after modifying the name_to_sysnum file.	/usr/sbin/reboot
s68_reboot_sys	Determines whether S68SEOS reboots the system after it modifies the name_to_sysnum file.	no
SEOS_unload_enabled	Determines whether the SEOS_syscall kernel module can be unloaded. Valid values include the following: **0**—Do not allow the unload. **1**—Allow the unload.	1
SEOS_use_streams	Determines whether to use the streams susbsystem for network interception (determines whether SEOS_load will autopush a module into streams). This token can be used for following platforms: HP-UX, Solaris, UnixWare, NCR, SINIX.	no

Token	Description	Default Value
STOP_enabled	Determines whether to use the STOP feature, which protects from stack overflow attacks. Valid values include the following: **0**—Off. **1**—On.	0
synchronize_fork	Determines how fork synchronization is managed. On **HP-UX platforms**, valid values include the following: **0**—Do not report forks to main daemon (to avoid an answer delay) **1**—Report forks from parent **2**—Report forks from child On **other platforms**, valid values include the following: **0**—Do not report forks to main daemon (to avoid an answer delay) **1**—Parent reports without synchronization **2**—Parent reports with synchronization	1
trace_enabled	Determines whether to use the SEOS_syscall circular trace buffer. Valid values include the following: **0**—Do not use tracing. **1**—Use tracing.	0

serevu

In the [serevu] section, the tokens determine the attributes of the serevu utility.

Token	Description	Default Value
admin_user	No longer used. The SPECIALPGM class provides the same function. For details, see the *Reference Guide*.	N/A

Token	Description	Default Value
config_file	Specifies the location of the serevu configuration file.	/opt/CA/ eTrustAccessControl/etc/ serevu.cfg
def_diff_time	Specifies the time interval during which serevu scans the relevant system log for failed logins. The value may be specified in seconds (that is, 300) or minutes (that is, 5m). For example, if the token is set to 300, serevu searches for failed logins that occurred during the previous 300 seconds. We recommend that this value be an even multiple of the value in the def_sleep_time token.	5m (5 minutes)
def_disable_time	Specifies the time that a user account is disabled because of too many failed login attempts. The value may be specified in seconds (that is, 300) or minutes (that is, 5m).	6m (6 minutes)
def_fail_count	Specifies the number of failed logins each user is entitled to, per period, in the token def_diff_time. Users with at least this number of failed logins over the specified time period are disabled. **Note:** We recommend that the number of failed logins always be the same as the value of allowed unsuccessful login attempts set on your system. (The system values for this are set in /etc/default/login. On Solaris systems, for instance, use the RETRIES token. On NCR systems, use the MAXTRYS token.) Default values are five for Solaris, three for HP-UX and AIX, and three **or** five for NCR. See your operating system documentation for more details.	5

Token	Description	Default Value
def_sleep_time	Specifies the time between successive serevu checks. The value may be specified in seconds (that is, 120) or minutes (that is, 2m).	2m (2 minutes)
save_disable_path	Specifies the location of the disabled user accounts list when serevu goes down.	/opt/CA/eTrustAccessControl/log/serevu_disable.users

For more information, see the serevu utility in the *Utilities Guide*.

sesu

In the [sesu] section, the tokens control logging on as a user other than yourself, without having to enter the password of the other user.

Token	Description	Default Value
AlwaysTargetShell	Determines whether to use the target shell (SysV style) or the invoker shell (BSD style). If yes, eTrust AC uses the target user shell. Valid values are yes and no.	no
FilterEnv	Specifies a list of environment variables that sesu does not pass to the shell when the target user is root. Separate variable names with spaces or tabs.	No default

Token	Description	Default Value
old_sesu	Determines whether the old or new sesu utility is used. Valid values include the following: **yes**—Use the old sesu utility as it was in previous versions. **no**—The new sesu utility calls the native su program (as defined in the SystemSu token) to ensure consistency between su and sesu. If the SystemSu token is not valid, sesu reverts to the old mechanism. **Note:** If this token is set to no, the tokens Path, AlwaysTargetShell, sys_env_file, and FilterEnv are ignored.	yes
Path	Specifies the value that sesu uses to set the PATH environment variable. If the token is not set, sesu does not set the PATH variable.	No default
sys_env_file	Specifies an ASCII file containing environment variable values for the sesu session. This token is relevant only when starting sesu with the "-" parameter (sesu -). The format for each line of the file is *variable* = *value*.	For IBM AIX: /etc/environment
SystemSu	Specifies the location of the /bin/su program. Update this token if you use a program in a location other than the default location. When sesu cannot find the authorization daemon, it executes the program specified in this token.	/bin/su
UseInvokerPassword	Determines whether sesu requires the invokers to specify their own passwords. If the token value is no, sesu does not require any password.	no

For more information, see the sesu utility in the *Utilities Guide*

sesudo

In the [sesudo] section, the tokens determines the attributes of the sesudo utility.

Token	Description	Default Value
echo_command	Determines whether sesudo displays the command before executing it. To echo the command, set the token value to yes.	no

For more information, see the sesudo utility in the *Utilities Guide*.

tng

In the [tng] section, the tokens control the integration of eTrust AC into the Unicenter TNG environment.

Token	Description	Default Value
defsesid	Specifies the default session group ID for users that do not have a specific session group ID defined. Session groups are used by eTrust™ Single Sign-On (eTrust SSO).	CAUNICENTER
ssf_numsubp	Specifies the number of subprocesses required for the sessfgate daemon to start processing incoming SSF requests.	1
sso_applname	For sites using the CA-Ticket functionality of eTrust Single Sign-On (SSO), specifies an eight-character string that **must** correspond to the keymgmt files found under the seos home directory in the folder data/keymgmt. The names of these files are under the SSO_APPLNAME_key. For example, if the default value of UNICENTR is taken, the name of the file becomes UNICENTR_key.	UNICENTR

Appendix D

The pmd.ini File

This appendix describes the tokens of the pmd.ini file.

The pmd.ini file contains various setup and initialization tokens used by eTrust AC when building and maintaining a PMDB. It consists of several sections and each section contains multiple tokens:

Section Name	Description
lang	Contains the parameters used by the language programs selang and seadm.
logmgr	Contains the parameters used by the PMDB logging facility.
maker-checker	Contains the parameters used by the maker-checker function.
passwd	Contains UID and GID default values.
pmd	Contains default parameters for a PMDB.
seos	Contains global eTrust AC tokens.

lang

The [lang] section contains the parameters used by the eTrust AC language programs—selang and seadm—when building and maintaining a PMDB.

Token	Description	Default Value
pre_user_exit	Specifies the path of the exit program to be executed before eTrust AC issues a language command to update the UNIX user database.	
post_user_exit	Specifies the path of the exit program to be executed after eTrust AC issues a language command to update the UNIX user database.	

Token	Description	Default Value
pre_group_exit	Specifies the path of the exit program to be executed before eTrust AC issues a language command to update the UNIX groups database.	
post_group_exit	Specifies the path of the exit program to be executed after eTrust AC issues a language command to update the UNIX groups database.	

logmgr

The [logmgr] section contains the parameters used by the PMDB logging facility.

Token	Description	Default Value
audit_back	Specifies the name of the PMDB audit backup file.	pmd_audit.bak
audit_log	Specifies the name of the PMDB audit log file.	pmd_audit
audit_group	Specifies the group that can read the PMDB audit files. If no group is specified, only root can read the audit files. eTrust AC does not verify the value of this token, so if you enter an invalid group name, eTrust AC does not assign any group permissions to the audit log files. To change the group ownership of an existing audit log file, do the following: 1. Use the selang command chgrp to set the group ownership of the files. 2. Change the UNIX permissions by entering: `chmod 640 /opt/CA/eTrustAccessControl/log/seos.audit`	none
audit_size	Specifies the size of the PMDB audit log file, in KB. Do not specify a size less than 50 KB.	50 KB

Token	Description	Default Value
error_back	Specifies the name of the PMDB error backup file.	pmd_error.bak
error_log	Specifies the name of the PMDB error log file.	ERROR_LOG
error_group	Specifies the group that can read the PMDB error files. If no group is specified, only root can read the error files. eTrust AC does not verify the value of this token, so if you enter an invalid group name, eTrust AC does not assign any group permissions to the error log files. To change the group ownership of an existing error log file, do the following: 1. Use the selang command chgrp to set the group ownership of the files. 2. Change the UNIX permissions by entering: `chmod 640 /opt/CA/eTrustAccessControl/log/seos.error`	none
error_size	Specifies the size of the PMDB error log file in KB. Do not specify a size less than 50 KB.	50 KB
max_log_size	Specifies the size of the PMDB general log file in KB.	50 KB
pmd_log_level	Determines the messages that are logged in the PMDB log file. Valid values include the following: **0**—Do not log any entries. **1**—List only error messages. **2**—List error and informational messages.	2

maker-checker

The [maker-checker] section contains parameters used by the maker-checker function.

Token	Description	Default value
transaction_lib	Specifies the path of the maker-checker policy.	none

passwd

The [passwd] section contains parameters for UIDs and GIDs.

Token	Description	Default value
AllowedGidRange	Specifies reserved numbers. The integers below the first number and above the second number are reserved GIDs, which eTrust AC cannot update. If only one integer is specified, all integers between one and the specified integer are reserved GIDs.	100,30000
AllowedUidRange	Specifies reserved numbers. The integers below the first number and above the second number are reserved UIDs, which eTrust AC cannot update. If only one integer is specified, all integers between one and the specified integer are reserved UIDs.	100,30000

pmd

The [pmd] section contains the attributes used by the sepmdd daemon when building and maintaining a PMDB.

Token	Description	Default Value
filter	Specifies the name of the filter file.	
force_auto_truncate	Specifies whether the sepmd utility truncates the update file with the –t parameter. If the token is set to **no**, *sepmd -t* does not truncate the update file. If the token is set to yes, *sepmd -t* truncates the update file even if there are no subscribers to the Policy Model.	no
group_file_name	Specifies the name of the group file for a new UNIX group. sepmdd saves the group entry of the new UNIX group in this file.	group
is_maker_checker	Specifies whether to use Dual Control. The valid values for this token are yes and no. If **yes** is selected, then the PMDB cannot be updated directly, but only through a transaction; and each transaction entered by one administrator must be processed by another administrator before the commands are implemented on the PMDB.	no
password_file_name	Specifies the name of the password file for new UNIX users. sepmdd stores the password entry of new UNIX users in this file.	passwd
passwd_pmd	The name of the PMDB from which sepass propagates a password.	

Token	Description	Default Value
_QD_timeout_	The maximum time, in seconds, that the daemon sepmdd waits when trying to update a subscriber database during its first scan of its subscriber list. 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. 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). sepmdd only checks this token and performs a two-scan process for updates originating from eTrust AC version 2.0 or higher.	3
send_unix_env	Indicates whether sepmd sends the contents of Policy Model password files and group files. If this token is set to **yes**, the *sepmd -n* option sends the contents of the Policy Model password files and group files. If this token is set to **no**, the *sepmd -n* option does not send the contents of the policy model password files and group files.	yes

Token	Description	Default Value
synch_uid	Determines whether sepmdd attempts to synchronize UIDs between a Policy Model and its subscribers. The valid values for this token are yes and no. If the token is **no**, sepmdd does not attempt to synchronize UIDs. Users are assigned the first available UID on each subscriber host. If the token is **yes**, sepmdd attempts to synchronize UIDs. For example, if a new UNIX user is created on the PMDB with a UID of 1000, sepmdd transfers that UID to the subscribers. If UID 1000 is already in use on one of the subscribers, then the update on that subscriber fails. sepmdd only tries to synchronize UIDs if the original command sent to the PMDB did not specify a UID for the user. If the original command did specify a UID, the specified UID is sent to all the subscribers.	yes
updates_in_chunk	Determines the maximum number of commands that the Policy Model sends to each of its subscribers in each cycle of a loop.	10
UseEncryption	Specifies whether update information saved to the updates.dat file is encrypted.	no
UseShadow	Determines whether to use a shadow file when you reference the PMDB native environment.	no
UseSystemFiles	Used by sepmdd and seagent in conjunction with the YpServerSecure and UseShadow tokens. ***WARNING!*** *Do not modify this token!*	no

Token	Description	Default Value
YpServerSecure	Specifies the name of the password shadow file (a security file on an NIS server) that is used for building the NIS password map. This token is relevant only if you set UseShadow to yes.	/etc/shadow

seos

The token of the [seos] section, which contains the global settings used by eTrust AC, is described in the following table.

Token	Description	Default Value
parent_pmd	Specifies the name of the parent Policy Model that can send updates to this PMDB. A Policy Model does not accept updates from another Policy Model unless this token is filled in.	

Appendix E

Password Synchronization with Mainframes

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

Tip: 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 already installed eTrust AC, you can run the installation script again to choose the Mainframe Password Synchronization option. Reinstalling 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 Computer Associates 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 */lbin. 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 mfscntrl 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 CAIGLBL0000 environment variable to the eTrust AC directory /seos/uni. If you need to define the variable manually, enter the command:

   ```
   setenv CAIGLBL0000 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 msfd. The script is located at *eTrustACDir*/lbin/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:

Character	Function
-	Matches to zero or more characters
?	Matches one character
[ ]	Delimit a range or value list 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:

```
MF->TSS REP*(sub(x)) PAS*(sub(y))*
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
    MF->ALT* (sub(x)) PASSWORD(sub(y))*
    PM->env seos; cu sub(x) password(sub(y))

    MF->ALT* sub(x) PASSWORD(sub(y))*
    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 will 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.

```
eTrust-Lang

eTrust> edituser root admin
(localhost)
Successfully updated USER root
eTrust> auth terminal name.companyname.com uid(root) acc(r w)
(localhost)
Successfully added root to name.companyname.com's ACL
eTrust>
```

   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.

```
eTrust-Lang

eTrust> edituser mfs_adm admin
(localhost)
Successfully created USER mfs_adm
eTrust> auth terminal name.companyname.com uid(mfs_adm) acc(r w)
(localhost)
Successfully added mfs_adm to name.companyname.com's ACL
eTrust> newres special pgm opt/CA/eTrustAccessControl/lbin/mfsd owner(nobody)\
unix(root) seosuid(mfs_adm)
(localhost)
Successfully created SPECIALPGM opt/CA/eTrustAccessControl/lbin/mfsd
eTrust>
```

   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):

```
eTrust-Lang

eTrust> hostpmd@
(pmd@localhost)
Successfully connected
INFO: Target host's version is 5.30 (5.30)
Unix OS info:hostname SunOS 5.8 Feb 2003 12:05:32 IST
eTrust>
```

2. Define user root or mfs_adm as an administrator in the Policy Model.

```
eTrust-Lang

eTrust> edituser mfs_adm admin
(pmd@localhost)
Successfully created USER mfs_adm
eTrust> ▌
```

   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 mfsd daemon is running):

```
eTrust-Lang

eTrust> newres terminal name.companyname.com owner(nobody)
(pmd@localhost)
Successfully created TERMINAL name.companyname.com
eTrust> auth terminal name.companyname.com uid(mfs_adm) acc(r w)
(pmd@localhost)
Successfully added mfs_adm to name.companyname.com's ACL
eTrust> ▌
```

   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):

```
eTrust-Lang

eTrust> newres mfterminal mfSYSID defaccess(none) owner(nobody)
(pmd@localhost)
Successfully created MFTERMINAL mfSYSID
eTrust> ▌
```

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.

3. 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.

```
eTrust-Lang

eTrust> newusr mfAdmin admin
(pmd@localhost)
Successfully created USER mfAdmin
eTrust> ▌
```

4. 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):

```
eTrust-Lang

eTrust> authorize MFTERMINAL mfSYSID uid(mfAdmin) access(read)
(pmd@localhost)
Successfully added mfAdmin to mfSYSID's ACL
eTrust> ▌
```

5. Subscribe the mainframes to the Policy Model (if you did not do this during installation).

```
eTrust-Lang
[131] % sepmd -sm pmdb unixhost.ca.com TSS sys1 user1
eTrust sepmd v5.30 (5.30) - Policy model management


Copyright 2003 Computer Associates International, Inc.


warning : couldn't fully qualify unixhost.ca.com
Subscribed unixhost.ca.com to policy model pmdb
[132] % sepmd -L pmdb
eTrust sepmd v5.30 (5.30) - Policy model management

Copyright 2003 Computer Associates International, Inc.



Initial offset:       0
Last offset:          0

unixhost.ca.com              Errors     Flag      Offset     Next Command
===========                  =======    ======    =======    ============
xxxxx.ca.com                 0                    0
[133] % _
```

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.

```
eTrust-Console

unixhost:bin> mfscntrl start cci
Starting cci daemon
unixhost:bin> █
```

To be sure that communication was established, run the utility:

```
eTrust-Console

unixhost:bin> /opt/CA/eTrustAccessControl/uni/cci/bin/ccii
Oid(unixhost,Cci.Remote.SERVER   ) Did( , ) type(L)
Oid(unixhost,Cci.Remote.SERVER   ) Did( , ) type(L)
Oid(MYMFSYSID,W410_LOGGER       09) Did(MYMFSYSID,RACF/CPF-CMD) type(R)
Oid(MYMFSYSID,W410_LOGGER       09) Did(MYMFSYSID,RACF/CPF-CMD) type(R)
Oid(MYMFSYSID,W410_LOGGER       23) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       23) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       22) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       22) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       21) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       21) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       20) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       20) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       19) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       19) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       18) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       18) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       17) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       17) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       16) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       16) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       15) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       15) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,RACF/CPF-CMD        ) Did(, ) type(R)
Oid(MYMFSYSID,RACF/CPF-CMD        ) Did(, ) type(R)
Oid(MYMFSYSID,W410_LOGGER       14) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       14) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       13) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       12) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       11) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       10) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       09) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       08) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       07) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       06) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       05) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       04) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       03) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER       02) Did(MYMFSYSID,CAS9VTAMCW410000) type(R)
Oid(MYMFSYSID,W410_LOGGER_SERVICE  ) Did(, ) type(R)
Oid(MYMFSYSID,W410_LOGGER_MAIN     ) Did(, ) type(R)
Oid(MYMFSYSID,MVS_START_SERVER     ) Did(, ) type(R)
Oid(MYMFSYSID,W410_SPAWN_SERVER    ) Did(, ) type(R)
unixhost:bin>

unixhost
```

If the eTrust AC directory /uni/cci/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:

```
eTrust-Console
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.

```
eTrust-Console
unixhost:bin> mfscntrl 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 Unicnter 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
     ```

Appendix F

eTrust Audit Integration

This appendix provides an overview of the integration between eTrust AC 5.3 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 server. 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.

About 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 running 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 in 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 selogrd whenever eTrust AC starts up.

   ```
   # eTrustACDir/bin/seini -s daemons.selogrd yes
   ```

3. Create a new file called selogrd.cfg

   ```
   # vi eTrustACDir/log/selogrd.cfg
   ```

4. Enter three lines in this file as follows

   ```
   hostrule
   host name_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.

   a. 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 usilsuv3:

        ```
        141.202.243.10     name     name.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.

Index