

CA Access Control

Endpoint Administration Guide for Windows

12.6



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or 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 YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

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

Copyright © 2011 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Third-Party Notices

CONTAINS IBM(R) 32-bit Runtime Environment for AIX(TM), Java(TM) 2 Technology Edition, Version 1.4 Modules

(c) Copyright IBM Corporation 1999, 2002

All Rights Reserved

Sample Scripts and Sample SDK Code

The Sample Scripts and Sample SDK code included with the CA Access Control product are provided "as is", for informational purposes only. They may need to be adjusted in specific environments and should not be used in production without testing and validating them before deploying them on a production system.

CA Technologies does not provide support for these samples and cannot be responsible for any errors that these scripts may cause.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Access Control Enterprise Edition
- CA Access Control
- CA Single Sign-On (CA SSO)
- CA Top Secret®
- CA ACF2™
- CA Audit
- CA Network and Systems Management (CA NSM, formerly Unicenter NSM and Unicenter TNG)
- CA Software Delivery (formerly Unicenter Software Delivery)
- CA Service Desk (formerly Unicenter Service Desk)
- User Activity Reporting (formerly CA Enterprise Log Manager)
- CA Identity Manager

Documentation Conventions

The CA Access Control documentation uses the following conventions:

Format	Meaning
Mono-spaced font	Code or program output
<i>Italic</i>	Emphasis or a new term
Bold	Text that you must type exactly as shown
A forward slash (/)	Platform independent directory separator used to describe UNIX and Windows paths

The documentation also uses the following special conventions when explaining command syntax and user input (in a mono-spaced font):

Format	Meaning
<i>Italic</i>	Information that you must supply
Between square brackets ([])	Optional operands

Format	Meaning
Between braces ({}).	Set of mandatory operands
Choices separated by pipe ().	Separates alternative operands (choose one). For example, the following means <i>either</i> a user name <i>or</i> a group name: <i>{username groupname}</i>
...	Indicates that the preceding item or group of items can be repeated
<u>Underline</u>	Default values
A backslash at end of line preceded by a space (\)	Sometimes a command does not fit on a single line in this guide. In these cases, a space followed by a backslash (\) at the end of a line indicates that the command continues on the following line. Note: Avoid copying the backslash character and omit the line break. These are not part of the actual command syntax.

Example: Command Notation Conventions

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

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

In this example:

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

File Location Conventions

The CA Access Control documentation uses the following file location conventions:

- *ACInstallDir*—The default CA Access Control installation directory.
 - Windows—C:\Program Files\CA\AccessControl\
 - UNIX—/opt/CA/AccessControl/

- *ACSharedDir*—A default directory used by CA Access Control for UNIX.
 - UNIX—/opt/CA/AccessControlShared
- *ACServerInstallDir*—The default CA Access Control Enterprise Management installation directory.
 - /opt/CA/AccessControlServer
- *DistServerInstallDir*—The default Distribution Server installation directory.
 - /opt/CA/DistributionServer
- *JBoss_HOME*—The default JBoss installation directory.
 - /opt/jboss-4.2.3.GA

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At <http://ca.com/support>, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at <http://ca.com/docs>.

Contents

Chapter 1: Introduction 13

About this Guide	13
Who Should Use this Guide.....	13

Chapter 2: Managing Endpoints 15

What Is CA Access Control?.....	15
What Is Protected?	15
How Is It Protected?.....	18
Expanding Native Security.....	19
Components	24
Database	25
Drivers	25
Services	25
selang	27
Endpoint Management	27

Chapter 3: Managing Users and Groups 29

Users and Groups	29
Where Information about Accessors Is Stored	30
How CA Access Control Finds a User Record	30
Integration with the Enterprise User Stores	31
Guidelines for Managing Accessors in Enterprise Stores	31
Users and Groups that Must be Defined in the Database	31
Restrictions on the Use of Enterprise Users.....	31
Restrictions on the Use of Enterprise Groups.....	32
Enable or Disable the Use of Enterprise Users and Groups	32
Enable or Disable the Creation of XUSER Records at Enterprise User Login.....	33
Enable or Disable Checking Enterprise Store before Creating XUSER Records on UNIX.....	34
Recycled Enterprise Store Accounts on Windows	34
Resolve Recycled Enterprise Accounts on Windows.....	35
Database Accessors.....	36
Predefined Users.....	37
Predefined Groups	38
Profile Groups	39
How CA Access Control Uses Profile Groups to Determine User Properties	39
Accessor Management.....	39

Manage Users or Groups	40
User Management Using selang	43
Group Management Using selang.....	43

Chapter 4: Managing Resources **45**

Resources	45
Resource Groups	45
Classes	46
Default Record for Class	46
User-Defined Classes.....	51
Windows Services Protection.....	53
Enable and Disable Windows Services Protection	54
Protect a Windows Service	54
Non-IPv4 telnet Connections Are Not Secured on Windows Server 2008	55
View Access Attempts to a Protected Windows Service.....	56
Windows Registry Protection.....	57
Protect a Windows Registry Entry	58
Protect File Streams	61
Internal File Protection.....	62
Internal File Rules.....	63
Default File Rules	64

Chapter 5: Managing Authorization **67**

Access Authorities	67
Setting Access Authority - Examples	67
Access Control Lists	68
Conditional Access Control Lists	69
defaccess—The Default Access Field	69
How Access Authority to a Resource Is Determined.....	70
Interaction Between User and Group Access Authorities.....	71
Accumulative Group Rights (ACGRR)	72
Security Levels, Categories, and Labels.....	72
Security Levels.....	72
Security Categories	73
Security Labels	73

Chapter 6: Protecting Accounts **75**

User Impersonation Protection.....	75
User Mode Interception.....	76
Kernel Mode Interception.....	77

How CA Access Control Responds to User Impersonation Requests	78
Enable User Impersonation Protection	79
Setting Up the Surrogate DO Facility.....	80
Defining SUDO Records (Task Delegation)	81
Checking User Inactivity	87

Chapter 7: Managing User Passwords **89**

Managing Password and Lockout Policies.....	89
Configure Password Quality Checking.....	90
Resolving Error Messages	91

Chapter 8: Monitoring and Auditing **93**

Security Auditors	93
Events Interception	94
Types of Intercepted Events.....	94
Interception Modes.....	94
Warning Mode	95
Monitoring Access Control Activity	100
Trace Record Filters.....	100
Filtering Trace Records.....	101
What CA Access Control Audits.....	101
Login Interception Limitations	102
What CA Access Control Audits in Full Enforcement Mode.....	103
What CA Access Control Audits in Audit Only Mode	103
How to Change What CA Access Control Writes to the Audit Log.....	103
Setting Audit Rules	104
Defining the Audit Events That CA Access Control Writes to the Audit Log	105
How CA Access Control Determines the Audit Mode for a User	106
Default Audit Modes for Users and Enterprise Users	108
Setting Audit Policies in Windows.....	109
The Auditing Process	111
How Auditing Works for Interception Events	112
How Auditing Works for Audit Events.....	113
Kernel and Audit Caches	114
Cache Reset.....	114
Viewing Audit Events.....	115
Audit Events in the Windows Event Log	116
Route Audit Events to the Windows Event Log	117
Route Audit Events to the Windows Event Log Channel	118
The Audit Log	119
Using Audit Logs.....	119

Audit Record Filters.....	120
Audit Display Filters	120
Audit Log Backup.....	124

Chapter 9: Scope of Administration Authority **127**

Global Authorization Attributes	127
ADMIN Attribute	127
AUDITOR Attribute.....	128
OPERATOR Attribute	128
PWMANAGER Attribute	128
SERVER Attribute.....	129
IGN_HOL Attribute	129
Group Authorization	129
Parentage	130
Group Authorization Attributes	130
Ownership	132
File Ownership	133
Authorization Examples	134
Single Group Authorization	134
Parent and Child Groups	135
Sub Administration.....	136
How to Grant Specific Administrative Privileges to Regular Users	136
The ADMIN Class.....	136
Environmental Considerations	138
Remote Administration Restrictions	138
UNIX Environment.....	139
Windows Environment.....	139
Default Permissions to Access the Database	140
Native Permissions to Access the Database.....	140

Chapter 10: Managing Policy Models **143**

The Policy Model Database	143
PMDB Location on Disk	144
Managing Local PMDBs.....	144
Managing Remote PMDBs	144
Architecture Dependency	146
Methods for Centrally Managing Policies	148
Automatic Rule-based Policy Updates	148
How Automatic Rule-based Policy Updates Work	148
How You Use a PMDB to Propagate Configuration Settings	149
How You Can Set Up a Hierarchy	150

Update Subscribers	151
Integrate PMDBs with Unicenter	161
Mainframe Password Synchronization.....	161
Mainframe Password Synchronization Prerequisite	162

Chapter 11: General Security Features **163**

Maintenance Mode Protection (Silent Mode)	163
Bypass Drivers	164
Toggle Driver Interception	166
Disable CA Access Control Kernel Interceptions	166
Stack Overflow Protection	167
Enable STOP	167
Configure STOP for Receiving Signature File Updates	168

Chapter 12: Configuring Settings **169**

Configuration Settings.....	169
Change Configuration Settings.....	169
Change Audit Configuration Settings	170

Chapter 1: Introduction

This section contains the following topics:

[About this Guide](#) (see page 13)

[Who Should Use this Guide](#) (see page 13)

About this Guide

This guide describes the concepts used by CA Access Control for Windows—a product that provides a total security solution for open systems. The guide describes Windows endpoint management tasks and concepts.

This guide is also provided with CA Access Control Enterprise Edition, which offers enterprise management and reporting capabilities, and advanced policy management features.

To simplify terminology, we refer to the product as CA Access Control throughout this guide.

Who Should Use this Guide

This guide was written for security and system administrators who are implementing and maintaining a CA Access Control-protected environment.

Chapter 2: Managing Endpoints

CA Access Control 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—CA Access Control can intercept the event in real time and evaluate its validity before passing control to the standard operating system (OS) functions.

This section contains the following topics:

[What Is CA Access Control?](#) (see page 15)

[Components](#) (see page 24)

[Endpoint Management](#) (see page 27)

What Is CA Access Control?

CA Access Control provides you with a powerful tool for managing security for your native platforms, making it possible to implement a security policy that can be customized entirely to an enterprise's security requirements. CA Access Control lets you provide security for users, groups, and resources beyond what is available in native operating systems. It lets you centrally manage security across the organization and integrate your Windows and UNIX security policies in a heterogeneous environment.

What Is Protected?

CA Access Control protects the following entities:

- **Files**

Is a user authorized to access a particular file?

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

- **Terminals**

Is a user authorized to use a particular terminal?

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

- **Signon time**

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

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

- **TCP/IP**

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

The advantage of an open system—a system in which both the computers and the networks are open—is also a disadvantage. Once a computer is connected to the outside world, one can never be sure who enters the system and what damage an alien user may do, whether intentionally or by mistake. CA Access Control 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. CA Access Control 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).

- **Aspects of administrator authority**

CA Access Control provides the means to both delegate superuser authorities to operators and restrict the privilege of the superuser account.

- **Registry keys**

Is a user authorized to access a particular registry key?

CA Access Control restricts a user's ability to access registry keys. You can give a user one or more types of access, such as READ, WRITE, and DELETE. The access can be specified with regard to an individual registry key or to a set of similarly named registry keys.

- **Programs**

Can a particular program be trusted? Is the user authorized to invoke it? Can the user access a specific resource using a program?

The security administrator can test programs 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 CA Access Control self-protection module (also referred to as the **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 CA Access Control does not allow it to run.

In addition, CA Access Control protects against various deliberate and accidental threats, including:

- **Kill attempts**

CA Access Control can be used to protect critical servers and services or daemons against kill attempts.

- **Password Attack**

CA Access Control protects against various types of password attacks, enforces the password-definition policies of your site, and detects break-in attempts.

- **Password Delinquency**

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

- **Account Management**

CA Access Control policies ensure that dormant accounts are dealt with appropriately.

How Is It Protected?

CA Access Control starts immediately after the operating system finishes its initialization. CA Access Control places hooks in system services that must be protected. In this way, control is passed to CA Access Control before the service is performed. CA Access Control decides whether the service should be granted to the user.

For example, a user may attempt to access a resource protected by CA Access Control. This access request generates a system call to the kernel to open the resource. CA Access Control intercepts that system call and decides whether to grant access. If permission is granted, CA Access Control passes control to the regular system service; if CA Access Control denies permission, it returns the standard permission-denied error code to the program that activated the system call, and the system call ends.

The decision is based on access rules and policies that are defined in the database. The database describes two types of objects: accessors and resources. *Accessors* are users and groups. *Resources* are objects to be protected, such as files and services. Each record in the database describes 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 (workstations) protected by CA Access Control.

Class Activation

The information about class status (that is, whether the class was active or inactive) is held in the database. Every attempt to access a resource is intercepted by CA Access Control, which checks the status in the database. If the class is inactive, access is allowed without further checking for authorization.

CA Access Control issues a list of active classes when the engine starts and when a user changes the class activity status. If a class is inactive, access to the resource is not intercepted, which reduces overhead.

Accessor Elements

Each user is represented by an *accessor element* (ACEE)—an in-memory reflection of the user's record in the database. CA Access Control builds the accessor element during the login process. The accessor element is associated with the user's process. Whenever the process requests a system service that is protected by CA Access Control, or issues an implicit request to access a resource, CA Access Control accesses the resource's record. It then determines whether the information in the previously created accessor element—such as the user's security level, mode, and group—lets the user access the resource.

Expanding Native Security

The following CA Access Control features expand native security.

Superuser Account Limitations

Users who administer and manage the operating systems are typically members of predefined accounts that are automatically created during system setup, such as the root account on UNIX systems, and the Administrator account on Windows systems. Each of the predefined accounts exists to perform a certain set of system functions.

Users acting as root or Administrator can perform a wide range of tasks, from creating, deleting, and modifying users to locking, reconfiguring, and shutting down servers.

One of the major security risks in these operating systems is that an unauthorized user can gain control of these accounts. If this happens, the user can cause enormous damage to the system.

CA Access Control lets you limit the rights granted to these accounts and to limit the rights of users who are members of the user groups that have these accounts as members. This reduces the vulnerability of your operating system.

CA Access Control Administrators

When you installed CA Access Control, you were asked to name one or more CA Access Control administrators. CA Access Control administrators have the authority to modify all or part of the rules database. You should have at least one full-authority administrator. This administrator can modify or create access rules freely and can designate other levels of administrators.

Once you have defined users for your system, you can assign administrative authority to other users by assigning the ADMIN attribute to them.

Note: A user with the ADMIN attribute possesses powerful authority. Consequently, the number of ADMIN users should be strictly limited. It is also a good policy to separate the roles of the native superuser and ADMIN, removing the ADMIN attribute from the superuser after you have set up one or more CA Access Control security administrators.

Because you always need at least one user with authority to manage the database, CA Access Control does not let you delete the last user that has the ADMIN attribute.

If you expect any of the CA Access Control administrators to be administering other hosts from this workstation, be sure that a rule in the database on that host gives them READ and WRITE access from this workstation.

Sub Administration

CA Access Control contains a *sub administration* feature. This lets administrators grant specific privileges that enable regular users to manage specific classes. These users are then called sub administrators.

For example, you can allow a specific user to manage users and groups only.

You can also specify a higher level of sub administration by granting access not only for specific classes, but for specified records in these classes.

Administration Rights for Regular Users

CA Access Control lets you grant ordinary users (that is, non-administrators) the necessary rights and privileges so that these users can perform administrative tasks without being members of the Administrators group. The ability to delegate tasks by granting administrative privileges in this granular way is a significant advantage of CA Access Control.

- A record in the SUDO class stores a command script to allow users to run the script with borrowed permissions.
- The data property value is the command script. This value can be modified by adding to it optional script parameter values.
- Each record in the SUDO class identifies a command for which a user can borrow permissions from another user.
- The key of the SUDO class record is the name of the SUDO record. This name is used instead of the command name when a user executes the commands in the SUDO record.

Enhanced File Protection

CA Access Control supports both logical and absolute file name formats. For example, if the file `foo.txt` is located under the directory `\tmp` on the logical drive `D` and the logical name `"D:"` is assigned to physical disk 1, partition 0, you can use either the logical or absolute file name to define a file to the CA Access Control database:

```
nr file D:\tmp\foo.txt
```

or

```
nr file \Device\HardDisk1\Partition1\tmp\foo.txt
```

Note: If the second format is used, the file remains protected even if the logical name of the disk is changed. The absolute file name format is also supported for CA Access Control generic file protection.

CA Access Control protects all file systems currently used in supported Windows operating systems. The two most commonly used are the Windows file system (NTFS) and the file allocation table (FAT). CA Access Control also supports CDFS (a file system especially for CDs).

CA Access Control supplies a total security solution to the file allocation table (FAT) and an extra layer of security to other file systems including NTFS and CDFS.

Generic File Protection

CA Access Control supports both logical and absolute file names. The absolute file name format is also supported for CA Access Control generic file protection.

Generic file protection lets you protect all the files that fit a specified wildcard pattern (regular expression). Any resource with a name matching the specified wildcard pattern is protected by the specified generic access rule. CA Access Control lets you protect files generically.

Should a resource match more than one generic access rule, CA Access Control chooses the rule that most closely matches the file.

With generic file protection, no more than a handful of security rules must be defined to protect many of the files requiring protection.

Password Protection

Native Windows security can protect passwords and enforce password quality in a number of ways. Windows offers the ability to:

- Enforce a maximum password age
- Enforce a minimum password length
- Save up to 24 generations of a user's passwords
- Lock out accounts after repeated login failures
- Force users to log on to Windows before changing their passwords

CA Access Control also enforces the same rules but through its own unique mechanisms. In addition, CA Access Control implements two-way password synchronization with mainframe computers.

Enhanced Password Protection

Native Windows security provides a significant amount of [protection for user passwords](#) (see page 21). However, CA Access Control significantly extends password protection so that the likelihood of a hacker succeeding in stealing a password is greatly reduced.

When using CA Access Control, you can create additional rules that force users to choose safer, more secure passwords. For instance, you can demand that users select a minimum number of alphabetic, numeric, special, lowercase, or uppercase characters. You can also ensure that the new password selected by a user does not contain, and is not contained by, the password being replaced.

Program Pathing

Program pathing is an access rule associated with a file that requires that the file is accessed only through a specific program. Program pathing greatly increases the security of sensitive files. CA Access Control lets you use program pathing to provide additional protection for the files in your system.

B1 Security Level Certification

CA Access Control includes the following B1 “Orange Book” features: security levels, security categories, and security labels.

- 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.
- 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.
- 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.

Note: For more information about B1 Orange Book features, see the *Implementation Guide*.

Setting Up Audit Procedures

CA Access Control 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 indicate whether access successes, failures, or both, should be logged; in addition, the AUDIT property for accessors can indicate whether login successes, failures, or both should be logged.

- If the resource or the accessor has the AUDIT(ALL) attribute, all events concerning resources protected by CA Access Control are logged, regardless of whether access failed or succeeded.
- If the access to a resource protected by CA Access Control is successful and the user or the resource has AUDIT(SUCCESS), the event is logged.
- If the access to a resource protected by CA Access Control fails and the user or the resource has AUDIT(FAIL), the event is logged.

Only a system auditor, a user to whom the AUDITOR attribute is assigned, can perform auditing tasks such as changing the auditing attribute that is assigned to users and resources.

If a resource is in warning mode, any access that violates access rules for the resource results in a warning mode audit record, which states that CA Access Control permitted access to the resource.

The audit records constitute a file called the *audit log* (seos.audit). The location for the audit log is specified in the registry, as is the location for the error log.

The audit log (and also the error log) is specified under the following registry key:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\logmgr
```

The audit log is a binary file and cannot be edited or changed. However, you can use CA Access Control Endpoint Management to view recorded events, to filter out events by time restrictions or event type, and so forth. (You can also use the seaudit utility to accomplish these same tasks.)

Consider archiving (backing up) old audit logs and error logs to let you scan the events at a later date.

Sending Audit Events to Unicenter TNG

Integration with Unicenter TNG is set up at installation.

You can choose to send audit data to Unicenter TNG, permit launching of CA Access Control from Unicenter TNG, or both. The two options are not interrelated.

Selecting the first option sets registry values under the subkey:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\UCTNG
```

The value Integration is set to 1 (yes) and the value EvtManagerServer receives the name of the Unicenter TNG host as a string value.

Audit events that are passed to Unicenter TNG appear in the Console logs in the Unicenter Enterprise Management\Enterprise Managers\Windows NT\Event window.

Audit Event	Display Color	Severity
Success	Blue	S
Denied	Orange	F
Fail	Orange	F
Warning	Blue	W
CA Access Control stopped (audit down)	Blue	I
CA Access Control started (audit start)	Blue	I

The second option permits launching CA Access Control from the Unicenter WorldView menu by pointing to the icon representing the TCP/IP Network in the Managed Objects window and selecting CA Access Control from the right-click menu.

CA Access Control also sends following information about events:

- Product name (CA Access Control + version number)
- User name
- Terminal name
- Class name
- Resource name
- Process name
- Event's time
- Full audit message in the format of CA Access Control auditing

The fields User name, Terminal name, Class name, Resource name, and Process name are not always sent, depending on event type.

Components

CA Access Control includes a database (seosdb), two drivers (seosdrv and drveng), a number of services (including the Watchdog, the Agent, the Engine (seosd), the Policy Model, and Task Delegation), and a graphical user interface.

Database

The database contains definitions of the following elements:

- Users and groups in your organization
- System resources that need protection
- Rules governing user and group access to system resources

Drivers

The drivers protect all the CA Access Control files and registry keys by performing the following tasks:

- Intercepting every request to open a file or registry key, terminate a process, and perform network activities
- Passing these requests to the CA Access Control Engine and receiving the decision of the Engine whether the request should be granted or denied
- Forwarding the decision to the original system call of the operating system, which then continues its processing based on the answer it received from the drivers.

Services

Watchdog

The Watchdog constantly checks that the other CA Access Control services are running. On the rare occasion when the Watchdog discovers that another service has stopped, it immediately starts the service again.

Agent

The Agent is responsible for the following tasks:

- Communicating with CA Access Control clients through a proprietary application protocol above TCP/IP
- Managing security for the CA Access Control user

Engine

The Engine is responsible for the following tasks:

- Managing the database, including controlling all database updates
- Deciding whether to grant access requests that it receives from the Driver and the Agent
- Checking that the Watchdog service is running, and restarting the Watchdog if it discovers that the Watchdog has stopped running

The Engine handles database access requests *and* makes the access decision, creating an efficient service.

Policy Model

Managing tens or hundreds of databases individually is not practical. Therefore, CA Access Control supplies the Policy Model service, a component that permits management of many computers from one computer. Using the Policy Model service is optional, but it greatly simplifies administration at large sites.

With the Policy Model service, use a Policy Model database (PMDB). Like other CA Access Control databases, the PMDB contains users, groups, protected resources, and rules governing access to the resources. In addition, the PMDB contains a list of subscriber stations. A subscriber station is one linked to the PMDB so that any change to the PMDB is automatically sent to the subscriber database.

You can create a basic security policy for your organization and implement all the necessary rules on a single database—the Policy Model database. The subscribers can include both Windows and UNIX stations, ensuring uniform rules with minimal administrative effort.

The system or security administrator updates the PMDB. The PMDB then propagates all updates from the PMDB to its subscribers in batch mode, freeing the administrator for other work.

A PMDB can have two types of subscribers: another PMDB or a local database. This PMDB also contains a list of subscribers to which it propagates database updates. This feature lets you build a hierarchy of PMDBs. The local database can be used to protect the users, groups, and resources defined on the station.

selang

The command-line language, *selang*, performs all the functions of CA Access Control. To use *selang* commands, open a command prompt window and start *selang*. You can also use *selang* in scripts.

For more information about *selang* and its commands, see the chapter “The *selang* Command Language” in the *Reference Guide*.

Endpoint Management

CA Access Control provides two ways to let you manage the resources in your enterprise and control who has access to them:

- **selang**—the CA Access Control command language.

The *selang* command language lets you make definitions in the CA Access Control database. The *selang* command language is the command definition language.

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

- **CA Access Control Endpoint Management**—the endpoint administration interface.

The web-based interface lets you administer remote endpoints through a central administration server.

Note: For more information about installing CA Access Control Endpoint Management, see the *Implementation Guide*.

Chapter 3: Managing Users and Groups

This section contains the following topics:

[Users and Groups](#) (see page 29)

[Where Information about Accessors Is Stored](#) (see page 30)

[Guidelines for Managing Accessors in Enterprise Stores](#) (see page 31)

[Database Accessors](#) (see page 36)

[Accessor Management](#) (see page 39)

Users and Groups

In CA Access Control, every action and access attempt is performed on behalf of a user, who is held responsible for submitting the request. Every process in the system is therefore associated with a certain user name. The user name identifies the user to CA Access Control.

A *user* is a person who can log on, or can be the owner of a batch or daemon program. In CA Access Control, every access attempt is performed by a user. CA Access Control can use user information from the CA Access Control database and from enterprise user stores. It stores user information in its database, in either a USER record or an XUSER record.

Note: An *enterprise user store* is a store in the operating system that stores users or groups, for example, /etc/passwd and /etc/groups on UNIX systems, or Active Directory on Windows.

A *group* is a collection of users. A group defines common access rules for users in the group. Groups can be nested (belong to other groups). CA Access Control can use group information from the CA Access Control database and from the enterprise user stores. Typically, you create groups and assign users to them, based on a role, for example, database_administrators.

The user records are the key accessor records. The main purpose for using groups in CA Access Control is to assign access authorities to all users in group at one time. Assigning access authorities at one time is easier and less error prone than assigning them separately to each user.

Where Information about Accessors Is Stored

The information that CA Access Control uses about users and groups is stored both in the CA Access Control database and in the host operating system. The host operating system information stores are called *enterprise user stores*, or just *enterprise stores*. By default, CA Access Control is configured so that it does not use the enterprise stores. You can, however, configure CA Access Control so that if it cannot find a user or group defined in its database, it looks for, and uses the information from, the users and the group memberships defined in the enterprise stores.

Note: CA Access Control uses information from the enterprise stores but only writes to them if you use `selang` command in the native environment.

When checking for authorization, CA Access Control always checks for accessors defined in its own database before it checks the enterprise store: if you have an enterprise user with the same name as a user defined in the CA Access Control database, the enterprise user is ignored by CA Access Control.

How CA Access Control Finds a User Record

When a user logs in, CA Access Control conducts the search in the following order, until it finds a record associated with the user:

1. CA Access Control searches for a user defined in its database.
2. CA Access Control searches its cache for an enterprise user of that name.

When the network is down, the operating system (OS) lets users log in using the OS cached credentials. The purpose of the CA Access Control cache is to let CA Access Control also use enterprise users' records in these cases.

3. CA Access Control uses the operating system to search the enterprise user stores for a user of that name.
4. If CA Access Control does not find a record associated with the user in its database or in the enterprise stores, CA Access Control assigns the user the attributes in the `_undefined` USER record.

Integration with the Enterprise User Stores

Typically, you configure CA Access Control to use the groups and users that are defined in the enterprise user stores.

If you do configure CA Access Control like this, by default, when an access rule that references an enterprise user or group is created, or when a user logs in to the operating system, CA Access Control creates a record in its database for that user or group, if one did not exist before. These records have the class XUSER (for enterprise users) or XGROUP (for enterprise groups). They hold the properties that CA Access Control requires to enforce access rules. You do not need to manage them, because CA Access Control creates them as required.

The only properties of an enterprise user or group that CA Access Control fetches from the enterprise user stores are the names and the group membership properties.

Guidelines for Managing Accessors in Enterprise Stores

If you decide to manage your accessors in enterprise user stores, you should consider the guidelines in the following sections.

Users and Groups that Must be Defined in the Database

CA Access Control needs some users and groups to be defined in its database, rather than in the enterprise user stores. These include:

- [Predefined users](#) (see page 37)
- [Predefined groups](#) (see page 38)
- A CA Access Control administrator
- Profile groups
- Logical users

Restrictions on the Use of Enterprise Users

CA Access Control imposes the following restrictions on the use of enterprise users:

- You cannot create, or refer to, an enterprise user in CA Access Control if it has the same name as a user defined in the database.
- You cannot create, delete or modify an enterprise user using the `selang AC` environment.

- You cannot use an enterprise user as a logical user.
- By default, you cannot create an enterprise user in CA Access Control unless the user is already defined in the enterprise user store. However, you can enable or disable this behavior on UNIX systems.

More information:

[Enable or Disable Checking Enterprise Store before Creating XUSER Records on UNIX](#)
(see page 34)

Restrictions on the Use of Enterprise Groups

CA Access Control imposes the following restrictions on the use of enterprise groups:

- You cannot create or delete an enterprise group within the selang AC environment.
- You cannot change the membership of an enterprise group within the selang AC environment.
- You cannot use an enterprise group as a [Profile Group](#) (see page 39).

Enable or Disable the Use of Enterprise Users and Groups

CA Access Control cannot by default use the groups and users defined in the enterprise user stores, but you can enable CA Access Control to do so. We recommend that you enable this feature unless you need compatibility with previous versions of CA Access Control.

To let CA Access Control use enterprise users and groups, set the configuration setting `osuser_enabled` to `yes`. To disable this behavior, set the value of `osuser_enabled` to `no`.

Example: Enable the Use of Enterprise Users and Groups on Windows

The following registry setting enables the use of enterprise users and groups on Windows:

- Key: HKLM\SOFTWARE\ComputerAssociates\AccessControl\OS_user
- Name: `osuser_enabled`
- Type: REG_DWORD
- Value: `yes`

Example: Enable the Use of Enterprise Users and Groups on UNIX

The following commands stop CA Access Control, enable the use of enterprise users and groups on UNIX, and restart CA Access Control:

```
secons -s  
seini -s OS_User.osuser_enabled yes  
seload
```

Enable or Disable the Creation of XUSER Records at Enterprise User Login

If CA Access Control is enabled to use enterprise users, by default it creates a record (in the XUSER class) for a user when that user logs in. Sometimes you do not want this, for example, if thousands of users log on at the same time each day.

To prevent CA Access Control creating XUSER records when users log in, change the value of the configuration setting `create_user_in_db` to 0 (zero). To re-enable this behavior set the value to 1 (one).

Example: Disable the Automatic Creation of XUSER Records on Enterprise User Login on Windows

The following registry setting disables the automatic creation of an enterprise user record in CA Access Control on Windows:

- Key: HKLM\Software\ComputerAssociates\AccessControl\OS_user
- Name: create_user_in_db
- Type: REG_DWORD
- Value: 0

Example: Disable the Automatic Creation of XUSER Records on Enterprise User Login on UNIX

The following commands stop CA Access Control, disable the automatic creation of a XUSER record on UNIX, and restart CA Access Control:

```
secons -s  
seini -s OS_User.create_user_in_db 0  
seload
```

Enable or Disable Checking Enterprise Store before Creating XUSER Records on UNIX

Sometimes you may want to create an enterprise user in CA Access Control when the user is not defined in the enterprise user store. On Windows you cannot create an enterprise user in CA Access Control unless the user exists in the Windows user store. On UNIX, the default behavior is the opposite to Windows. However, on UNIX, you can enable or disable this default behavior.

To disable checking (and therefore allow CA Access Control to create XUSER records when there is no enterprise user equivalent), change the value of the configuration setting `verify_osuser` to 0. To enforce checking, set the value to 1.

Example: Enable Creation of XUSER Records without Checking the Enterprise User Store

The following set of commands stops CA Access Control, enables the creation of XUSER records with no enterprise store equivalents, and restarts CA Access Control:

```
secons -s
seini -s OS_User.verify_osuser 0
seload
```

Recycled Enterprise Store Accounts on Windows

Recycled accounts are enterprise store users or groups that have been deleted and then recreated (using the same name). This is likely to happen when you remove a user from the user store (for example, when the user resigns) and then create a new account for a new user that has the same name as the old removed user.

Recycled accounts are a security concern because you do not necessarily want new accessors to have the same access permissions as those that were granted to the old account with the same name. To solve this problem, CA Access Control authorization is based on the SID. This means that when you create a new accessor, with the same name as a deleted accessor with existing access permissions, the new accessor does not automatically receive the old permissions of the old accessor.

Important! Recycled account accessors *do not* inherit the old access permissions. However, database access rules, which mention the accessor's name (not SID), may make it seem like these rules still apply. Use the `secons -checkSID` command to resolve this.

Resolve Recycled Enterprise Accounts on Windows

If an enterprise account (user or group) has associated database rules is then recycled (deleted and created with the same name), it may look like the old database rules still apply to the new account. However, as CA Access Control authorization is based on SID, these rules no longer apply and you need to create new rules for the new group. Before you can create the new rules, you have to resolve recycled accounts.

To resolve recycled enterprise accounts open a command prompt and run the following commands:

```
secons -checkSID -users  
secons -checkSID -groups
```

CA Access Control works through all the enterprise user accounts it has (XUSER records) and then all the group accounts (XGROUP records) and identifies accounts with an SID that differs from the SID of the enterprise account. It renames these accounts in CA Access Control using the following naming convention: *SID (accountName)*

You can now create the new rules for the recycled account.

Note: Recycled user accounts are resolved in this way when the user logs in or tries to access a resource. We recommend that when you create an enterprise account, run the `secons -checkSID` command as a scheduled task.

Example: A Recycled Group Account

Company ABCD has a group called *interns* in its enterprise store. The group has nine members and they are working on productA. The administrator makes the group known to CA Access Control and assigns it with access permissions to the files group members need to access, as follows:

```
nxcg interns owner(msmith)
auth file c:\products\productA\materials\* xgid(interns) access(all)
auth file c:\HR\interns\* xgid(interns) access(read)
```

When the interns complete their tenure with ABCD, the enterprise store administrator deletes the group. Three months later, a new group of interns with six members is created in the enterprise store, with the same name. The old rules in the CA Access Control database still exist so it seems like the new *interns* group inherited the permissions of the old group. However, these rules apply to the old interns group and the CA Access Control administrator needs to create new rules for the new group.

To do this, the administrator has to identify and resolve the recycled interns account, as follows:

```
secons -checkSID -groups interns
```

This renames the XGROUP resource, and any access rules references to it, to "*SID (domain\interns)*". Now, the administrator can create new rules for the new interns group that works on productB:

```
nxcg interns owner(msmith)
auth file c:\products\productB\materials\* xgid(interns) access(all)
auth file c:\HR\interns\* xgid(interns) access(read)
```

Note: For more information on the secons utility, see the *Reference Guide*.

Database Accessors

Regardless of how you decide to manage your users, some accessors must be defined in the CA Access Control database, as described in the following sections.

Predefined Users

CA Access Control predefines the following users, which you cannot delete:

+devcalc

(Windows) The user name under which CA Access Control runs the deviation calculation process, devcalc.

_dms

Installed on the advanced policy management server components' databases (DMS, DH reader, and DH writer), the `_dms` user is used by policyfetcher and devcalc to communicate with the DH and DMS.

nobody

The nobody user is a user record that cannot correspond to a real user. Use this record to create rules that do not give any user the associated permissions. For example, you can set *nobody* as the owner of resources, meaning that no user will get the permissions associated with owning that record.

+reportagent

The user name under which CA Access Control runs the Report Agent.

_seagent

`_seagent` is the user name under which CA Access Control runs some internal processes, such as:

- The PMDB process, sepmd
- (UNIX) The deviation calculation process, devcalc
- The user and group record update exit processes

The `_seagent` user has the SERVER attribute.

_sebuildla

(UNIX) The `_sebuildla` user is the user name under which CA Access Control runs the `sebuildla` utility to create a lookaside database for the CA Access Control daemon, `seosd`.

_seoswd

(UNIX) `_seoswd` is the user name used to run the `seoswd` watchdog daemon to monitor the file information and digital signatures of programs defined in the database as trusted programs.

_undefined

`_undefined` represents all users that are undefined in CA Access Control. You can use `_undefined` to include undefined users in ACLs.

Predefined Groups

CA Access Control comes with predefined groups. Except for the `_interactive` and `_network` groups, you add users to these groups in the same way as you do for any other group.

`_abspath`

If a user is in the `_abspath` group when logging in, that user must use absolute path names to invoke programs.

`_interactive`

A user is a member of the `_interactive` group only for the purposes of an access attempt. Users are members of the `_interactive` group if they are logged into the same host as the resource they are trying to access. CA Access Control dynamically and automatically manages the membership of the `_interactive` group—you cannot change the membership.

`_network`

This is the complementary group to `_interactive`. A user is a member of the `_network` group for the purposes of access only. Users are members of the `_network` group if they are trying to access a resource from a different host than the resource belongs to. CA Access Control dynamically and automatically manages the membership of the `_network` group—you cannot change the membership.

`_restricted`

For users in the `_restricted` group, all files, and on Windows registry keys too, are protected by CA Access Control. If a file or a Windows registry key does not have an access rule explicitly defined, access permissions are covered by the `_default` record for that class (FILE or REGKEY).

Note: Users in the `_restricted` group may not have sufficient authorization to do their work. If you plan to add users to the `_restricted` group, consider using Warning mode initially.

`_surrogate`

When a user uses a member of the `_surrogate` group as a surrogate, CA Access Control writes a full trace in the audit trail of the surrogate's actions, tagged with the original user's name.

Example: Adding a User to the `_restricted` Group Using `selang`

The following `selang` command adds the enterprise user `john_smith` to the `_restricted` group:

```
joinx john_smith group(_restricted)
```

Profile Groups

A *profile group* is a group defined in the CA Access Control database that contains default values for user properties. When you assign a user to a profile group, the profile group provides those values to the user unless they have already been set for the user.

You can specify a profile group for a user when you create the user, or you can assign the user to the profile group afterwards.

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 home directory of the user, the audit properties, the PMDB that defines the access authorities, and various password rules affecting a user who is associated with a profile group.

How CA Access Control Uses Profile Groups to Determine User Properties

The following process describes how CA Access Control uses profile groups to determine user properties:

1. CA Access Control checks if the user's record in the USER or XUSER class has a value for the property.

If the user's record has a value for the property, CA Access Control uses that value.

2. CA Access Control checks if the user is assigned to a profile group.

If the user is assigned to a profile group, the process continues. If the user is not assigned to a profile group, CA Access Control assigns the default property value to the user.

3. CA Access Control checks if the profile group has a value for that property.

If the profile group has a value for the property, CA Access Control assigns that value to the user. If the profile group does not have a value for the property, CA Access Control assigns the default property value to the user.

Note: If the audit property of a user or profile group is not set, the audit property of a group can affect the audit property of a user.

More information:

[How CA Access Control Determines the Audit Mode for a User](#) (see page 106)

Accessor Management

You can create, modify, and delete database or enterprise user or group records by using CA Access Control Endpoint Management or by using selang.

Manage Users or Groups

If you want to view or modify the properties of a particular accessor, or if you want to delete an accessor, you must first find that accessor.

To manage users or groups

1. In CA Access Control Endpoint Management, do as follows:
 - a. Click Users.
 - b. Click either the Users *or* Groups subtab.

Depending on your selection, the Users or the Groups page appears.

2. Complete the following fields in the Search section:

User/Group Name

Defines a mask for the accessors you want to find. You can enter the full name of the accessor you are after or you can use a mask. For example, use `*admin*` to list accessors whose name contains "admin".

Use an `*` (asterisk) to list all accessors and a `?` (question mark) to replace a single character.

User/Group Repository

Specifies the source from which you want to fetch a list of accessors. The source can be either:

- **Internal Accounts**—accessors defined in the CA Access Control database.
- **Enterprise Accounts**—accessors defined in specific enterprise user stores.



Show only AC accounts/profiles

Specifies whether to list only those accounts that have records in the CA Access Control database as follows:

- If you chose Internal Accounts, the application lists only those accounts that exist in the CA Access Control database (no native accounts).
- If you chose Enterprise Accounts, the application lists only those accounts that have a CA Access Control enterprise profile (XUSER or XGROUP records).

Click Go.

A list of accessors that exist in the repository you chose appears.

3. Do *one* of the following:
 - Click  in the View column to view the properties of the accessor.
 - Click  in the Delete column to delete the accessor.
 - Click the name of the accessor to modify the properties of the accessor.
 - Select the accessors you want to delete and click Delete.
 - Click Create User or Create Group to create a user or group record in the CA Access Control database.

Example: Search for Enterprise Users in a Repository

The following graphic shows you the result of looking for all users in the ABC-DM1 enterprise user store.

Search
Create User

Required

- User Name:** *

For multiple entities please use the wildcard *

User Repository:

Options: Show only AC accounts/profiles

User Environment

- With AC Profile
- Without AC Profile

Users list for: COMP001
Create User

Here are the results for XUSER with name: * at 08/07/09 00:22

Select and: Delete 1 - 10 of 12 > >>

<input type="checkbox"/> Select	Env.	Name	Comment	View	Delete
<input type="checkbox"/>		ABC-DM1\ac_ent_pers			
<input type="checkbox"/>		ABC-DM1\Administrator			
<input type="checkbox"/>		ABC-DM1\alice			
<input type="checkbox"/>		ABC-DM1\ASPNET			
<input type="checkbox"/>		ABC-DM1\bob			
<input type="checkbox"/>		ABC-DM1\entmgmt			
<input type="checkbox"/>		ABC-DM1\Guest			
<input type="checkbox"/>		ABC-DM1\IUSR_IIS_SVR1			
<input type="checkbox"/>		ABC-DM1\IWAM_IIS_SVR1			
<input type="checkbox"/>		ABC-DM1\rand			

1 - 10 of 12 > >>

Total of 12 objects.

User Management Using selang

Use the following selang commands for records of enterprise users:

- **newusr** and **editusr**—define a new enterprise user record
- **chusr** and **editusr**—change the CA Access Control properties of an enterprise user
- **find xuser**—list enterprise users that have a CA Access Control record
- **rmusr**—delete a user
- **show xuser**—display the CA Access Control properties of an enterprise user

Use the following selang commands for CA Access Control database user records:

- **newusr** and **editusr**—define a new user record
- **chusr** and **editusr**—change the properties of a user
- **rmusr**—delete a user
- **find user**—list database users
- **show user**—display the properties of a user

Example: Define a User in the Database Using selang

The following selang command defines a new user in the CA Access Control database with security level 100:

```
newusr internalUser level(100)
```

Example: Change a Property of an Enterprise User Using selang

The following selang command gives the AUDITOR property to an enterprise user Terry:

```
chxusr Terry auditor
```

Group Management Using selang

You can change any property of any group, except that you cannot change the name or the membership of enterprise groups (from within CA Access Control).

To change group properties or to assign access rights associated with groups, you can use CA Access Control Endpoint Management or the following selang commands:

- **join[-]** and **joinx[-]**

Change the membership of an internal group

Use **join** to add internal accessors to the group. Use **joinx** to add enterprise groups and users to an internal group. Use the **-** (minus) form of the commands to remove accessors.

- **editgrp, newgrp, chgrp**
Change the non-membership properties of an internal group
- **editxgrp, newxgrp, chxgrp**
Change the non-membership properties of an enterprise group
- **rmgrp, rmxgrp**
Remove a user group

Example: Define a Group in the Database Using selang

The following selang command defines a new group “sales” in the database. The full name of the group is “Sales Department”:

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

Example: Change a Property of a Group Defined in the Database Using selang

The following selang command makes CA Access Control audit all events for members of the group AC_admins:

```
chgrp AC_admins audit(all)
```

Example: Add an Enterprise Group to an ACL Using selang

The following selang command adds the enterprise group mygroup to the ACL of the myfile:

```
Authorize FILE (myfile) xgid(mygroup)
```

Example: Add an Enterprise User to a Group Defined in the Database Using selang

The following selang command adds the enterprise user mydomain\administrator to the group AC_admins which is defined in the database:

```
joinx mydomain\administrator group(AC_admins)
```

Example: Add an Enterprise Group to a Group Defined in the Database Using selang

The following selang command adds the enterprise group Guests to the _restricted group:

```
joinx Guests group(_restricted)
```

Chapter 4: Managing Resources

This section contains the following topics:

[Resources](#) (see page 45)

[Classes](#) (see page 46)

[Windows Services Protection](#) (see page 53)

[Windows Registry Protection](#) (see page 57)

[Protect File Streams](#) (see page 61)

[Internal File Protection](#) (see page 62)

Resources

A *resource* is an entity that can be accessed by an accessor and protected by an access rule, or the CA Access Control database record that corresponds to that entity. Examples of resources are files, programs, hosts, and terminals.

The main purpose of creating resource records in CA Access Control is to define access permissions for the resource that corresponds to the resource record. The access permissions that are required to access a resource are specified in the resource record's access control lists.

Resource Groups

A *resource group* is a resource that contains a list of other resources. A resource group is a member of one of the following classes: CONTAINER, GFILE, GSUDO, GTERMINAL, or GHOST.

Because a resource group is itself a resource, it has the same properties as its member resources. Therefore the advantage of using resource groups is that it simplifies administration. You can change the properties of all the member resources by changing the properties of the resource group.

Note: On Windows, CA Access Control takes into account resource group ownership when checking user authorization to a resource. This behavior was introduced in r12.0. In earlier releases, the authorization process considered only the resource's owner.

For example, you define a FILE resource with a default access of none and no owner. The FILE resource is a member of a GFILE resource with a named owner. In CA Access Control r12.0 and later, the named group owner has full access to the file. In earlier releases, nobody has access to the file.

Classes

In CA Access Control, the *class* of a record defines the properties that the record can have. All records in a class have the same properties, though different values for these properties.

Examples of classes are:

- **TERMINAL** class. This contains records for terminals, such as `tty1`, `tty`.
- **FILE** class. This contains records for files.
- **PROGRAM** class. This contains records of programs.

Each record contains values for the properties appropriate to the record class. For example, a record in the `XUSER` class includes such properties as the enterprise user's location and working hours, while a record in the `HOSTNET` class includes such properties as net services and IP address data.

CA Access Control includes predefined classes. You can also define new classes, called user-defined classes.

Default Record for Class

Most classes can include a default record (`_default`) specifying access types for resources of that class that are not defined in 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 (Deprecated)

The UACC class is no longer recommended. To specify the default values for records in a class, use the `_default` record.

Some earlier versions of CA Access Control 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`. CA Access Control checks for authorization in the following order. The primary question is this: Is the process `store_log` defined in the database? CA Access Control 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 CA Access Control. In that case, CA Access Control 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, CA Access Control 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.

Predefined Classes

The predefined classes can be categorized into the following types:

Class Type	Purpose
Accessor	Defines objects that access resources, such as users and groups
Definition	Defines objects that define security entities, such as security labels and categories
Installation	Defines objects that control the behavior of CA Access Control
Resource	Defines objects that are protected by access rules

The following table contains a list of all predefined classes.

Class	Class Type	Description
ADMIN	Definition	Lets you delegate administrative responsibilities to users who do not have the ADMIN attribute. You give these users global authorization attributes and limit their administration authority scope.
AGENT	Resource	Not applicable to CA Access Control
AGENT_TYPE	Resource	Not applicable to CA Access Control
APPL	Resource	Not applicable to CA Access Control
AUTHHOST	Accessor	Not applicable to CA Access Control
CALENDAR	Resource	Lets you define a Unicenter TNG calendar object for user, group, and resource enforced time restrictions.
CATEGORY	Definition	Lets you define a security category.
CONNECT	Resource	Lets you protect outgoing connections. 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.
CONTAINER	Resource	Lets you define 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	Lets you protect a file, a directory, or a file name mask.
GAPPL	Resource	Not applicable to CA Access Control
GAUTHHOST	Definition	Not applicable to CA Access Control

Class	Class Type	Description
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 an internal group.
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.
HNODE	Definition	The HNODE class contains information about the organization's CA Access Control hosts. Each record in the class represents a node in the enterprise.
HOLIDAY	Definition	Each record in this class defines one or more periods when users need extra permission to log in.
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.
HOSTNET	Resource	Each record in this class is identified by an IP address mask and contains access rules.
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 CA Access Control administration computer.
POLICY	Resource	Each record in the POLICY class defines the information required to deploy and remove a policy. It includes a link to the RULESET objects that contain a list of the selang commands for deploying and removing the policy.
PROCESS	Resource	Each record in this class defines an executable file.

Class	Class Type	Description
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.
PWPOLICY	Definition	Each record in the PWPOLICY class defines a password policy.
RESOURCE_DESC	Definition	Not applicable to CA Access Control
RESPONSE_TAB	Definition	Not applicable to CA Access Control
RULESET	Resource	Each record in the RULESET class represents a set of rules which define a policy.
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 authorization protection with a logical user ID. This allows you to set 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 contains access rules for an accessor that define who can use that accessor as a surrogate.
TCP	Resource	Each record in this class defines a TCP/IP service, for example, 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.
USER	Accessor	Each record in this class defines an internal user.
USER_ATTR	Definition	Not applicable to CA Access Control

Class	Class Type	Description
USER_DIR	Resource	Not applicable to CA Access Control
XGROUP	Resource	Each record in this class defines an enterprise group to CA Access Control.
XUSER	Resource	Each record in this class defines an enterprise user to CA Access Control.

Note: CA Access Control database classes TCP and SURROGATE are not active by default.

If you upgrade from an earlier release where the TCP class is active but you do not have any TCP records and have not changed the `_default` TCP resource, CA Access Control deactivates the class during upgrade. The same is true for the SURROGATE class.

If you upgrade from an earlier release where the SURROGATE class is active and you have defined SURROGATE records or have changed the value of any SURROGATE record from its default, CA Access Control retains the SURROGATE class configuration after the upgrade. The class remains active and kernel mode interception remains enabled.

Note: For more information about CA Access Control classes, see the *selang Reference Guide*.

User-Defined Classes

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

Example: User-Defined Class for a Database View

A site may use a database to store and display proprietary data.

You can define a user-defined class `DATABASE_VIEWS`, and define each database view to be a resource member of that class. Give the resource an ACL that defines the access authority required to create that database view. When a user attempts to create a database view, CA Access Control checks the access authority of the user, and permits or disallows the creation based on the ACL.

Wildcards in User-defined Classes Resources

By using wildcards in the name of a resource in a user-defined class, you can create a resource record that corresponds to multiple physical resources: any physical resource with a name that matches the wildcard pattern is protected by the access authorities associated with the resource record.

The wildcards you can use are:

- * for any number of any characters
- ? for any one character

If a physical resource name matches more than one resource record name, the longest non-wildcard match is used for that resource.

CA Access Control does *not* accept the following wildcard patterns as resource names:

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

User-Defined Class—Example

Suppose that your system serves a bank and you want to protect transfers of large amounts between accounts. You can use the following outline to set up this security.

1. Define a class to contain the records that describe transfers, called, for example, TRANSFERS.
2. For each monetary level transfer that you might want to protect, define a record in the TRANSFERS class.

For example, you might define records named Upto.\$1K, Upto.\$1M, Upto.\$10M, and Over.\$10M.

Define any other resources that you need to control transfers as members of the TRANSFERS class.

3. To give different users permission to perform different maximum transfers, grant or deny them access to the various records in the TRANSFERS class.
4. In addition, to handle programmatic transfers, insert in the bank's money-transfer program a call to the CA Access Control API, so that it checks the user's permission before it allows a transfer to proceed.

Windows Services Protection

CA Access Control lets you protect Windows services. A *Windows service* is a program that runs in the background on Windows, and is the Windows equivalent to a daemon on UNIX.

The CA Access Control Windows service protection intercepts service access events that originate from one of the following:

- Service management and information events.

CA Access Control intercepts the services.exe process for each service access. This includes starting or stopping a service. For example, net start *service*, net stop *service*, and so on, are all protected.

Intercepted events in this case are audited using the protected service's name.

- Service database management events.

CA Access Control intercepts registry calls to the service control management database to protect against service state queries or changes. This means that CA Access Control automatically protects the registry areas that are associated with the protected service. Effectively, CA Access Control protects the following registry keys when you define service protection:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\service_name  
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\service_name*
```

Intercepted events in this case are audited using the full registry path.

You protect a Windows service in the same way as you protect other resources, that is by creating assigning a resource to the service and adding accessors to the resource's access control lists. The resource class for a Windows service is WINSERVICE. A WINSERVICE resource has two access control lists: an ACL and an NACL. Valid access types for an entry in a WINSERVICE access control list are:

- Read
- Modify
- Start
- Stop
- Pause
- Resume

Enable and Disable Windows Services Protection

You can enable or disable the CA Access Control protection of Windows services.

To enable protection of Windows services, set the configuration setting `OperationMode` in the `Instrumentation\PlugIns\WinServiceplg` section of the CA Access Control registry to 1. To disable protection, set `OperationMode` to 0.

By default, CA Access Control enables protection of Windows services.

For CA Access Control to protect a Windows service, protection needs to be enabled and the `WINSERVICE` class needs to be active.

Protect a Windows Service

You can protect a Windows Service and so provide additional protection to Windows operations.

To protect a Windows Service

1. Ensure you have [enabled Windows services protection](#) (see page 54).
2. Ensure the `WINSERVICE` class is active. (It is active by default.)
3. Create a `WINSERVICE` record in CA Access Control, with the same name as the Windows service you want to protect.

Note: The Windows service name is shown on the General tab of the Windows service properties dialog, but is not the same as the "display name" on that tab.

4. Assign the accessors and their access authorization to the service.

The service is now protected.

Example: Restrict Access to the Print Spooler

On Windows the print spooler has the service name `spooler`. The following `selang` commands ensure the `WINSERVICE` class is active and sets the default access to the spooler to `read`.

```
setoptions class+(WINSERVICE)
editres WINSERVICE(spooler) defacc(R)
```

Non-IPv4 telnet Connections Are Not Secured on Windows Server 2008

On Windows Server 2008, CA Access Control cannot secure a telnet connection unless it uses IPv4.

To protect a localhost telnet connection—telnet from the localhost to the localhost—on Windows Server 2008, modify the `/etc/HOSTS` file as follows:

```
127.0.0.1      localhost
#           ::1          localhost
127.0.0.1      <your server name without domain suffix>
```

If your computer is on an IPv6 domain, add the following line:

```
127.0.0.1    <your server name with domain suffix>
```

View Access Attempts to a Protected Windows Service

When CA Access Control protects a Windows service, it intercepts, and records in the audit log, access attempts that are related to the service. These access attempts may be a result of using the services.exe process to manage the service (start, stop, and so on), or a result of registry access to the service database management area of the protected service. While the former access is audit contains only the service name, the latter (registry access) contains the full registry path. To view all access attempts related to a Windows service, you have to use wildcards.

To view access attempts to a protected Windows service, create an audit filter that filters audit records of class WINSERVICE and resource name **myService**

CA Access Control displays all audit records for the WINSERVICE resource you defined (whether access was attempted through the registry or through a service management interface).

Example: View All Access Attempts to the Print Spooler Service

This example assumes that you defined the Print Spooler service to CA Access Control with no access as follows:

```
er winservice spooler defaccess(none) owner(nobody)
```

You can then use the seaudit utility to list all access attempts to the Print Spooler service as follows:

```
seaudit -resource WINSERVICE *spooler* *
```

This command lists all audit records for the class WINSERVICE that were recorded for access attempts to the Print Spooler service. The resulting output can look as follows:

```
seaudit - Audit log lister
03 Apr 2008 16:53:48 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:53:48 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:53:50 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:53:50 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:53:53 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:53:53 D WINSERVICE bigHost1\Administrator Read 69 2 Spooler
c:\WINDOWS\system32\services.exe bigHost1.comp.com
03 Apr 2008 16:54:10 D WINSERVICE bigHost1\Administrator Read 69 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler
C:\WINDOWS\regedit.exe bigHost1.comp.com
03 Apr 2008 16:54:10 D WINSERVICE bigHost1\Administrator Read 69 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler
```



```
C:\WINDOWS\regedit.exe bigHost1.comp.com
03 Apr 2008 16:54:19 D WINSERVICE bigHost1\Administrator Read 69 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler
C:\WINDOWS\regedit.exe bigHost1.comp.com
03 Apr 2008 16:54:26 D WINSERVICE bigHost1\Administrator Read 69 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler
C:\WINDOWS\regedit.exe bigHost1.comp.com
03 Apr 2008 16:54:26 D WINSERVICE bigHost1\Administrator Modify 69 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Spooler
C:\WINDOWS\regedit.exe bigHost1.comp.com
```

Total records displayed 11

Windows Registry Protection

CA Access Control lets you protect entries in the Windows registry.

You provide protection to a registry key by assigning a resource of class REGKEY to the key. You can then specify access authorities on the key, as with other resources.

Specifying access rights on a key does not affect access to subkeys of the key, except for enumeration (listing) of subkeys, which requires read access to the key.

CA Access Control only supports the REGVAL resource in the AC environment on Windows Server 2003 and subsequent Windows systems. On these systems, CA Access Control protects registry values with the REGVAL class, and the REGKEY access authorization does not affect access to the key's values.

On earlier systems, CA Access Control does not support the REGVAL resource in the AC environment and the access authorization applied on a REGKEY record does affect access to the key's values.

REGKEY and REGVAL records have identical structures. Each record contains the following access control lists:

- ACL
- CALACL
- NACL
- PACL

REGVAL and REGKEY records both allow the same access types, which are as follows:

- READ
- WRITE
- DELETE
- NONE

Note: CA Access Control registry protection does not protect the registry operations of loading and unloading a hive. On Windows Server 2008 and subsequent systems, CA Access Control returns a value of REG_NONE if an accessor tries to access a protected registry value with access NONE. A value of REG_NONE confirms that a value is present but does not specify what the value is.

Protect a Windows Registry Entry

You can protect a Windows registry entry, and so provide additional protection to Windows operations.

To protect a Windows registry entry

1. If you want to use the REGKEY and REGVAL class records, ensure these classes are active. (They are active by default.)
2. Create a REGKEY or a REGVAL record with the name of the registry key or value you want to protect.

Note: Use the full registry path name to specify the key or value. You can use a wildcard to specify all sub-keys or sub-key values that are nested under a key.

The registry entry is now protected with the default access that CA Access Control provides for the record.

3. (Optional) Assign the users and groups, with their access authorization, to the appropriate access control list in the REGKEY or REGVAL record.

Example: Provide default access of NONE to a Registry Key

The following `seimg` command provides default access of NONE to a registry key:

```
er REGKEY HKEY_LOCAL_MACHINE\SOFTWARE\Test\Key1 defacc(NONE) owner(nobody)
```

As a result, the default access to `key1` is as follows:

Action	Systems earlier than Windows Server 2003	Windows Server 2003 systems and later	Windows Server 2008 systems and later
Enumerate sub-keys	Deny	Deny	Deny
Query, modify, rename, or delete key	Deny	Deny	Deny
Load or unload hive to key	Deny	Deny	Deny
Enumerate values	Deny	Deny	Permit
Read, create, rename, or delete values	Deny	Permit	Permit
Enumerate sub-keys of sub-keys	Deny	Permit	Permit
Create sub-keys	Permit	Permit	Permit
Query, modify, rename, or delete sub-keys	Permit	Permit	Permit
Load or unload hive to sub-keys	Permit	Permit	Permit

Example: Provide default access of READ to a Registry Key

The following selang command provides default READ access to a registry key:

```
er REGKEY HKEY_LOCAL_MACHINE\SOFTWARE\Test\Key1 defacc(READ) owner(nobody)
```

As a result, the default access to Key 1 is as follows:

Action	Systems earlier than Windows Server 2003	Windows Server 2003 and later	Windows Server 2008 and later
Enumerate sub-keys	Permit	Permit	Permit
Read key	Permit	Permit	Permit
Modify, rename, or delete key	Deny	Deny	Deny
Load or unload hive to key	Deny	Deny	Deny
Enumerate values	Permit	Permit	Permit
Read values	Permit	Permit	Permit
Create, rename, or delete values	Deny	Permit	Permit
Enumerate sub-keys of sub-keys	Permit	Permit	Permit
Create sub-keys	Permit	Permit	Permit
Query, modify, rename, or delete sub-keys	Permit	Permit	Permit
Load or unload hive to sub-keys	Permit	Permit	Permit
Enumerate sub-key values	Permit	Permit	Permit
Create sub-key values	Permit	Permit	Permit

Example: Provide default access of NONE to a Registry Key Wildcard

The following selang command provides default access of NONE to all subkeys in a registry key:

```
er REGKEY HKEY_LOCAL_MACHINE\SOFTWARE\Test\Key1\* defacc(NONE) owner(nobody)
```

The wildcard (*) does not apply to Key1, but to all subkeys of Key1; this means that any form of access is denied to all subkeys of Key1. Access is also denied to rename or delete Key1, due to the parent protection rule.

This command permits access to the values of Key1. The access to values of subkeys of Key1 (for example values of Key1\subkey1) varies between different Windows systems:

- On Windows Server 2003 and subsequent systems, this command denies access to enumerate the values of any subkey of key1, but grants access to create, rename, delete, and read the values.
- On systems earlier than Windows Server 2003, this command denies all access to the values of subkeys of Key1.

Example: Provide default access of NONE to a Registry Value

The following selang command protects a specific registry value with access NONE on Windows Server 2003 and subsequent systems:

```
er REGVAL HKEY_LOCAL_MACHINE\SOFTWARE\TestKey\value1 defacc(NONE) owner(nobody)
```

Note: On Windows Server 2008 and subsequent systems, CA Access Control returns a value of REG_NONE if an accessor tries to access a protected registry value with access NONE. A value of REG_NONE confirms that a value is present but does not specify what the value is.

Protect File Streams

A stream is a sequence of bytes. File streams contain file data, and provide additional information about a file. For example, you can create a stream that contains keywords or metadata.

Note: File streams are only available on the NTFS file system. For more information on file streams, see the Microsoft Developer Network (MSDN) Library website.

When you create a FILE rule, CA Access Control automatically protects the default data stream for the file. For example, a rule that protects the file c:\foo.txt also governs permissions to c:\foo.txt::\$DATA. However, CA Access Control does not automatically protect any non-default data streams; for these, you have to create additional file protection rules.

To protect file streams do either *one* of the following:

- To protect a specific stream, create a file rule in the format:
`drive:\path\filename.ext:stream`
- To protect a specific stream type of a particular stream, create a file rule in the format:
`drive:\path\filename.ext:stream:type`
- To protect all streams, create a generic file rule in the format:
`drive:\path\filename.ext:*`

Example: Protect All File Streams

The following selang command creates a generic file rule that protects all the streams in the file `c:\foo.txt`:

```
er file c:\foo.txt:* owner(nobody) defaccess(none)
```

Example: Protect A Specific Stream

The following selang command creates a file rule that protects the stream *mystream* in the file `c:\foo.txt`:

```
er file c:\foo.txt:mystream owner(nobody) defaccess(none)
```

Internal File Protection

During installation, CA Access Control writes rules to protect two types of internal files:

- Internal rules—Protect configuration files, log files, and database files.
You cannot delete internal rules.
- Default rules—Protect sensitive files such as root and server certificates that you use to encrypt and authenticate communication.
You can delete default rules after installation.

Internal File Rules

Internal file rules protect configuration files, log files, and database files. Internal file rules are not visible in selang and cannot be deleted. However, you can write FILE rules to override the internal file rules. If you delete these FILE rules, CA Access Control reverts to the internal file rules.

Except for database files, files that CA Access Control protects with internal file rules have the following access rights:

- Full access for CA Access Control internal processes
- Read and execute (where relevant) access for all other accessors

Database files that CA Access Control protects with internal file rules have the following access rights:

- CA Access Control internal processes have full access to the database
- The NT AUTHORITY\System user has read access to the database
- All other accessors have no access to the database

Note: The default access rights for all other accessors were changed in r12.5 SP3. In previous releases, all other accessors had read access by default to the database files.

CA Access Control protects the following files with internal file rules. The second column of the table lists the registry subkey and entry that specifies the file location, where applicable. CA Access Control creates its registry entries under the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl

Note: Some file locations are defined internally and do not have a corresponding registry entry. You cannot configure the location of these files.

File	Registry Subkey and Entry	Default File Location
seosdrv.sys	-	%SystemRoot%\system32\drivers\seosdrv.sys
cainstrm.sys	-	%SystemRoot%\system32\drivers\cainstrm.sys
drveng.sys	-	%SystemRoot%\system32\drivers\drveng.sys
pwdchange.dll	-	%SystemRoot%\system32\pwchange.dll
SUSRAUTH.dll	-	%SystemRoot%\system32\SUSRAUTH.dll
eACSubAuth.dll	-	%SystemRoot%\system32\eACSubAuth.dll
eACPasswordFltr.dll	-	%SystemRoot%\system32\eACPasswordFltr.dll

File	Registry Subkey and Entry	Default File Location
All database files	SeOSD\dbdir	<i>ACInstallDir</i> \Data\seosdb
All help files	lang\help_path	<i>ACInstallDir</i> \Data\help
All binaries	-	<i>ACInstallDir</i> \bin
seosd.trace	SeOSD\trace_file	<i>ACInstallDir</i> \log
seos.audit	logmgr\audit_log	<i>ACInstallDir</i> \log
seos.audit.bak	logmgr\audit_back	<i>ACInstallDir</i> \log
seos.error	logmgr\error_log	<i>ACInstallDir</i> \log
seos.error.bak	logmgr\error_back	<i>ACInstallDir</i> \log
seos.msg	message\filename	<i>ACInstallDir</i> \Data
stop.ini	STOP\STOPIniFileName	<i>ACInstallDir</i> \Data
stopsignature.dat	STOP\STOPSignatureFileName	<i>ACInstallDir</i> \Data
response.ini	SeOSD\ResponseFile	<i>ACInstallDir</i> \Data
audit.cfg	logmgr\AuditFiltersFile	<i>ACInstallDir</i> \Data

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

Default File Rules

CA Access Control creates default file rules during installation to protect sensitive files. Default file rules are visible in *selang* and can be deleted.

The following table lists the sensitive files that CA Access Control protects with default file rules, and the access rights and permitted accessors for the files.

In the table, *PMDBDir* is the directory in which the policy model databases (PMDBs) reside, and *pmd_name* is the name of each policy model. By default, *PMDBDir* is located at *ACInstallDir*\Data. The location of *PMDBDir* is defined in the following registry entry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\Pmd\_Pmd_directory_
```

File	Default Access	Permitted Accessors
<i>ACInstallDir</i> \data\crypto\crypto.dat	None	sechkey
<i>ACInstallDir</i> \data\crypto\def_root.pem*	None	sechkey
<i>ACInstallDir</i> \data\crypto\sub.key	None	sechkey
<i>ACInstallDir</i> \data\crypto\sub.pem	None	sechkey

File	Default Access	Permitted Accessors
<i>ACInstallDir\log\policyfetcher.log</i>	Read	+policyfetcher
<i>PMDBDir\pmd_name</i>	Read, Chdir	-
<i>PMDBDir\pmd_name*</i>	Read, Execute	-

Chapter 5: Managing Authorization

This section contains the following topics:

[Access Authorities](#) (see page 67)

[Setting Access Authority - Examples](#) (see page 67)

[Access Control Lists](#) (see page 68)

[How Access Authority to a Resource Is Determined](#) (see page 70)

[Interaction Between User and Group Access Authorities](#) (see page 71)

[Security Levels, Categories, and Labels](#) (see page 72)

Access Authorities

The main purpose of CA Access Control is to assign and enforce access authorities, also known as access rights.

An access authority always has the following components:

- The resource that the access applies to, for example, a file, host, or terminal
- The type of access, for example read, write, delete, log in, run
- The accessor, which is either a user or a group

A user has the authority to access a resource in a certain way because one or more of the following are true:

- The user has the access authority, as granted by the resource ACL
- The user is a member of a group that has access authority.
- The user is running a program that has the access authority. For example the user has the authority to run a program in the SPECIALPGM class, or to run a command in the SUDO class.

Note: For more information about access authority by class, see the *selang Reference Guide*.

Setting Access Authority - Examples

Example: Give an internal User Read Access

The following `selang` command adds the internal user `internal_user` to the ACL of terminal `tty30`, to give read access to the terminal:

```
authorize TERMINAL tty30 access(READ) uid(internal_user)
```

Example: Give an Enterprise User Read Access

The following selang command adds the enterprise user Terry to the ACL of terminal tty30, to give read access to the terminal:

```
authorize TERMINAL tty30 access(READ) xuid(Terry)
```

Example: Change an Access Authority of an Enterprise User to a Resource

The following selang command sets Terry's access to terminal tty30 to none, and so denies Terry access:

```
authorize TERMINAL tty30 access(NONE) xuid(Terry)
```

Example: Remove the Access Authority of an Enterprise User from a Resource

The following selang command removes Terry from the ACL in the terminal tty30:

```
authorize- TERMINAL tty30 xuid(Terry) access-
```

Terry now has the default access to the terminal.

Example: Give an Enterprise User Sub-administrator Access

The following selang commands set up the enterprise user Terry as a sub-administrator with the authority to manage users and files:

```
authorize ADMIN USER xuid(Terry)  
authorize ADMIN FILE xuid(Terry)
```

Access Control Lists

The access authorities to a resource are specified in an access control list. Every resource record has at least two access control lists:

ACL

Specifies the accessors that are granted access to the resource, together with the type of access that they are granted.

NACL

Specifies the accessors that are denied authorization to the resource, together with the type of access that they are denied.

The access authority can also depend on the circumstances around the access, such as whether the user is logged in locally or not.

Conditional Access Control Lists

Conditional Access Control Lists (CACLS) provide an extension to ACLs. When an accessor attempts to access a resource, if the resource's ACL and NACL do not define an access authority for the user, CA Access Control examines the conditional access control lists.

The conditional access control lists specify access to resource where the access is by a particular method, for example by using a specified program.

For example you can use a conditional access control list to define a program pathing rule.

CA Access Control allows the following conditional access control lists:

- Program Access Control Lists (PACLs)
- TCP class access control lists
- CALENDAR class access control lists

To define an entry in a conditional access control list entry, you can use the `via` option of the `selang authorize` command.

In common with other access control lists, each entry in a conditional access control list specifies the accessors that are granted access to the resource, together with the type of access that they are granted. In addition, an entry in a conditional access control list specifies the condition under which the authority is assigned. For a PACL, the condition is the name of a program which the accessor needs to run to have the access.

Example: Using a PACL

To allow the enterprise user `sysadm1` to become superuser only by running the program `secured_su`, you can specify the corresponding conditional access rule using the following `selang` command:

```
authorize SURROGATE user.root xuid(sysadm1) via(pgm(secured_su))
```

defaccess—The Default Access Field

The record for a resource can include a default access field, `defaccess`. The value of the `defaccess` field specifies the access authority that is allowed to accessors who are not covered by any of the resource access control lists.

How Access Authority to a Resource Is Determined

When an accessor attempts to access a resource, CA Access Control checks the access authority by running through one or more checks in a pre-determined order, until it gets a result. If any check produces an access result (deny or allow access), CA Access Control does not check any further, but instead returns the result.

The order in which it runs through these checks is important. For each resource, CA Access Control checks the access records in the following order by default:

1. The resource's time based restrictions
2. The resource's ownership (owners are allowed access)
3. B1 checks
4. The resource's NACL
5. The resource's ACL
6. The resource's PACL
7. The resource's defaccess field

The order of the last two checks is determined by the setting of the accpac option. You can disable the use of resource PACL by using the selang command setoptions setpacl-

One access control list can contain more than one entry that affects a user. For example, it can contain an entry that mentions a user explicitly, and also entries for each of the groups to which the user belongs. CA Access Control checks all the possible entries at each level before it goes to the next level. For more information about how it resolves conflicting rules at each level, see [Interaction Between User and Group Access Authorities](#) (see page 71).

Example: The Resultant Permission on a File

For the following table, assume that an accessor named user1 attempts to read the resource file1.

In the following table CA Access Control is following the default setting of the accpac option to use the PACL.

Entry in NACL for user1	Entry in ACL for user1	Entry in PACL for user1	Entry in defaccess	Resulting Permission
Read	(Any)	(Any)	(Any)	Read denied
(Not defined)	None	(Any)	(Any)	Read denied
(Not defined)	Read	(Any)	(Any)	Read granted

Entry in NACL for user1	Entry in ACL for user1	Entry in PACL for user1	Entry in defaccess	Resulting Permission
<i>(Not defined)</i>	<i>(Not defined)</i>	via pgm securereader	<i>(Any)</i>	Read allowed through the securereader program
<i>(Not defined)</i>	<i>(Not defined)</i>	<i>(Not defined)</i>	Read	Read granted

Where an entry is shown as *(Not defined)*, this means that no entry for user1 exists in that access control list.

Where an entry is shown as *(Any)*, this means that the entry in that access control list does not matter, because CA Access Control does not check it.

The order that CA Access Control checks is from left to right. Notice that for all rows, the cells to the right of a cell with a defined access have the value *(any)*. Conversely all the cells to the left of a cell that contains a defined access have the value *(not defined)*.

Interaction Between User and Group Access Authorities

You can explicitly grant or deny access authorities to a user, and also to groups to which the user belongs. Sometimes these can conflict. The following example shows what results if conflicting access authorities are assigned to the same resource when a user is a member of two groups (Group 1 and Group 2).

It assumes that the [accumulative group rights](#) (see page 72) option is set (the default setting).

Access Authority for User	Access Authority for Group 1	Access Authority for Group 2	Resulting Access Authority
Access denied	<i>(Any)</i>	<i>(Any)</i>	Access denied
Access granted	<i>(Any)</i>	<i>(Any)</i>	Access granted
<i>(Not defined)</i>	Access granted	<i>(Not defined)</i>	Access granted
<i>(Not defined)</i>	<i>(Not defined)</i>	Access granted	Access granted
<i>(Not defined)</i>	Access granted	Access granted	Access granted
<i>(Not defined)</i>	Access denied	<i>(Any)</i>	Access denied
<i>(Not defined)</i>	<i>(Any)</i>	Access denied	Access denied

Where an entry is shown as *(Not defined)*, this means that no entry for the user or group is defined.

Where an entry is shown as *(Any)*, this means that the access authority does not matter, because CA Access Control does not check it.

Accumulative Group Rights (ACCGRR)

The *accumulative group rights* option (ACCGRR) affects how CA Access Control checks a resource's ACL. If ACCGRR is enabled, CA Access Control checks the ACL for the authorities granted from all the groups to which the user belongs. If ACCGRR is disabled, CA Access Control checks the ACL to see if any of the applicable entries contain the value none. If so, access is denied. Otherwise CA Access Control ignores all group entries except the first applicable one in the access control list. By default the option is enabled.

To enable the ACCGRR option, you can use the following `setlang` command:

```
setoptions accgrr
```

To disable the ACCGRR option, you can use the following `setlang` command:

```
setoptions accgrr-
```

Security Levels, Categories, and Labels

Security levels and security categories provide additional ways to restrict access to a resource, complementary to the use of access control lists.

Security labels are a means to bundle security levels and categories together, to manage them more easily.

Security Levels

A *security level* is an integer between 0 and 255 that you can assign to accessors and resources. An accessor cannot access a resource if the accessor has a security level less than the security level assigned to the resource, even if the user is granted access authority in the resource's access control list. If a resource has a zero security level, security level checking is not checked for that resource.

An accessor with a security level of zero cannot access any resource that has a non-zero security level.

Security Categories

A *security category* is the name of record in the CATEGORY class. You can assign a security category to accessors and to resources. An accessor can access a resource only if the accessor is assigned to all of the security categories assigned to the resource.

Security Labels

A *security label* is the name of a record in the SECLABEL class. A security label bundles together a security level and a set of security categories. Assigning a security label to an accessor or a resource gives the accessor or resource the combined security level and security categories associated with the security label. A security label overrides any specific security level and category assignments in an accessor or resource.

Example: Use of a Security Label High_Security

Assume High_Security is a security label that contains a security level 255 and the security categories MANAGEMENT and CONFIDENTIAL.

if you assign a user user1 to the security label High_Security, user1 has a security level of 255 and also has the security categories MANAGEMENT and CONFIDENTIAL.

Chapter 6: Protecting Accounts

This section contains the following topics:

[User Impersonation Protection](#) (see page 75)

[Setting Up the Surrogate DO Facility](#) (see page 80)

[Defining SUDO Records \(Task Delegation\)](#) (see page 81)

[Checking User Inactivity](#) (see page 87)

User Impersonation Protection

When you enable the SURROGATE class in CA Access Control, you enable user impersonation protection. User impersonation protection lets you specify that a user or group can only change their SID (security identifier) to another SID if a specific rule permits the change. This prevents a user from impersonating another user's identity if they are not authorized to do so.

Note: A security identifier is a numeric value that identifies a user or group to the operating system.

For example, you define a CA Access Control rule that prevents any user from impersonating Administrator. User Tom tries to run a program that performs some tasks as Administrator. CA Access Control does not permit the program to execute because Tom does not have permission to impersonate Administrator.

You can run user impersonation protection in two modes:

- User mode interception
- Kernel mode interception

User Mode Interception

If you enable user mode interception, CA Access Control intercepts only the impersonation requests that originate from the Windows RunAs utility. User mode interception is available on all supported Windows versions.

Note: User mode interception is enabled by default when you enable user impersonation protection, that is, when you enable the SURROGATE class.

The advantages of user mode interception include:

- CA Access Control identifies the user who made the original impersonation request.
In many Windows applications, including the RunAs utility, the NT AUTHORITY\SYSTEM user impersonates the requesting user and makes the impersonation request. User mode interception identifies the user executing the utility, not the NT AUTHORITY\SYSTEM user who makes the request. For example, if Tom executes RunAs to impersonate Administrator, the NT AUTHORITY\SYSTEM user makes the impersonation request and CA Access Control identifies Tom as the requesting user.
- CA Access Control intercepts impersonation requests only when a user executes the RunAs utility.
This minimizes performance impact.

A disadvantage of user mode interception is that CA Access Control does not intercept every impersonation request from every Windows process.

Kernel Mode Interception

If you enable kernel mode interception, CA Access Control intercepts every impersonation request from all Windows processes. Kernel mode interception is not available on all supported Windows versions.

Note: For more information about the Windows versions for which kernel mode interception is not available, see the *Release Notes*.

An advantage of kernel mode interception is that it lets you protect every impersonation request that is made on a Windows computer.

The disadvantages of kernel mode interception include:

- If the NT AUTHORITY\SYSTEM user impersonates the requesting user and makes the impersonation request, CA Access Control does not identify the user who made the original impersonation request.

For example, RunAs, ftp, and telnet requests are all made by the NT AUTHORITY\SYSTEM user. If Tom executes RunAs to impersonate Administrator, the NT AUTHORITY\SYSTEM user makes the impersonation request and CA Access Control identifies NT AUTHORITY\SYSTEM as the requesting user.

- CA Access Control intercepts every impersonation request that the OS makes as part of its normal operation, which may have a performance impact.

Although CA Access Control caches impersonation requests, the authorization engine must still authorize many impersonation events.

How CA Access Control Responds to User Impersonation Requests

Each record in the SURROGATE class defines restrictions that protect a user from impersonation attempts. CA Access Control treats an impersonation request as an abstract object that can only be accessed by authorized users. A record in the SURROGATE class represents each user or group who has surrogate (impersonation) protection.

When a user or group makes a request to impersonate another user or group, CA Access Control does the following:

1. Checks the access authority of the SURROGATE record for the user or group. Depending on the SURROGATE record, *one* of the following happens:
 - The SURROGATE record for the user or group specifically permits or denies the impersonation.

CA Access Control uses the access authority of the SURROGATE record to permit or deny the impersonation request.
 - The user or group does not have a SURROGATE record.

The process goes to Step 2.
2. Checks the access authority of the default SURROGATE record for the user or group, as follows:
 - If the requester is a user, CA Access Control gives the user the access type that is defined in the USER._default SURROGATE record.
 - If the requester is a group, CA Access Control gives the user the access type that is defined in the GROUP._default SURROGATE record.

Note: The default access authority of the USER._default, GROUP._default, and _default SURROGATE records is read. This means that CA Access Control permits any request to impersonate a user or group, unless a SURROGATE record for the user or group prohibits the impersonation request. To change this behavior, change the access authority of the USER._default and GROUP._default records. You can also set the same default for users and groups by changing the access authority of the _default SURROGATE record.

Enable User Impersonation Protection

User impersonation protection lets you set rules to permit or deny requests to impersonate specific users and groups.

To enable user impersonation protection

1. (Optional) Enable kernel mode interception, as follows:

- a. Stop CA Access Control.
- b. Change the value of the following registry value to 1:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\
SeOSD\SurrogateInterceptionMode
```

- c. Restart CA Access Control.

Note: User mode interception is enabled by default.

2. Open a `seimg` command prompt window.
3. Enable the SURROGATE class:

```
setoptions class+(SURROGATE)
```

4. Define `seimg` rules for SURROGATE records for your CA Access Control implementation.

5. (Kernel mode interception only) Define a rule that lets the SYSTEM user impersonate the user that makes the impersonation request:

```
auth SURROGATE USER.Administrator uid("NT AUTHORITY\SYSTEM") acc(R)
```

Windows identifies many utilities and services (for example, Run As) as user "NT AUTHORITY\SYSTEM" and not as the user running the utility. You must define a rule for the SYSTEM user to let users who run these utilities impersonate another user.

Example: Permit Any Impersonation Request

The following `seimg` rule lets any user impersonate another user, unless a record in the database explicitly prevents the impersonation:

```
editres SURROGATE _default defaccess(READ)
```

Example: Prevent Impersonation of a Specific User

The following selang rule prevents any user impersonating Administrator, unless a record in the database explicitly permits the user impersonation:

```
newres SURROGATE USER.Administrator defaccess(NONE)
```

Example: Permit a Group to Impersonate a User

The following rule permits members of the Administrators group to impersonate Administrator:

```
authorize SURROGATE USER.Administrator gid("Administrators")
```

Setting Up the Surrogate DO Facility

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

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 starts the "Print Spooler" service as if the user were System, enter the following selang command:

```
newres SUDO StartSpooler data("net start spooler")
```

This newres command defines StartSpooler as a protected action that some users may receive System authority to perform.

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

In addition, users can be authorized to perform the StartSpooler action by using the authorize command. For example, to allow the user *operator1* to start the "Print Spooler" service, enter the following selang command:

```
authorize SUDO StartSpooler 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 starting the "Print Spooler" service, enter the `selang` command:

```
authorize SUDO StartSpooler uid(operator2) access(None)
```

Executing the `sesudo` utility performs the protected action. For example, the user `operator1` would start the "Print Spooler" service using the following command:

```
sesudo -do StartSpooler
```

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 `StartSpooler` action and then invokes the command "net start spooler" with System credentials.

Note: For more information about the `sesudo` utility, see the *Reference Guide*.

Defining SUDO Records (Task Delegation)

A record in the SUDO class stores a command script so that users can run the script with borrowed permissions. The ability to borrow permissions is tightly controlled by the SUDO record, as well as by the `sesudo` command that executes the scripts.

Note: If you create a SUDO record for an interactive Windows application, you must set the interactive flag for the SUDO record. If you do not set the interactive flag, the application runs in the background and you cannot interact with it. For more information, see the *Troubleshooting Guide*.

In a SUDO record, the comment property is used for a special purpose, and often it is known by its alternate name: the data property.

The comment property's value is the command script, with the optional addition of one or more script parameter values that are to be prohibited or permitted. The entire comment property value must be enclosed in single quotes, and executables should be referenced by their complete path names in order to prevent Trojan horses from taking their place.

This is the format for the comment property:

```
comment('cmd[;[prohibited-values][;permitted-values]]')
```

Because the lists of prohibited and permitted values are optional, a simple comment property value can be the following:

```
newres SUDO NET comment('net use')
```

The simple value in the command means that the command `sesudo NET` will execute the command `'net use'`. No particular script parameter values are prohibited; all are permitted.

Wildcards and powerful variables give you flexibility in specifying prohibited and permitted parameters. The wildcards you can use are the standard Windows wildcards. Prohibited and permitted parameters can also contain the following variables:

Variable	Description
\$A	An alpha value
\$G	An existing CA Access Control group name
\$H	(UNIX only) A parameter that starts with the user's home directory
\$N	A numeric value
\$O	The CA Access Control name of the user running <code>sesudo</code>
\$U	An existing CA Access Control user name
\$e	An empty entry. Use this to specify a SUDO command with no parameters for the rule.
\$f	An existing file name
\$g	An existing Windows group name
\$h	An existing host name
\$r	An existing file with Windows read access
\$u	An existing Windows user name
\$w	An existing file with Windows write access
\$x	An existing file with Windows execute access

If you append a list of *prohibited* parameter values to the script:

- Separate the script from the prohibited parameter values with a semicolon, but keep them all inside the single quotes. For example, if you want to prevent the user from using `-start` but you permit the user to use all other parameters, enter the following command:

```
newres SUDO scriptname comment('cmd;-start')
```

where *cmd* represents your script.

Alternatively, if you do not allow any parameter values, but rather want all parameters defaulted, define the SUDO record as follows:

```
newres SUDO scriptname comment('cmd;*')
```

- If a script parameter has more than one prohibited value, use the space character as a separator. For example, if you want to prevent the user from using `-start` and `-stop` but you permit the user to use all other parameters, enter the following command:

```
newres SUDO scriptname comment('cmd;-start -stop')
```

- If more than one script parameter has prohibited values, use the pipe character (`|`) as a separator between sets of prohibited values. For example, if you want to prevent the user from using `-start` and `-stop` for the script's first parameter and from using any existing Windows user name for the second parameter (see the previous list of variables), enter the following command:

```
newres SUDO scriptname comment('cmd;-start -stop | $u')
```

If the script has more parameters than you list, then your last set of prohibited parameters applies to all the remaining parameters.

If you append a list of *permitted* parameter values to the script,

- The `sesudo` utility checks that the parameter values:
 - Do *not* match any of the corresponding *prohibited* values.
 - Match at least one of the corresponding *permitted* values.

This means that if a parameter value is in the prohibited list, it will not be permitted even if it is also specified in the permitted list.

- Separate the list of *permitted* values from the list of *prohibited* values with a semicolon, but keep them all inside the single quotes. Even if you have no list of prohibited values, you still need the semicolon; otherwise what you intend to permit will be prohibited. For example, if you want to allow only the value `NAME` as a parameter value for the script, enter the following command:

```
newres SUDO scriptname comment('cmd;;NAME')
```

- Just as in the other list,
 - If a script parameter has more than one permitted value, use the space character as a separator.
 - If more than one script parameter has permitted values, use the pipe character (`|`) as a separator between sets of permitted values.

For example, if you have two parameters, and the first must be numeric but must not be a Windows user name, and the second must be alphabetic but must not be a Windows group name, enter the following command:

```
newres SUDO scriptname comment('cmd;$u | $g ;$N | $A')
```

If the script has more parameters than you list, then your last set of permitted parameters applies to all the remaining parameters.

Thus, the overall format for the comment property is this: first the script; then the prohibited values, parameter by parameter; then the permitted values, parameter by parameter:

```
comment('cmd; \  
param1_prohib1 param1_prohib2 ... param1_prohibN | \  
param2_prohib1 param2_prohib2 ... param2_prohibN | \  
...  
paramN_prohib1 paramN_prohib2 ... paramN_prohibN ; \  
param1_permit1 param1_permit2 ... param1_permitN | \  
param2_permit1 param2_permit2 ... param2_permitN | \  
...  
paramN_permit1 paramN_permit2 ... paramN_permitN')
```

The `sudo` utility checks each parameter entered by the user in the following manner:

1. Test if parameter N matches permitted parameter N. (If permitted parameter N does not exist, the last permitted parameter is used.)
2. Test if parameter N matches prohibited parameter N. (If prohibited parameter N does not exist, the last prohibited parameter is used.)

If all the parameters match permitted parameters, and none match prohibited parameters, `sudo` executes the command.

Example: Set Up Task Delegation that Permits a User to Run `net send`

The following procedure shows you how you let user Takashi execute the `net send` command and prevent him from executing the `net start` command:

1. In CA Access Control Endpoint Management click the Users tab, then click the Authorization and Delegation subtab.

The Authorization and Delegation menu options appear on the left.

2. Click Task Delegations.

The Task Delegations page appears.

3. Click Create Task.

The Create Task page appears.

4. Complete the dialog fields as follows:

Field	Value
Name	NET
Data	net;start;send *
Owner	nobody
Default Access	None (option cleared)
Authorized Accessors	USER: Takashi Allow: Execute

Click Save.

The new task delegation (SUDO) record is created.

5. Test the task delegation rule:

a. Log in as Takashi.

b. Open the command prompt and execute the following:

```
sesudo -do NET start
```

The following message appears:

```
sesudo: you are not allowed to use 'start' as parameter number 1.
```

Note: *net start* will not execute because it was defined as a prohibited value.

c. Execute the following value:

```
sesudo -do NET send comp message
```

The command should execute.

Example: Authorize a User to Execute Privileged Operations using an Interactive Application

A user can perform highly privileged operations using any snap-in MSC module, as the following example shows:

1. In CA Access Control Endpoint Management click the Users tab, then click the Authorization and Delegation subtab.

The Authorization and Delegation menu options appear on the left.

2. Click Task Delegations.

The Task Delegations page appears.

3. Click Create Task.

The Create Task page appears.

4. Complete the dialog fields as follows:

Field	Value
Name	services
Data	c:\winnt\system32\mmc.exe
Owner	nobody
Options	Interactive (option selected)
Default Access	None (option cleared)
Authorized Accessors	USER: Tori Allow: Execute

Click Save.

The new task delegation (SUDO) record is created. The Interactive option provides the desktop user interface that can be used by whoever is logged in when the service is started. This is available only if the service is running as a LocalSystem account.

5. Test the task delegation rule:
 - a. Log in as Tori.
 - b. Open the command prompt and execute the following:

```
sesudo -do services
```
 - c. mmc.exe will start.

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.

In `selang`, use the following command to specify inactivity globally:

```
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:

```
editgrp 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:

```
editusr userName inactive (numdays)
```

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

```
editusr userName resume
```

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

```
editgrp userName resume
```

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

```
setoptions inactive-
```

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

```
editgrp groupName inactive-
```

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

```
editusr userName inactive-
```


Chapter 7: Managing User Passwords

This section contains the following topics:

[Managing Password and Lockout Policies](#) (see page 89)

[Configure Password Quality Checking](#) (see page 90)

[Resolving Error Messages](#) (see page 91)

Managing Password and Lockout Policies

Passwords are the most popular device for authentication, but password protection methods have well-known problems: trivial passwords are easy to guess; passwords that last for years and cyclic passwords are eventually broken; and passwords sent in clear text over a network can be trapped by listeners.

Windows has a set of password rules and policies that force users to use passwords that avoid most of these common pitfalls. CA Access Control has additional rules that ensure that users select even more secure passwords.

You can specify the following rules in CA Access Control:

- A new password cannot match previous passwords. The number of previous passwords that CA Access Control stores is specified in the password policy.
- A new password cannot contain the user name.
- A new password cannot contain the password that it is replacing.
- A new password cannot match the password that it is replacing. CA Access Control disregards letter case.
- A new password must have at least the minimum number of alphanumeric characters, special characters, digits, lowercase characters, and uppercase characters specified in the password policy.

- A new password must not have more repetitive characters than is specified in the password policy.
- A new password cannot be one of the restricted words in the dictionary included in CA Access Control. The dictionary is specified in the Dictionary value in the registry subkey:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\passwd
```

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. With frequently changed passwords, they could overflow the password history stack and then re-use a previous password.

Configure Password Quality Checking

To configure password quality checking

1. In CA Access Control Endpoint Management click the Configuration tab.

The configuration menu options appear on the left.

2. Click Class Activation in the Miscellaneous section options.

The Class Activation page appears.

3. Select PASSWORD in the User Identity Control section, and click Save.

This activates password quality checking.

4. Click User Password Policy in the Policies section options.

The User Password Policy page appears.

5. Define the rules to be used for the password checks, and click Save.

The rules you define for password checks are now enforced when passwords are changed.

6. (UNIX only) Update the new passwords by using the `sepass` utility.

Note: For more information about `sepass` utility, see the *Reference Guide*.

Example: Define Password Checking Rules

The following `setlang` commands activate password quality checking and define password rules that enforce a minimum of:

- Six alphanumeric characters
- Three lowercase characters
- Two numeric characters

```
setoptions class+ (PASSWORD)  
setoptions password(rules(alpha("6") lowercase("3") numeric("2")))
```

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

Resolving Error Messages

If you are setting passwords for users on Windows NT systems, the following message may appear:

The password is shorter than required.

This error means that the password does not meet the policy requirements. This is caused by any of the following:

- The password is shorter or longer than the required length.
- The password has been used recently and exists in the Windows NT Change History field.
- The password does not have enough unique characters.
- The password does not meet other password policy requirements (such as those set with CA Access Control password policies).

To avoid this error, make sure you set a password which meets all applicable requirements.

Chapter 8: Monitoring and Auditing

This section contains the following topics:

- [Security Auditors](#) (see page 93)
- [Events Interception](#) (see page 94)
- [Monitoring Access Control Activity](#) (see page 100)
- [What CA Access Control Audits](#) (see page 101)
- [The Auditing Process](#) (see page 111)
- [Viewing Audit Events](#) (see page 115)
- [The Audit Log](#) (see page 119)

Security Auditors

One of the most important tasks of security auditors and system administrators is auditing or monitoring system activity to detect suspicious or malicious activity. Security auditing plays an essential role in a secure environment, and the security auditing features in CA Access Control include the following:

- Providing a reliable indication of who has accessed the system, what resources have been accessed, how the resource has been accessed (for example, read a file), and when resources have been accessed
- Notifying and alerting appropriate users in case of an attempted security breach, even if the attempt failed
- Indicating what changes have been made to the security rules, and by whom
- Providing a means to test the effect of access rules before they are enforced

CA Access Control auditing is modeled after real-world auditing: security auditors act independently of system and security administrators, although you can change your implementation so that this is not the case if some other model is more appropriate for your environment.

A security auditor is a user to whom the AUDITOR attribute is assigned. Users defined as security auditors are permitted to perform auditing tasks such as changing the audit rules that are assigned to users and resources. They are also authorized to use the CA Access Control auditing utilities without being required to have the ADMIN attribute.

Events Interception

CA Access Control intercepts an event if the following two conditions are met:

- The appropriate class is active.
- A rule anticipating this event exists in the database.

For example, you can use the following generic rule to audit all file accesses to files that reside in c:\data\payroll:

```
newres FILE c:\data\payroll\*
```

You also need to make sure that the FILE class is active (the default).

Types of Intercepted Events

CA Access Control intercepts two types of events:

- Interception Events
Information from an interception event is cached as part of the process for future use by an audit event.
- Audit Events

Interception Modes

Based on the interception mode, CA Access Control intercepts, checks for authorization, and logs audit records of access request events. CA Access Control has the following modes of interception:

- Full Enforcement mode
- Audit Only mode
- No Interception mode

Note: Warning mode (see definition on page 95) is not an interception mode; it works in Full Enforcement mode only and is designed for short term use during implementation.

Audit Only Mode

Audit Only mode records all intercepted events without checking or enforcing access rules. Use this mode to collect data for compliance requirements or regulations. In Audit Only mode, CA Access Control intercepts the event and writes an audit event but does not process the request for authorization and does not enforce rules. As a result, CA Access Control permits all access requests it intercepts. This means that the authorization result recorded in the audit log for all events is *P* (permitted).

The following restrictions apply to Audit Only mode:

- No audit records are sent to Unicenter.
In Audit Only mode all events are permitted (*P*). Permitted events are not sent to Unicenter.
- The audit properties of the resource and the user are *not* taken into consideration.
Audit Only mode records *all intercepted* events regardless of resource- or user-specific settings.

Set Up Audit Only Mode

Audit Only mode records all intercepted events without checking or enforcing access rules. Use this mode to collect data for compliance requirements or regulations.

To set up Audit Only mode, set the SeOSD\GeneralInterceptionMode CA Access Control registry entry to 1.

Important! If you use Audit Only mode, make sure that you have enough disk space for the audit logs and that the size limit of the audit log is large enough. You should also consider options for [audit log backup](#) (see page 124).

Warning Mode

Warning Mode is a property that you can apply to a resource, and an option that you can apply to a class. If Warning mode is applied to a resource or a class and an access violates an access rule, CA Access Control writes an audit log entry with the return code W, but permits the access to the resource. If a class is in Warning mode, all the resources in that class are in Warning mode.

Warning Mode only has an effect if CA Access Control is in Full Enforcement mode.

Note: Full Enforcement mode is the only mode CA Access Control for UNIX supports. CA Access Control for Windows also supports Audit Only mode.

You can use Warning mode when you introduce or modify an access policy. If you do this, you can examine the audit log to preview the results of your intended policy before you put that policy into effect. You can display the audit log by using the `seaudit` command.

If a class has the property *warning*, you can put the class into Warning mode. If a resource group or class is in Warning mode, when an access rule is violated, CA Access Control allows the access and writes an entry in the audit log that references the resource (not the resource group or class).

The Warning mode settings on a resource and on a class are independent: if you put a resource into Warning mode, it remains in Warning mode, even if it belongs to a class and you remove Warning mode from that class.

Note: You can only put resources or classes into Warning mode if they have the property *warning*; not all resources or classes have this property.

More information:

[Audit Only Mode](#) (see page 94)

Put a Resource into Warning Mode

You put a resource into Warning mode to monitor the effects of access rules, without needing to enforce these rules.

Note: As well as putting individual resources into Warning mode, you can [put a class into Warning mode](#) (see page 97).

To put a resource into Warning mode

1. In CA Access Control Endpoint Management edit the resource you want to put into Warning mode.

The appropriate Modify page appears.

2. Click the Audit tab.

The Audit Modes page for the resource appears.

3. Select Warning Mode, and click Save.

The resource you modified is now in Warning mode.

Note: In Warning mode, CA Access Control always writes warning records to the audit log when access is permitted but access rules are violated: you do not need to set the audit property on the resource for this to happen.

Use the `sereport` utility (report number 6) to see all resources in Warning mode.

Example: Put a File into Warning Mode

The following `selang` example puts the file `c:\myfile` into Warning mode:

```
chres FILE c:\myfile warning
```


Example: Clear Warning Mode from a File

The following selang example takes the file c:\myfile out of Warning mode:

```
chres FILE c:\myfile warning-
```

Warning mode is now not active for the myfile, so CA Access Control enforces the access rules for myfile.

Example: Put a Terminal into Warning Mode

The following selang example puts the terminal myterminal into warning mode:

```
chres terminal myterminal warning
```

CA Access Control permits access by any authorized user from the terminal myterminal, but logs an audit record for any user that normally would be denied access from that terminal.

Put a Class into Warning Mode

Rather than putting individual records into Warning mode, you can put all records in a class into Warning mode. You might use Warning mode to monitor the effects of access rules, without needing to enforce these rules.

To put a class into Warning mode

1. In CA Access Control Endpoint Management, do as follows:
 - a. Click Configuration.
 - b. Click Class Activation.

The Class Activation page appears.

2. Select the check box in the Warning column for the class you want to put into Warning mode.
3. Click Save.

A confirmation message appears, letting you know that CA Access Control options have been successfully updated.

Find Out Which Resources Are in Warning Mode

You should use Warning mode as a temporary measure when implementing CA Access Control. Once you are comfortable that users have the required access to the resources they require, you should turn off Warning mode and CA Access Control will start enforcing the associated rules.

To find out which resources are in Warning mode, you can create a report that shows all resources with Warning mode.

To create a report, enter the following command:

```
sereport -f pathname.html -r 6
```

CA Access Control creates the report.

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

Find Out Which Classes Are in Warning Mode

You should use Warning mode as a temporary measure when implementing CA Access Control. Once you are comfortable that users have the required access to the resources they require, you should turn off Warning mode and CA Access Control will start enforcing the associated rules.

To find out which classes are in Warning mode, you can get CA Access Control to display this data.

To display this data, enter the following selang command:

```
setoptions cwarnlist
```

CA Access Control displays a table showing the classes that are in Warning mode.

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

How to Perform System Maintenance

At certain times you may need to perform system maintenance to upgrade the system, install a new application, and so on. During system maintenance you should set CA Access Control rules in Warning mode. Once you are comfortable that the maintenance did not affect user access to resources that they require, you should turn off Warning mode and CA Access Control will start enforcing the associated rules.

To use Warning mode when you perform system maintenance, do the following:

1. Set the appropriate classes to Warning mode before you start the maintenance, using the following selang rule:

```
setoptions class(NAME) flags(W)
```

2. Perform the maintenance.
3. Run the seretrust utility after you perform the maintenance.

The seretrust utility generates the selang commands required to retrust programs and secure files defined in the database.

4. Run the selang command to retrust the programs defined in the database.
5. Remove the Warning mode from the classes to enable policy enforcement, using the following selang rule:

```
setoptions class(NAME) flags-(W)
```

6. Review CA Access Control audit log files.

The audit log contains warnings for the resources that were affected by the maintenance.

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

Monitoring Access Control Activity

The CA Access Control trace is a real-time log that can show every action taken by CA Access Control. Trace records are accumulated in *ACInstallDir*\log\seosd.trace (where *ACInstallDir* is the directory where you installed CA Access Control).

Or they are accumulated in whatever file you specify as the *trace_file* value in the registry subkey:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\SeOSD\
```

Although you can filter the records from the trace file, the trace mechanism is designed for system monitoring and not for security auditing.

By default, CA Access Control only generates trace messages during CA Access Control initialization. Once CA Access Control is initialized, it stops the trace mechanism and trace messages are not generated.

Trace Record Filters

CA Access Control generates two types of trace records:

- User trace records—Record actions completed by the user, for example, user1 accessed file c:\tmp\tmp.exe.
- General trace records—Record actions completed by the system, for example, the Watchdog set a program to be non-trusted.

Trace records are written to the seos.trace file, and can be filtered using the trcfilter.ini file.

If you set a user to be traceable, each time a trace record is written for that user, a matching audit record is written to the seos.audit file. Audit records are filtered by the audit.cfg file.

Note: Audit records generated by trace events are not cached, and always go through the full enforcement flow.

The following `selang` command sets a user to be traceable:

```
editusr userName audit(trace)
```

To view trace or audit records, use the `seaudit` utility.

Filtering Trace Records

Using a trace filter file, you can specify that certain types of activity should not appear in the trace file. The trace filter file is specified with the *trace_filter* value in the registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\SeOSD
```

The default value is *ACInstallDir*\log\trcfilter.ini (where *ACInstallDir* is the directory where you installed CA Access Control).

Important! CA Access Control creates the trace filter file at installation with the single line: **seosd.trace**. Never delete this record.

Each line in the trace filter file represents an access or an activity that should *not* be traced. For example, to eliminate tracing the users' access to Microsoft Word, add the following line to the trace filter file:

```
*winword.exe*
```

What CA Access Control Audits

For security auditing, CA Access Control keeps audit records for intercepted events according to the audit rules defined in the database and the enforcement mode it operates in. The records in the audit log accumulate according to these audit rules.

Full auditing provides audit records for all intercepted events of any of the following:

- File access (FILE class)
- Program execution (PROGRAM class)
- Registry access (REGKEY and REGVAL classes).
- Impersonation control (SURROGATE class)
- Network control (CONNECT, TCP, HOST, GHOST, HOSTNET, and HOSTNP classes)
- Log in (TERMINAL class)
Note: Intercepted login events are not cached; they always follow the auditing process for interception events.
- Service protection (WINSERVICE class)
- Password verification failure (PASSWORD class)
- Process termination (PROCESS class)

The decision whether to log an event depends on the CA Access Control interception mode.

Login Interception Limitations

Login interception on Windows is supported only by CA Access Control sub-authentication method.

You cannot set login interception through the kernel. As a result, you should consider the following:

- Since the sub-authentication component works on the Domain Controller (DC) level, and it is up to the OS to decide which DC authenticates the user's login events (and triggers the CA Access Control sub-authentication module), in a Windows domain environment, CA Access Control needs to be installed on every DC.
- When working in a Windows domain environment, CA Access Control login policy (TERMINAL rules) need to be located on the DCs and not necessarily on the target server.

For example, if you would like to protect or audit login events made by domain users on a file server, which is part of the Windows domain but is not a DC, the CA Access Control login policy needs to be defined on the DC and not on the target file server. This is because when a domain user accesses the shared file directory, a login authorization occurs on the DC, not the file server.

- When there is more than one DC, CA Access Control login authorization could be processed on any one of the DCs. As a result, we recommended you synchronize CA Access Control login policy between all DCs.

You can implement this through either the Policy Model mechanism, where all DCs are subscribers to a PMDB, or by adding all DCs into a host group and deploying a common policy using advanced policy management.

- Some user properties, which correspond to login events, are updated at runtime-during event authorization. These properties might be out-of-sync because the login authorization happens only on one of the DCs. These properties are *Gracelogins*, *Last accessed*, and *Last access time*.

That said, it is possible that, for example, the user's property *Last access time* value will be different between DCs because CA Access Control sub-authentication was triggered on one of the DCs, not on all of them.

- To enforce local users (that is, not domain users) login events, CA Access Control needs to be installed on the local computer that the local user needs access to. This is because the local computer is used as the domain computer (the domain is the local computer).
- Remote Desktop Protocol (RDP)/Terminal Services login events are enforced on the target server as it was in previous CA Access Control versions. However, for RDP login events, CA Access Control login policy should be defined on the target server.

What CA Access Control Audits in Full Enforcement Mode

In Full Enforcement mode (regular operation), CA Access Control logs events as follows:

- If Warning mode is turned *off* for the intercepted resource, CA Access Control enforces rules and logs the events based on the *audit* property of the resource or user.

Audit Property	Events Logged
ALL	All
SUCCESS	Access permitted
FAIL	Access denied

- If Warning mode is turned *on* for the intercepted resource, a record is written to the audit log if an access request violates an access rule (if the rules were enforced, the request would have failed). The audit record mentions that the violation was permitted because Warning mode is in effect.

Rules are not enforced in this mode.

What CA Access Control Audits in Audit Only Mode

In Audit Only mode, CA Access Control does not process requests for authorization or enforce rules. All intercepted login events for the accessor and all intercepted events for resources protected by CA Access Control are logged, regardless of whether access failed or succeeded.

How to Change What CA Access Control Writes to the Audit Log

You can change what CA Access Control writes to the audit log in two ways:

- Use the AUDIT property of the resources or accessors to define the audit events that CA Access Control writes to the audit log.

Note: You can use the AUDIT property for a GROUP or XGROUP to set the audit property for all members of the group. However, you cannot use the AUDIT property to set the audit mode for group members if a user's audit mode is defined in a USER record, XUSER record, or profile group.

- Use the audit configuration file `audit.cfg` to filter the events CA Access Control sends to the audit log. You cannot use the `audit.cfg` file to add events to the audit log.

To reduce the number of audit records, you can also control consecutive audit events written to the log file. This customization is based on time interval between consecutive matching audit events (that is, an access to a resource with the same process ID, thread ID, rule ID, user ID, and access mask). The time interval, in seconds, can be set by setting the value of the AuditRefreshPeriod registry entry. By default, the AuditRefreshPeriod is set to zero (0), which means that all events are written to the log file.

Setting Audit Rules

For security auditing, CA Access Control keeps audit records for events of access denial or access grants according to the audit rules defined in the database.

Every accessor and resource has an AUDIT property that can be set to one or more of the following values:

FAIL

Logs access failures by the accessor to the resource.

SUCCESS

Logs successful accesses by the accessor to the resource.

LOGINFAIL

Logs every logon failure by the accessor. (This value does not apply to resources.)

LOGINSUCCESS

Logs every successful logon by the accessor. (This value does not apply to resources.)

ALL

Logs the same information as FAIL, SUCCESS, LOGINFAIL, and LOGINSUCCESS for accessors or FAIL and SUCCESS for resources.

NONE

Logs nothing concerning the accessor or resource.

Whenever you create or update an accessor or resource record in the database, you can specify the AUDIT property. You can also specify whether email notification of logged events should be sent and to whom.

The records in the audit log accumulate according to these audit rules. The decision whether to log an event is based on the following:

- If the resource or accessor has AUDIT(ALL), all login events for the accessor and all events concerning resources protected by CA Access Control are logged, regardless of whether access failed or succeeded.
- If access to a resource protected by CA Access Control is successful and the accessor or resource has AUDIT(SUCCESS), the event is logged.
- If access to a resource protected by CA Access Control fails and the accessor or resource has AUDIT(FAIL), the event is logged.

In addition, if you set a user to be traceable, each time a trace record is written for that user, a corresponding audit record is written to the audit log.

Defining the Audit Events That CA Access Control Writes to the Audit Log

CA Access Control writes access success and failures to the audit log. You define which access events CA Access Control writes to the audit log, by changing the value of the AUDIT property for the resource or accessor that you want to audit. You can also use this method to specify that CA Access Control logs every trace event to the audit log.

You use the AUDIT property to specify the audit events that CA Access Control writes to the audit log. Use `selang` or CA Access Control Endpoint Management to set the AUDIT property for resources and accessors as follows:

Value of AUDIT	What CA Access Control Logs	Applicable Objects
FAIL	Access failures	Users and resources
SUCCESS	Access successes	Users and resources
LOGINFAIL	Login failures	Users
LOGINSUCCESS	Login successes	Users
ALL	Equivalent to FAIL, SUCCESS, LOGINFAIL, LOGINSUCCESS, INTERACTIVE	Users and resources
TRACE	Equivalent to ALL plus all system events	Users
INTERACTIVE	User sessions on UNIX computers	Users
NONE	No logging	Users and resources

Note: If the audit property of a user is not set, the AUDIT value of a group or profile group can affect the audit mode CA Access Control uses for the user.

How CA Access Control Determines the Audit Mode for a User

The audit mode for a user specifies which audit events CA Access Control sends to the audit log for that user. The following process describes how CA Access Control determines the audit mode for a user:

1. CA Access Control checks if the user's record in the USER or XUSER class has a value for the AUDIT property.

If the user's record has a value for the AUDIT property, CA Access Control uses that value as the audit mode for the user.

2. CA Access Control checks if the user is assigned to a profile group. If the user is assigned to a profile group, CA Access Control checks if the profile group's record in the GROUP class has a value for the AUDIT property.

If the user is assigned to a profile group and the profile group's record has a value for the AUDIT property, CA Access Control uses that value as the audit mode for the user.

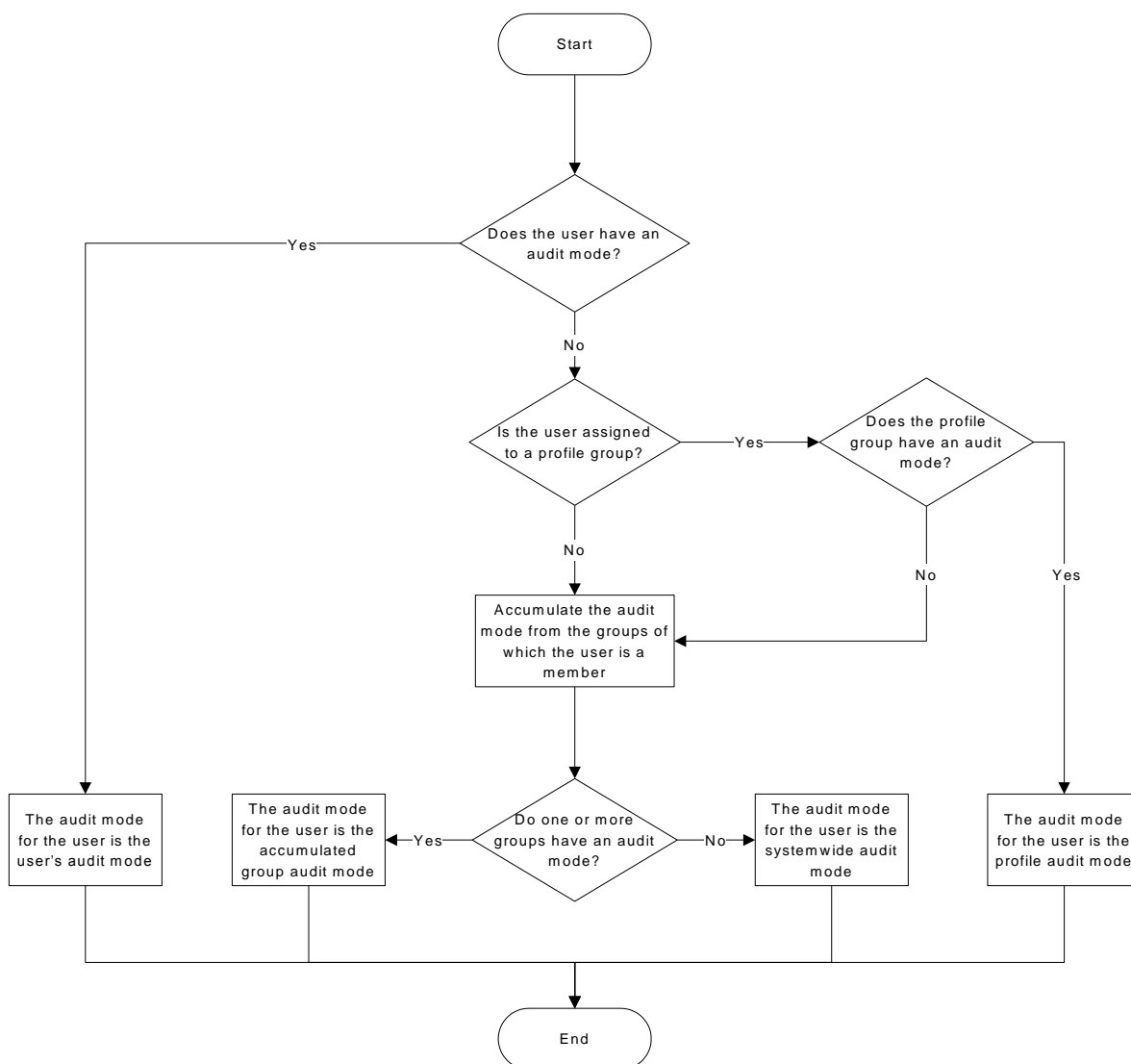
3. CA Access Control checks if the user is a member of a group. If the user is a group member, CA Access Control checks if the group's record in the GROUP or XGROUP class has a value for the AUDIT property.

If the user is group member and the group's record has a value for the AUDIT property, CA Access Control uses that value as the audit mode for the user. If the user is not a member of a group, or if the group's record does not have a value for the AUDIT property, CA Access Control assigns the systemwide audit mode to the user.

Note: The user's audit mode accumulates if a user is a member of more than one group and the groups have different audit modes. The audit mode for the user is the sum of all the audit modes for the groups of which they are members.

Note: If CA Access Control uses the value of a group's AUDIT property to determine the audit mode for a user, and you change the group's audit mode while the user is logged in, the audit mode for the logged-in user also changes. The user does not have to log off for the change in group audit mode to take effect.

The following diagram shows how CA Access Control determines the audit mode for a user:



Example: Audit by Groups

User Jan is a member of Group A and Group B. Group A has an audit mode of FAIL and Group B has an audit mode of SUCCESS. Because Jan is a member of both groups, Jan has the accumulated audit mode of FAIL and SUCCESS.

More information:

[How CA Access Control Uses Profile Groups to Determine User Properties](#) (see page 39)

Default Audit Modes for Users and Enterprise Users

When you create a user (USER object), CA Access Control assigns the default AUDIT_MODE to the object. The default value of the AUDIT_MODE property is Failure, SuccessLogin, SuccessFailure.

When you create an enterprise user (XUSER object), by default CA Access Control does not assign a default AUDIT_MODE value to the object.

Note: (UNIX) To change the default value of the AUDIT_MODE property for USER objects, edit the value of DefaultAudit in the [newusr] section of the lang.ini file.

Change to Default Audit Value for Some Users

Before r12.0 SP1 CR1, the default audit mode was None for the following accessors:

- Users that do not have a defined AUDIT value in their corresponding USER class record, and that are not associated with a profile group that has a defined AUDIT value.
- Any user that is not defined in the database (represented by the _undefined user record).

Note: If you use enterprise users, CA Access Control does not consider any users as undefined. Properties of the _undefined user are not relevant in this case.

From r12.0 SP1 CR1, the default audit mode for these accessors is Failure, LoginSuccess, and LoginFailure. To retain earlier behavior, set the value of the AUDIT property to None for these users.

Changing the Value of AUDIT Property for GROUP Records

If you have a GROUP record that has two functions:

- A profile that defines an audit policy for one set of users
- A container for a second set of users

From r12.0 SP1 CR1 onwards, the GROUP record also defines the audit policy for the second set of users. To avoid problems that this behavior change may cause, create a separate GROUP for the second set of users.

Setting Audit Policies in Windows

In addition to setting access rules for accessors and resources, you can specify Windows events that you want to write to the audit log. You can specify such audit policies for the entire organization, on a group basis, on a profile group basis, or on a user-by-user basis.

Example: Set an Audit Policy for All Members of a Profile Group

The following example shows you how you can set an audit policy for all users that are part of profile group:

1. Create a new profile group with the audit mode you require. For example:

```
newgrp profileGroup audit(failure) owner(nobody)
```
2. Create a new user and attach it to the profile group you created. For example:

```
newusr user1 profile(profileGroup) owner(nobody)
```
3. Remove the user's audit setting. For example:

```
chusr user1 audit-
```

You can now check whether this setting is effective:

1. Log on as the new user:

```
runas /user:user1 cmd.exe
```
2. From user1's command prompt window, enter the following:

```
secons -whoami
```

This command displays the information that is used for authorization and is held in the ACEE for user1.

```
ACEE audit mode is: Failure; Originated from Profile group definition
```

This message confirms that the audit policy is derived from the profile group the user is attached to.

Example: Set a Audit Policy for Group Members

In this example, a fictional company named Forward Inc wants to use CA Access Control to protect all files in the /production directory. The /production directory has full access permissions in the native environment.

Forward Inc wants to deny and audit any attempts to access the /production directory. However, Forward Inc permits read access to the /production directory for developers. This access is not audited. An attempt by a developer to write to the /production directory is denied and audited.

Developers can request full access to the /production directory. Forward Inc audits any activity that a user with full access performs in the /production directory.

The following process describes the steps Forward Inc takes to implement the previous scenario:

1. Create a group named Developers in the native environment. Join all the developers to this group.
2. Create a group named Dev_Access_All in the native environment. Do not join any users to this group.
3. Define a generic access rule for the /production directory, as follows:

```
authorize FILE /production/* access(none) uid(*)
```

This rule sets the default access as none.
4. Define a generic audit rule for the /production directory, as follows:

```
editres FILE /production/* audit(failure)
```

This rule audits any failed attempt to access the /production directory.
5. Define an access rule for the Developers group, as follows:

```
authorize FILE /production/* access(read) xgid(Developers)
```

This rule permits members of the Developers group to have read access to the /production directory.

Note: The rule you set in Step 4 helps ensure that CA Access Control audits any failed access attempt by any user, including members of the Development group.
6. Define an access rule for the Dev_Access_All group, as follows:

```
authorize FILE /production/* access(all) xgid(Dev_Access_All)
```

This rule permits members of the Dev_Access_All group to have full access to the /production directory.
7. Define an audit rule for the Dev_Access_All group, as follows:

```
chxgrp Dev_Access_All audit(all)
```

This rule audits every action a member of the Dev_Access_All group performs.

8. When a member of the Developers group needs full access to the /production directory, add the user to the Dev_Access_All group in the native environment.

The user has full access to the /production directory, and CA Access Control audits every action the user performs.

Note: The user must start a new logon session for the change in group membership to take effect.

9. When the user has completed their task in the /production directory, remove the user from the Dev_Access_All group in the native environment.

The user now has read access to the /production directory. CA Access Control denies and audits any other access attempt on the /production directory by the user.

Note: The user must start a new logon session for the change in group membership to take effect.

The Auditing Process

To configure CA Access Control for your auditing requirements, you must first understand how auditing works. Auditing lets you keep track of access requests (events) that CA Access Control intercepted. You can use this data to meet with compliance requirements, to analyze and refine your access rules for your security requirements, or to monitor access requests.

The process CA Access Control follows to record audit events in the log depends on the type of event it intercepts:

- [Interception events](#) (see page 112)

Note: Intercepted login events (TERMINAL class), and audit records generated by user traces, are not cached; they always follow the auditing process for interception events.

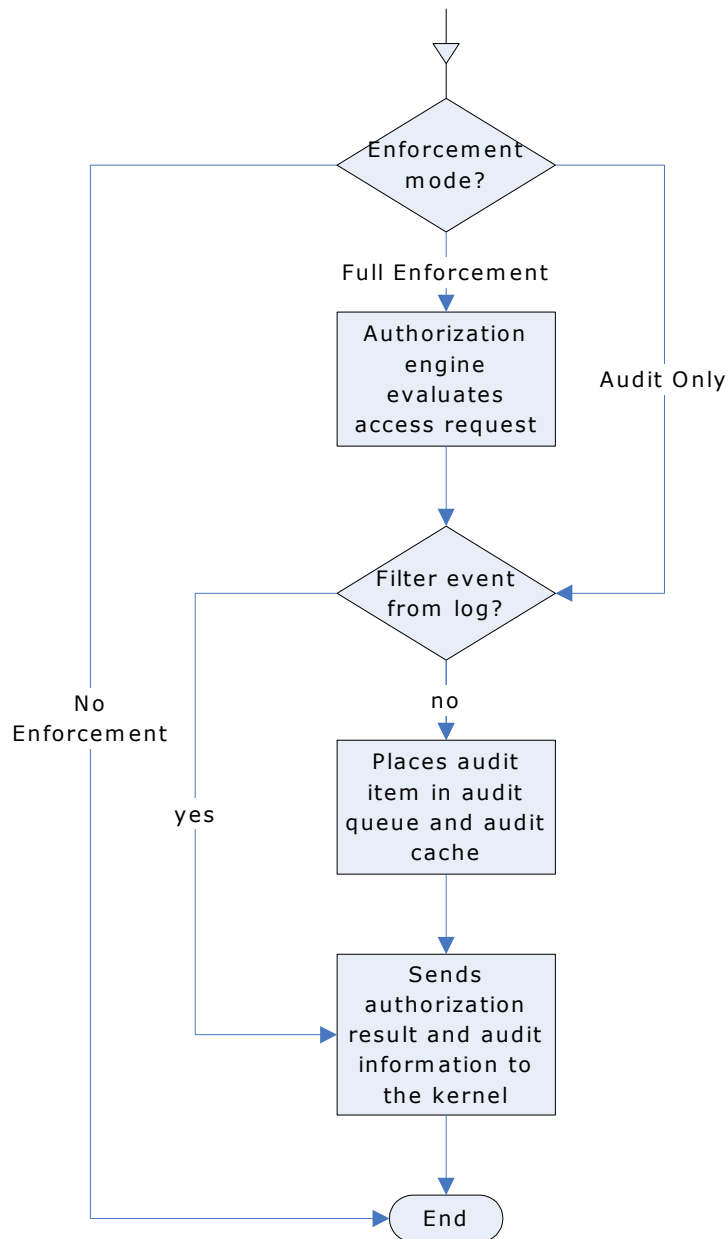
- [Audit events](#) (see page 113)

Note: CA Access Control intercepts an event only if the appropriate class is active, and the database contains a rule anticipating this event.

How Auditing Works for Interception Events

An *interception event* is an event that CA Access Control encounters for the first time and for which no authorization information or audit information exists in the kernel cache.

To log audit records, CA Access Control performs the following actions and causes these effects for an interception event:



- In No Enforcement mode, events are not intercepted or audited.
- In Full Enforcement mode, CA Access Control does the following:
 1. The authorization engine places an audit item based on the authorization result in the audit queue and in the audit cache.

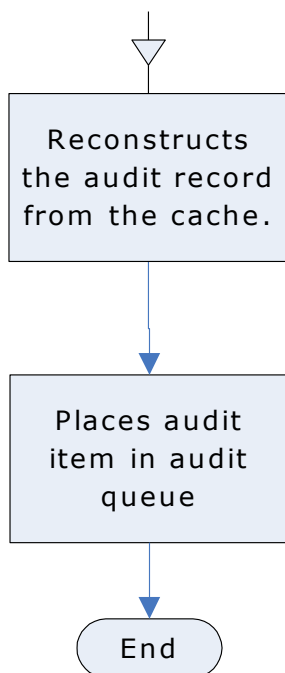
CA Access Control writes an audit item only if the audit property for the resource or accessor is set to audit the resulting event and the audit filter file is not set to filter this event.
 2. The authorization engine returns an informative answer on the authorization result and the audit related information to the kernel.
- In Audit Only mode, CA Access Control does not process the request for authorization. Audit information is always written, regardless of the audit property of the resource and user.

CA Access Control writes an audit item only if the audit filter file is not set to filter this event. The authorization result in this mode is always *P* (permitted).

Note: Intercepted login events (TERMINAL class), and audit records generated by user traces, are not cached; the authorization engine always writes audit records for these events.

How Auditing Works for Audit Events

The following diagram and steps demonstrate how auditing works for audit events:



Once the kernel notifies CA Access Control about the cached interception event, CA Access Control performs the following actions to log the audit event:

1. Reconstructs the audit data using the audit cache out of the information sent by the kernel
2. Puts the audit item in the audit queue

Kernel and Audit Caches

The *kernel cache* contains data about previously intercepted events. The kernel identifies such cached intercepted events (audit events) and sends them to CA Access Control for processing. Essentially, CA Access Control uses the kernel cache to intercept events that follow the same pattern as a previously intercepted event.

The *audit cache* contains data that lets CA Access Control reconstruct reoccurring audit records and send them to the audit queue without needing to follow the authorization process. This means that intercepted events, for which enough information already exists in the cache (audit events), are processed quickly and added to the audit queue. The authorization engine provides the data that is stored in the kernel and audit caches from the result of the initial event it intercepted (the interception event).

Cache Reset

CA Access Control clears both the kernel and audit caches in the following cases:

- Database changes
CA Access Control clears the entire cache when database information changes. New or modified access rules make an existing cache potentially inaccurate.
- Time checkpoint reached
CA Access Control clears the entire cache when a time checkpoint affects an authorization result for any event. At the time that a DAYTIME restriction property or a HOLIDAY class record changes, the authorization result may change too and the cache becomes potentially inaccurate.

- PROGRAM resource change
CA Access Control clears the entire cache when the watchdog identifies that a PROGRAM resource has changed and become un-trusted. An un-trusted program affects the result of an authorization request regarding that program. This makes the cache potentially inaccurate.
- Audit cache filling
CA Access Control clears 10% of cache items (the least recently used items) when the audit cache fills up.

Once the cache is cleared, information from new interception events is needed to refill the cache and let CA Access Control intercept an audit event.

Viewing Audit Events

CA Access Control sends audit events to the audit logs. You view the audit logs using the following CA Access Control tools:

- CA Access Control Endpoint Management
- The seaudit utility

You can configure CA Access Control to also send audit events to the Windows event log. The event log stores audit events from various applications in a single collection. You use the Windows Event Viewer to view audit events in the event log.

Audit Events in the Windows Event Log

The Windows event log stores audit events from various sources in a single collection. If you configure CA Access Control to route audit events to the event log, each time seosd writes an audit event to the CA Access Control audit log, a corresponding event is sent to the event log.

The audit.cfg file filters audit events from both the audit log and the event log. If an audit event is not written to the audit log, it is not sent to the event log.

The Windows 2008 event log also routes audit events into containers called channels, depending on the volume, audience, and originating application of the audit events. The CA Access Control channel is named CA-AccessControl-AuthorizationEngine/Audit.

If you have deployed CA Access Control on a Windows 2008 server, you can choose to send audit events to:

- the event log
- the channel
- both the event log and the channel
- neither the event log or the channel

Route Audit Events to the Windows Event Log

If you configure CA Access Control to route audit events to the Windows event log, each time seosd writes an audit event to the CA Access Control audit log, a corresponding event is sent to the event log. You can also configure CA Access Control to send Policy Model audit events to the event log.

To route events to the event log

1. Stop CA Access Control using the following command:

```
secons -s  
CA Access Control stops.
```

2. Set the value of the SendAuditToNativeLog configuration setting in the logmgr section to 1.

Audit events are sent to the Windows event log.

3. (Optional) Set the value of the SendAuditToNativeLog configuration setting in the Pmd section to 1.

Audit events for policy models are sent to the Windows event log.

4. Restart CA Access Control using the following command:

```
seosd -start  
CA Access Control restarts.
```

Example: Route Audit Events to the Event Log

The following example routes audit events to the event log. You must be in the remote configuration environment (env config) to use this command:

```
er config ACROOT section(logmgr) token(SendAuditToNativeLog) value(1)
```

Example: Route Policy Model Audit Events to the Event Log

The following example routes Policy Model audit events to the event log. You must be in the remote configuration environment (env config) to use this command:

```
er config ACROOT section(Pmd) token(SendAuditToNativeLog) value(1)
```

More information:

[Change Configuration Settings](#) (see page 169)

Route Audit Events to the Windows Event Log Channel

Valid for Windows Server 2008 only

If you configure CA Access Control to route audit events to the Windows event log channel, each time seosd writes an audit event to the CA Access Control audit log, a corresponding event is sent to the event log channel. The CA Access Control event log channel is named CA-AccessControl-AuthorizationEngine/Audit.

You can also configure CA Access Control to send Policy Model audit events to the event log channel. The Policy Model event log channel is named CA-AccessControl-Policy Models/Audit.

To route events to the event log channel

1. Stop CA Access Control using the following command:

```
secons -s  
CA Access Control stops.
```

2. Set the value of the SendAuditToNativeChannel token in the logmgr registry subkey to 1.

Audit events are sent to the Windows event log channel.

3. (Optional) Set the value of the SendAuditToNativeChannel token in the Pmd registry subkey to 1.

Policy Model audit events are sent to the Windows event log channel.

4. Restart CA Access Control using the following command:

```
seosd -start  
CA Access Control restarts.
```

Example: Route Audit Events to the Event Log Channel

The following example routes audit events to the event log channel. You must be in the remote configuration environment (env config) to use this command:

```
er config ACROOT section(logmgr) token(SendAuditToNativeChannel) value(1)
```

Example: Route Policy Model Audit Events to the Event Log Channel

The following example routes Policy Model audit events to the event log channel. You must be in the remote configuration environment (env config) to use this command:

```
er config ACROOT section(Pmd) token(SendAuditToNativeChannel) value(1)
```

The Audit Log

The audit log is stored in a file. The value *audit_log* in the following Windows registry subkey specifies the location of the audit log file:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\logmgr
```

The default value for this key is:

```
C:\Program Files\CA\AccessControl\log\seos.audit
```

By default, CA Access Control automatically backs up the audit log when it reaches 1024 KB. You can change this size by changing the value *audit_size* in the subkey:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\logmgr
```

You can also choose to back up the audit log periodically (daily, weekly, or monthly) by changing the value *BackUp_Date* in the Windows registry subkey:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\logmgr
```

Note: For more information about these registry subkeys, see the *Reference Guide*.

Using Audit Logs

CA Access Control provides two built-in tools for viewing, filtering, and searching the audit logs:

- CA Access Control Endpoint Management
- The seaudit utility

You can display every record in the audit log, or you can use filters to select particular records from the audit log.

The remainder of this chapter describes how to view the records in the audit log when using audit filters in CA Access Control Endpoint Management.

Audit Record Filters

The `audit.cfg` file filters audit records on a host by defining records that should not be sent to the audit file. Each line in the file represents a rule for filtering out audit information (that is, the records that match the criteria in the line will not appear in the audit file). This filter helps to limit the size of the `seos.audit` file by keeping only the records needed. You can edit the `audit.cfg` file to suit your enterprise requirements.

By default, the `audit.cfg` file is located in the `ACInstallDir/etc` directory (UNIX) or `ACInstallDir\data` directory (Windows). You can change the location of the `audit.cfg` file by editing the `[logmgr] AuditFiltersFile` token in the `seos.ini` file (UNIX), or the `AuditFiltersFile` entry in the `logmgr` registry key (Windows).

The CA Access Control Engine, `seosd`, reads the `audit.cfg` file at startup. When a message is sent to the audit file, `seosd` checks if the message matches one of the rules in the `audit.cfg` file. If the message matches a rule, the message is not written to the audit file.

Note: For more information about the `audit.cfg` file, see the *Reference Guide*.

Audit Display Filters

The number of records in the audit log can become enormous. To reduce the number of records that *display*, use filters to specify the types of records for display. You can filter events by various criteria, including time or event type.

Note: You can also filter the audit records CA Access Control *writes* to the audit file using the audit configuration settings (`audit.cfg` file).

You can create a filter in CA Access Control Endpoint Management simply by giving it a name and choosing at least one switch. You can then select additional switches, and have the option of assigning one or more options. You can also filter records with the `seaudit` utility.

CA Access Control Endpoint Management provides several predefined filters, and you can create your own filters.

Filter Wizard, Choose Name and Switches Page

The Choose Name and Switches page of the Filter Wizard lets you define the name of the audit display filter you want to create and the switches that you want to apply to this filter.

This window contains the following fields:

Filter Name

Defines the name of the audit display filter you want to create.

Audit Event Records

Specifies whether you want the filter to display all audit records or only those switches that are selected.

If you select to list all records, the switches on this page do not apply.

List INET audit records of Host and Service

Specifies whether to list the INET audit records of the TCP requests received from the specified hosts for the specified services. Host and service are masks that identify which set of hosts and services are searched for.

Show LOGINs for user on terminal

Specifies to list the following:

- LOGIN records for the specified user on the specified terminal. Both *user* and *terminal* are masks you define.
- Records created by the authorization engine when an invalid password is entered multiple times.

List RESOURCEs audit of class on resource for users

Specifies whether to list resource records. You can define the following later:

- *Class*—a mask that identifies the class to which the accessed resource belongs.
- *Resource*—a mask that identifies the names of the resources that were accessed.
- *User*—a mask that identifies the name of the users who accessed the resources.

List updates to database

Lists database update audit records. You can define:

- *Cmd*—a mask identifying the selang commands to search for.
- *Class*—a mask identifying the classes to search for.
- *Object*—a mask identifying the records to search for.
- *User*—a mask identifying the users who executed the commands.

List startup/shutdown messages

Specifies whether to list the startup and shutdown messages from the CA Access Control services.

List WATCHDOG audit records

Specifies whether to list the Watchdog audit records.

Show only trace records

Specifies whether to only list records sent to the audit log by the tracing facility.

Filter Wizard, Edit Options Page

The Edit Options page of the Filter Wizard lets you define the options you want to apply to audit display filter.

This window contains the following fields:

Listing's Starting Today

Specifies today as the start date. Records logged before today are not listed.

Listing's Starting Date

Specifies the start date. Records logged before the specified date are not listed.

Listing's Starting Time

Specifies the start time. Records logged before the specified time are not listed.

Listing's Ending Date

Specifies the end date. Records logged after the specified date are not listed.

Listing's Ending Time

Specifies the end time. Records logged after the specified time are not listed.

Show internet address not host name

Specifies that Internet addresses be listed instead of host names in TCP/IP records.

Hide failures

Specifies that failures are not listed.

Hide any granted accesses

Specifies that successful (granted) accesses are not listed.

Hide logout records

Specifies that logout records are not listed.

Hide NOTIFY audit records

Specifies that NOTIFY audit records are not listed.

Hide passwords attempts and actions

Specifies that password attempt records are not listed.

Hide warning records

Specifies that warning records are not listed.

Show port numbers not names

Specifies that port numbers be listed instead of service names.

Show only records originated from *host*

Specifies that only records originating from the specified host are listed. This option is applicable only when connected to a UNIX workstation.

Predefined Filters

CA Access Control comes with the following predefined filters:

All records

Displays every record in the audit log. No filtering takes place.

Today's records

Shows every record created today.

Last 2 days records

Shows every record created yesterday and today.

Last 7 days records

Shows every record created during the last seven days.

Connections to CA Access Control services

Shows records that indicate when users connect to CA Access Control services such as CA Access Control Endpoint Management or selang.

Note: When connecting to a UNIX workstation, the name for this filter becomes Login Records. The records represent user logins.

Administration activity

Shows all records that update the CA Access Control or operating system databases. Updates to the databases include adds, deletes, and changes to all types of records.

Create a User-Defined Filter

You can build as many filters as you need. Create a custom filter when you want to view only a particular set of audit records.

To create a user-defined filter

1. In CA Access Control Endpoint Management click the Audit Events tab.
The Audit Records Viewer - Filter Settings section shows the list of Saved Filters.
2. In the Saved Filters section, click Create Filter.
The Audit Filter Wizard appears.
3. Complete the wizard pages.

Choose Name and Switches

Specifies the [switches](#) (see page 121) you want to use in your filter.

Edit Switches

Specifies settings for the switches you selected. Essentially these are masks that you can define for the audit events you want to filter.

Edit Options

Specifies the [options](#) (see page 122) you want to set for audit filtering.

Click Finish.

The new audit filter you defined is saved and loaded.

Audit Log Backup

CA Access Control lets you automatically backup the audit log file for archiving.

The name of the audit log backup file is set in the logmgr\audit_back CA Access Control registry entry.

You can use the following methods for backing up the audit log file:

- Size-triggered backups
- Date-triggered backups

The method and settings you choose for backing up your audit log file should depend on:

- Whether you need backup copies of the log file
- How much auditing data is likely to be generated in your environment
- System performance issues (for example, larger audit log files increase processing time)

Note: By default, CA Access Control protects audit log backup files if you configure settings to keep timestamped backups. This is the same default protection that the size-triggered audit backup file receives. To remove these files, you need to set permissive rules in the database.

Set the Size at which the Audit Log will be Backed Up Automatically

You can set a limit on the size of the audit log file. When the file reaches the defined size, CA Access Control automatically creates a backup copy of the file and clears the log. This means that the file is automatically backed up regularly.

To set the size at which the audit log will be backed up automatically, set the maximum size you require, in KB, in the logmgr\audit_size CA Access Control registry entry.

Note: You can define the name of the backup file by setting the logmgr\audit_back CA Access Control registry entry.

Important! If the logmgr/BackUp_Date CA Access Control registry entry is set to yes (no is the default), each size-triggered backup copy of the audit log is suffixed with a timestamp. In all other cases, including when date-triggered backups are configured, each backup copy *overwrites* the previously written backup copy.

Example: Set automatic backup of audit log file when it reaches 5 MB

This example shows you how you set your audit log file to be backed up when it reaches 5 MB (5120 KB). To do this, set the logmgr\audit_size CA Access Control registry entry to **5120**.

When the audit log file reaches 5 MB, CA Access Control will create a backup copy of the file, named seos.audit.bak by default, and clear the log.

Example: Set automatic backup of audit log file when it reaches 1 MB with a custom name and a timestamp

This example shows you how you set your audit log file to be backed up when it reaches 1 MB (1024 KB), using a custom name for the backup file and adding a timestamp to the name.

To do this, set the following CA Access Control registry entries as shown:

- logmgr\audit_size=1024
- logmgr\audit_back=log\ac_audit.old
- logmgr\BackUp_Date=yes

When the audit log file reaches 1 MB, CA Access Control will create a backup copy of the file, and clear the log. The name of the backup log file name will be: ac_audit.old.*timestamp*, where *timestamp* is the date and time in the format DD-Mon-YYYY.hhmmss. For example:

```
ac_audit.old.06-Feb-2007.144330
```

Set the Time Interval at which the Audit Log will be Backed Up Automatically

You can define a time interval (daily, weekly, or monthly) at which CA Access Control automatically creates a backup copy of the audit log file and clears the log.

To set the time interval at which the audit log is backed up automatically, set the interval in the logmgr\BackUp_Date CA Access Control registry entry. The interval can be one of the following:

daily

Backs up the audit log file once a day.

weekly

Backs up the audit log file once a week.

monthly

Backs up the audit log file once a month.

Note: You can define the name of the backup file by setting the logmgr\audit_back CA Access Control registry entry.

Important! If the audit log reaches the size limit defined in the logmgr\audit_size CA Access Control registry entry before the backup interval is reached, CA Access Control creates a backup copy of the file without a timestamp. Each such backup copy can potentially overwrite any previous copy.

Example: Set a daily backup of the audit log file

This example shows you how you set your audit log file to be backed up daily. To do this, set the logmgr\BackUp_Date CA Access Control registry to **daily**.

Once a day CA Access Control creates a backup copy of the file, and clears the log. The backup log file name has the *.timestamp* suffix, where *timestamp* is the date and time in the format DD-Mon-YYYY.hhmmss. For example:

```
seos.audit.bak.06-Feb-2007.144330
```

Chapter 9: Scope of Administration Authority

This section contains the following topics:

[Global Authorization Attributes](#) (see page 127)

[Group Authorization](#) (see page 129)

[Ownership](#) (see page 132)

[Authorization Examples](#) (see page 134)

[Sub Administration](#) (see page 136)

[Environmental Considerations](#) (see page 138)

[Default Permissions to Access the Database](#) (see page 140)

[Native Permissions to Access the Database](#) (see page 140)

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 lets a user execute almost all commands in CA Access Control. Users who are defined in the database with the ADMIN attribute can define and update users, groups, and resources in the database. This is the most powerful attribute in CA Access Control, but it does have limitations:

- 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 change the type of auditing that is done on a user, group, or resource (audit mode). If you have the ADMIN attribute and need to change the auditing characteristics of a user, group, or resource, assign yourself the AUDITOR attribute.
- Users with the ADMIN attribute cannot delete superuser (the root account on UNIX or the Administrator account on Windows), but they can set root to be a non-ADMIN user.

AUDITOR Attribute

Users with the AUDITOR attribute can monitor system usage. Explicit privileges of a user with the AUDITOR attribute include the following:

- Users can display information in the database.
Auditors can execute the selang commands *showusr*, *showgrp*, *showres*, and *showfile*.
- Users can set the audit mode for existing records.
Auditors can execute the selang commands *chusr*, *chgrp*, *chres*, and *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 lets a user use the *secons* utility.

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

PWMANAGER Attribute

The PWMANAGER attribute gives a regular user the authority to use the *chusr* or *sepass* command to change the passwords of other users.

Note: To let the PWMANAGER change the ADMIN user's password, set the *cng_adminpwd* option of the *setoptions* command. For more information, see the *selang Reference Guide*.

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

CA Access Control, like many other security models, does not permit a regular user to ask: “Can user A access resource X?” The only question a regular user can ask is: “Can I access resource X?” However, a process that supplies services to many users, such as a database server service or an in-house application, should be permitted to ask for authorization on behalf of other users.

The SERVER attribute allows a process to ask for authorization for users. Users with the SERVER attribute set can issue the SEOSROUTE_VerifyCreate API.

Note: For more information about the server attribute and CA Access Control APIs, see the *SDK Guide*.

IGN_HOL Attribute

The IGN_HOL attribute allows users to log in during any period defined in a holiday record. Each record in the HOLIDAY class defines one or more periods when users need extra permission to log in. With the IGN_HOL attribute, users can log in at any time, regardless of the periods defined in holiday records.

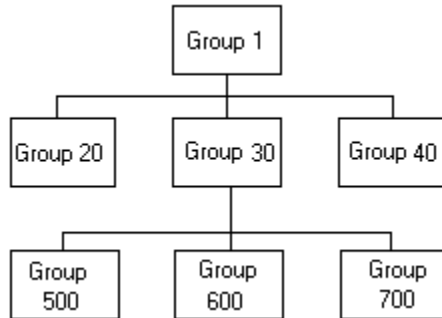
Note: For more information about the HOLIDAY class, see the *Reference Guide*.

Group Authorization

It is necessary to understand the concept of parentage before discussing group authorization attributes.

Parentage

The concept of subordinate and superior groups, also known as parentage, is important when discussing group administration privileges. One group can be the parent-superior-of one or more groups. A *child* or subordinate group can have only one parent. Assigning a parent to a group is optional. Consider the following diagram:



Group 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.

A group can own its own records. However, within a group that owns records, only certain privileged users can manage the records. These special users have a group authorization attribute set in their own user records. The group authorization attributes are the following:

- GROUP-ADMIN
- GROUP-AUDITOR
- GROUP-OPERATOR
- GROUP-PWMANAGER

The join command-which only a properly authorized user can issue-sets these attributes. The join command serves the purpose of both putting a user into a group, and specifying the user's group authorization attribute (if any).

The privileged members of the group may or may not be authorized to manage the user records that define the members of the group, depending on who owns those records.

More information:

[Ownership](#) (see page 132)

GROUP-ADMIN Attribute

Users with a group administration authorization attribute can create a certain set of records. In order to create a record, the group administrator has to specify the owner of the record.

The owner of the records must be the group in which the user has a group authorization attribute. If that group is the parent of other groups, the owner can also be from one of the sub groups. The whole set of records is called the group scope. The authorization examples provided illustrate the concept of group scope.

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

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 superuser (the root account on UNIX or the Administrator account on Windows) 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 let CA Access Control assign the user who defines the record as the owner of the record.

Accessors own a record if *any* of the following are true:

- They are defined as the owner of the record.
- They are members of a group that is defined as the owner of the record *and* they have joined the group with the GROUP-ADMIN property.
- They are owners of a resource group record that the resource is a member of.

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 and to any resource group record that the record is a member of.

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 a superuser (the root account on UNIX or the Administrator account on Windows) 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

CA Access Control 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.

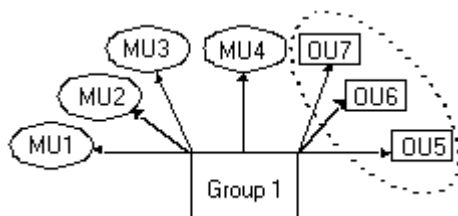
On UNIX, when a user creates a file, UNIX assigns the user as the owner of the file. CA Access Control allows UNIX file owners to define FILE records, unless this feature is explicitly disabled. If you do not want file owners to define FILE records, make sure that the use_unix_file_owner token in the [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

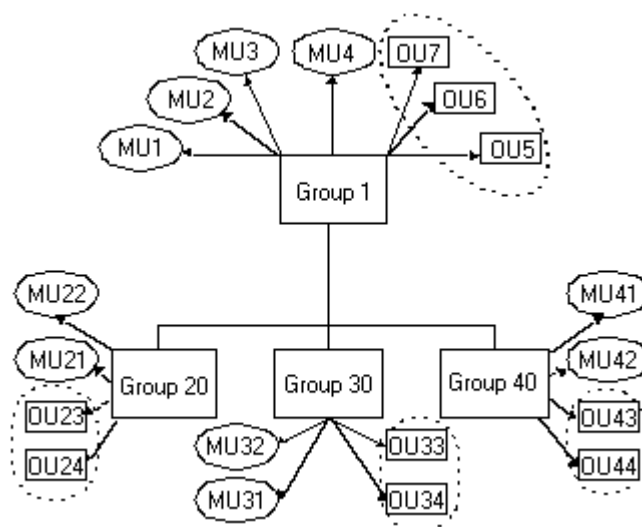
In the following diagram, four users are members of Group 1: MU1, MU2, MU3, and MU4. Group 1 also owns three users-OU5, OU6, and OU7. The member MU4 has the GROUP-ADMIN attribute.



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.

Sub Administration

Security administrators (users with the ADMIN attribute) can grant specific administrative privileges to regular users. These regular users are then called sub administrators. Sub-administrators have privileges to manage only specified CA Access Control classes or objects. For example, a sub administrator can be authorized to manage only user and group objects. You can set a higher level of sub administration by authorizing the sub admin user the administrative privileges for specific objects in a class.

Sub administrators of users, groups and resources can use `selang` to perform administrative tasks related to these resources.

How to Grant Specific Administrative Privileges to Regular Users

Because administrators—users with the ADMIN attribute—can execute almost all actions in CA Access Control, you may want to delegate specific administrative tasks to sub administrators. To do this, you need to grant those users with privileges to classes in the CA Access Control database that control the specific administrative tasks the user needs to perform as follows:

1. Identify one or more classes that control the tasks you want to delegate.

For example, CA Access Control uses the USER and GROUP classes to create accessor resources. If you want to delegate accessor management, you then need to use the USER and GROUP records of the ADMIN class.

2. Authorize one or more sub administrator to the applicable resource of the ADMIN class.

For example, to let a sub administrator view and modify user records, grant the user with *read* and *modify* access to the USER record of the ADMIN class.

The ADMIN Class

Sub administrators—users listed in the access control list (ACL) of records in the class ADMIN—have privileges similar to users with the ADMIN attribute. However, the privileges of users in the ACL for records in the class ADMIN are limited to the particular class represented by the record. For example, the SURROGATE record in the ADMIN class determines which users can administer records of the SURROGATE class.

Note: For more information about CA Access Control 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 superuser (the root account on UNIX or the Administrator account on Windows), but they can set root to be NOADMIN.
- ADMIN class users cannot make resources inaccessible to themselves, so:
 - ADMIN class users cannot assign a security level to a resource that is higher than their own security level.
 - ADMIN class users cannot assign a security category or security label that they do not have.

These limitations are part of the B1 security level certification.

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 CA Access Control 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 CA Access Control 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:

```
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:

```
authorize TERMINAL tty13 uid(ADMIN1) access(r,w)
```

UNIX Environment

For managing users and groups in UNIX, users in CA Access Control with global or group authorization attributes have the same privileges and limits for UNIX as they do for CA Access Control.

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.)

Windows Environment

Valid in the native Windows environment

When CA Access Control is running, if you use `selang` to change a resource in the native Windows environment, the CA Access Control Agent changes the resource in the appropriate Windows repository. You do not need any additional Windows permissions to change the resource. This means that when users in CA Access Control with global or group authorization attributes perform `selang` commands in the native Windows environment, they have the same privileges and limits for Windows as they do for CA Access Control.

When CA Access Control is not running, if you use `selang` to change a resource in the native Windows environment, you must follow these rules:

- You must include the `-l` option in the `selang` command
- You must have the ADMIN attribute or sub administration privileges
- You must have sufficient Windows permissions to change the resource

This restriction occurs because a `selang` process, not the CA Access Control Agent, changes the resource in the Windows repository.

For example, user Emma wants to use the `chfile` `selang` command in the native Windows environment to change the owner of the file `C:\tmp.txt`. If CA Access Control is running, Emma requires sufficient CA Access Control permissions to change the file owner, but does not require additional Windows permissions. If CA Access Control is not running, Emma requires both CA Access Control and Windows permissions to change the file owner.

Default Permissions to Access the Database

CA Access Control protects the internal database, seosdb, with internal file rules when it is running. Internal file rules are not visible in selang and cannot be deleted. You can write FILE rules to override the internal file rules. If you delete these FILE rules, CA Access Control reverts to the internal file rules.

The following internal file rules protect the database when CA Access Control is running:

- CA Access Control internal processes have full access to the database
- The NT AUTHORITY\System user has read access to the database
- All other accessors have no access to the database

Note: The default access rights for all other accessors were changed in r12.5 SP3. In previous releases, all other accessors had read access by default to the database files.

By default CA Access Control services run automatically after you install CA Access Control or reboot the endpoint. Consequently, the only user who can access the database out of the box is NT AUTHORITY\System. The CA Access Control administrators that you define during installation can also use a utility such as selang to update the database.

Native Permissions to Access the Database

When CA Access Control is stopped, access rights to the database files are determined by native Windows permissions. Permissions are inherited from the parent directory in which CA Access Control is installed. Because of this inheritance, when CA Access Control is stopped the default access to the database files is read.

To protect CA Access Control when it is stopped, you can change the Windows permissions for the database files to suit your enterprise requirements. Before you change the permissions, consider the following:

- The NT AUTHORITY\System user *must* have Windows permissions to read and write to the database files.

The CA Access Control authorization engine inherits privileges from the NT AUTHORITY\System user. If this user cannot access the database, the engine does not have sufficient native privileges to update the database.

- Users who need read and write access to CA Access Control when it is stopped must have Windows permissions to read and write to the database files.

Users who need read and write access include users who back up, restore, or upgrade CA Access Control.

- Users that can use `selang` when CA Access Control is stopped (`selang -l` option) must have the following permissions:
 - The ADMIN attribute or sub administration privileges
 - Windows permissions to read and write to the database files
 - Windows permissions to change native repositories, if required

For example, to use the config environment to change CA Access Control registry entries when CA Access Control is stopped, you must have sufficient Windows privileges to change the registry.

Only CA Access Control administrators (users with the ADMIN attribute or with sub administration privileges) can use `selang` to maintain the database when CA Access Control is stopped. If the CA Access Control administrators cannot access the database when CA Access Control is stopped, no user can perform offline database maintenance and a deadlock may occur.

Chapter 10: Managing Policy Models

This section contains the following topics:

[The Policy Model Database](#) (see page 143)

[Architecture Dependency](#) (see page 146)

[Methods for Centrally Managing Policies](#) (see page 148)

[Automatic Rule-based Policy Updates](#) (see page 148)

[Integrate PMDBs with Unicenter](#) (see page 161)

[Mainframe Password Synchronization](#) (see page 161)

The Policy Model Database

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

Note: In Windows Task Manager, the Policy Model service appears as `sepmdd.exe`.

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

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

Policy Model names are case-sensitive on Windows for compatibility with UNIX. When specifying PMDB names in commands, make sure you use the correct case.

Note: You cannot use non-English characters in PMDB and host names.

Although PMDB names are case-sensitive, you cannot have two PMDBs on the same computer with only the letter case being different. This is because CA Access Control uses the PMDB name as part of the file path but Windows is case-insensitive and so does not permit this. For example, `myPMDb` and `MYpmdb` are two different Policy Model databases but cannot live on the same system.

Note: For information about administrating a PMDB (`sepmdd` utility), see the *Reference Guide*. For information about managing PMDBs remotely using `selang`, see the *selang Reference Guide*.

PMDB Location on Disk

All PMDBs on a computer reside in a common directory. The `_pmd_directory_` value in the following subkey of the Windows registry specifies the name of the directory:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\Pmd
```

The default value of `_pmd_directory_` in the NTFS root directory is: `ACInstallDir\data`, where `ACInstallDir` is the directory where you installed CA Access Control (by default `C:\Program Files\CA\AccessControl\`).

Each PMDB occupies a subdirectory in the common directory. The files in the subdirectory contain all the data required to define the Policy Model. Policy Model configuration settings are stored in a `Pmd` subkey of the CA Access Control registry settings. The name of the subkey is the name of the Policy Model.

Managing Local PMDBs

CA Access Control offers a utility for administering PMDBs:

sepmd

A PMDB administration utility that lets you:

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

Note: For a comprehensive discussion of `sepmd`, see the *Reference Guide*.

Managing Remote PMDBs

CA Access Control also offers you a range of `selang` commands that you can use in the `pmd` environment. These commands let you manage PMDBs remotely:

backuppmd

Backs up a PMDB.

createpmd

Creates a PMDB.

deletepmd

Deletes a PMDB.

findpmd

Displays the names of all PMDBs on the computer.

listpmd

Lists the following information about a PMDB:

- Subscribers and their status
- PMDB description and its status
- Commands in the update file and their offsets
- Contents of the error log

pmd

A PMDB administration command that lets you:

- Remove a subscriber from the list of unavailable subscribers
- Clear the Policy Model error log
- Start and stop the Policy Model service
- Lock and unlock the Policy Model
- Truncate the update file

restorepmd

Restores a PMDB from its backup files.

subs

A PMDB subscription command that lets you:

- Add an existing subscriber to a parent PMDB
- Add a new subscriber to a parent PMDB
- Assign a parent PMDB to a database (CA Access Control or another PMDB)

subspmd

Assigns a parent PMDB to the local database.

unsubs

Removes a subscriber from the PMDB.

Note: For a comprehensive discussion of *selang* commands you can use in the *pmd* environment, see the *selang Reference Guide*.

Architecture Dependency

When deploying CA Access Control, you should consider the hierarchy of your environment. At many sites, the network includes a variety of architectures. Some policy rules, such as the list of trusted programs, are architecture-dependent. On the other hand, most rules are independent of the system's architecture.

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

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

Example: A Two-tiered Deployment Hierarchy

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

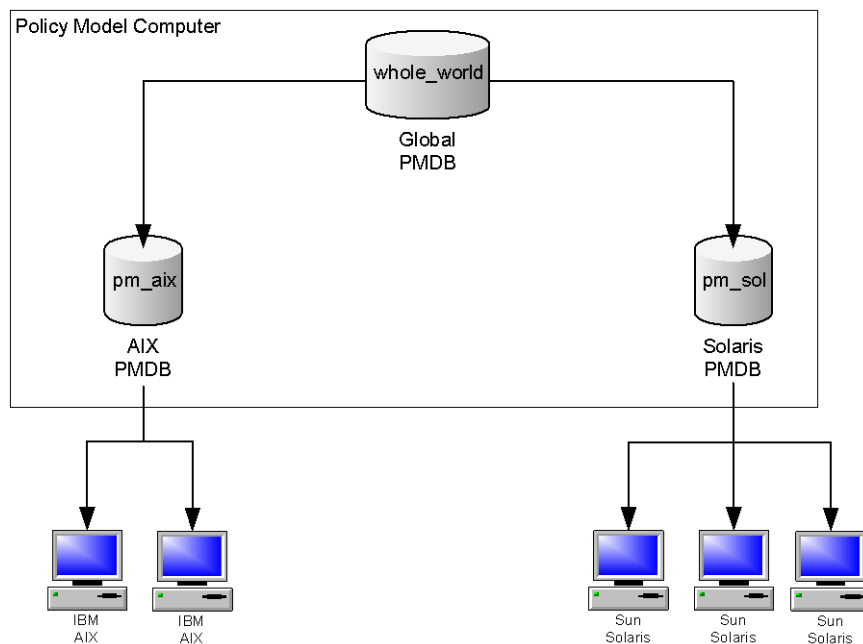
In the example, the site consists of IBM AIX and Sun Solaris systems. Since the list of trusted programs on IBM AIX differs from the one on Sun Solaris, the PMDBs need to consider architecture dependency.

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

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

3. Define the PMDB pm_sol, to contain all the Sun Solaris specific rules.

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



4. When you enter platform-independent commands in whole_world, such as adding a user or setting a SURROGATE rule, all databases at the site are automatically updated.
5. When you add a trusted program to pm_aix, only IBM AIX computers are updated, without affecting the Sun Solaris systems.

Methods for Centrally Managing Policies

CA Access Control lets you manage several databases from a single computer in the following ways:

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

Note: Dual control is only available with this method and on UNIX only. Information about dual control for automatic rule-based policy updates is found in the *Endpoint Administration Guide for UNIX*. Information about automatic rule-based policy updates can also be found in the *Endpoint Administration Guide for Windows*.

- **Advanced policy management**—Policies (groups of rules) you deploy are propagated to all databases based on host or host group assignment. You can also undeploy (remove) policies and view deployment status and deployment deviation. You need to install and configure additional components to use this functionality.

Note: Information about advanced policy management is found in the *Enterprise Administration Guide*.

Automatic Rule-based Policy Updates

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

Note: This method of managing policies is limited to letting you make single-rule policy updates across your hierarchy. Other functionality is only available through implementing advanced policy management and reporting.

How Automatic Rule-based Policy Updates Work

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

1. A rule is defined for any PMDB with at least one subscriber.
2. The PMDB sends the command to all subscriber databases.

3. The subscriber database applies the propagated command.
 - a. If the subscriber database does not respond, the PMDB sends the command at a regular interval (by default, every 30 minutes) until the subscriber database has been updated.
 - b. If a subscriber database is responding, but refuses to apply the command, the PMDB places the command in the [Policy Model error log](#) (see page 155).
4. If the subscriber database is a parent to other subscribers, it then sends the command to its subscribers.

Example: Removing a user from all computers in a hierarchy

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

How You Use a PMDB to Propagate Configuration Settings

When you edit a Policy Model's configuration, the new configuration values are propagated to the Policy Model's subscribers.

The following process describes how configuration updates are propagated to a Policy Model's subscribers:

1. You edit one or more of the Policy Model's configuration values.
2. The Policy Model writes the new configuration values to the virtual configuration file.

Note: The virtual configuration file does not contain values for the `audit.cfg` file. The Policy Model does not write any changes you make to this file to the virtual configuration file.
3. The Policy Model sends the new configuration values to its subscribers.
4. `selang` commands update each subscriber with the new configuration values.

Virtual Configuration File

Each Policy Model has a virtual configuration file that contains the configuration values for its subscribers. The virtual configuration file is located in the PMD directory, and is named `cfg_configname`, where `configname` is the name of the Policy Model configuration.

The virtual configuration file does not contain the configuration values held in the `audit.cfg` file.

How New Subscribers Are Configured

The Policy Model configures each new subscriber with the existing configuration values. The existing configuration values are stored in the virtual configuration file.

Note: The virtual configuration file does not store configuration values from the `audit.cfg` file. Any changes you make to the `audit.cfg` file prior to creating a new subscriber are not propagated to the new subscriber.

The following process describes how a Policy Model configures new subscribers:

1. You create a new subscriber to the Policy Model.
2. The Policy Model reads the values in its virtual configuration file.
3. The Policy Model adds the configuration values from its virtual configuration file to the `updates.dat` file. The `updates.dat` file also contains the access rules for the Policy.
4. The Policy Model sends the `updates.dat` file to the new subscriber.
5. `selang` commands configure the new subscriber with the values in the `updates.dat` file.

How You Can Set Up a Hierarchy

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

The most straightforward way to set up the PMDB hierarchy is as you install CA Access Control, so it is worth thinking through how you want to structure the hierarchy before you begin the installation. Make sure that all the hosts in the PMDB hierarchy are part of the same network, since the parent PMDB and its subscribers must be able to communicate with each other. That is, the parent must be able to connect with each of its subscribers by name, and every subscriber must be able to connect to the parent PMDB by name.

Note: For more information about installing CA Access Control, see the *Implementation Guide*.

If you want to change the configuration that you created during installation or if you did not create a PMDB structure during installation, you can change or create the PMDB configuration at any time. You can do this in one of the following ways:

- With CA Access Control Endpoint Management
- With the `sepmdb` utility

To create a PMDB hierarchy and enable automatic rule-based policy updates after installation, do the following:

1. Create and configure the master PMDB.
2. (Optional) Create and configure subscriber PMDBs.
3. Define parent PMDBs for the subscribing computers, called *endpoints*.

Update Subscribers

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

1. The Policy Model tries to fully qualify subscriber names as they are added or deleted from the Policy Model.
2. The PMDB service, *sepmdd*, attempts to update a subscriber database.
3. If the maximum time elapses and the service does not succeed in updating a subscriber, it skips that particular subscriber and tries to update the rest of the subscribers on its list.
4. After it completes its first scan of the subscriber list, *sepmdd* then performs a second scan, in which it tries to update the subscribers that it did not succeed in updating during its first scan.

Note: Whenever a PMDB encounters an error while propagating updates to subscribers, the *sepmdd* service creates an entry in the [Policy Model error log file](#) (see page 155). This file, `ERROR_LOG`, is located in the [PMDB directory](#) (see page 144).

Update a Policy Model Database

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

You can specify the PMDB using *selang* or CA Access Control Endpoint Management. To specify a target database using *selang*, use the `hosts` command in the *selang* command shell:

```
hosts pmd_name@pmd_host
```

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

Example: Specify a target PMDB

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

```
hosts policy1@myPMD_host
```

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

Clean Up the Update File

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

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

```
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.

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

Propagate and Synchronize Passwords

Once you set up a PMDB hierarchy, you can use it to keep user passwords synchronized throughout your system when the user passwords are changed with the Windows User Manager or software other than CA Access Control.

Note: CA Access Control also supports mainframe password synchronization.

To propagate and synchronize passwords

1. Create a PMDB Hierarchy.
2. Enter the name of the appropriate parent PMDB as the passwd_pmd entry value in the registry on every station on which users or administrators may change passwords.

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\AccessControl\passwd_pmd
```

The PMDB then propagates the password change to all its subscribers. If the passwd_pmd value is blank, CA Access Control checks the secondary_pmd value and sends new and updated passwords to the PMDB listed in this value, unless it also is blank.

Note: If the PMDB sends a user password to a subscriber in which the user is not defined, settings are not changed and the user remains undefined for the subscriber.

Remove a Subscriber

If you no longer want to propagate updates to a particular subscriber, you should remove it.

To remove a subscriber

1. Remove the computer from the subscription list:

```
secmd -u PMDB_name computer_name
```

The computer is removed from the Policy Model subscription list.

2. Shut down seosd on the computer that you removed from the subscription list:

```
secons -s
```

The seosd service is shut down.

3. Delete the value of the parent_pmd registry value in the following registry key on the computer you removed from the subscription list:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\AccessControl
```

The computer will stop accepting updates from the parent PMDB.

4. Restart seosd.

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

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

Filter Updates

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

To filter updates

1. Configure PMDBs to serve as parents to subsets of subscribers.
2. Modify the *Filter* registry entry in the registry key of the parent PMDB, to point to a filter file you set up on the same computer.

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

Policy Model Filter File

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

- The form of access permitted or denied.
For example, EDIT or MODIFY
- The environment affected:
For example, AC or native
- The class of the record.
For example, USER or TERMINAL
- The objects, within the class, that the rule covers.
For example, User1, AuditGroup, or COM2
- The properties that the record grants or cancels.
For example, OWNER and FULL_NAME in the filter line means that any command having those properties is filtered. You must enter each property exactly as it appears in the *Reference Guide*.
- Whether such records should be forwarded to the subscriber database or not:
PASS or NOPASS

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

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

Example: Filter file

The following example describes a line from a filter file:

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

In this example, if we name the file with this line `Printer1_Filter.flt` and edit the registry for PMDB PM-1 so that `filter=C:\Program Files\CA\AccessControl\Printer1_Filter.flt`, then PMDB PM-1 will not propagate to its subscribers any records that create new users with the `FULL_NAME` and `OBJ_TYPE` property.

Policy Model Error Log File

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

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

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

```
ACInstallDir/bin sepmd -e pmdname
```

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

```
ACInstallDir/bin sepmd -c pmdname
```

Important! The error log in CA Access Control r5.1 and later versions has a format that is not compatible with the format of earlier versions. `sepmd` cannot handle error logs from these earlier versions. When you upgrade to a version that has this format, the old error log is copied to `ERROR_LOG.bak`; a new log file is created when you start `sepmdd`.

Example: PMDB Update Error Message

The following example shows a typical error message:

```

date      time      pmdb name  subscriber  command  offset  flag
  ↓        ↓        ↓           ↓           ↓        ↓       ↓
20 Nov 03 19:53:17 (pmdb1): fargo  nu u5 0  Retry
ERROR: Connection failed (10071) ← major level (type of error)
Host is unreachable (12296) ← minor level (cause of error)
                        ↑
                    return code

```

- The top line always consists of the date, time, and subscriber. The command that generated the error appears next, followed by the offset (in decimal format), which indicates the location of the failed update inside the updates file. Lastly, the flag indicates whether the PMDB retries the update automatically or continues without it.
- The second line shows an example of a major level message (what type of error occurred) and its return code.
- The third line displays an example of a minor level message (why the error occurred), and its return code.

Example: Error Message

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

The following error has only one message level:

```
Fri Dec 29 10:30:43 2003 CIMV_PROD:Release failed. Return code = 9241
```

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

Native Policy Model Repositories

You can store all native environment user and group object types in the PMDB. By storing this information in the PMDB, you can receive information about objects using show commands (such as show user or show group). The returned objects are an image of the actual objects that are defined in Windows or UNIX subscribers.

After connecting to a Policy Model, a user can choose the following environments:

- AC
- Native

- NT
- UNIX
- Config

Note: Native operates exactly as Windows while you are working in a Windows operating system, or exactly as UNIX while you are working on a UNIX operating system.

To use native environment repositories, use the following commands:

- Enter the following commands at the selang prompt:

```
env NT; find
```

Your results list all the native environment object types.

Note: For descriptions of these object types, see the Windows environment classes and properties in the *Reference Guide*.

- Enter the following commands to receive a list of NT and Active Directory USER properties:

```
env NT; ruler user
```

- Enter the following commands to receive a list of NT and Active Directory GROUP properties:

```
env NT; ruler group
```

If a Policy Model is a subscriber of another (parent) Policy Model, it receives data from a parent through propagation and saves in the database all user and group properties, so you can see them and change them.

Note: For more information, see the `sepmdb` utility in the *Reference Guide*.

Policy Model Backup

When you back up a PMDB, you copy the data in the Policy Model database to another directory. This includes:

- policy information
- the list of the Policy Model's subscribers
- configuration settings
- registry entries
- the `updates.dat` file

You cannot restore a PMDB from backup files that use another platform, operating system, or version of CA Access Control. Ensure you back up the Policy Model to a host running the same platform, operating system, and version of CA Access Control.

Back Up a PMDB Using sepmd

When you back up the PMDB, you copy the data from the Policy Model database to a specified directory. You should store the backed up PMDB files in a secure location, preferably protected by CA Access Control access rules.

You can use the sepmd utility to back up a PMDB on a local host. You can also use selang commands to back up a PMDB on a remote host.

Note: You can back up a PMDB recursively. A recursive backup backs up all the PMDBs in a hierarchy to the host you specify, and modifies the PMDB subscribers so that the subscription still works when the backup is moved to the host. You can only use a recursive backup if the master and child PMDBs are deployed on the same host.

To back up a PMDB using sepmd

1. Lock the PMDB using the following command:

```
sepmd -bl pmdb_name
```

The PMDB is locked and cannot send commands to its subscribers.

2. Do one of the following:

- Back up the PMDB using the following command:

```
sepmd -bh pmdb_name [destination_directory]
```

- Back up the PMDB recursively using the following command:

```
sepmd -bh pmdb_name [destination_directory] [backup_host_name]
```

Note: If you do not specify a destination directory, the backup is saved to the following directory:

```
ACInstallDir\data\policies_backup\pmdb_name
```

3. Unlock the PMDB using the following command:

```
sepmd -ul pmdb_name
```

The PMDB is unlocked and can send commands to its subscribers.

Back Up a PMDB Using `selang`

When you back up the PMDB, you copy the data from the Policy Model database to a specified directory. You should store the backed up PMDB files in a secure location, preferably protected by CA Access Control access rules.

You can use `selang` commands to back up a PMDB on a local or remote host. You can also use the `sepmdb` utility to back up a PMDB on a local host.

Note: You can back up a PMDB recursively. A recursive backup backs up all the PMDBs in a hierarchy to the host you specify, and modifies the PMDB subscribers so that the subscription still works when the backup is moved to the host. You can only use a recursive backup if the master and child PMDBs are deployed on the same host.

To back up a PMDB using `selang`

1. (Optional) If you are using `selang` to connect to the PMDB from a remote host, connect to the PMDB host using the following command:

```
host pmdb_host_name
```

2. Move to the PMD environment using the following command:

```
env pmd
```

3. Lock the DMS using the following command:

```
pmd pmdb_name lock
```

The PMDB is locked and cannot send commands to its subscribers.

4. Back up the DMS database using the following command:

```
backuppmd pmdb_name [destination(destination_directory)] [hir_host(host_name)]
```

Note: If you do not specify a destination directory, the backup is saved to the following directory:

```
ACInstallDir\data\policies_backup\pmdbName
```

5. Unlock the PMDB using the following command:

```
pmd pmdb_name unlock
```

The PMDB is unlocked and can send commands to its subscribers.

Policy Model Restoration

When a Policy Model is restored, CA Access Control copies the backup PMDB files into the specified directory. Everything that is in the original PMDB files is copied to the new PMDB directory, including:

- policy information
- the list of the Policy Model's subscribers
- configuration settings
- registry entries
- the updates.dat file

If there is an existing PMDB in the destination directory, CA Access Control deletes the existing files before copying the restoration files into that directory.

You cannot restore a PMDB from backup files that use another platform, operating system, or version of CA Access Control. Ensure you back up the Policy Model to a host running the same platform, operating system, and version of CA Access Control.

Restore a PMDB

When you restore a PMDB, CA Access Control copies the data from the PMDB backup files into the directory you specify. CA Access Control must be running on the terminal you do the restoration on.

Note: If you back up and restore the PMDB on different terminals, the PMDB does not automatically update the terminal resource in the restored PMDB database. You must add the new terminal resource to the restored PMDB. To add the new terminal resource, stop the restored PMDB, run the `selang -p pmdb` command, then start the restored PMDB.

To restore a PMDB, run *one* of the following on the terminal that you want to restore the PMDB on:

- `sepmc -restore` utility
- `selang restore pmd` command

Note: For more information about the `sepmc` utility, see the *Reference Guide*. For more information about `selang` commands, see the *selang Reference Guide*.

Integrate PMDBs with Unicenter

Integrating your PMDB with Unicenter TNG lets you use the PMDB to create rules that secure Unicenter TNG objects from being manipulated by the various Unicenter TNG components (such as the command processor, Event Management, and Workload Management).

You must perform the integration manually.

To integrate a PMDB with Unicenter TNG

1. Create the PMDB.
2. Migrate Unicenter Security options into the PMDB with the following command:

```
MigOpts pmdb-name
```

where *pmdb-name* is the name of your PMDB.

Note: This step is required only if you used Unicenter Security and selected Security Data Migration under the Unicenter Integration during the CA Access Control installation. If you did not use Unicenter Security, then you never established any security options and there is nothing to migrate into your PMDB.

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

```
defclass.bat. pmdb-name
```

where *pmdb-name* is the name of your PMDB

Note: This step is required only 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 CA Access Control installation.

Mainframe Password Synchronization

CA Access Control supports password synchronization among mainframes running CA Top Secret, CA ACF2, or RACF security products (and CA Common Services CAICCI package) and Windows or UNIX computers running CA Access Control. Synchronization is accomplished using the standard CA Access Control password Policy Model method.

Any password change a mainframe user makes is propagated to all the machines in the password Policy Model hierarchy.

Mainframe Password Synchronization Prerequisite

To work with Mainframe Password Synchronization on the server that has TNG/TND/NSM installed, CA Access Control requires a prerequisite TNG/TND/NSM fix - T129430. Please contact support for getting the fix.

Chapter 11: General Security Features

This section contains the following topics:

[Maintenance Mode Protection \(Silent Mode\)](#) (see page 163)

[Bypass Drivers](#) (see page 164)

[Disable CA Access Control Kernel Interceptions](#) (see page 166)

[Stack Overflow Protection](#) (see page 167)

Maintenance Mode Protection (Silent Mode)

CA Access Control has a maintenance mode, also known as silent mode, for protection when the CA Access Control services are down for maintenance. In this mode, CA Access Control denies events while these services are down.

When CA Access Control is running, it intercepts security sensitive events and checks whether the event is allowed. Without activating maintenance mode, all events are permitted when CA Access Control services are down. With active maintenance mode, events are denied when CA Access Control services are down, stopping user activity while the system is maintained.

Maintenance mode can be tuned, and it is disabled by default.

When the CA Access Control security services are down:

- If maintenance mode is active, all security sensitive events are denied, except for special cases and for events executed by the maintenance user.
- If maintenance mode is disabled, CA Access Control does not intervene and execution is passed to the operating system.

When maintenance mode is activated and security is down, the prevented events are not logged in the audit log file.

To enable maintenance mode, follow these steps:

1. Make sure the CA Access Control services are down.
2. Using a registry editor, navigate to registry key

`\HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\FsiDrv`

and change the following values:

- SilentModeEnabled = 1
- SilentModeAdmins = *special_admins*

The *special_admins* variable defines a list of user names that are allowed to access the computer while CA Access Control services are down.

Use a new line for each user. Whether specified or not, *SYSTEM* is always a maintenance mode user.

Note: On Windows 2000 and Windows NT you cannot use regedit to edit the SilentModeAdmins key; use Regedt32.exe instead.

3. Start CA Access Control services with “seosd -start” command from the command shell, or using an option from Windows Start menu.

Now, if CA Access Control services are down, only users that are listed under SilentModeAdmins registry key will have access to the computer, and all other users will receive a deny to any attempt of activity.

Bypass Drivers

To specify that some drivers can operate without needing to submit operations for CA Access Control authorization checks, define a bypass for these drivers. For example, define a bypass for your antivirus program driver so that it can open files for scanning without CA Access Control authorization checks. Without the bypass, the driver might cause a deadlock with CA Access Control.

Note: A bypass for a current version of Trend Micro™ PC-cillin Antivirus is configured out-of-the-box.

To bypass drivers

1. Set the BypassDriversCount registry entry value to the number of drivers you want to define a bypass for.

You can find this entry in the FsiDrv key of the CA Access Control registry.

Note: You must stop CA Access Control before you can change CA Access Control registry entries.

2. For each driver you want to bypass:
 - a. Create a registry entry of type REG_SZ named *DriverName_drvNumber*
The first entry should be *DriverName_0* and the last *DriverName_X*, where *X* is *BypassDriversCount - 1*.
 - b. Edit each *DriverName_drvNumber* entry so that its value is the name of the program driver you want to bypass.
The value should be the name of the driver only (for example, *thisdrv.sys*).
3. Restart CA Access Control.
CA Access Control reloads and bypasses the drivers you defined in the registry.

Example: Bypass Drivers to Resolve Compatibility Issues

This example resolves a compatibility issue an antivirus product has with CA Access Control by defining the antivirus drivers (*avDriverA.sys* and *avDriverB.sys*) for bypass. You set registry entries for driver bypass in the CA Access Control registry tree under the *FsiDrv* key:

HKLM\SOFTWARE\ComputerAssociates\AccessControl\FsiDrv

Set the registry entries as follows:

Name	Type	Data
BypassDriversCount	REG_DWORD	2
DriverName_0	REG_SZ	avDriverA.sys
DriverName_1	REG_SZ	avDriverB.sys

The *BypassDriversCount* registry entry value of 2 tells CA Access Control to look for two drivers to bypass. Each *DriverName_drvNumber* registry entry value defines a driver to bypass.

Toggle Driver Interception

You can activate or deactivate the interception of the CA Access Control filter driver.

Note: When the interception is deactivated, CA Access Control protection that is not enforced by the filter driver still applies. This includes password quality checks, login events, Windows services events, STOP, and so on.

To activate interception, set UseFsiDrv to 1; to deactivate, set UseFsiDrv to 0.

You can find this configuration setting in the AccessControl section of the CA Access Control registry.

After you change this registry value, restart CA Access Control services.

Disable CA Access Control Kernel Interceptions

You can disable the following CA Access Control interceptions at the kernel level:

- network interception
- process interception
- registry interception
- file interception

Even when the network, process, registry, and file classes are disabled and you are not using those classes to intercept kernel activity, the network, process, registry, and file interception processing code is initiated at boot time and working at run time, affecting performance. To improve performance, you can disable one or more interceptions from initiating at boot time.

To disable CA Access Control interceptions at the kernel level

1. Create one or more of the following registry entries of type REG_DWORD and set the value of one or more entries to 1.
 - DisableNetworkInterception—disables network interception
 - DisableProcessInterception—disables process interception
 - DisableRegistryInterception—disables registry interception
 - DisableFileInterception—disables file interception

The entries must be created under the following registry key:

HKLM\SYSTEM\CurrentControlSet\Services\drveng\Parameters

2. Reboot the computer.

CA Access Control reloads without initializing the disabled interception types.

Stack Overflow Protection

Stack Overflow Protection (STOP) is a feature that prevents hackers from creating and exploiting stack overflow to break into systems. Stack overflow enables hackers to execute arbitrary commands on remote or local systems, many times as the administrator. They do this by exploiting bugs in the operating system or other programs. These special types of bugs permit users to overwrite the program stack, changing the next command to be executed.

STOP works by intercepting crucial operating system calls to each application on the computer. Each call is then given an initial analysis before being sent for further analysis if it seems suspicious. Further analysis is performed using data from the STOP configuration and signature files.

Enable STOP

STOP lets you prevent hackers from creating and exploiting stack overflow to break into your systems. You can enable STOP when installing CA Access Control. Alternatively, you can enable STOP manually.

To enable STOP

1. Enter the following command:

```
secons -s
```

CA Access Control shuts down.

2. Set the STOP *OperationMode* registry entry to 1.

The registry entry can be found in the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\Instrumentation\PlugIns\StopPlg
```

When CA Access Control is started, STOP modules will load and STOP will be enabled on the computer.

3. (Optional) Adjust STOP configuration using registry entries in the following keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\AccessControl\Instrumentation\PlugIns\StopPlg
```

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\STOP
```

Note: For more information about STOP registry settings, see the *Reference Guide*.

4. the following command:

```
seosd -start
```

CA Access Control starts up.

Configure STOP for Receiving Signature File Updates

You can make sure that all computers in your environment have the latest STOP information required for preventing stack overflow. You can do this by updating the STOP signature file on a central computer and setting up your computers to regularly retrieve the file.

To configure STOP for receiving signature file updates

1. Enter the following command:

```
secons -s  
CA Access Control shuts down.
```

2. Set the *STOPSignatureBrokerName* registry entry to the host name the computer you want to CA Access Control to retrieve the signature file from.

The registry entry can be found in the following key:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\STOP
```

When you start CA Access Control (and then at a defined interval), it retrieves the STOP signature file from the specified computer.

3. Set the *STOPUpdateInterval* registry entry to the interval at which you want the signature file updated.

CA Access Control retrieves the signature file from the specified computer at the specified interval.

4. (Optional) Adjust STOP configuration using registry entries in the following key:

```
HKEY_LOCAL_MACHINE\Software\ComputerAssociates\AccessControl\STOP
```

Note: For more information about STOP registry settings, see the *Reference Guide*.

5. the following command:

```
seosd -start  
CA Access Control starts up.
```

Note: You can retrieve the signature file from any host using the eACSigUpdate utility. For more information about this utility, see the *Reference Guide*.

Chapter 12: Configuring Settings

CA Access Control lets you manage CA Access Control endpoint configuration settings remotely. To do this you can use CA Access Control Endpoint Management or the selang config environment.

This section contains the following topics:

[Configuration Settings](#) (see page 169)

[Change Configuration Settings](#) (see page 169)

[Change Audit Configuration Settings](#) (see page 170)

Configuration Settings

CA Access Control stores endpoint and Policy Model configuration settings it uses in:

- The Windows registry on Windows computers
- Initialization files (.ini) on UNIX computers

Note: For information about the configuration settings you can make and what they mean, see the *Reference Guide*.

Change Configuration Settings

To affect how CA Access Control and any Policy Models work, you need to make changes to the configuration settings.

To change configuration settings

1. In CA Access Control Endpoint Management, do as follows:

- a. Click Configuration.
- b. Click Remote Configuration.

The Remote Configuration page appears.

2. In the Remote Configuration Sections pane on the left, expand the configuration tree as required to reveal the section that contains the configuration setting you want modify, then click that section.

The Section: *sectionName* System Tokens page appears, displaying all the configuration settings in it.

3. Locate and edit the configuration settings as required, then click Save Tokens.

The changed configuration setting is saved.

Change Audit Configuration Settings

To affect how CA Access Control generates and stores audit records, you need to make changes to the settings in the audit configuration files. You use `selang` commands to change the settings in the audit configuration files.

To change audit configuration settings

1. (Optional) If you are using `selang` to connect to a remote host, connect to the host using the following command:

```
host host_name
```

2. Move to the `config` environment using the following command:

```
env config
```

3. Use the `editres config` command to modify the configuration settings as required.

The audit configuration settings are changed.

Example: Modify Audit Configuration File

The following example adds a line to the audit configuration file:

```
er CONFIG audit.cfg line+("FILE;*;Administrator;*;R;P")
```