CA Access Control

UNIX エンドポイント管理ガイド

r12.5

CA 製品リファレンス

このマニュアルが参照している CA の製品は以下のとおりです。

- CA Access Control Premium Edition
- CA Access Control
- CA Single Sign-On（CA SSO）
- CA Top Secret®
- CA ACF2™
- CA Audit
- CA Network and Systems Management （CA NSM、旧 Unicenter NSM and Unicenter TNG）
- CA Software Delivery （旧 Unicenter Software Delivery）
- CA Enterprise Log Manager
- CA Identity Manager

ドキュメントの表記規則

CA Access Control のドキュメントには、以下の規則があります。

形式	意味
等幅フォント	コードまたはプログラムの出力
斜体	強調または新規用語
太字	表示されているとおりに入力する必要のある要素
スラッシュ（/）	UNIX および Windows のパスの記述で使用される、プラットフォームに依存しないディレクトリの区切り文字

また、本書では、コマンド構文およびユーザ入力の説明に（等幅フォントで）以下の特殊な規則を使用します。

形式	意味
斜体	ユーザが入力する必要のある情報
角かっこ（[]）で囲まれた文字列	オプションのオペランド

形式	意味
中かっこ({})で囲まれた文字列	必須のオペランド セット
パイプ(|)で区切られた選択項目	代替オペランド(1 つ選択)を区切ります。 たとえば、以下の例は「ユーザ名またはグループ名のいずれか」を意味します。 {username|groupname}
...	前の項目または項目のグループが繰り返し可能なことを示します
下線	デフォルト値
スペースに続く、行末の円記号(¥)	本書では、コマンドの記述が 1 行に収まらない場合があります。このような場合、行末の空白とそれに続く円記号(¥)は、そのコマンドが次の行に続くことを示します。 **注**: このような円記号はコピーしないでください。また、改行はコマンドに含めないようにしてください。 これらの文字は、実際のコマンド構文の一部ではありません。

例: コマンドの表記規則

以下のコードは、本書でのコマンド表記規則の使用方法を示しています。

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

この例の内容

- 標準的な等幅フォントで表示されているコマンド名(ruler)は表示されているとおりに入力します。
- 斜体で表示されている className オプションは、クラス名(USER など)のプレースホルダです。
- 2 番目の角かっこで囲まれた部分を指定しなくても、コマンドは実行できます。この部分は、オプションのオペランドを示します。
- オプションのパラメータ(props)を使用する場合は、キーワード all を選択するか、またはカンマで区切られたプロパティ名を 1 つ以上指定します。

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

マニュアルの変更点

以下のマニュアルの更新は、本書のリリース r12.0 SP1 以降に行われたものです。

- ユーザまたはグループの管理 (38 ページ) – この更新されたトピックでは、画面のサンプルが示されます。
- 所有者権限 (197 ページ) – この更新されたトピックでは、承認プロセスで、リソースが属するリソース グループの所有者が考慮されることが確認されます。 このコード変更は、CA Access Control r12 で導入されました。
- クラスを警告モードに設定する (180 ページ) – この更新されたトピックでは、CA Access Control エンドポイント管理 の手順について説明します。
- CA Access Control がユーザの監査モードを決定する方法 (175 ページ) – この更新されたトピックでは、システム全体に適用される、新しい監査モードについて説明します。
- 設定の変更 (245 ページ) – この更新されたトピックでは、CA Access Control エンドポイント管理 の手順について説明します。

このリリースから、以下の章が削除されました。

- サンプル ポリシーのデプロイ – この更新された章は、現在、「エンタープライズ管理ガイド」にあります。

  サンプル ポリシーは、CA Access Control の変数を使用するため、もはや selang を使用した直接デプロイに適さなくなりました。

目次

第 1 章: 概要

このセクションには、以下のトピックが含まれています。

本書の内容

本書では、CA Access Control UNIX 版に採用されているさまざまな概念について説明します。CA Access Control UNIX 版は、オープン システムに統合的なセキュリティ ソリューションを提供する製品です。 本書では、UNIX エンドポイントの管理タスクと概念について説明します。

また、エンタープライズ管理機能、レポート機能、および拡張ポリシー管理機能を備えた CA Access Control Premium Edition についても説明します。

用語を簡潔に示すために、本書の全体を通してこの製品を CA Access Control と呼びます。

本書の対象読者

本書は、CA Access Control で保護される環境の実装およびメンテナンスを担当するセキュリティ管理者およびシステム管理者を対象にしています。

第 2 章: エンドポイントの管理

CA Access Control は、オペレーティング システムと動的に連携し、アクティブで統合的なセキュリティ ソリューションをオープン システムに提供するソフトウェアです。 ファイルのオープン、ユーザ ID の変更、ネットワーク サービスの取得など、セキュリティ保護が必要な操作をユーザが要求するたびに、CA Access Control は各イベントをリアルタイムでインターセプトし、その妥当性を検証してから、オペレーティング システム(OS)標準機能に制御を渡します。

このセクションには、以下のトピックが含まれています。

CA Access Control とは何か

CA Access Control は、ネイティブ プラットフォームのセキュリティ管理を行うための強力なツールを提供し、企業のセキュリティ要件に合わせて完全にカスタマイズできるセキュリティ ポリシーの実装を可能にします。 CA Access Control を使用すると、ネイティブのオペレーティング システムでは実現できない強力なセキュリティをユーザ、グループ、およびリソースに対して提供できます。 また、組織全体のセキュリティを集中管理し、マルチプラットフォーム環境において Windows と UNIX のセキュリティ ポリシーを統合できます。

UNIX に保護が必要な理由

多くのオペレーティング システムには、さまざまな技術を使用したアクセス制御機能が用意されています。 確立されたメインフレーム オペレーティング システムである IBM の z/OS には、SAF(System Authorization Facility、システム許可機能)が組み込まれています。SAF は、ユーザの権限を検証するためにオペレーティング システム自体が発行する一連のコールです。

z/OS 環境では、アクセス制御ソフトウェアによって SAF コールのリターン コードが設定されます。z/OS はこのコードに従ってアクセスを許可または拒否します。 設定されるリターン コードは、セキュリティ管理者がセキュリティ データベースに定義したアクセス ルールおよびアクセス ポリシーに基づいて決定されます。

OS/2 などの他のオペレーティング システムでも、アクセス制御のために同様の技術が使用されています。 OS/2 の SES(Security Enabling Services)というアクセス制御モジュールは、z/OS の SAF と同じ概念に基づいています。

しかし、残念ながら、UNIX ベースのオペレーティング システムはこのように設計されていません。 許可の決定は主にファイル アクセスに対して行われ、ファイルの i-node エントリの 9 ビット(rwx-rwx-rwx)を使用して、オペレーティング システム自体によって決定されます。 SAF とは異なり、イベント インターセプトの exit ポイントは用意されていません。 したがって、メインフレーム タイプのセキュリティ パッケージの機能よりも複雑なセキュリティ機能を実行するには、さらに高度なセキュリティが必要です。

機能

CA Access Control には、アクセス ルール データベース、監査ログ、管理ツールなどの一般的なセキュリティ機能に加えて、保護の対象となるオペレーティング システム イベントをインターセプトする機能が用意されています。 CA Access Control は、さまざまなオペレーティング システムで動作する必要があるため、メモリ内のイベントをインターセプトします。 システム ファイルは変更されないので、オペレーティング システムに対する変更はありません。

保護の対象

CA Access Control は、以下のエンティティを保護します。

- **ファイル**

  特定のファイルにアクセスする権限があるか?

  CA Access Control は、ファイルへのユーザのアクセスを制限します。 ユーザに対して、READ、WRITE、EXECUTE、DELETE、RENAME などのアクセス権限を 1 種類以上与えることができます。 アクセス権限は、個々のファイルに対して、または類似した名前を持つファイルの集合に対して指定できます。

- **端末**

  特定の端末を使用する権限があるか?

  このチェックは、ログイン プロセスで行われます。 CA Access Control データベースに個々の端末または端末グループを定義し、アクセス ルールにより、その端末または端末グループの使用を許可されているユーザまたはユーザ グループを指定できます。 端末を保護することによって、強力な権限を持つユーザ アカウントのログインに未許可の端末が使用されることを確実に防止します。

- **ログイン時間**

  ユーザには、特定の曜日の特定の時間にログインする権限があるか?

  通常、エンド ユーザは平日の勤務時間帯にのみ端末を使用します。そのため、平日の曜日と時間帯によるログイン制限、および休日のアクセス制限を行うことによって、ハッカーやその他の無許可のアクセサから端末を保護できます。

- TCP/IP

  相手の端末には、ローカル コンピュータから TCP/IP サービスを受け取る権限があるか? 相手の端末には、ローカル コンピュータに TCP/IP サービスを供給する権限があるか? 相手の端末は、ローカル端末のすべてのユーザからサービスを受け取ることを許可されているか?

  オープン システムの長所は、コンピュータとネットワークの両方がオープンであるという点ですが、これは同時に短所でもあります。 いったんコンピュータが外部に接続されると、故意または過失により、外部ユーザがシステムに侵入したり、そのユーザが行った行為が損害をもたらしたりする危険が発生します。 CA Access Control には、「ファイアウォール」が用意されており、ローカルの端末やサーバが不特定の端末へサービスを提供することを防止します。

- **複数ログイン権限**

  ユーザは他の端末からログインできるか?

  同時ログインとは、ユーザが複数の端末からシステムにログインできることを意味します。 CA Access Control では、1 人のユーザが複数の端末から同時にログインすることを防止できます。 これにより、すでにログインしているユーザのアカウントで外部からの侵入者がログインすることを防止できます。

- **ユーザ定義エンティティ**

  標準エンティティ(TCP/IP サービスや端末など)および機能エンティティ(トランザクションの実行やデータベース内のレコードへのアクセスなどの抽象オブジェクト)の両方を定義して保護できます。

- **管理者権限**

  CA Access Control には、管理者権限をオペレータに委任する方法、および管理者権限自体を制限する方法が用意されています。

- **ユーザ ID 一時変更**

  ユーザには、そのユーザ ID を一時変更する権限があるか?

  UNIX の setuid システム コールは、オペレーティング システムが提供するサービスの中で最も慎重に扱うべきサービスの 1 つです。setuid システム コールは CA Access Control によってインターセプトされ、ユーザに ID の置換を実行する権限があるかどうかをチェックします。 ユーザ ID 一時変更権限のチェックには Program Pathing などが使用されます。Program Pathing では、特定のプログラムを使用した場合にのみ、ユーザはユーザ ID を一時的に変更することができます。この Program Pathing は、root ユーザになって root のアクセス権を取得できるユーザを制御する際に特に重要です。

- **グループ一時変更**

  ユーザには、newgrp(グループ一時変更)コマンドを発行する権限があるか?

  グループ一時変更の保護は、ユーザ ID 一時変更保護と同様に行われます。

- **setuid プログラムおよび setgid プログラム**

  特定の setuid または setgid プログラムを信頼できるか? ユーザには、このプログラムを起動する権限があるか?

  セキュリティ管理者は、setuid または setgid 実行可能ファイルとなっているプログラムをテストし、これらのプログラムにアクセス権の不正取得に利用される可能性があるセキュリティ ホールがないことを確認できます。 テストで安全とみなされたプログラムは、trusted プログラムとして定義されます。 CA Access Control の自己防衛機能モジュール(CA Access Control Watchdog ともいう)は、ある特定の時点で制御の対象になっているプログラムを認識し、そのプログラムが、trusted と分類された後に変更または移動されたかどうかをチェックします。 trusted プログラムが変更または移動された場合、その時点で trusted とはみなされなくなり、CA Access Control はプログラムの実行を許可しません。

さらに、CA Access Control では、以下のような作為的または偶発的な脅威に対して防御を行います。

- **強制終了**

  CA Access Control では、重要なサーバやサービス、またはデーモンを強制終了から保護できます。

- **パスワード攻撃**

  CA Access Control はさまざまなタイプのパスワード攻撃からパスワードを保護します。サイトのパスワード定義ポリシーを適用し、パスワードの盗用による侵入を検知します。

- **不適切なパスワード**

  CA Access Control のポリシーでは、十分な品質のパスワードを作成して使用することをユーザに強制するルールが定義されます。 CA Access Control では、ユーザが基準に合ったパスワードを作成して使用することを確実にするために、最長および最短のパスワード有効期限の設定、特定の語句の使用制限、文字の繰り返しの禁止、およびその他の制限事項の適用を行うことができます。 パスワードを長期間継続して使用することは認められません。

- **アカウント管理**

  CA Access Control のポリシーによって、休止状態のアカウントの適切な処理が保証されます。

- **ドメイン管理**

  CA Access Control は、NIS ドメインおよび NIS 以外のドメインの両方にパスワード保護を実装してセキュリティを強化できます

保護の方法

CA Access Control は、オペレーティング システムの初期化が終了するとただちに開始されます。 CA Access Control によって、保護の必要なシステム サービスにフックが設定されます。 このようにして、サービスが実行される前に CA Access Control に制御が渡されます。 CA Access Control によって、サービスの使用をユーザに許可するかどうかが決定されます。

たとえば、CA Access Control によって保護されているリソースにユーザがアクセスしようとするとします。 このアクセス要求によって、カーネルに対してリソースのオープンを指示するシステム コールが生成されます。 そのシステム コールは CA Access Control によってインターセプトされ、アクセスを許可するかどうかが決定されます。 アクセスが許可された場合は、通常のシステム サービスに制御が渡されます。アクセスが許可されない場合は、システム コールをアクティブにしたプログラムに、permissiondenied 標準エラー コードが返され、システム コールの処理が終了します。

これは、データベースに定義されたアクセス ルールとポリシーに基づいて決定されます。データベースには、アクセサとリソースという 2 種類のオブジェクトが定義されています。アクセサとは、ユーザおよびグループのことです。 リソース とは、ファイルやサービスなど、保護対象のオブジェクトのことです。 データベース内の各レコードには、アクセサまたはリソースが定義されています。

各オブジェクトはクラスに属します。クラスは、同じタイプのオブジェクトの集合です。 たとえば、TERMINAL は、CA Access Control によって保護されている端末（ワークステーション）であるオブジェクトを含むクラスです。

クラスのアクティブ化

CA Access Control には、CLASS がデータベース内でアクティブまたは非アクティブのいずれであるかに関する情報が格納されます。 CA Access Control を起動すると、アクティブなクラスのリストが SEOS_syscall に渡されます。したがって、CA Access Control が常にこれらのクラスをインターセプトする必要はありません。 CA Access Control がクラスをインターセプトするのは、ユーザがクラスのアクティビティ ステータスを変更した場合のみです。 クラスがアクティブでない場合、リソースへのアクセスはインターセプトされません。

FILE、HOST、TCP、CONNECT、および PROCESS クラスについては、アクティブでないクラスのインターセプトを省略できます。

アクセサ エレメント

各ユーザは、アクセサ エレメント(ACEE)として表されます。ACEE は、データベースに格納されているユーザのレコードをメモリ内に反映したものです。 CA Access Control は、ログイン プロセス時にアクセサ エレメントを作成します。 アクセサ エレメントは、ユーザのプロセスと関連付けられます。 CA Access Control によって保護されているシステム サービスをプロセスが要求するたびに、またはプロセスがリソースにアクセスするために暗黙的な要求を発行するたびに、CA Access Control はそのリソースのレコードにアクセスします。 そして次に、以前に作成されたアクセサ エレメントの情報(ユーザのセキュリティ レベル、モード、グループなど)から、ユーザがリソースへのアクセスを許可されているかどうかを判断します。

ネイティブ セキュリティの拡張

以下の CA Access Control の機能により、ネイティブ セキュリティが拡張されます。

スーパーユーザ アカウントの制限

通常、UNIX システムの root アカウントや Windows システムの Administrator アカウントなどオペレーティング システムを管理するユーザ(管理者)はシステム セットアップ時に自動的に作成される、事前定義されたアカウントです。 事前定義された各アカウントは、一連のシステム機能のセットを実行します。

root または Administrator のアカウントを持つユーザは、ユーザの作成、削除、および変更から、サーバのロック、環境設定の変更、およびシャットダウンまで、広範なタスクを実行できます。

これらのオペレーティング システムにおけるセキュリティ上の主なリスクの 1 つは、権限のないユーザがこれらのアカウントの持っている制御権を手に入れる可能性があることです。 このような事が発生した場合、システムは重大な危険にさらされることになります。

CA Access Control では、これらのアカウントに与える権限を制限して、これらのアカウントをメンバとして持つユーザ グループに属するユーザの権限を制限することができます。これにより、オペレーティング システムの脆弱性をカバーします。

CA Access Control 管理者

CA Access Control のインストール時には、1 人以上の CA Access Control 管理者の名前を設定する必要があります。 CA Access Control 管理者には、ルール データベースのすべてまたは一部を変更する権限があります。 すべての権限を持つ管理者を最低 1 人は設定する必要があります。 この管理者は、アクセス ルールを自由に変更または作成することができ、管理者のレベルを指定できます。

システムのユーザを定義した後、管理者以外のユーザに ADMIN 属性を割り当てることによって、管理者権限を割り当てることができます。

注: ADMIN 属性が割り当てられたユーザには、強力な権限が与えられます。 このため、ADMIN ユーザの数は厳しく制限する必要があります。 また、1 人以上の CA Access Control 管理者の設定が終了した後に、スーパーユーザから ADMIN 属性を削除して、ネイティブのスーパーユーザの役割と CA Access Control 管理者 の役割を分離する方法もお勧めします。

CA Access Control では、常に最低 1 人のユーザがデータベースを管理する権限を持つ必要があるため、ADMIN 属性を持つ最後のユーザを削除することはできません。

CA Access Control 管理者がこのワークステーションから他のホストを管理する可能性がある場合は、そのホスト上のデータベースに、このワークステーションからの READ アクセス権と WRITE アクセス権の両方を管理者に与えるルールが定義されていることを確認してください。

サブ管理

CA Access Control には、サブ管理機能があります。 CA Access Control 管理者はこの機能を使用することで、一般ユーザに対して特定のクラスを管理できるようにする特定の権限を与えることができます。 このようなユーザをサブ管理者といいます。

たとえば、特定のユーザに対して、ユーザとグループを管理できる権限を与えることができます。

また、特定のクラスに対してだけではなく、そのクラスの指定されたレコードに対してアクセス権を許可することにより、より高いレベルのサブ管理を指定することもできます。

一般ユーザに与える管理者権限

CA Access Control では、管理者 グループのメンバでなくても管理タスクを実行できるように、必要な権限を一般ユーザ(管理者以外)に与えることができます。 このような細かい方法でタスクを委任できる(つまり、管理権限を付与できる)機能は、CA Access Control の最も重要な機能の 1 つです。

- SUDO クラスのレコードには、コマンド スクリプトが格納されています。ユーザは、付与された権限でそのスクリプトを実行できます。
- data プロパティの値はコマンド スクリプトです。 この値は、省略可能なスクリプトパラメータ値を追加して変更することができます。
- SUDO クラスの各レコードは、あるユーザが別のユーザの権限を借用できるようにするためのコマンドを識別します。
- SUDO クラス レコードのキーは、SUDO レコードの名前です。 この名前は、ユーザが SUDO レコードでコマンドを実行する際に、コマンド名の代わりに使用されます。

Program Pathing

Program Pathing は、ファイルにアクセスするには特定のプログラムを介さなければならないことを要求する、ファイルに関連するアクセス ルールです。 Program Pathing により、機密ファイルのセキュリティを大幅に強化できます。 CA Access Control の Program Pathing を使用すると、システム内のファイルに対する保護を強化できます。

B1 セキュリティ レベル認証

CA Access Control には、セキュリティ レベル、セキュリティ カテゴリ、およびセキュリティ ラベルという「Orange Book」の B1 レベルの機能があります。

- データベースのアクセサとリソースには、セキュリティ レベルを割り当てることができます。 セキュリティ レベルは、1 から 255 までの整数です。 アクセサのセキュリティ レベルが、リソースに割り当てられたセキュリティ レベル以上である場合にのみ、アクセサはリソースにアクセスできます。
- データベースのアクセサとリソースは、1 つ以上のセキュリティ カテゴリに属することができます。 リソースに割り当てられているすべてのセキュリティ カテゴリにアクセサが属している場合のみ、そのアクセサはリソースにアクセスできます。
- セキュリティ ラベルは、特定のセキュリティ レベルを 0 個以上のセキュリティ カテゴリの集合に関連付けるための名前です。 ユーザをセキュリティ ラベルに割り当てると、セキュリティ ラベルに関連付けられたセキュリティ レベルおよびセキュリティ カテゴリの両方がユーザに設定されます。

注: Orange Book の B1 レベルの機能の詳細については、「実装ガイド」を参照してください。

エンドポイント管理

CA Access Control では、2 つの方法で、企業内のリソースを管理し、リソースにアクセスするユーザを制御することができます。

- selang – CA Access Control コマンド言語。

  selang コマンド言語を使用すると、CA Access Control データベースに定義を作成することができます。 selang コマンド言語は、コマンド定義言語です。

  注: selang の使用法の詳細については、「selang リファレンス ガイド」を参照してください。

- CA Access Control エンドポイント管理 – エンドポイント管理インタフェース。

  この Web ベースのインタフェースでは、中央の管理サーバからリモートのエンドポイントを管理することができます。

  注: CA Access Control エンドポイント管理のインストールの詳細については、「実装ガイド」を参照してください。

第 3 章: ユーザおよびグループの管理

このセクションには、以下のトピックが含まれています。

ユーザおよびグループ

CA Access Control では、アクションまたはアクセスのすべての試みが、要求を送信するユーザに代わって、実行されます。 したがって、システムのすべてのプロセスは、特定のユーザ名に関連付けられます。 CA Access Control のユーザは、ユーザ名によって識別されます。

ユーザとは、ログインできる人、またはバッチおよびデーモン プログラムの所有者すべてを指します。 CA Access Control では、アクセス試行のすべてがユーザによって実行されます。 CA Access Control は、CA Access Control データベースのユーザ情報とエンタープライズ ユーザ ストアのユーザ情報を使用できます。 ユーザ情報は、データベースの USER レコードまたは XUSER レコードのいずれかに格納されます。

注: エンタープライズ ユーザ ストアとは、ユーザやグループが格納されているオペレーティング システム内のストア (たとえば、UNIX システムの /etc/passwd や /etc/groups、Windows の Active Directory など) です。

グループは、ユーザの集合です。 グループでは、グループ内のすべてのユーザに適用する共通のアクセスルールを定義します。 グループはネストする (他のグループに属する) こともできます。 CA Access Control は、CA Access Control データベースのグループ情報とエンタープライズ ユーザ ストアのグループ情報を使用できます。 通常は、ロール (database_administrators など) に基づいて、グループを作成し、そのグループにユーザを割り当てます。

ユーザ レコードは、重要なアクセサ レコードです。 CA Access Control でグループを使用する主な目的は、一度にグループ内のすべてのユーザにアクセス権限を割り当てることです。 アクセス権限を個々のユーザに別々に割り当てるよりも一度に割り当てるほうが簡単で、エラーが発生する可能性も低くなります。

アクセサに関する情報の格納場所

CA Access Control が使用するユーザ情報とグループ情報は、CA Access Control データベースとホスト オペレーティング システムの両方に格納されます。 ホスト オペレーティング システム内の情報は、エンタープライズ ユーザ ストア、または単にエンタープライズ ストアと呼ばれます。 デフォルトでは、CA Access Control は、エンタープライズ ストアを使用しないように設定されています。 ただし、CA Access Control データベースに定義されているユーザまたはグループが見つからない場合は、エンタープライズ ストアに定義されているユーザおよびグループ メンバシップを検索して、その情報を使用するように、CA Access Control を設定することもできます。

注: CA Access Control は、エンタープライズ ストアの情報を使用しますが、エンタープライズ ストアに書き込みを行うのはネイティブ環境で selang コマンドが使用された場合のみです。

権限をチェックする際、CA Access Control は必ず自身のデータベースに定義されているアクセサをチェックしてから、エンタープライズ ストアを調べます。CA Access Control データベースに定義されているユーザと同じ名前のエンタープライズ ユーザがいる場合、CA Access Control はそのエンタープライズ ユーザを無視します。

CA Access Control によるユーザ レコードの検索方法

ユーザがログインすると、CA Access Control は、そのユーザに関連付けられたレコードを見つけるまで、以下の順序で検索を実施します。

1. CA Access Control は、自身のデータベースに定義されているユーザを検索します。
2. CA Access Control は、自身のキャッシュで、そのエンタープライズ ユーザを検索します。

   ネットワークが停止した場合は、オペレーティング システム(OS)により、ユーザは OS 内にキャッシュされた認証情報を使用してログインできます。 CA Access Control キャッシュの目的は、このような場合に CA Access Control がエンタープライズ ユーザのレコードも使用できるようにすることです。
3. CA Access Control は、オペレーティング システムを使用して、エンタープライズ ユーザ ストアで、そのエンタープライズ ユーザを検索します。
4. CA Access Control がデータベース内またはエンタープライズ ストア内でユーザに関連付けられたレコードを見つけられない場合、CA Access Control はユーザに _undefined USER レコード内の属性を割り当てます。

エンタープライズ ユーザ ストアとの統合

通常は、エンタープライズ ユーザ ストアに定義されているグループとユーザを使用するように CA Access Control を設定します。

デフォルトで、このように CA Access Control を設定しておけば、エンタープライズ ユーザまたはエンタープライズ グループを参照するアクセス ルールが作成されたときや、ユーザがオペレーティング システムにログインしたときに、ユーザまたはグループのレコードが事前に存在していなかった場合に、CA Access Control は自身のデータベースにそのユーザまたはグループのレコードを作成します。 これらのレコードには、XUSER クラス(エンタープライズ ユーザの場合)または XGROUP クラス(エンタープライズ グループの場合)が割り当てられます。 これらのクラスは、CA Access Control がアクセス ルールを適用する場合に必要とするプロパティを保持しています。 CA Access Control が必要に応じて作成するため、手動で管理する必要はありません。

CA Access Control がエンタープライズ ユーザ ストアから取得するエンタープライズ ユーザまたはエンタープライズ グループのプロパティは、名前と、グループ メンバシップ のプロパティのみです。

エンタープライズ ストアでアクセサを管理するためのガイドライン

エンタープライズ ユーザ ストアでアクセサを管理する場合は、以下のセクションに記載されているガイドラインを確認してください。

データベースに定義する必要があるユーザおよびグループ

CA Access Control では、一部のユーザおよびグループを、エンタープライズ ユーザ ストアではなく、自身のデータベースに定義する必要があります。 これらのユーザおよびグループは、以下のとおりです。

- 事前定義済みユーザ(35 ページ)
- 事前定義済みグループ(36 ページ)
- CA Access Control 管理者
- プロファイル グループ
- 論理ユーザ

エンタープライズ ユーザの使用制限

CA Access Control は、エンタープライズ ユーザの使用に以下の制限を適用します。

- エンタープライズ ユーザの名前が、データベースに定義されるユーザと同じ場合、CA Access Control でそのエンタープライズ ユーザを作成、または参照することはできません。
- selang AC 環境を使用して、エンタープライズ ユーザを作成、削除、または変更することはできません。
- エンタープライズ ユーザを論理ユーザとして使用することはできません。
- デフォルトでは、ユーザがエンタープライズ ユーザ ストアに事前に定義されていない限り、CA Access Control でエンタープライズ ユーザを作成することはできません。 ただし、UNIX システム上でこの動作を有効または無効にすることができます。

詳細情報:

UNIX 上で XUSER レコードを作成する前のエンタープライズ ストア チェックの有効化/無効化(32 ページ)

エンタープライズ グループの使用制限

CA Access Control は、エンタープライズ グループの使用に以下の制限を適用します。

- selang AC 環境内で、エンタープライズ グループを作成または削除することはできません。
- selang AC 環境内で、エンタープライズ グループのメンバシップを変更することはできません。
- エンタープライズ グループをプロファイル グループ(37 ページ)として使用することはできません。

エンタープライズ ユーザおよびグループの有効化/無効化

デフォルトでは、CA Access Control は、エンタープライズ ユーザ ストアに定義されているグループおよびユーザを使用できませんが、CA Access Control による使用を有効に設定することができます。 CA Access Control の以前のバージョンとの互換性が必要な場合を除き、この機能は有効にすることをお勧めします。

CA Access Control がエンタープライズ ユーザとグループを使用できるようにするには、設定 osuser_enabled を 1 に設定します。 この動作を無効にするには、osuser_enabled の値を 0(ゼロ)に設定します。

例: Windows 上でエンタープライズ ユーザとグループを有効にする

Windows 上でエンタープライズ ユーザとグループの使用を有効にするには、以下のレジストリ設定を指定します。

- キー: HKLM¥ComputerAssociates¥AccessControl¥OS_user
- 名前: osuser_enabled
- タイプ: REG_DWORD
- 値: 1

例: UNIX 上でエンタープライズ ユーザとグループを有効にする

以下のコマンドを実行して、CA Access Control を停止してから、UNIX 上でエンタープライズ ユーザとグループの使用を有効にし、CA Access Control を再起動します。

```
secons -s
seini -s OS_User.osuser_enabled 1
seload
```

エンタープライズ ユーザのログイン時の XUSER レコードの作成の有効化/無効化

CA Access Control で、エンタープライズ ユーザの使用が有効になっている場合、デフォルトでは、ユーザがログインしたときにそのユーザのレコードが(XUSER クラスに)作成されます。 ただし、毎日同じ時刻に数千人のユーザがログインする場合など、このレコードを作成したくないこともあります。

ユーザがログインしたときに CA Access Control が XUSER レコードを作成しないようにするには、設定 create_user_in_db の値を 0(ゼロ)に変更します。 この動作を再び有効にするには、この値を 1 に設定します。

例: エンタープライズ ユーザが Windows にログインしたときの XUSER レコードの自動作成を無効にする

Windows 上で CA Access Control でのエンタープライズ ユーザ レコードの自動作成を無効にするには、以下のレジストリ設定を指定します。

- キー: HKLM¥ComputerAssociates¥AccessControl¥OS_user
- 名前: create_user_in_db
- タイプ: REG_DWORD
- 値: 0

例：エンタープライズ ユーザが UNIX にログインしたときの XUSER レコードの自動作成を無効にする

以下のコマンドを実行して、CA Access Control を停止してから、UNIX 上で XUSER レコードの自動作成を無効にし、CA Access Control を再起動します。

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

UNIX 上で XUSER レコードを作成する前のエンタープライズ ストア チェックの有効化/無効化

ユーザがエンタープライズ ユーザ ストアに定義されていない場合は、CA Access Control でエンタープライズ ユーザを作成することができます。 Windows では、ユーザが Windows のユーザ ストアに存在しない限り、CA Access Control でエンタープライズ ユーザを作成することはできません。 UNIX のデフォルト動作は Windows とは逆です。 ただし、UNIX では、このデフォルト動作を有効または無効にすることができます。

チェックを無効にする（したがって、同等のエンタープライズ ユーザが存在しない場合に CA Access Control が XUSER レコードを作成できるようにする）には、verify_osuser の設定値を 0 に変更します。 チェックを適用するには、この値を 1 に設定します。

例：エンタープライズ ユーザ ストアをチェックせずに XUSER レコードの作成を有効にする

以下のコマンド セットを実行すると、CA Access Control は停止し、エンタープライズ ストアに同等のレコードがない XUSER レコードの作成が有効になり、CA Access Control の再起動が実行されます。

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

Windows での再利用エンタープライズ ストア アカウント

再利用アカウントとは、削除された後で(同じ名前を使用して)再作成されたエンタープライズ ストアのユーザまたはグループです。 これは、たとえば、ユーザ ストアからユーザを削除した後で(ユーザが退職した場合など)、その削除されたユーザと同じ名前の新規ユーザの新規アカウントを作成するときに発生します。

再利用アカウントは、セキュリティ ホールです。名前が同一の以前のアカウントに付与されていたアクセス許可と同じアクセス許可が新規のアクセサに必ずしも必要とは限らないためです。 この問題を解決するためには、CA Access Control の許可は SID に基づいています。 つまり、アクセス許可が付与されていた削除済みのアクセサと名前が同一の新規アクセサを作成しても、その新規アクセサに対しては、以前のアクセサに付与されていた古い許可は自動的には付与されません。

重要: 再利用アカウント アクセサは、古いアクセス許可を継承しません 。 ただし、(SID ではなく)アクセサの名前を指定するデータベース アクセス ルールでは、これらのルールが適用されると思われることがあります。 この問題は、secons -checkSID コマンドを使用して解決します。

Windows での再利用エンタープライズ アカウントの解決

関連付けられたデータベース ルールを持つエンタープライズ アカウント(ユーザまたはグループ)が再利用(つまり、削除されてから、同じ名前で作成)された場合、古いデータベース ルールが新規アカウントにも適用されると思うユーザもいるでしょう。 しかし、CA Access Control の許可は SID に基づいているので、これらのルールは適用されません。新規ユーザ/グループ用の新規ルールを作成する必要があります。 新規ルールを作成するには、事前に再利用アカウントを解決しておく必要があります。

再利用エンタープライズ アカウントを解決するには、コマンド プロンプトを開き、以下のコマンドを実行します。

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

CA Access Control は、所有しているすべてのエンタープライズ ユーザ アカウント(XUSER レコード)を確認してから、すべてのグループ アカウント(XGROUP レコード)を確認し、エンタープライズ アカウントの SID とは異なる SID を持つアカウントを識別します。 CA Access Control で、命名規則 SID (accountName) に従って、これらのアカウントの名前を変更します。

これで、再利用アカウントの新規ルールを作成できます。

注: 再利用ユーザ アカウントは、ユーザがログインしたり、リソースにアクセスしようとしたときに、このように解決されます。 エンタープライズ アカウントを作成する場合は、secons -checkSID コマンドを、スケジュール タスクとして実行することをお勧めします。

例: 再利用グループ アカウント

ABCD 社のエンタープライズ ストアに、interns というグループがあります。 このグループには、9 人のメンバが所属し、productA に取り組んでいます。 管理者は、以下のように、このグループを CA Access Control に認識させ、グループのメンバがアクセスする必要があるファイルへのアクセス許可をこのグループに割り当てます。

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

interns が ABCD 社での就労期間を完了すると、エンタープライズ ストア管理者はこのグループを削除します。 3 か月後、6 人のメンバから成る新しい interns グループがエンタープライズ ストアに同じ名前で作成されます。 CA Access Control データベース内の古いルールはまだ存在するので、新しい interns グループはこの古いルールの許可を継承するように思われます。 しかし、これらのルールは古い interns グループに適用されるものなので、CA Access Control 管理者は新規グループ用の新規ルールを作成する必要があります。

このためには、管理者は、以下のように interns 再利用アカウントを識別して解決する必要があります。

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

これにより、XGROUP リソースの名前と、このリソースへのアクセス ルール参照の名前が、「SID (domain¥interns)」に変更されます。 これで、管理者は、productB に取り組む新規 interns グループ用の新規ルールを作成できます。

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

注: secons ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

データベース アクセサ

ユーザをどのように管理するかに関係なく、以下に説明するように、CA Access Control データベースに定義する必要があるアクセサがあります。

事前定義済みユーザ

CA Access Control は、以下のユーザを事前定義します。これらのユーザを削除することはできません。

_dms

拡張ポリシー管理サーバ コンポーネントのデータベース(DMS、DH リーダ、および DH ライタ)にインストールされている _dms ユーザは、policyfetcher および devcalc が DH および DMS と通信する場合に使用されます。

_seagent

_seagent は、CA Access Control が以下のような内部プロセスを実行するときのユーザ名です。

- PMDB プロセス、sepmdd
- (UNIX)偏差計算プロセス、devcalc
- ユーザおよびグループ レコード更新 の exit プロセス

_seagent ユーザには SERVER 属性が割り当てられています。

_undefined

_undefined は、CA Access Control で定義されていないすべてのユーザを表します。 _undefined を使用して、未定義のユーザを ACL に含めることができます。

+devcalc

(Windows)CA Access Control が偏差計算プロセス devcalc を実行するときのユーザ名。

+reportagent

CA Access Control がレポート エージェントを実行するときのユーザ名。

nobody

nobody ユーザは、実際のユーザに対応させることのできないユーザ レコードです。このレコードは、関連する許可をどのユーザにも付与しないルールを作成する場合に使用します。 たとえば、nobody をリソースの所有者として設定し、どのユーザも、そのレコードの所有に関連する許可を取得しないようにすることができます。

事前定義済みグループ

CA Access Control には、事前定義済みグループが用意されています。 _interactive グループと _network グループを除き、これらの事前定義済みグループには、他のグループと同じようにユーザを追加できます。

_abspath

ログイン時に _abspath グループに属しているユーザは、プログラムを起動する場合に絶対パス名を使用する必要があります。

_interactive

ユーザは、アクセスの目的でのみ、_interactive グループのメンバになります。 ユーザは、アクセスしようとしているリソースと同じホストにログインしている場合、_interactive グループのメンバになります。 CA Access Control は、_interactive グループのメンバシップを動的かつ自動的に管理します。このメンバシップを変更することはできません。

_network

これは、_interactive の補完グループです。 ユーザは、アクセスの目的でのみ、_network グループのメンバになります。 ユーザは、リソースが属するホストとは別のホストにアクセスしようとする場合、_network グループのメンバになります。 CA Access Control は、_network グループのメンバシップを動的かつ自動的に管理します。このメンバシップを変更することはできません。

_restricted

_restricted グループのユーザに対しては、ファイルはすべて（Windows の場合はレジストリ キーも） CA Access Control によって保護されます。 ファイルまたは Windows のレジストリ キーで、アクセス ルールが明示的に定義されていない場合、アクセス許可は、そのクラス（FILE または REGKEY）の _default レコードが適用されます。

注: _restricted グループに属するユーザには、処理を実行するための十分な権限が付与されない可能性があります。 このため、ユーザを _restricted グループに追加する場合は、最初に警告モードの使用を検討してください。

_surrogate

ユーザが _surrogate グループのメンバを代理として使用する場合、CA Access Control は、その代理のアクションの監査証跡として元のユーザの名前が付けられた完全なトレースを書き込みます。

例: selang を使用して _restricted グループにユーザを追加する

以下の selang コマンドは、エンタープライズ ユーザ john_smith を _restricted グループに追加します。

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

プロファイル グループ

プロファイル グループは、ユーザ プロパティのデフォルト値が収められている、CA Access Control データベースに定義されるグループです。 ユーザをプロファイル グループに割り当てた場合、そのユーザにすでに値が設定されていない限り、プロファイル グループはそのデフォルト値をユーザに提供します。

ユーザのプロファイル グループは、ユーザの作成時に指定できます。または、後でプロファイル グループにユーザを割り当てることもできます。

プロファイル グループを使用すると、管理者は、グループに割り当てる新規ユーザに対して、特定の権限が指定された標準設定を効率よく作成できます。 このセットアップでは、ユーザのホーム ディレクトリ、監査プロパティ、アクセス権限を定義する PMDB、およびプロファイル グループに関連付けられているユーザに影響を与えるさまざまなパスワード ルールなどを指定することができます。

CA Access Control がプロファイル グループを使用してユーザ プロパティを決定する方法

以下のプロセスでは、CA Access Control がプロファイル グループを使用してユーザ プロパティを指定する方法について説明します。

1. CA Access Control は、USER クラスまたは XUSER クラスのユーザのレコードにプロパティの値があるかどうかをチェックします。

   ユーザのレコードがプロパティの値を持っている場合は、CA Access Control はその値を使用します。

2. CA Access Control は、ユーザがプロファイル グループに割り当てられているかどうかをチェックします。

   ユーザがプロファイル グループに割り当てられている場合は、プロセスは続行します。 ユーザがプロファイル グループに割り当てられていない場合、CA Access Control はデフォルトのプロパティ値をユーザに割り当てます。

3. CA Access Control は、プロファイル グループがそのプロパティの値を持っているかどうかをチェックします。

   プロファイル グループがプロパティの値を持っている場合、CA Access Control はその値をユーザに割り当てます。 プロファイル グループがプロパティの値を持っていない場合、CA Access Control はデフォルトのプロパティ値をユーザに割り当てます。

   注: ユーザまたはプロファイル グループの監査プロパティが設定されていない場合、グループの監査プロパティはユーザの監査プロパティに影響を与える場合があります。

詳細情報:

CA Access Control がユーザの監査モードを決定する方法(175 ページ)

アクセサ管理

CA Access Control エンドポイント管理 または selang を使用して、データベースまたはエンタープライズ ストアのユーザまたはグループのレコードを作成、変更、および削除することができます。

ユーザまたはグループの管理

特定のアクセサのプロパティを表示または変更する場合や、アクセサを削除する場合は、まずそのアクセサを見つける必要があります。

ユーザまたはグループの管理方法

1. CA Access Control エンドポイント管理 内で、以下の操作を実行します。
   a. [ユーザ]をクリックします。
   b. [ユーザ]または[グループ]サブタブのいずれかをクリックします。

   選択したサブタブに応じて、[ユーザ]ページまたは[グループ]ページが表示されます。
2. [検索]セクションの以下のフィールドに入力します。

   ユーザ名/グループ名

   表示したいアクセサのマスクを定義します。 対象とするアクセサのフル ネームを入力するか、マスクを使用することができます。 たとえば、名前に「admin」を含むアクセサをリストするには、*admin* を使用します。

   すべてのアクセサをリストするには、アスタリスク(*)を、 1 文字を置換するには、疑問符(?)を使用します。

   ユーザ リポジトリ/グループ リポジトリ

   アクセサ リストの取得元のソースを指定します。 ソースとして、以下のいずれかを指定できます。

   - 内部アカウント - CA Access Control データベースに定義されているアクセサ。
   - エンタープライズ アカウント - 特定のエンタープライズ ユーザ ストアに定義されているアクセサ。

AC アカウント/プロファイルのみを表示

以下のように、CA Access Control データベース内にレコードがあるアカウントのみをリストするかどうかを指定します。

- [内部アカウント]を選択した場合は、CA Access Control データベース内に存在するアカウントのみをリストします(ネイティブ アカウントは含まれません)。
- [エンタープライズ アカウント]を選択した場合は、CA Access Control エンタープライズ プロファイル(XUSER レコードまたは XGROUP レコード)を持つアカウントのみをリストします。

[実行]をクリックします。

選択したリポジトリに存在するアクセサのリストが表示されます。

3. 以下のいずれかの操作を行います。
   - [表示]列で[]をクリックして、アクセサのプロパティを表示します。
   - [削除]列で[]をクリックして、アクセサを削除します。
   - アクセサの名前をクリックして、アクセサのプロパティを変更します。
   - 削除するアクセサを選択し、[削除]をクリックします。
   - [ユーザの作成]または[グループの作成]をクリックし、CA Access Control データベースに新規のユーザ レコードまたはグループ レコードを作成します。

例：リポジトリ内でのエンタープライズ ユーザの検索

以下の図は、ABC-DM1 エンタープライズ ユーザ ストア内での全ユーザの検索結果を示しています。

selang を使用したユーザ管理

エンタープライズ ユーザのレコードには、以下の selang コマンドを使用します。

- newxusr および editxusr – 新規のエンタープライズ ユーザ レコードを定義します。
- chxusr および editxusr – エンタープライズ ユーザの CA Access Control プロパティを変更します。
- find xuser – CA Access Control レコードを持つエンタープライズ ユーザをリストします。
- rmxusr – ユーザを削除します。
- show xuser – エンタープライズ ユーザの CA Access Control プロパティを表示します。

CA Access Control データベース ユーザ レコードには、以下の selang コマンドを使用します。

- newusr および editusr – 新規のユーザ レコードを定義します。
- chusr および editusr – ユーザのプロパティを変更します。
- rmusr – ユーザを削除します。
- find user – データベース ユーザをリストします。
- show user – ユーザのプロパティを表示します。

例: selang を使用してデータベースにユーザを定義する

以下の selang コマンドは、CA Access Control データベースに、セキュリティ レベルを 100 とする新規ユーザを定義します。

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

例: selang を使用してエンタープライズ ユーザのプロパティを変更する

以下の selang コマンドは、エンタープライズ ユーザ Terry に AUDITOR 属性を割り当てます。

```
chxusr Terry auditor
```

selang を使用したグループ管理

エンタープライズ グループの名前およびメンバシップを除く、任意のグループのすべてのプロパティを変更できます(名前とメンバシップの変更は、CA Access Control 内からは変更できません)。

グループ プロパティを変更したり、グループに関連付けるアクセス権を割り当てるには、CA Access Control エンドポイント管理 または以下の selang コマンドを使用できます。

- join[-] および joinx[-]

  内部グループのメンバシップを変更します。

  内部アクセサをグループに追加するには、join を使用します。 エンタープライズ グループおよびユーザを内部グループに追加するには、joinx を使用します。 アクセサを内部グループから外すには、コマンドにマイナス(-)記号を付けます。

- editgrp、newgrp、chgrp

  内部グループのメンバシップ以外のプロパティを変更します。

- editxgrp、newxgrp、chxgrp

  エンタープライズ グループのメンバシップ以外のプロパティを変更します。

- rmgrp、rmxgrp

  内部グループ、 エンタープライズ グループを削除します。

例: selang を使用してデータベースにグループを定義する

以下の selang コマンドは、データベースに新規グループ「sales」を定義します。 グループのフル ネームは「Sales Department」です。

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

例: selang を使用して、データベースに定義されているグループのプロパティを変更する

以下の selang コマンドによって、CA Access Control は、グループ AC_admins のメンバに対するすべてのイベントを監査します。

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

例: selang を使用して、ACL にエンタープライズ グループを追加する

以下の selang コマンドは、myfile という ACL にエンタープライズ グループ mygroup を追加します。

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

例：selang を使用して、データベースに定義されているグループにエンタープライズ ユーザを追加する

以下の selang コマンドは、データベースに定義されているグループ AC_admins に、エンタープライズ ユーザ mydomain¥administrator を追加します。

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

例：selang を使用して、データベースに定義されているグループにエンタープライズ グループを追加する

以下の selang コマンドは、_restricted グループにエンタープライズ グループ Guests を追加します。

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

第 4 章: リソースの管理

このセクションには、以下のトピックが含まれています。

リソース

リソースとは、アクセサがアクセスでき、アクセス ルールによって保護されるエンティティ、またはそのエンティティに対応する CA Access Control データベース レコードです。 リソースの例には、ファイル、プログラム、ホスト、端末などがあります。

CA Access Control でリソース レコードを作成する主な目的は、リソース レコードに対応するリソースのアクセス許可を定義することです。 リソースへのアクセスに必要なアクセス許可は、リソース レコードのアクセス制御リストに指定します。

リソース グループ

リソース グループは、その他のリソースから成るリストを含むリソースです。 リソース グループとは、CONTAINER、GFILE、GSUDO、GTERMINAL、または GHOST のいずれかのクラスのメンバです。

リソース グループはそれ自体がリソースであるため、そのメンバリソースに同じプロパティが割り当てられます。 したがって、リソース グループを使用するメリットは、管理の簡略化です。 リソース グループのプロパティを変更することで、すべてのメンバ リソースのプロパティを変更できます。

クラス

CA Access Control では、レコードに割り当てることのできるプロパティはレコードのクラスによって定義されます。 1 つのクラス内のすべてのレコードに、同じプロパティが割り当てられます。ただし、これらのプロパティの値は異なります。

クラスの例は、以下のとおりです。

- TERMINAL クラス。 tty1、tty などの端末のレコードが含まれます。
- FILE クラス。 ファイルのレコードが含まれます。
- PROGRAM クラス。 プログラムのレコードが含まれます。

各レコードには、レコード クラスに適したプロパティの値が保存されます。 たとえば、XUSER クラスのレコードにはエンタープライズ ユーザの勤務地や勤務時間などのプロパティが保存され、HOSTNET クラスのレコードにはネット サービスや IP アドレス データなどのプロパティが保存されます。

CA Access Control には、事前定義されたクラスが含まれています。 また、ユーザ定義クラスと呼ばれる新規クラスを定義することもできます。

クラスのデフォルト レコード

ほとんどのクラスには、デフォルト レコード(_default)を含めることができます。このレコードは、クラスのリソースのうち、データベースに対応するレコードが定義されていないリソースのアクセス タイプを指定します。

他のリソース レコードと同様に、_default レコードには、ACL および defaccess フィールドを含めることができます。 _default レコードは、USER、GROUP、CATEGORY、SECLABEL、および SEOS を除くすべてのクラスに作成できます。

UACC クラス(廃止予定)

UACC クラスの使用はお勧めしません。 クラス内のレコードに対するデフォルト値を指定するには、_default レコードを使用してください。

CA Access Control の一部の旧バージョンでは、他のクラスの _default レコードに似たレコードに対して、UACC という別のクラスを使用していました。 UACC クラスの使用はお勧めしません。_default レコードを使用する場合、UACC クラスの対応するレコードはチェックされません。 今後のバージョンでは、UACC クラスはサポートされなくなる可能性があります。

たとえば、ユーザ Henderson がプロセス store_log の強制終了(kill)を試みたとします。この場合、CA Access Control では、以下の順序で権限がチェックされます。 まず最初に、プロセス store_log がデータベースに定義されているかどうかがチェックされます。CA Access Control は、データベースで PROCESS クラスの store_log というレコードを検索します。

- 該当するレコードが見つからない場合、このプロセスは CA Access Control に定義されていません。 この場合、CA Access Control は、PROCESS クラスの _default レコード、または UACC クラスの PROCESS レコードのいずれかを使用して、Henderson が store_log を強制終了(kill)できるかどうかを判断します。
  - ユーザ Henderson が _default レコードの ACL に定義されている場合は、ACL に指定された権限が適用されます。

- Henderson が _default レコードの ACL に定義されていない場合は、_default レコードの defaccess プロパティに指定された権限が適用されます。この権限は、_default の ACL に明示的に指定されていないすべてのユーザに適用されます。

- プロセス store_log がデータベースに定義されている場合は、ユーザ Henderson がデータベースでプロセス store_log の ACL に定義されているかどうかが問題になります。
  - ユーザ Henderson がプロセス store_log の ACL に定義されている場合は、ACL に指定された権限が適用されます。
  - ユーザ Henderson が ACL に定義されていない場合は、store_log リソースのデフォルト アクセス プロパティに指定された権限が適用されます。 この権限は、リソースのデフォルト アクセスといいます。

注: _default のデフォルト アクセス(defaccess)が NONE に設定されている場合、または、_default が未指定で UACC クラスの対応するリソースのデフォルトが NONE である場合は、クラスに定義されていないリソースにアクセスを試みたアクセサは、リソースへのアクセスを拒否されます。

_default(または UACC)のデフォルト アクセス権として最上位の権限(ALL、または場合によっては READ か EXECUTE)が設定されている場合、明示的に保護されていないリソースには、すべてのユーザがアクセスできます。

事前定義されたクラス

事前定義されたクラスは、以下のタイプに分類できます。

クラス タイプ	目的
アクセサ	ユーザ、グループなど、リソースにアクセスするオブジェクトを定義します。
定義	セキュリティ ラベルやセキュリティ カテゴリなど、セキュリティ エンティティを定義するオブジェクトを定義します。
インストール	CA Access Control の動作を制御するオブジェクトを定義します。
リソース	アクセス ルールによって保護されるオブジェクトを定義します。

以下の表は、事前定義クラスの一覧です。

クラス	クラス タイプ	説明
ADMIN	定義	ADMIN 属性を持たないユーザに管理責任を委任します。 これらのユーザにグローバル権限属性を付与し、管理者権限の適用範囲を制限します。

クラス	クラス タイプ	説明
AGENT	リソース	CA Access Control には適用されません。
AGENT_TYPE	リソース	CA Access Control には適用されません。
APPL	リソース	CA Access Control には適用されません。
AUTHHOST	アクセサ	CA Access Control には適用されません。
CALENDAR	リソース	時間制限が適用されるユーザ、グループ、およびリソースの Unicenter TNG カレンダ オブジェクトを定義します。
CATEGORY	定義	セキュリティ カテゴリを定義します。
CONNECT	リソース	外部接続を保護します。 このクラスのレコードは、どのユーザがどのインターネット ホストにアクセスできるかを定義します。 CONNECT クラスをアクティブにする前に、streams モジュールがアクティブであることを確認します。
CONTAINER	リソース	他のリソース クラスにあるオブジェクトのグループを定義します。これにより、複数の異なるオブジェクトのクラスに 1 つのルールを適用する際のアクセス ルールの定義が簡略化されます。
FILE	リソース	ファイル、ディレクトリ、またはファイル名マスクを保護します。
GAPPL	リソース	CA Access Control には適用されません。
GAUTHHOST	定義	CA Access Control には適用されません。
GFILE	リソース	このクラスの各レコードは、ファイルまたはディレクトリのグループを定義します。 グループを定義するには、ユーザをグループに追加する場合と同じ方法で、ファイルまたはディレクトリ(FILE クラスのリソース)を GFILE リソースに明示的に追加します。
GHOST	リソース	このクラスの各レコードは、ホストのグループを定義します。 グループを定義するには、ユーザをグループに追加する場合と同じ方法で、ホスト(HOST クラスのリソース)を GHOST リソースに明示的に追加します。
GROUP	アクセサ	このクラスの各レコードは、内部グループを定義します。
GSUDO	リソース	このクラスの各レコードは、あるユーザが実行しても、別のユーザが実行しているかのように見せかけることができるコマンドのグループを定義します。 sesudo コマンドはこのクラスを使用します。

クラス	クラス タイプ	説明
GTERMINAL	リソース	このクラスの各レコードは、端末のグループを定義します。
HNODE	定義	HNODE クラスには、組織の CA Access Control ホストに関する情報が含まれます。 クラスの各レコードは、組織内のノードを表します。
HOLIDAY	定義	このクラスの各レコードは、ユーザのログインに特別な許可を必要とする期間を 1 つ以上定義します。
HOST	リソース	このクラスの各レコードは、ホストを定義します。 ホストは、ホスト名または IP アドレスによって識別されます。 オブジェクトには、ローカル ホストがこのホストからサービスを受信できるかどうかを決定するアクセス ルールが保存されます。 HOST クラスをアクティブにする前に、streams モジュールがアクティブであることを確認します。
HOSTNET	リソース	このクラスの各レコードは、IP アドレス マスクによって識別され、アクセスルールを格納します。
HOSTNP	リソース	このクラスの各レコードは、ホストのグループを定義します。グループに属しているホストは、すべて同じ名前パターンになります。 各 HOSTNP オブジェクトの名前にはワイルドカードが含まれています。
LOGINAPPL	定義	LOGINAPPL クラスの各レコードは、ログイン アプリケーションの定義、ログイン プログラムを使用してログインできるユーザの指定、およびログイン プログラムの使用方法の制御を行います。
MFTERMINAL	定義	MFTERMINAL クラスの各レコードは、メインフレーム CA Access Control 管理コンピュータを定義します。
POLICY	リソース	POLICY クラスの各レコードは、ポリシーのデプロイおよび削除に必要な情報を定義します。 これらのレコードには、ポリシーをデプロイおよび削除するための selang コマンドのリストを含む RULESET オブジェクトへのリンクが含まれます。
PROCESS	リソース	このクラスの各レコードは、実行可能ファイルを定義します。
PROGRAM	リソース	このクラスの各レコードは、条件付きアクセス ルールに従って使用できる trusted プログラムを定義します。 trusted プログラムとは、改ざんされないように Watchdog 機能で監視されている setuid または setgid プログラムのことです。
PWPOLICY	定義	PWPOLICY クラスの各レコードは、パスワード ポリシーを定義します。
RESOURCE_DESC	定義	CA Access Control には適用できません。

クラス	クラス タイプ	説明
RESPONSE_TAB	定義	CA Access Control には適用できません。
RULESET	リソース	RULESET クラスの各レコードは、ポリシーを定義するルールのセットを表します。
SECFILE	定義	このクラスの各レコードは、変更されてはならないファイルを定義します。
SECLABEL	定義	このクラスの各レコードは、セキュリティ ラベルを定義します。
SEOS	インストール	このクラスのレコードはアクティブ クラスとパスワード ルールを指定します。
SPECIALPGM	インストール	SPECIALPGM クラスの各レコードは、Windows では、バックアップ機能、DCM 機能、PBF 機能、および PBN 機能を登録し、UNIX では、xdm 機能、バックアップ 機能、メール 機能、DCM 機能、PBF 機能、および PBN 機能を登録します。または、特別な権限保護を必要とするアプリケーションを論理ユーザ ID に関連付けます。 これにより、誰が実行しているかではなく何が実行されているかに従って、アクセス許可を効率的に設定できます。
SUDO	リソース	sesudo コマンドで使用されるこのクラスは、あるユーザ（一般ユーザなど）が実行しても、別のユーザ（root ユーザなど）が実行しているかのように見せかけることができるコマンドを定義します。
SURROGATE	リソース	このクラスの各レコードには、アクセサを代理として使用できるユーザを定義する、アクセサのアクセス ルールが含まれます。
TCP	リソース	このクラスの各レコードは、メール、http、ftp などの TCP/IP サービスを定義します。
TERMINAL	リソース	このクラスの各レコードは、端末（ユーザがログインに使用できるデバイス）を定義します。
UACC	リソース	各リソース クラスのデフォルト アクセス ルールを定義します。

クラス	クラス タイプ	説明
USER	アクセサ	このクラスの各レコードは、内部ユーザを定義します。
USER_ATTR	定義	CA Access Control には適用できません。
USER_DIR	リソース	CA Access Control には適用できません。
XGROUP	リソース	このクラスの各レコードは、CA Access Control に対するエンタープライズユーザを定義します。
XUSER	リソース	このクラスの各レコードは、CA Access Control に対してエンタープライズグループを定義します。

注: CA Access Control クラスの詳細については、「selang リファレンス ガイド」を参照してください。

ユーザ定義クラス

CA Access Control では、新しいクラスを定義し、そのクラスに適切なレコードを作成することによって抽象オブジェクトを保護できます。

例: データベース ビューのユーザ定義クラス

データベースを使用して独自のデータを格納および表示しているサイトがあるとします。

ユーザ定義クラス DATABASE_VIEWS を定義し、各データベース ビューをそのクラスのリソース メンバとして定義することができます。 リソースに、そのデータベース ビューを作成する場合に必要なアクセス権限を定義する ACL を割り当てます。 ユーザがデータベース ビューを作成しようとしたときに、CA Access Control は、ユーザのアクセス権限をチェックし、ACL に基づいて作成を許可または拒否します。

ユーザ定義クラスのリソースでのワイルドカードの使用

ユーザ定義クラスのリソースの名前にワイルドカードを使用することで、複数の物理リソースに対応するリソース レコードを作成できます。ワイルドカードのパターンと一致する名前を持つ物理リソースはすべて、リソース レコードに関連付けられたアクセス権限によって保護されます。

使用できるワイルドカードは、以下のとおりです。

- * - 任意の複数文字に対応します。
- ? - 任意の 1 文字に対応します。

物理リソースの名前が複数のリソース レコード名と一致する場合、そのリソースには、ワイルドカードを除く、最も長い一致が使用されます。

CA Access Control では、リソース名として以下のワイルドカード パターンは使用できません。

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

ユーザ定義クラス - 例

銀行のサービスを提供しているシステムで、口座間での高額の送金を保護する場合を考えます。 このセキュリティを設定するには、以下の手順に従います。

1. 送金を表すレコードを格納するためのクラス(たとえば、TRANSFERS)を定義します。
2. 保護する必要がある金額レベルの送金ごとに、TRANSFERS クラスにレコードを定義します。

   たとえば、Upto.$1K、Upto.$1M、Upto.$10M、および Over.$10M という名前のレコードを定義します。

   送金を制御する必要があるその他のリソースを、TRANSFERS クラスのメンバとして定義します。
3. ユーザごとに、最大送金額の異なる実行権限を与えるには、TRANSFER クラスの各種レコードへのアクセスを許可または拒否します。
4. さらに、プログラムによる送金を処理するため、ユーザのアクセス許可をチェックしてから送金処理を許可するように、銀行の送金プログラムに CA Access Control API コールを挿入します。

第 5 章: 許可の管理

このセクションには、以下のトピックが含まれています。

アクセス権限

CA Access Control の主な目的は、アクセス権限 (アクセス権とも呼ばれます) を割り当て、適用することです。

アクセス権限には、常に以下のコンポーネントがあります。

- アクセスの適用先のリソース (ファイル、ホスト、端末など)。
- アクセスのタイプ (読み取り、書き込み、削除、ログイン、実行など)。
- アクセサ (ユーザまたはグループのいずれか)。

以下の 1 つ以上に当てはまる場合、ユーザに対してリソースにアクセスする権限が割り当てられます。

- ユーザがリソースの ACL によって許可されている。
- ユーザが、アクセス権限が割り当てられたグループのメンバ。
- ユーザが、アクセス権限が割り当てられたプログラムを実行してアクセス。 たとえば、ユーザには、SPECIALPGM クラス内のプログラムを実行する権限、または SUDO クラス内のコマンドを実行する権限が割り当てられている。

注: クラス別のアクセス権限の詳細については、「selang リファレンス ガイド」を参照してください。

アクセス権限の設定 - 例

例：内部ユーザへ読み取りアクセス権限を付与する

以下の selang コマンドは、端末 tty30 の ACL に内部ユーザ internal_user を追加し、端末への読み取りアクセス権限を付与します。

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

例：エンタープライズ ユーザへ読み取りアクセス権限を付与する

以下の selang コマンドは、端末 tty30 の ACL にエンタープライズ ユーザ Terry を追加し、端末への読み取りアクセス権限を付与します。

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

例：リソースに対するエンタープライズ ユーザのアクセス権限を変更する

以下の selang コマンドは、端末 tty30 への Terry のアクセスを none に設定し、Terry のアクセスを拒否します。

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

例：エンタープライズ ユーザのアクセス権限をリソースから削除する

以下の selang コマンドは、端末 tty30 の ACL から Terry を削除します。

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

これで、Terry には、端末へのデフォルトのアクセス権が割り当てられます。

例：エンタープライズ ユーザにサブ管理者アクセスを付与する

以下の selang コマンドは、エンタープライズ ユーザ Terry を、ユーザとファイルを管理する権限を持つサブ管理者として設定します。

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

アクセス制御リスト

リソースに対するアクセス権限は、アクセス制御リストに指定されます。 各リソース レコードには、少なくとも 2 つのアクセス制御リストが割り当てられます。

ACL

リソースへのアクセスが許可されるアクセサと、そのアクセサが許可されるアクセスのタイプを指定します。

NACL

リソースへのアクセスが拒否されるアクセサと、そのアクセサが拒否されるアクセスのタイプを指定します。

アクセス権限は、ユーザがローカルでログインするかどうかなど、アクセスに関する状況によっても異なります。

条件付きアクセス制御リスト

条件付きアクセス制御リスト(CACL)は、ACL の拡張機能です。 アクセサがリソースへのアクセスを試みたときに、リソースの ACL と NACL にそのユーザのアクセス権限が定義されていない場合、CA Access Control は条件付きアクセス制御リストを確認します。

条件付きアクセス制御リストでは、アクセスが特定の方法による(たとえば、指定されたプログラムの使用による)場合のリソースへのアクセスを指定します。

たとえば、条件付きアクセス制御リストを使用して、Program Pathing ルールを定義できます。

CA Access Control では、以下の条件付きアクセス制御リストを使用することができます。

- プログラム アクセス制御リスト(PACL)
- TCP クラス アクセス制御リスト
- CALENDAR クラス アクセス制御リスト

条件付きアクセス制御リストのエントリを定義するには、selang authorize コマンドの via オプションを使用します。

他のアクセス制御リストと同様に、条件付きアクセス制御リストの各エントリでは、リソースへのアクセスが許可されるアクセサと、許可されるアクセスのタイプを指定します。 さらに、条件付きアクセス制御リストのエントリでは、権限を割り当てる条件も指定します。 PACL の条件とは、アクセサがアクセスをするために実行する必要があるプログラムの名前です。

例: PACL の使用

エンタープライズ ユーザ sysadm1 がプログラム secured_su を実行することによってスーパーユーザになれるようにするには、以下の selang コマンドを使用して、条件付きアクセス ルールを指定します。

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

defaccess - デフォルト アクセス フィールド

リソースのレコードには、デフォルト アクセス フィールド defaccess を含めることができます。 defaccess フィールドの値には、リソース アクセス制御リストのいずれでもカバーされないアクセサに許可するアクセス権限を指定します。

リソースに対するアクセス権限を決定する方法

アクセサがリソースへのアクセスを試みると、CA Access Control は、結果が得られるまで、事前定義された順序で 1 つ以上のチェックを実行することでアクセス権限をチェックします。 チェックによってアクセスの結果(アクセスの拒否または許可)が得られると、CA Access Control はそれ以上チェックを実行せず、代わりに結果を返します。

これらのチェックを実行する順序は重要です。 リソースごとに、CA Access Control はデフォルトでは以下の順序でアクセス レコードをチェックします。

1. リソースの時刻ベースの制限
2. リソースの所有権(所有者はアクセスが許可される)
3. B1 チェック
4. リソースの NACL
5. リソースの ACL
6. リソースの PACL
7. リソースの defaccess フィールド

最後の 2 つのチェックの順序は、accpacl オプションの設定によって決まります。 リソース PACL の使用を無効にするには、selang コマンドの setoptions setpacl- を使用します。

1 つのアクセス制御リストに、同じユーザに影響する複数のエントリが含まれていることがあります。 たとえば、ユーザを明示的に指定するエントリと、そのユーザが属する各グループに対するエントリが含まれることがあります。 CA Access Control は、各レベルで有効なすべてのエントリをチェックしてから、次のレベルに進みます。 各レベルで競合するルールを解決する方法の詳細については、「ユーザのアクセス権限とグループのアクセス権限との相互作用 (58 ページ)」を参照してください。

例: ファイルのアクセス許可の結果

以下の表は、アクセサ user1 がリソース ファイル 1 の読み取りを試みることを前提としています。

以下の表では、CA Access Control は accpacl オプションのデフォルトの設定に従って PACL を使用します。

user1 に対する NACL 内のエントリ	user1 に対する ACL 内のエントリ	user1 に対する PACL 内のエントリ	defaccess 内のエントリ	結果的に付与されるアクセス許可
Read	(任意)	(任意)	(任意)	読み取り拒否
(未定義)	None	(任意)	(任意)	読み取り拒否
(未定義)	Read	(任意)	(任意)	読み取り許可
(未定義)	(未定義)	via pgm secureader	(任意)	secureader プログラムの実行によって読み取り許可
(未定義)	(未定義)	(未定義)	Read	読み取り許可

エントリが (未定義) と表示されている場合、これは、user1 に対するエントリがアクセス制御リストに存在しないことを意味します。

エントリが (任意) と表示されている場合、これは、CA Access Control によるチェックが行われず、アクセス制御リスト内のエントリは関係ないことを意味します。

CA Access Control は、左から右にチェックします。 すべての行で、アクセスが定義されているセルの右側に位置するセルの値は、(任意) になることに注意してください。 逆に、アクセスが定義されているセルの左側にあるセルの値はすべて (未定義) になります。

ユーザのアクセス権限とグループのアクセス権限との相互作用

ユーザ、およびユーザが属するグループに対して、アクセス権限を明示的に許可または拒否することができます。 場合によってはこれらのアクセス権限が競合することがあります。 以下の例では、ユーザが 2 つのグループ（Group 1 と Group 2）のメンバであるときに競合するアクセス権限が同じリソースに割り当てられた場合、どのような結果になるかを示します。

累積グループ権限（58 ページ） オプションが設定されていることを前提とします（デフォルトの設定）。

ユーザのアクセス権限	Group 1 のアクセス権限	Group 2 のアクセス権限	最終的なアクセス権限
アクセス拒否	（任意）	（任意）	アクセス拒否
アクセス許可	（任意）	（任意）	アクセス許可
（未定義）	アクセス許可	（未定義）	アクセス許可
（未定義）	（未定義）	アクセス許可	アクセス許可
（未定義）	アクセス許可	アクセス許可	アクセス許可
（未定義）	アクセス拒否	（任意）	アクセス拒否
（未定義）	（任意）	アクセス拒否	アクセス拒否

エントリが（未定義）と表示されている場合、これは、ユーザまたはグループに対するエントリが定義されていないことを意味します。

エントリが（任意）と表示されている場合、これは、CA Access Control によるチェックが行われず、アクセス権限は関係ないことを意味します。

累積グループ権限（ACCGRR）

累積グループ権限オプション（ACCGRR）では、CA Access Control がリソースの ACL をチェックする方法を制御します。 ACCGRR が有効な場合、CA Access Control は、ACL で、ユーザが属するすべてのグループで許可されている権限をチェックします。 ACCGRR が無効な場合、CA Access Control は、ACL で適用可能なエントリのいずれかに値 none が含まれているかどうかをチェックします。 none が含まれている場合、アクセスは拒否されます。 none が含まれていない場合、CA Access Control は、ACL 内の最初の適用可能なグループ エントリを除くすべてのグループ エントリを無視します。 このオプションはデフォルトで有効です。

ACCGRR オプションを有効にするには、以下の selang コマンドを使用できます。

```
setoptions accgrr
```

ACCGRR オプションを無効にするには、以下の selang コマンドを使用できます。

```
setoptions accgrr
```

セキュリティ レベル、セキュリティ カテゴリ、およびセキュリティ ラベル

セキュリティ レベルとセキュリティ カテゴリは、リソースへのアクセスを制限する追加の方法を提供して、アクセス制御リストを補完します。

セキュリティ ラベルは、セキュリティ レベルとセキュリティ カテゴリを 1 つにまとめて、管理を簡易化する手段です。

セキュリティ レベル

セキュリティ レベルは、ユーザおよびリソースに割り当てることのできる 0 から 255 までの整数です。 リソースのアクセス制御リストでユーザにアクセス権限が付与されていても、アクセサのセキュリティ レベルがリソースのセキュリティ レベルより低い場合、そのアクセサはそのリソースにアクセスできません。 リソースのセキュリティ レベルがゼロの場合、そのリソースに対してセキュリティ レベルのチェックは実行されません。

セキュリティ レベルがゼロのアクセサは、セキュリティ レベルがゼロ以外のリソースにアクセスできません。

セキュリティ カテゴリ

セキュリティ カテゴリは、CATEGORY クラスにあるレコードの名前です。 セキュリティ カテゴリは、アクセサとリソースに割り当てることができます。 リソースに割り当てられているすべてのセキュリティ カテゴリにアクセサが割り当てられている場合のみ、そのアクセサはリソースにアクセスできます。

セキュリティ ラベル

セキュリティ　ラベルは、SECLABEL クラスにあるレコードの名前です。 セキュリティ ラベルによって、セキュリティ ラベルと複数のセキュリティ カテゴリを 1 つにまとめることができます。 セキュリティ ラベルをアクセサまたはリソースに割り当てると、そのセキュリティ ラベルに関連付けられたセキュリティ レベルとセキュリティ カテゴリの組み合わせが、アクセサまたはリソースに設定されます。 セキュリティ ラベルは、アクセサまたはリソースに設定された特定のセキュリティ レベルおよびセキュリティ カテゴリよりも優先されます。

例: セキュリティ ラベル High_Security の使用

High_Security は、セキュリティ レベル 255 と、セキュリティ カテゴリ MANAGEMENT および CONFIDENTIAL を含むセキュリティ ラベルであるとします。

ユーザ user1 をセキュリティ ラベル High_Security に割り当てた場合、user1 には、セキュリティ レベル 255 と、セキュリティ カテゴリ MANAGEMENT および CONFIDENTIAL が設定されます。

第 6 章: アカウントの保護

このセクションには、以下のトピックが含まれています。

アカウントを保護する理由

ユーザ アカウントは、頻繁にパスワード攻撃の標的になります。 root アカウントの保護では、ユーザの切り替え (su) 要求を監視し、Surrogate DO (SUDO) 機能を使用して、スーパーユーザ権限に関するジレンマを解消します。 CA Access Control には、serevu (ユーザ無効デーモン) および PAM (Pluggable Authentication Module) という 2 段階のパスワード保護システムが用意されています。 また、ユーザの非アクティブ状態が一定期間続いた場合に自動ロックアウトを指定してアカウントを保護することもできます。

安全なユーザの置換

UNIX の su コマンドを使用すると、ターゲット ユーザのパスワードを使用して、そのユーザに切り替えることができます。 ユーザ ID の切り替えを行うユーザは、ターゲット ユーザのパスワードを記憶するか、書き留めるか、または簡単なパスワードを使用するようにターゲット ユーザに依頼する必要があります。 このような行為は、いくつかのパスワード ポリシーに反します。 また、su コマンドは、どのユーザがこのコマンドを呼び出したのかを記録しないため、アカウントの所有者を偽装するユーザを実際のユーザと区別することはできません。

CA Access Control に含まれている sesu ユーティリティは、この UNIX の su コマンドの機能強化版です。 sesu では、認証の手段として、ターゲット ユーザのパスワードではなく、ユーザ自身のパスワードを要求するように設定できます。 認証プロセスは、SURROGATE クラスに定義されているアクセス ルールに基づいて実行されます。また、コマンドを実行するユーザのパスワードに基づいて実行されるようにすることもできます。

su の実行とは異なり、sesu の実行では、ターゲット ユーザのパスワードを知っているかどうかは関係ありません。 その代わり、sesu の実行は、データベースに指定されている権限に依存します。各ユーザのログイン ID が記憶されるため、アクションを実行したユーザの記録が残ります。

ユーザが _surrogate グループのユーザの代理ユーザになる場合、CA Access Control はユーザのアクションの完全なトレースを新規ユーザとして監査証跡に送信します。

このプログラムは、誤って使用されることを防ぐために、ファイル システム内でマークされており、誰もこれを実行できません。 このため、セキュリティ管理者は、このプログラムを実行する前に、それが実行可能ファイルであることをマークし、ユーザ ID を root に設定する必要があります。

重要: sesu ユーティリティを使用する前に、すべてのユーザを CA Access Control データベースに定義し、前提条件を設定してください。 これは、CA Access Control に定義されていないユーザに対してシステム全体が開放されることを防止するためです。

ユーザ ID の置換ルールの設定

ユーザが他のユーザを置換できないようにする、または他のユーザに置換できるようにするには、ユーザ ID の置換ルールを設定する必要があります。 これらのルールは、SURROGATE クラスのリソースを使用して制御されます。 ユーザ置換ルールを定義するには、SURROGATE レコードを作成する必要があります。

ユーザ ID の置換ルールを設定するには、以下の手順に従います。

1. CA Access Control エンドポイント管理の[ユーザ]タブをクリックし、[権限および委任]サブタブをクリックします。

   [権限および委任]メニュー オプションが左側に表示されます。

2. [ユーザ ID の置換]をクリックします。

   [ユーザ ID の置換]ページが表示されます。

3. [ユーザ ID の置換の作成]をクリックします。

   [ユーザ ID の置換の作成]ページが表示されます。

4. タブ ページのフィールドに入力し、[保存]をクリックします。

   注: SURROGATE クラス プロパティの詳細については、「selang リファレンス ガイド」を参照してください。

ユーザ置換 sesu のセットアップ方法

デフォルトでは、sesu ユーティリティはファイル システム内でマークされているため、誰もこれを実行できません。 sesu をユーザが使用できるようにする前に、データベース ルールを設定し、それが必ず安全に使用されるようにします。 次に、システムの su ユーティリティをロックし、ユーザが強制的に CA Access Control の sesu ユーティリティを使用するようにする必要があります。

sesu をセットアップするには、以下の手順に従います。

1. 基本的なユーザ置換ルールを設定します(63 ページ)。
2. システムの su ユーティリティを CA Access Control の sesu ユーティリティにします(63 ページ)。
3. ユーザがシステムの su ユーティリティを実行できないようにします(66 ページ)。

注: このセットアップを完了すると、CA Access Control の実行時にシステムの su ユーティリティは実行されず、ユーザは安全な sesu ユーティリティを使用するように強制することができます。 CA Access Control が実行されていない場合は、システムの su ユーティリティが使用されます。

基本的なユーザ置換ルールの設定

sesu ユーティリティの使用を開始する前に、一般的なユーザ置換ルールをデータベースに定義する必要があります。 これらのルールによって、不明なユーザが特権ユーザのアカウントへユーザ置換するのを防ぐ一方で、特定のユーザまたはプロセスが必要なユーザ置換操作を行えるようにします。

基本的なユーザ置換ルールを設定するには、以下の手順に従います。

1. 以下の属性を指定して、root ユーザ(USER.root)用の SURROGATE リソースを作成します。
   - 所有者 - nobody
   - デフォルト アクセス - none
   - すべての管理者にフル コントロールを付与する

   これにより、特に許可のない限り、すべてのユーザが root にユーザ置換できなくなります。 すべての管理者に、root にユーザ置換する権限が明示的に付与されます。

   注: 個々の管理者に別々に権限を付与するか、管理者のグループを使用してすべての管理者に権限を付与することができます。
2. 以下の属性を指定して、root グループ(GROUP.other)用の SURROGATE リソースを作成します。
   - 所有者 - nobody
   - デフォルト アクセス - none
   - すべての管理者にフル コントロールを付与する

   これにより、特に許可のない限り、すべてのユーザが root グループに置換できなくなります。 すべての管理者に、root グループに置換する権限が明示的に付与されます。

   注: ほとんどの UNIX システムでは、root グループは other または sys です。

3. USER._default 用のユーザ置換ルールを以下のように変更します。
   - 所有者 - nobody
   - デフォルト アクセス - none
   - root に、未定義のユーザになる権限を付与する
   - 管理者のグループに、未定義のユーザになる権限を付与する

   これにより、特に許可のない限り、すべてのユーザがユーザ置換できなくなります。また、特に拒否のない限り、root および root グループにユーザ置換する権限が付与されます。

   注: root に権限を与えることは、dtlogin などのプログラムがセッションの所有者をデフォルトの X ウィンドウ所有者である root（ユーザ ID = 0）からほかのユーザに変更する場合などに特に必要になります。 権限を付与しないと、ログインに失敗します。これは、CA Access Control が明示的に許可されていないユーザ置換操作をブロックしているためです。

4. GROUP._default 用のグループ置換ルールを以下のように変更します。
   - 所有者 - nobody
   - デフォルト アクセス - none
   - root に、未定義のグループを置換する権限を付与する
   - 管理者のグループに、未定義のグループを置換する権限を付与する

   これにより、特に許可のない限り、すべてのユーザがグループに置換できなくなります。また、特に拒否のない限り、root および root グループにグループを置換する権限が付与されます。

例: selang での基本的なユーザ置換ルールの設定

以下の selang コマンドを使用して、基本的なユーザ置換ルールを環境に設定します。

```
nr surrogate USER.root defacc(n) own(nobody)
auth surrogate USER.root gid(sys_admin_GID) acc(a)
nr surrogate GROUP.other defacc(n) own(nobody)
auth surrogate GROUP.other gid(sys_admin_GID) acc(a)
cr surrogate USER._default defacc(n) own(nobody)
cr surrogate GROUP._default defacc(n) own(nobody)
auth surrogate USER._default uid(root) acc(a)
auth surrogate GROUP._default uid(root) acc(a)
auth surrogate USER._default gid(sys_admin_GID) acc(a)
auth surrogate GROUP._default gid(sys_admin_GID) acc(a)
```

システムの su ユーティリティを CA Access Control の sesu ユーティリティに置換

デフォルトでは、sesu ユーティリティはファイル システム内でマークされているため、誰もこれを実行できません。 sesu ユーティリティを使用してユーザが他のユーザのアカウントを代理使用できるようにするには、まず sesu ユーティリティを有効にし、次にシステムの su を sesu ユーティリティに置換する必要があります。

システムの su ユーティリティを CA Access Control の sesu ユーティリティに置換するには、以下の手順に従います。

注: 以下の手順を実行するには、root または権限を持つ他のユーザである必要があります。

1. 以下のコマンドを使用して、ユーザが sesu ユーティリティを実行できるようにします。

   chmod +s /opt/CA/AccessControl/bin/sesu

2. 以下のコマンドを使用して、システムの su ユーティリティが格納されている場所を確認します。

   which su

3. 以下のコマンドを使用して、システムの su ユーティリティの名前を変更します。

   mv su_dir/su su_dir/su.ORIG

   ここで、su_dir は su があるディレクトリです。

4. sesu ユーティリティを su コマンドにリンクします。

   ln -s /opt/CA/AccessControl/bin/sesu su_dir/su

   これで、ユーザは引き続き su コマンドを実行できますが、実際に実行されるのは sesu ユーティリティになります。

5. 以下のコマンドを使用して、CA Access Control を停止します。

   secons -s

6. 以下のコマンドを使用して、CA Access Control の設定を変更します。

   seini -s sesu.SystemSu su_dir/su.ORIG
   seini -s sesu.UseInvokerPassword yes

   トークン SystemSu が設定されます。これにより、CA Access Control が実行されていない状態では sesu は元のシステム su ユーティリティを呼び出せるようになります。

   トークン UseInvokerPassword が設定され、CA Access Control はユーザに、root のパスワードまたは他のユーザのパスワードではなく、自分のパスワードの入力を求めます。 ユーザ置換が許可されるには、ユーザの再認証が必要になります。

7. 以下のコマンドを使用して、CA Access Control を再ロードします。

   seload

ユーザのシステム su ユーティリティ実行の阻止

sesu ユーティリティが構成されていても、誰でも、以前と同様に、root またはユーザのパスワードを使用して、su.ORIG（名前を変更したシステム su ユーティリティ）を実行できます。 これを防止するには、PROGRAM クラスを使用して、CA Access Control の実行時に、明示的に su.ORIG の実行を阻止します。

注: CA Access Control のインストールおよび構成時に seuidpgm を使用した場合、この手順に従う必要はありません。 su は変更（su.ORIG に名前変更）されたため、実行されません。

ユーザのシステム su ユーティリティ実行の阻止方法

1. selang の以下のコマンドを使用して、CA Access Control が名前を変更した su ユーティリティを監視するようにします。

   ```
   nr program su_dir/su.ORIG defacc(x) own(nobody)
   ```

2. root としてログインし、以下のコマンドを使用して、ファイルのアクセスおよび変更時間を変更します。

   ```
   touch su_dir/su.ORIG
   ```

   CA Access Control は su.ORIG を監視しているため、変更が加えられた su.ORIG の実行は阻止されます。

Surrogate DO 機能のセットアップ

多くの場合、オペレータ、プロダクション担当者、およびエンド ユーザは、スーパーユーザのみが実行できるタスクを実行する必要があります。 このタスクには、以下のタスクが含まれます。

- CD-ROM のマウント
- バックアップ スクリプトの使用
- プリンタのセットアップ

これまでの方法では、これらのタスクを実行する必要があるすべてのユーザに、スーパーユーザのパスワードを知らせていました。これはサイトのセキュリティを脅かすことにつながります。 このため、安全な代替策としてパスワードの公開を禁止すると、システム管理者はユーザからの正当な要求によってさまざまなルーチン タスクを実行しなければならず、システム管理者の負荷が大きくなります。

Surrogate DO（sesudo）ユーティリティは、このジレンマを解消します。 このユーティリティは、SUDO クラスに定義されているアクションの実行をユーザに許可します。SUDO クラスの各レコードにはスクリプトが保存されていて、スクリプトを実行できるユーザとグループが指定されています。それらのユーザやグループに、目的に応じて必要な許可が与えられます。

たとえば、ユーザが root ユーザであるかのように CD-ROM をマウントする SUDO リソースを定義するには、以下のコマンドを入力します。

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

この newres コマンドによって、一部のユーザだけが root の実行権限を使用できる保護されたアクションとして、MountCd が定義されます。 この例では、root がターゲットユーザの ID であることを明確にするために、targuid(root) パラメータを使用しています。このターゲット ユーザには、root の実行権限が与えられています。 実際には、SUDO レコードのデフォルトのターゲット ID は root であるため、この例で指定しているパラメータは不要になります。

重要: data プロパティには、完全な絶対パス名を使用してください。 相対パス名を使用すると、保護されていないディレクトリに仕掛けられたトロイの木馬プログラムが、誤って実行される可能性があるからです。

さらに、authorize コマンドを使用して、MountCd アクションを実行する権限をユーザに与えることもできます。 たとえば、ユーザ operator1 に CD-ROM のマウントを許可するには、以下のコマンドを入力します。

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

また、authorize コマンドを使用して、保護されたアクションの実行をユーザに対して明示的に禁止することもできます。 たとえば、ユーザ operator2 による CD-ROM のマウントを禁止するには、次のコマンドを入力します。

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

sesudo ユーティリティを実行すると、保護されたアクションが実行されます。 たとえば、ユーザ operator1 が以下のコマンドを使用して CD-ROM をマウントするとします。

```
sesudo MountCd
```

この sesudo ユーティリティは、最初に SUDO アクションの実行権限がユーザにあるかどうかをチェックし、そのユーザにリソースの権限がある場合は、そのリソースに定義されているコマンド スクリプトを実行します。 この例に示した sesudo は、MountCd アクションの実行権限が operator1 にあるかどうかをチェックした後に、mount /usr/dev/cdrom /cdr コマンドを起動します。

実行前に sesudo でユーザのパスワードを要求する場合は、PASSWORD パラメータを指定したコマンドを使用して、SUDO レコードを定義または変更します。 このパラメータを使用しない場合、ユーザがコマンドを実行できるかどうかは、SUDO オブジェクトのアクセス ルールに基づいて決定されます。

注: sesudo ユーティリティ、および SUDO レコードの管理(editres コマンド)の詳細については、「リファレンス ガイド」を参照してください。

SUDO レコードの定義

SUDO クラスのレコードには、コマンド スクリプトが格納されています。ユーザは、借用した権限でそのスクリプトを実行できます。 権限を利用できるかどうかは、スクリプトを実行する sesudo コマンドと SUDO レコードの両方で厳密に制御されます。

SUDO レコードでは、comment プロパティを特別な目的に使用します。通常、このような comment プロパティを data プロパティといいます。

data プロパティの値は、コマンド スクリプトです。必要に応じて、禁止(prohibited)または許可(permitted)するスクリプト パラメータ値を 1 つ以上追加します。 data プロパティ値全体は一重引用符で囲む必要があります。トロイの木馬の侵入を防ぐために、実行可能ファイルは完全パス名で参照する必要があります。

data プロパティの形式は、以下のとおりです。

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

prohibited および permitted の値のリストは省略できるため、data プロパティの値は以下のような簡単な値にすることもできます。

```
newres SUDO MountCd data('mount /dev/cdrom /cdr')
```

この例では、コマンドに指定されている簡単な値によって、sesudo MountCd コマンドで mount /dev/cdrom /cdr というスクリプトが実行されます。 特定のスクリプト パラメータ値が禁止されていないため、すべての値が許可されます。

ワイルドカードと強力な変数を使用すると、prohibited パラメータおよび permitted パラメータを柔軟に指定できるようになります。 使用できるワイルドカードは、UNIX の標準的なワイルドカードです。 使用できる変数は以下のとおりです。

変数	説明
$A	英字の値
$G	既存の CA Access Control グループ名
$H	ユーザのホーム パス パターン
$N	数値
$O	実行するユーザの名前
$U	既存の CA Access Control ユーザ名
$e	パラメータが指定されていない SUDO コマンド
$f	既存のファイル名
$g	既存の UNIX グループ名
$h	既存のホスト名
$r	UNIX の読み取り許可が付与された既存の UNIX ファイル名
$u	既存の UNIX ユーザ名
$w	UNIX の書き込み許可が付与された既存の UNIX ファイル名
$x	UNIX の実行許可が付与された既存の UNIX ファイル名

prohibited パラメータ値のリストをスクリプトに追加する場合は、以下のようにします。

- スクリプトと prohibited パラメータの値をセミコロンで区切り、全体を一重引用符で囲みます。 たとえば、ユーザによる-9 の使用を禁止し、それ以外のすべてのパラメータの使用を許可する場合は、以下のコマンドを入力します。

  ```
  newres SUDO scriptname data('cmd;-9')
  ```

  ここで、cmd はユーザのスクリプトを表します。

  また、パラメータ値を許可せず、すべてのパラメータをデフォルトに設定する場合は、SUDO レコードを以下のように定義します。

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

- 1 つのスクリプト パラメータに対して複数の prohibited 値を指定する場合は、スペース文字を区切り記号として使用します。 たとえば、ユーザによる-9 および -HUP の使用を禁止し、それ以外のすべてのパラメータの使用を許可する場合は、以下のコマンドを入力します。

  newres SUDO scriptname data('cmd;-9 -HUP')

- 複数のスクリプト パラメータに対して prohibited 値を指定する場合は、パイプ (|) を区切り記号として使用して、それぞれの prohibited 値セットの間を区切ります。 たとえば、スクリプトの最初のパラメータに-9 および-HUP の使用を禁止し、2 番目のパラメータに既存の UNIX ユーザ名 (前出の変数の一覧を参照) の使用を禁止する場合は、以下のコマンドを入力します。

  newres SUDO scriptname data('cmd;-9 -HUP | $u')

  指定したパラメータよりスクリプトのパラメータが多い場合は、指定した最後の prohibited パラメータのセットが、残りすべてのパラメータに適用されます。

permitted パラメータ値のリストをスクリプトに追加する場合は、以下の操作を行います。

- sesudo ユーティリティでは、パラメータ値が、対応する prohibited 値のいずれとも一致しないこと、および対応する permitted 値の少なくとも 1 つと一致することの 2 つがチェックされます。
- permitted 値のリストと prohibited 値のリストをセミコロンで区切り、全体を一重引用符で囲みます。 prohibited 値のリストを指定しない場合でも、セミコロンは必要です。セミコロンがないと、permitted 値として指定した値が、prohibited 値として処理されます。 たとえば、スクリプトのパラメータ値として値 NAME のみを許可する場合は、以下のコマンドを入力します。

  newres SUDO scriptname data('cmd;;NAME')

- 他のリストの指定も同様に行います。
  - 1 つのスクリプト パラメータに対して複数の permitted 値を指定する場合は、スペース文字を区切り記号として使用します。
  - 複数のスクリプト パラメータに permitted 値を指定する場合は、パイプ (|) を区切り記号として使用して、それぞれの permitted の値セットの間を区切ります。

  たとえば、2 つのパラメータがあるとします。最初のパラメータには UNIX のユーザ名でない数字を指定し、2 番目のパラメータには UNIX のグループ名でない英字を指定する必要がある場合は、以下のコマンドを入力します。

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

  スクリプトのパラメータが指定したパラメータより多い場合は、指定した最後の permitted パラメータのセットが、残りすべてのパラメータに適用されます。

したがって、data プロパティ全体の形式は、スクリプト、パラメータごとの prohibited 値、パラメータごとの permitted 値の順になります。

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

パスワード攻撃の防止

最も一般的なタイプの不正アクセスは、パスワードを推測するハッカーによるアクセスです。 CA Access Control には、パスワード攻撃を検出し防止するために serevu および pam_seos という 2 つのツールが用意されています。

さらにパスワード攻撃を防止するもう 1 つの方法は、パスワード ポリシー ルールを設定して、現在の環境で使用されているパスワードを制御することです。

serevu

serevu デーモンは、指定された回数を上回るログインを試みたユーザのアカウントをロックします。 このように、指定回数を超えてログインしようとすることを拒否することで、パスワード攻撃や「辞書攻撃」を防止します。

通常、ユーザ ロックアウト ユーティリティの使用に伴う危険は、システムがサービス妨害攻撃に対して無防備になることです。 サービス妨害攻撃の一般的なタイプの 1 つに、システム管理者のアカウントへの不正な侵入があります。 侵入が数回試みられると、システム管理者のアカウントは無効となり、システム管理者はログインできなくなります。 同じような攻撃をすべての重要なユーザ アカウントが受けると、システムが使用不能に陥る可能性があります。この場合、システムを回復する手段はありません。 このような状況を防ぐために、serevu デーモンには以下の 2 つの操作モードが用意されています。

- アカウントは指定した期間無効になり、その後自動的に回復します。
- アカウントは、永久に無効になります。

serevu によって root が無効になることはないため、システムがロックアウトされることはありません。

注: serevu デーモンの詳細については、「リファレンス ガイド」を参照してください。

注: root を辞書攻撃から守るために、root ユーザのパスワードの取り扱いには特に注意が必要です。

pam_seos

pam_seos は、CA Access Control の高度なアカウント管理機能で使用される PAM (Pluggable Authentication Module) です。 CA Access Control では、すべてのログイン プログラムのログイン時に pam_seos が呼び出されます。 このモジュールは、要求に応じて必要な機能を提供するために動的にロードできる共有オブジェクトです。

pam_seos は、次の 3 つのアクションを実行するように設定できます。

- ログイン エラーの検出

  アカウント管理コンポーネントは、失敗したすべてのログインの試みを検出し、監査ファイル、および失敗したログインを記録する特別なファイルの両方に検出結果を記録します。 このモジュールは、CA Access Control がアクセスを拒否したケースを検出するのではなく、UNIX のエラーを検出します。

  CA Access Control では、失敗したログインの試みが特別なファイルに書き込まれます。 serevu ユーティリティは、このファイルを読み込み、その情報を使用して、ユーザのアクセス権を無効にするかどうか、また無効にする場合はいつ無効にするかを決定します。

- デバッグ モードの提供

  CA Access Control がログインを拒否する場合、通常は、ログイン セッション中に拒否理由が表示されることはありません。 pam_seos モジュールのデバッグ モードが設定されている場合、CA Access Control には、ログインの拒否理由に関する短い説明が表示されます。 たとえば、「猶予ログイン」は、ユーザのログイン回数が残っていないことを意味します。

- 有効期限切れのパスワードおよび猶予ログインのチェック

  パスワード管理コンポーネントは、segrace ユーティリティを起動し、ユーザのパスワードの有効期限および猶予ログインの回数をチェックします。 ユーザのパスワードが有効期限切れになり、猶予ログインの回数が残っていない場合、segrace は sepass ユーティリティを起動し、ユーザによるパスワードの変更を許可します。

注: CA Access Control が segrace を起動するのは、パスワードの変更が必要な場合のみです。

インストール プログラムは、関連する行を pam.conf 環境設定ファイルに追加し、古い環境設定ファイルを /etc/pam.conf.bak として保存します。

pam_seos モジュールの環境設定は、seos.ini ファイルを使用して行います。 必要な機能に応じて、[pam_seos]セクションにある以下のトークンを設定します。

パスワードの有効期限および猶予ログインのチェックを使用するには、seos.ini ファイルに以下のトークンを設定します。

call_segrace = Yes

ログイン デバッグ モードを使用するには、seos.ini ファイルに以下のトークンを設定します。

debug_mode_for_user = Yes

serevu で pam_seos のログイン エラー検出機能を使用するには、seos.ini ファイルに以下のトークンを設定します。

serevu_use_pam_seos = Yes

制約と制限

このセクションで説明した保護方法には、以下の制約と制限があります。

- Sun Solaris では、ログインの試みに 5 回失敗すると、serevu に通知されます。
- pam_seos モジュールの実装は、PAM をサポートしている Sun Solaris、HP-UX、および Linux のバージョンに限定されます。

ユーザの非アクティブ状態のチェック

ユーザの非アクティブ状態をチェックする機能を使用して、不在または会社を退職したユーザのアカウントを使用した不正なアクセスからシステムを保護します。 非アクティブ状態の日とは、ユーザがログインしていない日を指します。 ユーザ アカウントが一時停止されて、ログインできなくなるまでの、非アクティブ状態の日数を指定できます。 一時停止したアカウントは、手動で再びアクティブにする必要があります。

注: 非アクティブ状態のチェックでは、パスワード変更はアクティビティとしてカウントされます。 ユーザのパスワードが変更された場合、非アクティブ状態を理由としてそのユーザのアカウントを一時停止することはできません。

非アクティブ日数は、USER クラスまたは GROUP クラスのレコードの inactive プロパティを使用して設定できます。 GROUP クラスのレコードでの設定は、そのグループがプロファイル グループであるユーザのみに適用されます。 また、SEOS クラスの INACT プロパティを使用して、システム全体のすべてのユーザに非アクティブ状態を設定することもできます。

selang では、以下のコマンドを使用して、非アクティブ状態をグローバルに指定します。

```
setoptions inactive (numdays)
```

非アクティブ日数をグループに設定するには、以下のコマンドを使用します(この設定は、そのグループに対するシステム全体の非アクティブ設定よりも優先されます)。

```
editgrp groupName inactive (numdays)
```

非アクティブ日数をユーザに設定するには、以下のコマンドを使用します(この設定は、そのユーザに対するグループおよびシステム全体の設定よりも優先されます)。

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

一時停止しているユーザ アカウントを再びアクティブにするには、以下のコマンドを使用します。

```
editusr userName resume
```

一時停止しているプロファイル グループを再びアクティブにするには、以下のコマンドを使用します。

```
editgrp userName resume
```

システム全体レベルで非アクティブ ログイン チェックを無効にするには、以下のコマンドを使用します。

```
setoptions inactive
```

グループに対する非アクティブ ログイン チェックを無効にするには、以下のコマンドを使用します。

```
editgrp groupName inactive
```

ユーザに対する非アクティブ ログイン チェックを無効にするには、以下のコマンドを使用します。

```
editusr userName inactive
```

第 7 章: ユーザ パスワードの管理

このセクションには、以下のトピックが含まれています。

パスワードの管理

パスワードは最も一般的な認証手段ですが、パスワードの保護には、以下のようなよく知られた問題があります。

- 簡単なパスワードは推測されやすい。
- 長い間同じパスワードを使用したり、同じパスワードを繰り返し使用していると、解読されやすくなる。
- ネットワークを介して平文で送信されたパスワードは、盗まれる危険性がある。

パスワード ポリシーの定義

最も重要なパスワード ルールは、明示的または間接的に自分のパスワードが他人に知られないように、簡単なパスワードを使用しないことです。 適切なパスワード セキュリティを実現する唯一の方法は、トレーニングと教育です。 CA Access Control によって、ユーザ教育が必要がなくなるわけではありません。しかし、ユーザが最低限の品質を持つパスワードを使用するようにルールとポリシーを強化できます。 ルールには、以下の項目を指定できます。

- 新しいパスワードは以前に使用したパスワードと同じにすることはできません。
- 新しいパスワード中にユーザ名を使用することはできません。
- 新しいパスワードは変更前のパスワードを含むことはできません。
- 新しいパスワードには変更前のパスワードの一部を使用することはできません。
- 大文字と小文字の区別に関係なく、新しいパスワードと変更前のパスワードを同じにすることはできません。
- 新しいパスワードには、パスワード ポリシーで指定されている英数字、特殊文字、数字、小文字、および大文字を、それぞれ最低文字数以上使用する必要があります。
- 新しいパスワードで繰り返し使用される文字の数が、パスワード ポリシーで指定されている数を超えないようにする必要があります。

- seos.ini ファイルの Dictionary トークンに指定されている辞書ファイルで使用が禁止されている単語を、新しいパスワードに使用することはできません。
- パスワードごとに、最長有効期限を指定する必要があります。つまり、有効期限を過ぎたパスワードは失効し、ユーザが新しいパスワードを選択する必要があります。
- パスワードごとに、最短有効期限を指定する必要があります。 最短有効期限を指定すると、ユーザが短期間に何度もパスワードを変更することを防止できます。 パスワードを短期間に何度も変更すると、パスワード履歴リストがオーバーフローし、使用済みパスワードが再利用される場合があります。

重要: パスワード ルールは、sepass のみに影響し、ネイティブのパスワード ツールには影響しません。 passwd を sepass へのリンクに置き換えていることを確認してください。

パスワード品質チェックの設定

パスワード品質チェックを設定するには、以下の手順に従います。

1. CA Access Control エンドポイント管理の[環境設定]タブをクリックします。

   [環境設定]メニュー オプションが左側に表示されます。

2. [その他]セクションのオプションで[クラスのアクティブ化]をクリックします。

   [クラスのアクティブ化]ページが表示されます。

3. [ユーザ識別コントロール]セクションで [PASSWORD]を選択して、[保存]をクリックします。

   これで、パスワード品質チェックがアクティブになります。

4. [ポリシー]セクションのオプションで、[ユーザ パスワード ポリシー]をクリックします。

   [ユーザ パスワード ポリシー]ページが表示されます。

5. パスワードのチェックに使用するルールを定義し、[保存]をクリックします。

   パスワードのチェックに定義したルールは、パスワードが変更されたときに適用されます。

6. (UNIX のみ)sepass ユーティリティを使用して、新しいパスワードを更新します。

   注: sepass ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

例: パスワード チェック ルールを定義する

以下の selang コマンドは、パスワード品質チェックをアクティブにし、以下の最小文字数を適用するパスワード ルールを定義します。

- 英数字: 6 文字
- 小文字: 3 文字
- 数字: 2 文字

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

注: setoptions コマンドの形式の詳細については、「リファレンス ガイド」を参照してください。

パスワードの変更

CA Access Control には、ACInstallDir/bin/sepass という実行可能プログラムが用意されています(ここで、ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは、/opt/CA/AccessControl となります)。ほとんどのユーザは、/bin/passwd ではなくこのプログラムを使用して、パスワードを変更する必要があります。

- sepass プログラムによってのみ、新しいパスワードが CA Access Control のパスワード ポリシーに適合していることを確認できます。 同様に、sepass プログラムによってのみ、新しいパスワードとパスワードの変更日付を使用してデータベースを更新できます。 また、このプログラムは、/bin/passwd と同じ機能を実行します。
- CA Access Control によるパスワード品質チェックを無視する場合を除いて、元の実行可能プログラム /bin/passwd を使用しないでください。 無視する場合は、元の /bin/passwd を続けて使用できるため、CA Access Control では、パスワードに対する品質チェックを実行せずに、システムのパスワードが許可されます。

selang を使用してパスワードを変更することもできます。 パスワードをユーザに割り当てるには、以下のコマンドを入力します。

```
chusr userName password(string)
```

注: (管理者として)別のユーザのパスワードを変更したときに、パスワード チェックが有効である場合、ユーザは次回のログイン時にパスワードを変更する必要があります。

パスワードの有効期限と猶予ログイン

interval パラメータには、パスワードを使用できる最長日数を設定します。 指定した日数が経過すると、CA Access Control は、現在のパスワードが期限切れであることをユーザに通知します。 通知を受けたユーザは、ただちにパスワードを更新するか、猶予ログイン回数に達するまで古いパスワードを引き続き使用することができます。 猶予ログイン回数に達すると、ユーザはシステムにアクセスできなくなるので、システム管理者に連絡して、新しいパスワードを設定する必要があります。

パスワード期間の指定

setoptions コマンドを使用して、システム全体のレベルでパスワード期間を指定します。パスワード期間を過ぎると、すべてのユーザは新しいパスワードの入力を要求されます。ユーザのログイン スクリプトに segrace ユーティリティが含まれている場合、または segrace を呼び出すように PAM を設定している場合（ネイティブ オペレーティング システムが PAM をサポートしている場合）、CA Access Control では、指定された日数に達すると、現在のパスワードが期限切れになったことがユーザに通知されます。 通知を受けたユーザは、ただちにパスワードを更新するか、猶予ログイン回数に達するまで古いパスワードを引き続き使用することができます。 猶予ログイン回数に達すると、ユーザはシステムへのアクセスが拒否されます。システム管理者に連絡して新しいパスワードを設定する必要があります。

システム全体のレベルでパスワード期間を設定または取り消すには、以下のコマンドを使用します。

```
setoptions password({interval(NumDays)|interval-})
```

NumDays の値には、ゼロ（0）または正の整数を指定する必要があります。 期間を 0 に設定すると、ユーザに対するパスワード期間のチェックは無効になります。 パスワードに有効期限を設定しない場合は、interval を 0 に設定します。 期間 0 は、セキュリティ要件の低いユーザに対してのみ設定します。

interval- パラメータは、パスワード期間の設定を取り消します。 このパラメータの値がユーザのプロファイル グループに含まれている場合は、その値が使用されます。 それ以外の場合は、setoptions コマンドで設定したデフォルト値が使用されます。 このパラメータは chusr コマンドまたは editusr コマンドでのみ使用できます。

個々のユーザまたはグループのパスワード期間の設定

特定のユーザまたはプロファイル グループに対してパスワード期間を設定することもできます。 これらの設定内容は、システム全体に適用するユーザまたはグループのパスワード期間よりも優先されます。 指定した日数に達すると、CA Access Control は、現在のパスワードが期限切れになったことをユーザに通知します。 通知を受けたユーザは、ただちにパスワードを更新するか、猶予ログイン回数に達するまで古いパスワードを引き続き使用することができます。 猶予ログイン回数に達すると、ユーザはシステムへのアクセスが拒否されます。システム管理者に連絡して新しいパスワードを設定する必要があります。

ユーザのパスワード期間を設定または取り消すには、次のコマンドを入力します。

```
editusr {interval(NumDays) | interval-}
```

グループのパスワード期間を設定または取り消すには、次のコマンドを入力します。

```
editgrp password{(interval(NumDays)) | (interval-)}
```

NumDays の値には、ゼロ(0)または正の整数を指定する必要があります。 期間をゼロ(0)に設定すると、パスワード期間のチェックが無効になります。 パスワードに有効期限を設定しない場合は、期間を 0(ゼロ)に設定します。 期間 0 は、セキュリティ要件の低いユーザに対してのみ設定します。

interval- パラメータは、パスワード期間の設定を取り消します。 設定を取り消しても、期間の値がユーザ レコード内に設定されている場合は、ユーザ レコード内の値が使用されます。 それ以外の場合は、setoptions コマンドで設定したデフォルト値が使用されます。 このパラメータは、setoptions、chgrp、または editgrp コマンドのみで使用します。

猶予ログイン

パスワード チェックが有効になっている場合、CA Access Control は、ユーザがログインを試みるたびに、ユーザのパスワードの有効期限をチェックします。 パスワードの有効期限が切れても、ユーザにはその後数回はログインする「猶予」が与えられます。その後はログインできなくなります。

猶予ログインのオプションでは、ユーザのパスワード有効期限が切れた後、ユーザが使用できなくなるまでの間に許容される最大ログイン回数を設定します。 猶予ログイン回数には、0 ～ 255 の値を指定する必要があります。 猶予ログイン回数に達するとユーザはシステムへのアクセスを拒否されるため、システム管理者に連絡して新しいパスワードを設定する必要があります。 猶予ログイン回数が 0 に設定されている場合、ユーザはログインできません。 デフォルトのログイン数は 5 回です。

この方式を使用して、ユーザにパスワードの変更を強制することができます。 ユーザのパスワードをリセットし、ユーザがパスワードを変更できる 1 回の猶予ログインを付与します。

猶予ログインの追跡

パスワードの有効期限が切れた後にエンド ユーザが猶予ログインの回数を把握できるようにするには、ユーザの .login、.profile、または .cshrc ファイルに segrace ユーティリティへの呼び出しを追加します。 このように設定すると、segrace ユーティリティにより、猶予ログインの残り回数を示すメッセージが表示されます。 segracex ユーティリティを使用すると、ユーザのパスワードが有効期限切れであるかどうかを GUI で表示できます。

注: segrace ユーティリティおよび segracex ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

システム全体に適用する猶予ログイン回数のデフォルト値を設定するには、以下のコマンドを入力します。

```
setoptions password(rules(grace(nLogins)))
```

特定ユーザの猶予ログインを設定または取り消すには、以下のコマンドを入力します。

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

プロファイル グループの猶予ログインを設定または取り消すには、以下のコマンドを入力します。

```
chgrp groupName {grace(nLogins) | grace-}
```

chusr コマンドまたは chgrp コマンドでユーザに対して設定した値は、システム値よりも優先されます。

注: GROUP クラスの grace プロパティ、およびグローバル猶予ログイン設定では、ユーザのパスワードの有効期限が切れた後の猶予ログインの回数を設定します。 ただし、USER クラスの grace プロパティではパスワードの有効期限がすぐに切れるように設定されています。猶予ログインは、ユーザのパスワードの有効期限が切れた後、(GROUP レコードまたはシステムのデフォルトを使用して) 自動的に設定されます。 パスワードの有効期限は、グループには設定できません。ユーザにのみ設定できます。

第 8 章: ファイルおよびプログラムの保護

このセクションには、以下のトピックが含まれています。

ファイルおよびディレクトリへのアクセス制限

CA Access Control では、UNIX システムの権限チェック機能を保持しつつ、強化されたアクセス制御機能を追加します。

CA Access Control は、以下の各ファイル アクセス操作をインターセプトし、その特定の操作がユーザに許可されていることを確認してから、UNIX に制御を戻します。 かっこ内はアクセス タイプです。

- ファイルを作成する (create)
- 読み取りの目的でファイルを開く (read)

  注: ユーザがファイルに関する情報を取得する操作 (ls -l など) を実行できるかどうかを制御するために read 権限を必要とする場合は、設定 STAT_intercept を 1 に設定します。 詳細については、「リファレンス ガイド」を参照してください。
- 書き込みの目的でファイルを開く (write)
- ファイルを実行する (execute)
- ファイルを削除する (delete)
- ファイル名を変更する (delete、rename)
- アクセス許可ビットを変更する (chmod)
- 所有者を変更する (chown)
- タイム スタンプを変更する – たとえば touch コマンドの実行 (utime)
- ACL をサポートしているシステムのネイティブ ACL を (acledit コマンドを使用して) 編集する (sec)
- ディレクトリを変更する (chdir)

CA Access Control のアクセス チェックがネイティブ UNIX の認証と異なる点は以下のとおりです。

- CA Access Control の権限チェックは、ユーザの実効ユーザ ID(euid)ではなく、ログインに使用した元のユーザ ID に基づいて行われます。 たとえば、userA が su コマンドを実行して別のユーザの代理になった場合、userA がアクセスできるのは、userA がアクセスを許可されているファイルのみです。 su コマンドを実行して別のユーザになっても、UNIX の認証とは異なり、su コマンドを実行したユーザに、ターゲット ユーザのファイルにアクセスする権限が自動的に与えられることはありません。
- CA Access Control では、システム上のすべてのファイルにアクセスできる権限がスーパーユーザ(root)に自動的に与えられることはありません。 スーパーユーザは、システムの他のすべてのユーザと同様に、権限チェックの対象です。
- 権限チェックは、CA Access Control の標準アクセス リスト、条件付きアクセス リスト、日時制限、セキュリティ レベル、セキュリティ カテゴリ、およびセキュリティ ラベルに基づいて行われます。
- 管理者がユーザに対してファイルへのアクセスを明示的に許可しなかった場合、CA Access Control は、そのユーザがファイルへのアクセス権限を持つグループに属しているかどうかをチェックします。
- 個々のファイル アクセスは、CA Access Control の通常の監査手続きによって監査されます。
- ファイルを削除する場合、UNIX では親ディレクトリに対する WRITE アクセス権限がユーザに必要であるのに対して、CA Access Control では、指定したファイルに対する DELETE アクセス権がユーザに必要です。
- ファイルの名前変更を行う場合は、ソース ファイルに対する DELETE アクセス権限と、ターゲット ファイルに対する RENAME アクセス権限がユーザに必要です。また、UNIX では、親ディレクトリに対する WRITE アクセス権限もユーザに必要になります。
- すべてのユーザには、デフォルト設定とは関係なく/etc/passwd ファイルおよび /etc/group ファイルに対する永久的な READ アクセス権(最低限)が与えられます。 この権限の付与によって、システムが中断される可能性を抑えられます。
- CA Access Control データベースの FILE オブジェクトの所有者には、このオブジェクトによって保護されているファイルへのフル アクセス権が常に与えられます。
- chdir アクセス タイプは chdir コマンドを明示的に制御し、UNIX の場合とは動作が異なります。

ファイル保護システムに関する制限事項は、以下のとおりです。

- _restricted グループのメンバでないユーザについては、以下のファイルとディレクトリのみが CA Access Control で保護されます。
  - データベースに個々の名前で定義されているファイルおよびディレクトリ
  - データベースに定義された名前パターン(/etc/* など)と一致するファイルおよびディレクトリ

  グループ _restricted に属するユーザについては、すべてのシステム ファイルが CA Access Control で保護されます。 データベースに定義されていないファイルに対する権限は、FILE クラスの _default レコードに基づきます。
- CA Access Control では、保護が必要なリソースを示すすべてのファイル名およびディレクトリ名(ワイルドカードを使用したパターンを含む)のテーブルが保持されます。 このテーブルを格納するために使用できるメモリの量には制限があります。 通常、データベース内に個々の名前で定義できるファイルおよびディレクトリの最大数は 4096 で、名前パターンの最大数は 512 です。
- 一部のファイルは、明示的なアクセス ルールがない場合でも保護対象となります。このようなファイルには、CA Access Control データベース ファイル、監査ログ、および環境設定ファイルがあります。

  注: 詳細については、「リファレンス ガイド」の FILE クラスの説明を参照してください。

CA Access Control では、ファイルに対して以下のアクセス タイプを使用できます。

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

- UTIME
- WRITE

ファイル保護システムは、機密データが保存されている特定のファイルを保護する場合に便利です。 たとえば、CA Access Control を使用して保護できるファイルは、以下のとおりです。

- /etc/passwd
- /etc/group
- /etc/hosts
- /etc/shadow

データベースを保護(サーバ デーモンのみにアクセスを許可)し、さらにサイトにあるその他すべての機密ファイルを保護するには、CA Access Control を使用する必要があります。

常にアクセス制御が必要な一部のファイルは、たとえユーザが明示的にルールを指定しない場合でも、ルールによって制御されます。

ファイル保護の機能

seosd デーモンが起動すると、データベースに定義されている各個別ファイル オブジェクトに対して UNIX の stat コマンドが実行されます。 次に、各ファイル オブジェクトのエントリを含むテーブルがメモリ内に作成されます。 さらに、個別ファイルごとに i-node およびデバイスがテーブルに保存されます。CA Access Control では、この情報を使用して、ファイルへのハード リンクを保護することもできます。これは、ハード リンクの保護がデバイスおよび i-node に基づいて行われるためです。 データベースには、ファイルの i-node およびデバイスに関する情報は保存されません。

CA Access Control で新しいファイル ルールを作成するときは、以下の処理が行われます。

- ファイルが UNIX 環境にある場合は、最初に、ファイルに対して stat コマンドが実行され、次に、ファイルの i-node およびデバイスの情報と共に新規エントリがファイル テーブルに追加されます。
- ファイルが UNIX 環境に存在しない場合は、ファイル名の新規エントリがファイル テーブルに追加されます(i-node およびデバイスの情報は含まれません)。 この新規エントリは、包括的なファイル オブジェクトに対するエントリと同じです。 同時に、カーネルの内部テーブルには、i-node およびデバイス情報の作成時にこのファイルをチェックする必要があるという指示が保存されます。 その後にファイルが作成されると、カーネルによってファイルの作成がインターセプトされ、seosd がファイル テーブル内のファイルのエントリを更新できるように、ファイルの i-node およびデバイス情報が seosd に通知されます。

ファイルを削除すると、CA Access Control により、seosd ファイル テーブル内のエントリが削除されます。ただし、エントリは、再作成する場合に備えて、データベースに保持されます。

ファイルの保護

selang で保護ファイルを定義するには、以下のコマンドを入力します。

```
newres FILE filename
```

たとえば、/tmp/binary.bkup ファイルを登録するには、以下のコマンドを入力します。

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

注: アクセス タイプを指定せずにファイル ルールを定義すると、デフォルト アクセス権に NONE が割り当てられます。 この場合、ファイルの所有者のみがファイルにアクセスできます。

大部分の保護ファイルは、スーパーユーザによるアクセスから保護する必要があります。これを行わないと、スーパーユーザのパスワードを知っているユーザに、保護ファイルへのアクセス権が自動的に与えられます。 同時に、ファイル所有者以外のすべてのユーザによるファイルへのアクセスも防止できます。

同じような名前の複数のファイルを保護するには、ワイルドカードを含むファイル名パターンを使用します。 ワイルドカードは、* (0 個以上の文字)および? (/ を除く、任意の 1 文字)です。

指定したパターンは、ファイルの完全パス名と照合されます。その結果、パターン /tmp/x* は、/tmp/x1、/tmp/xxx、および /tmp/xdir/a という名前のファイルに一致します。

CA Access Control で指定できないパターンは、/*、/tmp/*、および /etc/* です。

重要: ファイル名パターンは非常に影響力の大きいツールであるため、練習でむやみにいろいろなパターンを試さないでください。

たとえば、以下のコマンドでは、/tmp ディレクトリ内の、a で始まり b で終わるファイル名を持つすべてのファイル(/tmp/axyz/axyzb のようなファイルも含まれる)を保護対象として定義しています。

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

ファイル リソース名でのワイルドカードの使用

ファイル リソース名にワイルドカードを使用することで、複数のファイルに対応するファイル レコードを作成できます。ワイルドカードのパターンと一致する名前を持つファイルはすべて、レコードに関連付けられたアクセス権限によって保護されます。

使用できるワイルドカードは、以下のとおりです。

- * - 任意の複数文字に対応します。
- ? - 任意の 1 文字に対応します。

物理リソースの名前が複数のリソース レコード名と一致する場合、そのリソースには、ワイルドカードを除く、最も長い一致が使用されます。

CA Access Control では、ファイル リソース名として以下のパターンは使用できません。

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

例: ファイル リソースでのワイルドカードの使用

ファイル リソース /usr/lpp/bin/* は、/usr/lpp/bin 下にある(深くネストされている)すべてのファイルおよびサブディレクトリを保護します。

ファイル アクセスの制限

スーパーユーザによるファイルへのアクセスを selang で制限するには、newres コマンドの長いバージョンを使用します。 たとえば、スーパーユーザ、および myuser 以外のユーザによる /tmp/binary.bkup ファイルへのアクセスを防止するには、以下のコマンドを使用します。

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

このコマンドは、以下のことを示します。

1. /tmp/binary.bkup を保護ファイルとして定義します。
2. ユーザ myuser をファイルの所有者に設定して、myuser にファイルへのアクセス権を与えます。
3. ファイルのデフォルト アクセス権を NONE に設定して、他のユーザによるファイルへのアクセスを防止します。 ファイルへのアクセスを他のユーザに許可するには、そのファイルのアクセス ルールを明示的に定義する必要があります。

重要: root 権限で selang コマンドを起動し、別のユーザを所有者として明示的に指定せずに FILE レコードを定義すると、定義したレコードの所有者は root になります。所有者である root ユーザ(つまり、root ユーザとしてログインするすべてのユーザ)には、ファイルに対する完全で自由なアクセス権が与えられています。

注: seos.ini ファイルのトークン use_unix_file_owner を yes に設定できます。 この設定によって、UNIX の一般ユーザは、所有しているファイルに対するアクセス ルールを定義できます。

ファイル アクセスの防止

所有者のない FILE レコードを定義すると便利な場合があります。 所有者のない FILE レコードを selang で定義するには、特別な所有者である「nobody」を使用します。

たとえば、/tmp/binary.bkup ファイルを保護ファイルとして定義し、すべてのユーザがこのファイルにアクセスできないようにするには、以下の selang コマンドを入力します。

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

この newres コマンドを実行すると、たとえこのコマンドを定義したユーザ(root であるかどうかに関係なく)であっても、ファイルにアクセスすることはできません。 すべてのユーザをファイルにアクセスできないようにした後、通常は、そのファイルへのアクセス権を 1 人以上のユーザに明示的に与える必要があります。

保護ファイルに対するユーザ アクセスを明示的に許可するには、authorize コマンドを使用します。 たとえば、/tmp ディレクトリ内の Jo で始まるすべてのファイルに対する更新アクセス権限をユーザ「userJo」に与えるには、以下のコマンドを入力します。

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

注: CA Access Control では、独自のデータベースに定義されているファイルのみが保護されます。

ファイル情報の取得に対するユーザ制限

ユーザに、ファイルまたはディレクトリに対する読み取りアクセス許可を付与していない場合は、デフォルトで、start 関数を使用してファイルに関する情報を取得することができます。 たとえば、ファイル /tmp/abc に対する読み取りアクセス許可を持たないユーザは、以下の操作を実行できます。

```
ls -l /tmp/abc
```

読み取り アクセス許可を持たないユーザが、ファイル情報を取得することを防止するには、STAT_intercept 設定を 1 に設定します。

注: STAT_intercept 設定の詳細については、「リファレンス ガイド」を参照してください。

デフォルト アクセス権の表示

一致するレコードが見つからない場合に、_restricted グループ内のユーザのデフォルト アクセス権を表示するには、クラスの _default レコードに対して selang の showres コマンドを使用します。

たとえば、CA Access Control データベースには存在しないファイルに対して _restricted グループ内のユーザが持っているデフォルト アクセス権を表示するには、selang の showres コマンドを使用して、FILE クラスの _default リソースを表示します。

```
showres FILE _default
```

注: その他のすべてのユーザには、特定の CA Access Control データベース ルールで定義されたアクセス権が付与されます。

条件付きアクセス制御リストの使用

ファイルアクセスに使用するプログラムを条件として、ファイルへのアクセス権を指定することができます。 このようにファイル アクセスを条件付きにすることを、Program Pathing と呼びます。

注: ファイルにアクセスする条件として指定されたプログラムがシェル スクリプトの場合は、そのシェル スクリプトの 1 行目に #!/bin/sh が含まれていなければなりません。

以下のコードは、パスワード変更プログラム /bin/passwd を使用するすべてのプロセスに対して、/etc/passwd ファイルの更新を許可する例です。 /bin/passwd 以外から行われた、/etc/passwd ファイルに対するアクセスの試みは、すべてブロックされます。

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

この newres コマンドは、/etc/passwd ファイルを CA Access Control に定義し、ファイルの所有者も含む任意のユーザに対してファイルの読み取りを許可します。 下の authorize コマンドは、/bin/passwd プログラムの制御下でアクセスが行われた場合に、すべてのユーザにファイルへのアクセスを許可します。 この方法でパスワード ファイルを保護すると、ユーザが /bin/passwd プログラムを使用していない場合は、トロイの木馬による /etc/passwd ファイルへのエントリの追加や「users」グループのユーザによるパスワード ファイルの更新がブロックされます。

条件付きアクセス リストは、データベース管理システム(DBMS)のファイルに対するアクセスを制御する場合にも役に立ちます。 通常は、データベース ベンダによって提供されているプログラムやユーティリティを使用した場合にのみ、ユーザが DBMS ファイルにアクセスできるようにします。 以下のコマンドについて考えてみましょう。

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

この 2 つの authorize コマンドは、DBMS バイナリ ディレクトリに属しているプログラム pgm1 またはプログラム pgm2 のいずれかによってアクセスが行われた場合、すべての CA Access Control ユーザに、DBMS システムのファイルである xyz へのアクセスを許可します。 ユーザのオペランドにアスタリスクが使用されていることに注意してください。 このアスタリスクは、CA Access Control に定義されているすべてのユーザを指定します。 このアスタリスクの使用は、デフォルト アクセスの概念に類似しています。ただし、デフォルト アクセスは CA Access Control に定義されていないユーザにも適用される点において異なります。 CA Access Control データベースで定義されていないユーザに対して _undefined グループを使用できます。

また、Unicenter TNG カレンダの ACL プロパティを使用すると、Unicenter TNG カレンダのステータスに基づいて、現在のリソースに対するアクセスを特定のユーザおよびグループに対して許可または拒否できます。 Unicenter TNG カレンダの ACL プロパティには、標準プロパティと制限プロパティの 2 種類があります。

たとえば、以下のコマンドを実行すると、basecalendar という標準カレンダの条件付きアクセス制御リストに george というユーザが追加されます。

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

また、以下のコマンドを実行すると、Unicenter TNG カレンダから george というユーザが削除されます。

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

拒否アクセス制御リストの使用

NACL（Nagative Access Control List）を使用して、ユーザまたはグループ固有のアクセス タイプを拒否できます。

CA Access Control のコマンド言語（selang）で、以下のコマンドを入力して、アクセスを拒否します。

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

_abspath グループによるトロイの木馬のブロック

$PATH 変数のすべての相対パス名、特に「カレント ディレクトリ」を意味するドット（.）パス名は、セキュリティ上の弱点です。 以下の例を考えてみましょう。

- root の PATH 変数の先頭には、カレント（.）ディレクトリがあります。
- 悪意のあるユーザが破壊的なプログラム（トロイの木馬）を作成し、/tmp/ls として保存します。
- しばらくすると、悪意のあるユーザの意図したとおりに、root ユーザが /tmp ディレクトリから ls コマンドを発行します。 その結果、通常の ls コマンドが実行される代わりに、完全な管理者権限を持つ root ユーザによって、/tmp ディレクトリに保存されていたトロイの木馬が実行されます。

このセキュリティ上の脆弱性を排除するために、CA Access Control には_abspath というユーザ グループが用意されています。 _abspath グループのすべてのメンバは、プログラム起動時に相対パス名を使用することが禁止されます。

他のグループと同様に、ユーザを _abspath グループに追加できます。 この設定は次回のログインから有効になり、ユーザはプログラムにアクセスするときに相対パス名を使用できません。

ネイティブ UNIX セキュリティとの同期

CA Access Control のアクセス許可はネイティブ UNIX のアクセス許可よりも複雑ですが、ネイティブ UNIX のアクセス許可を CA Access Control のアクセス許可に同期させることができます。 つまり、アクセス許可を一致させることができます。 ただし、この同期にはいくつかの制限事項があります。

- 同期は遡及して適用できません。 同期がいったん有効になると、新しく発行される CA Access Control の承認コマンドをすべて制御できます。ただし、既存のアクセス ルールは制御されません。
- CA Access Control で与えられたアクセス許可は UNIX に渡されますが、UNIX で与えられたアクセス許可は CA Access Control に渡されません。
- UNIX 自体のアクセス許可システムの制約により、UNIX では、CA Access Control の簡略化されたアクセス許可より複雑な許可は、適用できない場合があります。 アクセス制御リスト(ACL)を備えたバージョンの UNIX でも、CA Access Control の ACL の複雑な機能をすべて反映することができない場合があります。

CA Access Control に同期させることができる ACL を備えた UNIX のプラットフォームは、Sun Solaris、HP-UX、および Tru64 です。

このような ACL がない場合でも、従来からある UNIX の rwx 権限を CA Access Control の権限に、ある程度まで同期させることができます。

同期は、authorize コマンドの UNIX オプションと seos.ini ファイルの SyncUnixFilePerms トークンの組み合わせによって制御されます。

- UNIX オプションを指定することによって、authorize コマンドは、CA Access Control のみでなく UNIX でも実装を行います。 このコマンドでは、それまでアクセス許可がなかった場合でも UNIX のアクセス許可を設定できます。

  (UNIX オプションを使用しない場合、selang のコマンドは UNIX セキュリティに影響を与えません。 また、CA Access Control 権限は、UNIX によって制御されている箇所では無効になります。 したがって、selang で UNIX によるアクセス制御を解除する唯一の方法は、authorize コマンドの UNIX オプションを使用することです)。

- authorize コマンドの UNIX オプションは、SyncUnixFilePerms トークンが seos.ini ファイルの[seos]セクションで適切に設定されている場合にのみ動作します。 このトークンでは、以下の値が使用できます。

  - no は、ACL 権限を同期させないことを指定します。 デフォルト値です。
  - warn は、ACL 権限を同期させないが、権限とネイティブ UNIX 権限が競合した場合に警告を発行することを指定します。
  - traditional は、グループの rwx 権限を CA Access Control の ACL に従って調整することを指定します(個々のユーザの権限は UNIX にコピーされません)。
  - acl は、UNIX の ACL を CA Access Control の ACL に従って調整することを指定します。
  - force は、UNIX 環境のアクセス属性を CA Access Control の defaccess 権限に従って調整することを指定します。

  SyncUnixFilePerms トークンの値の変更を有効にするには、seosd デーモンを再起動する必要があります。

例: 同期

以下の例では、/var/temp/newdata というファイルと fowler というユーザを使用し、FILE クラスのレコードにファイルがすでに定義されていると想定しています。

1. seos.ini ファイルを編集できるように、seosd デーモンを停止します。

   ```
   # secons -s
   ```

2. seos.ini ファイルを編集する権限を持つユーザとしてログインし、この seos.ini ファイルの[seos]セクションで SyncUnixFilePerms 行を以下のように編集します。

   ```
   SyncUnixFilePerms = acl
   ```

   acl は、UNIX オプションを使用すると、UNIX の ACL が CA Access Control の ACL に応じて調整されることを意味します。 UNIX オプションの機能は、トークンが acl に設定されている限り有効です。

3. seosd デーモンを再起動します。

   # seosd

4. selang を起動し、以下の selang コマンドを発行します。

   authorize FILE /var/tmp/newdata uid(fowler) access(r w) unix

   このコマンドにより、fowler には、/var/tmp/newdata に 対する Read および Write アクセス権限が与えられます。UNIX オプションを指定すると、対応するネイティブ UNIX 権限が与えられます。

HP-UX の制限

HP-UX の ACL では、CA Access Control の ACL を正しく反映できない場合があります。

- HP-UX の ACL では、ユーザとグループの組み合わせごとにアクセス権限を割り当てます。 つまり、割り当てられたアクセス権限は、ユーザのプライマリ グループも指定されている場合のみ、指定されたユーザに適用されます。

  また、CA Access Control では、ユーザとグループの組み合わせではなく、ユーザまたはグループごとにアクセス権限を割り当てます。

  したがって、CA Access Control 権限が HP-UX のユーザとグループの組み合わせにマップされると、ユーザまたはグループのいずれかが「*」または「any」に設定されます。

- HP-UX は、ボリューム マネージャ(LVM)の制御下にあるファイル システムの ACL はサポートしません。 そのため、一部の重要な HP-UX マシンでは、root ファイル システムでのみ ACL の同期が許可されます。

- HP-UX の ACL は 16 エントリまでに制限されています。 CA Access Control の同期では、使用可能なエントリを可能な限り効率的に使用しますが、16 エントリでは、すべての CA Access Control の ACL を完全に反映できない場合があります。

Sun Solaris の制限

Sun Solaris の場合は、ネイティブ UNIX の ACL が /tmp ディレクトリに実装されていません。

機密ファイルの監視

Watchdog 機能は、指定したファイル以外に setuid/setgid プログラムのバイナリを保護することができます。 seoswd ユーティリティ(Watchdog デーモン)は、以下の 2 点を継続的にチェックしています。

- seosd デーモンが稼動中であり、応答しているかどうか (必要に応じて、Watchdog デーモンによって seosd デーモンが再起動されます)。
- trusted プログラムまたはファイルがユーザによって変更されたかどうか (変更されている場合は、seoswd によりファイルの実行が阻止されます)。

seosd デーモンが fork で複製されるたびに、seoswd プログラムが自動的に実行され、Watchdog 機能が開始されます。

注: seoswd の詳細については、「リファレンス ガイド」を参照してください。

seos.ini ファイルには、Watchdog 機能のスキャン操作とタイムアウトの値を制御する複数のトークンがあります。 また、このファイルには、これらの値に関する最新のドキュメントも含まれています。

注: seos.ini ファイルの詳細については、「リファレンス ガイド」を参照してください。

Watchdog 機能を使用すると、setuid プログラムおよび setgid プログラムに対して行う内容と同じバックグラウンド チェックを通常のファイルに対して実行できます。このチェックには、ファイルが変更された際の監査レコードの生成も含まれます。

たとえば、セキュリティ管理者のみが /etc/inittab ファイルの変更を許可される環境設定を考えてみましょう。 CA Access Control でファイルを監視し、変更があった場合に警告が生成されるようにするには、以下の selang コマンドを使用します。

```
newres SECFILE /etc/inittab
```

これで、/etc/inittab ファイルへの変更が継続的に監視されます。

setuid プログラムおよび setgid プログラムの保護

ユーザ ID 設定(setuid)プログラムは、UNIX で最も頻繁に使用されるプログラムの 1 つです。 setuid プログラムを起動するプロセスによって、setuid プログラムの所有者の ID が自動的に取得されます。 setuid プログラムの所有者が root である場合、setuid プログラムを起動すると、一般ユーザは自動的にスーパーユーザになります。 setuid プログラムが起動すると、スーパーユーザが実行できるすべての操作をプロセスで実行できるようになります。したがって、setuid プログラムが正しく使用されるようにすることが非常に重要です。 setuid プログラム内のバック ドアまたはシェルにより、システム上のすべてのファイルに対するアクセス権がユーザに与えられます。

CA Access Control では、PROGRAM クラスを使用して setuid プログラムおよび setgid プログラムを保護しています。 CA Access Control は、インストール時にデフォルトですべてのプログラムの実行を許可します。 データベースで trusted プログラムを定義した後、プログラムが trusted プログラムとして定義されていない場合は setuid プログラムまたは setgid プログラムの実行を禁止するように、CA Access Control の動作を変更できます。 たとえば、/bin/ps(プロセス状態表示プログラム)を setgid プログラムとして(本来の目的どおりに)実行できるようにするには、以下の selang のコマンドを使用します。

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

/bin/ps プログラムが、CA Access Control の trusted プログラムとして登録されます。 次に、CRC、i-node 番号、サイズ、デバイス番号、所有者、グループ、アクセス許可ビット、最終変更日時、および(オプションで)レコード内のその他のデジタル署名が確認され、データベースの PROGRAM クラスのレコードに格納されます。

Watchdog 機能により、プログラムの CRC、サイズ、i-node、および他の特性が定期的にチェックされます。 このいずれかの値が変更された場合、Watchdog 機能では、trusted プログラム リストからプログラムを削除してプログラムへのアクセスを拒否するように、seosd に対して自動的に指示が出されます。 これにより、setuid プログラムを変更または移動して、プログラムを不正に使用することができなくなります。 上記の newres コマンドの例では、データベースに定義されていないユーザも含め、すべてのユーザに /bin/ps コマンドの実行を許可していることに注意してください。

untrusted setuid プログラムは、UNIX ベースのオペレーティング システムにおいて最も危険なセキュリティ ホールになる可能性があります。 trusted プログラムのアクセス ルールを使用することで、セキュリティ管理者は、テストとチェックを終えた特定の trusted プログラムのみに setuid の使用を許可し、プログラムの完全性を保証することができます。 ただし、すべてのユーザが、trusted 実行可能ファイルを自動的に起動できるわけではありません。アクセス ルールによって、その setuid プログラムへのアクセス権限を与えるユーザおよびグループを明示的に指定する必要があります。 たとえば、以下の 2 つの selang のコマンドでは、/bin/su の実行権限のみをシステム部門のユーザ(sysdept グループ)に与えています。

```
newres PROGRAM /bin/su defaccess(NONE)
authorize PROGRAM /bin/su gid(sysdept) access (EXEC)
```

データベースで定義されているすべてのユーザを指定するには、アスタリスク(*)を使用します。 たとえば、CA Access Control に定義されているすべてのユーザに su コマンドの実行を許可するには、以下のコマンドを入力します。

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

この説明は、setgid 実行可能ファイルの場合にも当てはまります。

nr コマンドおよび er コマンドを使用して、PROGRAM クラスに setuid プログラムおよび setgid プログラムを登録できます。 重要な setuid 以外および setgid 以外のプログラムも、同様に PROGRAM クラスに登録できます。 これらのプログラムの FILE ルールを定義して、権限のないユーザによる更新を防ぎます。 プログラムが Untrusted の場合にプログラムの実行を許可する場合は(アップグレード後に、プログラムは再度 Trusted 状態にされずに実行されます)、blockrun プロパティを no に設定します。

- blockrun プロパティが yes に設定されている場合、プログラムは再度 Trusted 状態になるまで実行されず、関連する PACL によって許可されているファイルへのアクセスを許可されません。 PACL は、プログラムが再度 Trusted 状態になるまで事実上無効になります。
- blockrun プロパティが no に設定されている場合、プログラムは実行されます。 ただし、プログラムは関連する PACL によって許可されているリソースにアクセスすることはできません。

blockrun プロパティの値を yes に設定するには、以下の editres/newres コマンドを使用します。

```
er program /bin/ps blockrun
```

blockrun プロパティの値を no に設定するには、以下の editres/newres コマンドを使用します。

```
er program /bin/ps blockrun-
```

デフォルトでは、PROGRAM クラスに登録されているすべてのプログラムの blockrun プロパティが yes に設定されます。 この値は、seos.ini ファイルの SetBlockRun トークンを使用して変更できます。 詳細については、seos.ini ファイルの説明を参照してください。

setuid/setgid プログラムの自動定義

CA Access Control では、すべての setuid プログラムおよび setgid プログラムを自動的に定義できます。 ユーティリティ プログラム /bin/seuidpgm を使用して一連のコマンドを作成し、すべての setuid プログラムとその実行権限を定義します。

たとえば、ファイル システム全体で setuid プログラムおよび setgid プログラムをスキャンし、生成された selang のコマンドを /tmp/pgm_script ファイルに書き込むには、以下の selang コマンドを入力します。

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

seuidpgm によって生成された出力ファイルは、発行する前に、必要に応じて編集および変更することができます。

注: seuidpgm ユーティリティの詳細については、「リファレンス ガイド」を参照してください。 setuid プログラムまたは setgid プログラム以外のプログラムを同様に保護する方法については、「 リファレンス ガイド」の SECFILE クラスの説明を参照してください。

条件付きアクセス

アクセス権限に関するもう 1 つの高度な機能は、条件付きアクセス ルールです。 たとえば、securedSU という非常に安全性の高いバージョンの su コマンドがあり、このコマンドは、ユーザがスーパーユーザになる前に指紋の読み取り装置で本人確認を行うと仮定します。

このプログラムを実行している場合のみスーパーユーザになることを UserX に許可する 1 つの方法として、以下のように条件付きアクセス ルールを設定する方法があります（ルールを設定する前に、USER.root に defaccess(none) も設定する必要があります）。

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

login コマンドの保護

/bin/login の使用は、スーパーユーザのみに限定することを強くお勧めします。 スーパーユーザに限定しない場合、他のユーザのパスワードを知っているユーザは、他のユーザとしてログインし、他のユーザのパスワードを指定して、ユーザ ID 変更と端末に関するすべての制限を省略することができます。

/bin/login を使用する権限を selang で変更するには、以下のコマンドを使用します。

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

通常のプログラムの保護

CA Access Control では、setuid プログラムおよび setgid プログラムを保護するのと同じ方法で通常のプログラムも保護できます。 そのためには、PROGRAM クラスの blockrun プロパティを、選択する値に設定します。

注: 有効なオプションの詳細については、「リファレンス ガイド」を参照してください。

カーネル モジュールのロードとアンロードの保護

カーネル モジュールは UNIX オペレーティング システムのコンポーネントです。実行中のカーネルにロードすることで拡張を行い、不要になったときにはアンロードすることができます。 これにより、ベース カーネルの通常機能全般をカバーするうえで必要なメモリ リソースを無駄にせず、必要に応じて機能をロードするという柔軟性が実現します。

CA Access Control では、カーネル モジュールの保護を有効および無効に設定できます。 カーネル モジュールの保護を有効にすると、CA Access Control は、カーネル モジュールをロードおよびアンロードするシステム コールをインターセプトし、要求されたアクセスを、データベースにある関連付けられたレコード、つまり KMODULE クラス内のレコードと照合します。 カーネル モジュール レコードに対するアクセスが要求された場合、CA Access Control では、要求されたアクセスは「ロード」または「アンロード」のいずれかです。

Linux 以外のすべてのシステムでは、KMODULE レコードの名前は(完全パスではなく)カーネル モジュール ファイルの名前と同じにする必要があります。 これは、モジュールの名前がファイルの名前と同じであるためです。 Linux では、KMODULE レコードの名前は、カーネル モジュールの名前のみと同じにする必要があるので、実際のファイル名と異なる場合もあります。 Linux でファイル名を変更しても、Linux が使用するモジュール名は変更されず、KMODULE レコードは有効のままです。

カーネル モジュールのロードでのファイル パス チェックを有効にした状態で、要求されたアクセスがロードされると、CA Access Control は、以下の追加チェックを実行します。

- KMODULE レコードの filepath プロパティに、有効な絶対ファイル パスのみが保持されていること。
- パス名 filepath 内のファイルに、KMODULE レコード名と一致するモジュールがあること。
- カーネル モジュールが KMODULE プロパティ(Linux 以外のシステムでは filepath、Linux システムでは signature)と一致すること。

  注: CA Access Control は、Linux システム上のカーネル モジュール ファイルに対して一意のシグネチャを生成し、このシグネチャをカーネル モジュール レコードの signature プロパティの値として挿入します。 CA Access Control は、アクセスのたびにシグネチャをチェックします。 シグネチャを手動で入力する必要はありません。CA Access Control が自動的にシグネチャを算出して挿入します。 ただし、seretrust ユーティリティを使用してこれを実行することもできます。

詳細情報:

カーネル モジュールのロードでのファイル パス チェックの有効化および無効化(103 ページ)

カーネル モジュールの保護

カーネル モジュールのロードとアンロードを保護することで、オペレーティング システムを保護することができます。

カーネル モジュールを保護するには、以下の手順に従います。

1. カーネル モジュールの保護が有効になっていることを確認します。
2. CA Access Control に KMODULE レコードを作成します。

   カーネル モジュールを作成するには、以下を定義する必要があります。

   - カーネル モジュールの名前

     Linux 以外のすべてのシステムでは、KMODULE レコードの名前は(完全パスではなく)カーネル モジュール ファイルの名前と同じにする必要があります。これは、モジュールの名前がファイルの名前と同じであるためです。 Linux では、KMODULE レコードの名前は、カーネル モジュールの名前のみと同じにする必要があるので、実際のファイル名と異なる場合もあります。

- レコードの所有者(デフォルトでは、モジュールを作成しているユーザ)
- (オプション)カーネル モジュール ファイルの絶対ファイル パス、またはモジュールに複数のバージョンがある場合は、ファイル パスのリスト

注: HP システムおよび Solaris システムでは、特別なカーネル モジュール _ALL_MODULES を定義して、すべてのカーネル モジュールのアンロードを保護できます。

3. モジュールのロードおよびアンロードを実行する権限を付与するユーザまたはグループを定義します。

例: selang コマンドを使用してカーネル モジュールを保護する

以下の selang コマンドは、CA Access Control に対してカーネル モジュール serial.o を定義して認証し、エンタープライズ ユーザ kadmin にこのモジュールのロードおよびアンロードを実行する権限を付与します。

```
newres kmodule serial.o owner(kadmin) defaccess(none) \
filepath(/lib/modules/2.2.19/serial.o:/lib/modules/2.2.20/serial.o)
authorize kmodule serial.o access(load, unload) xuid(kadmin)
```

カーネル モジュールの保護の有効化および無効化

カーネル モジュールの保護が有効である場合、CA Access Control は、CA Access Control データベースに定義されているカーネル モジュールのロードおよびアンロードをチェックします。

デフォルトでは、CA Access Control は、カーネル モジュールの保護を有効にします。

カーネル モジュールの保護を有効または無効にするには、setoptions コマンドなどを使用して、KMODULE クラスを有効または無効にします。

例: selang を使用してカーネル モードの保護を有効にする

以下の selang コマンドは、カーネル モードの保護を有効にします。

```
setoptions class+(kmodule)
```

例: selang を使用してカーネル モードの保護を無効にする

以下の selang コマンドは、カーネル モードの保護を無効にします。

```
setoptions class-(KMODULE)
```

カーネル モジュールのロードでのファイル パス チェックの有効化および無効化

カーネル モジュールの保護が有効である場合は、カーネル モジュールのロードでのファイル パス チェックも有効にできます。 これが有効な場合、CA Access Control は、ロード対象のカーネル モジュールが KMODULE レコードの filepath プロパティと一致すること(Linux 以外のシステムの場合)、または KMODULE レコードの signature プロパティ(Linux システムの場合)と一致することをチェックします。

ファイル パス チェックを有効にするには、環境設定ファイル seos.ini の seosd セクションで、special_check トークンを yes に設定します(デフォルトは no です)。

CA Access Control は、ファイル パス チェックとカーネル モード保護の両方が有効である場合のみ、ファイル パス チェックを実行します。

例: seini ユーティリティを使用してカーネル モジュールのロードでのファイル パス チェックを有効にする

カーネル モジュールのロードでのファイル パス チェックを有効にするには、以下のように、seini ユーティリティと secons ユーティリティを使用します。

```
seini -s seosd.special_check yes
secons -rl
```

kill コマンドに対するバイナリ

データベース サーバやアプリケーション デーモンなどのミッションクリティカルなプロセスは、サービス妨害攻撃から保護する必要があります。 ネイティブ UNIX セキュリティ システムでは、プロセス ユーザ ID に基づいてプロセスの保護を行います。 これは、ネイティブ UNIX の環境では、root ユーザはすべてのプロセスに対して何でも実行できることを意味します。 CA Access Control は、プロセスで実行されている実行可能ファイルに基づいたルールを定義することによって、UNIX プロセスの保護を補強します。 CA Access Control プロセスの保護は、そのプロセスのユーザ ID に依存しません。 CA Access Control で保護するすべてのプロセスを PROCESS クラスのレコードに定義する必要があります。

たとえば、ASCII ビューアである /bin/more の強制終了(kill)を阻止するには、以下の手順に従います。

1. selang を起動します。
2. 以下の selang コマンドを入力します。

   newres PROCESS /bin/more defaccess(N) owner(nobody)

   このコマンドは、強制終了(kill)することができないプロセスとして /bin/more を定義します。したがって、デフォルトのアクセス権限は、none(N)です。
   owner(nobody) を設定すると、たとえこのルールを定義したユーザであっても、/bin/more プロセスを強制終了(kill)することはできません。
3. selang を終了します。
4. 手順 2 で定義したルールをテストします。
   a. 以下のコマンドを入力します。

      /bin/more /tmp/seosd.trace

   b. /tmp/seosd.trace ファイルのサイズが、/bin/more をただちに終了させない程度の大きさであると仮定して、Ctrl キーを押しながら Z キーを押して、/bin/more プロセスを一時停止します。
   c. 以下のコマンドを入力して、一時停止したジョブの強制終了(kill)を試みます。

      kill %1

      この試みは失敗し、CA Access Control では、「アクセス許可が拒否されました」というメッセージが表示されます。

例外として、特定のユーザが /bin/more プロセスを強制終了(kill)できるようにするには、以下の selang コマンドを入力します。

authorize PROCESS /bin/more uid(username)

注: システム上の他のバイナリ実行可能ファイルの強制終了(kill)を防止する場合も、同じ手順に従います。

CA Access Control では、通常の kill シグナル(SIGTERM)、およびアプリケーションでマスクできない kill シグナル(SIGKILL および SIGSTOP)が保護されます。 CA Access Control は、SIGHUP や SIGUSR1 などの他のシグナルをプロセスに渡し、kill シグナルを無視するか、kill シグナルに反応するかを決定します。

第 9 章: ログイン コマンドの制御

このセクションには、以下のトピックが含まれています。

ログイン プロセスの制御

CA Access Control には、ログインの保護機能が 2 種類 (端末による保護機能とアプリケーションによる保護機能) 用意されています。 TERMINAL クラスを使用すると、ログインできるユーザと端末またはホストを指定できます。

注: TERMINAL クラスの詳細については、「リファレンス ガイド」を参照してください。

LOGINAPPL クラスを使用すると、特定のログイン アプリケーション (telnet、FTP、rlogin など) を使用してログインできるユーザまたはグループを指定することもできます。 クラスのアクセス ルールを設定することによって、各ログイン アプリケーションに固有のルールを定義します。 たとえば、すべてのユーザにホストへの FTP 接続を許可し、限られた数のユーザにシステムへの telnet 接続を許可し、すべてのユーザにシステムへの rlogin 接続を禁止する、といったルールを定義できます。 LOGINAPPL クラスの各レコードで、特定のログイン アプリケーションに関するアクセス ルールを定義します。

例: LOGINAPPL

たとえば、FTP アプリケーションの使用を匿名ユーザのみに許可するには、以下の手順に従います。

1. 以下の selang コマンドを使用して、FTP のデフォルト アクセス権を none に変更します。

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

2. 以下の selang コマンドを使用して、ユーザ anonymous に FTP の使用を許可します。

   ```
   auth LOGINAPPL FTP uid(anonymous) access(X)
   ```

account というグループに属するユーザが telnet のみを使用するように制限するには、以下の手順に従います。

1. 以下の selang コマンドを使用して、rlogin と rsh の使用を禁止します。

   auth LOGINAPPL(RLOGIN RSH) gid(account) access(N)

2. 以下の selang コマンドを使用して、account というグループに telnet の使用を許可します。

   auth LOGINAPPL TELNET gid(account) acc(X)

注: 上記の例では、RLOGIN および RSH の制限について説明しましたが、他のログイン プログラムも制限できます。

新規のログイン プログラムを追加または使用する場合は必ず、新しい LOGINAPPL レコードを追加する必要があります。

ログイン インターセプト シーケンスは、常に、setgid イベントまたは setgroup イベントで始まります。これらのイベントをトリガといいます。 このシーケンスは、ユーザの ID を実際にログインしたユーザに変更する setuid イベントで終わります。

ログイン アプリケーションが発行する各種システム コールは、CA Access Control でログイン アクティビティを監視するために使用されます。 標準のログイン アプリケーションについては、これらのログイン順序があらかじめ設定されています。 これらのログイン順序は、CA Access Control のトレース ファイルを調べることによって確認できます。

注: LOGINAPPL クラス、およびシーケンスの設定方法に関する詳細については、「 selang リファレンス ガイド」を参照してください。

包括的なログイン アプリケーションの制御

CA Access Control では、包括的なログイン アプリケーションを制御および保護することもできます。つまり、特定のルールを汎用パターンに一致させるログイン アプリケーションのグループを保護できます。 包括的なログイン アプリケーションを定義するには、LOGINAPPL クラスを使用します。

包括的なログイン アプリケーションの定義

包括的なログイン アプリケーションを selang で定義するには、LOGINPATH パラメータを除き、通常のログイン制限を設定するときと同じコマンドを使用します。
LOGINPATH パラメータには、[、]、*、? のうち 1 つ以上の文字を使用した正規表現で構成された包括的なパスを含める必要があります。 たとえば、包括的な telnet アプリケーションを定義するには、以下のコマンドを使用します。

er LOGINAPPL GENERIC_TELNET loginpath(/usr/sbin/in.tel*)

包括的なログイン プログラムのインターセプト

通常のログイン制限を使用する場合、適用されるルールは明らかです。たとえば、インターセプトされたログイン プログラムが loginpath プロパティに指定されている LOGINAPPL オブジェクトがデータベースに存在する場合は、そのオブジェクトのルールが適用されます。

ただし、包括的な LOGINAPPL オブジェクトの場合、CA Access Control では以下の処理が行われます。

1. seosd は、インターセプトされたログイン アプリケーションに完全に一致するもの（LOGINAPPL オブジェクトに一致するログイン パス）を検索します。見つかった場合は、そのオブジェクトのルールが適用されます。
2. 見つからなかった場合は、包括的なログイン パスを使用して、一致する LOGINAPPL オブジェクトの検索が続行されます。
3. 一致するオブジェクトが複数ある場合は、より詳細に一致するオブジェクトのルールが適用されます。

端末を使用するユーザ権限の定義

侵入者によるシステムへのアクセスをブロックする最も効果的な方法の 1 つは、端末（つまり、ログイン元）を保護することです。 ユーザがログインしたホストまたは端末（X 端末やコンソールなど）がログイン元になります。

最新アーキテクチャの端末は、UNIX が開発された当時のテレタイプ マシンとは異なります。 ほとんどのサイトでは、擬似端末サーバ（PTS）または X Window マネージャによって「擬似端末」が割り当てられます。セキュリティ システムに関して、端末の名前は意味のない記号になっています。 CA Access Control は、端末とみなされるものを保護します。 CA Access Control では、以下の 3 つの方法のいずれかで端末を定義した場合は、ログインの段階で端末の保護が実行されます。

- ユーザが X 端末から XDM ログイン ウィンドウを使用してログインする場合、CA Access Control では、ホスト名に（/etc/hosts、NIS、または DNS から）変換された X 端末の IP アドレスが、ログイン要求に使用された端末として認識されます。 ホスト名への変換に失敗した場合、またはホスト名ではなく IP アドレスを使用したい場合、CA Access Control では IP アドレスの使用を保護することもできます。
- ユーザがダム端末からログインする場合は、TTY 名によって端末が識別されます。
- ユーザがネットワークから（telnet、rlogin、rsh などを使用して）ログインする場合は、ホスト名に（/etc/hosts、NIS、または DNS によって）変換された要求元 IP アドレスが端末名として認識されます。

特定のホストに対するログイン ルールは、TERMINAL クラスで該当するホストを定義し、適切なユーザおよびグループをオブジェクトのアクセス リストに追加することによって定義できます。 各ログイン元の TERMINAL オブジェクトで日時制限を設定し、そのホストまたは端末からログインできる曜日や時間帯を制限することもできます。 また、TERMINAL クラスにワイルドカードを使用して、パターン(ホスト名または IP アドレス)と一致するホストを定義することもできます。

通常、スーパーユーザやシステム管理者などの上位の権限を持つユーザについては、セキュリティで保護された場所にある端末を使用するように制限する必要があります。 このように制限すると、スーパーユーザとしてシステムに侵入を試みる侵入者やハッカーが、自分のリモート端末からシステムに侵入することができません。システムにアクセスするには、セキュリティで保護された場所にある許可された端末の 1 台を使用しなければなりません。

ネットワーク経由のログインでは、ユーザが実際にホスト コンソールからアクセスしているという保証はありません。 ユーザは、ホストに接続された端末からアクセスしているか、要求元ホストからサービスを受信することを許可されているネットワーク内の他のノードからアクセスしている可能性があります。 別のホストからのログインをユーザに許可すると、そのユーザは特定の端末のみでなく、その端末によって許可されている他の端末からもログインできるようになります。 部門間の独立性を確実に維持するには、端末グループを定義し、各部門のユーザが、自分の所属する部門の端末グループからのみ操作を行えるように制限します。

端末の権限は、他のリソースと異なり、情報にアクセスする権限が多く与えられているユーザの端末ほど、権限を制限する必要があります。 セキュリティが確保されていないリモート端末からは誰も root ユーザとしてログインできないように、スーパーユーザの端末アクセス権を最も制限する必要があります。

CA Access Control では、端末を定義するときに、端末定義の所有者を明示的に指定する必要があります。 理由は、セキュリティ管理者として root ユーザがデフォルトで端末の所有者になった場合、その端末はスーパーユーザとしてのログインが可能な端末になるからです。 多くの場合、これは望ましい状態ではありません。 このような間違いによって知らないうちにセキュリティホールができないように、CA Access Control では、端末を定義するときに、端末の所有者を定義する必要があります。

端末 tty34 を定義するには、以下のコマンドを使用します。

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

このコマンドは、端末 tty34 のレコードを作成し、デフォルトのアクセス権を NONE に設定して、所有者として userA を定義しています。 端末の所有者である userA には、端末 tty34 からシステムにログインする権限が自動的に与えられます。

すべてのユーザに対して端末 tty34 からのログインを禁止するには、所有者として「nobody」を指定します。

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

特定の端末からログインする権限をユーザに与えるには、以下のコマンドを入力します。

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

このコマンドにより、USR1 は端末 tty34 からログインできる権限が与えられます。

端末を使用する権限をグループに与えることもできます。 たとえば、以下のコマンドでは、端末 tty34 を使用する権限をグループ DEPT1 のメンバに与えています。

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

端末のグループ（端末グループ）を定義するには、以下のコマンドを入力します。

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

メンバ端末を端末グループ TERM.DEPT1 に追加するには、以下のコマンドを入力します。

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

この端末グループを使用する権限を USR1 に与えるには、以下のコマンドを入力します。

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

このコマンドでは、tty34 および tty35 の両方を使用する権限を USR1 に与えています。

root ユーザの端末の制限

考慮する必要があるもう 1 つの問題として、TERMINAL クラスのデフォルト ルールがあります。 初期実装段階のデフォルト ルールでは、定義されていないアクセスはすべて許可されます。 TERMINAL クラスの場合は、これが問題になる可能性があります。

たとえば、サイトに数百台の端末があるとします。ほとんどのユーザは任意の端末からログインできるようにしますが、root ユーザは事前定義された 2 台の端末からのみログインできるように設定します。

最初に、TERMINAL クラスのデフォルトを READ に設定すると、root ユーザを含むすべてのユーザが、データベース内に特定の TERMINAL レコードがない任意の端末からログインできるようになります。 スーパーユーザが任意の端末からログインできることは望ましくありません。 また、TERMINAL クラスのデフォルトを NONE に設定すると、データベースで各端末を定義する必要があるため、作業負荷が非常に大きくなる場合があります。

CA Access Control では、この問題を解決するために、TERMINAL クラスの _default レコード内にアクセス制御リストを定義できるようになっています。 以下のコマンドは、最小限の操作で root ユーザを 2 つの端末に制限する方法を示しています。

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

最初の 2 つのコマンドでは、term1 と term2 を root が所有する端末として定義しています。そのため、これらの端末は、スーパーユーザでのログインが可能な端末になります。 newres TERMINAL _default コマンドと chres コマンドでは、デフォルト アクセス権として READ を設定しています。その結果、データベースで定義されていない端末には、すべてのユーザがアクセス可能です。 authorize コマンドでは、未定義の端末に対するスーパーユーザのアクセスを明示的に禁止しています。

注: UACC クラスは現在も存在し、リソースのデフォルト アクセス権を指定するときに使用できます。 ただし、リソースのデフォルト アクセス権を指定するには、_default レコードを使用する方が簡単です。

推奨する制限

TERMINAL クラスのデフォルト アクセス権が READ の場合、loopback 端末、ローカル ホスト端末、および端末ホスト名の使用を制限する必要があります。 これらの端末の使用をユーザに許可すると、他のすべてのユーザは、ターゲット ユーザのパスワードを知っている場合に自分のユーザ ID を置換することができます。 例として次の場合を考えてみましょう。

- ユーザ U には端末 T を使用する権限があります。
- 端末 T でスーパーユーザのログインは許可されていません。
- ユーザ U には、ユーザ ID を root に置換する権限がありません。

- ユーザ U がスーパーユーザのパスワードを入手しました。
- すべてのユーザに loopback 端末からログインする権限が与えられています。

ユーザ U は、ユーザ ID として root を指定し、root のパスワードを入力して、telnet loopback コマンドを実行するだけで、この一連のアクセス ルールを簡単に省略できます。 このようにして、ユーザは、スーパーユーザのログインが許可されていない端末 T からスーパーユーザ セッションを開始できます。 ユーザは、ローカル ホスト、または端末のホスト名を利用することによって、同様にアクセス ルールを無視できます。

このような 3 つの脆弱性を制限するには、以下の定義を使用します。

```
newres TERMINAL loopback defaccess(N) owner(nobody)
newres TERMINAL localhost defaccess(N) owner(nobody)
chres TERMINAL hostname defacc(N) owner(nobody)
```

このセキュリティ違反は、ローカル ホストからの telnet、FTP などの TCP 要求を制限することによっても防止できます。

また、TERMINAL グループのデフォルト アクセス権を NONE に設定して、TERMINAL ルールおよび GTERMINAL ルールを指定することによっても防止できます。

パスワード チェックとログインの制限

CA Access Control では、/bin/login 実行可能ファイルは置き換えられません。 CA Access Control の実行中も、/etc/passwd、shadow password ファイル、または NIS passwd マップとのパスワードの照合が続行されます。 さらに、CA Access Control では、以下のセクションで説明するチェックも実行されます。

ログイン チェック

ログイン プロセスが認証の段階を通過した後、CA Access Control はプロセスをインターセプトして、以下の点をチェックします。

- パスワードの有効期限は有効か。

  期限が切れている場合、ユーザは、アクセスが拒否される前に、猶予ログインの回数と警告を受け取ります。 アクセスが拒否された場合、セキュリティ管理者はユーザのパスワードを再度割り当てる必要があります。 猶予ログインの回数は、ユーザ パスワード ポリシーによって決定されます。ユーザ パスワード ポリシーは、setoptions コマンドを使用してグローバルに指定するか、chgrp コマンドを使用してプロファイル グループに対して指定できます。

  注: setoptions コマンドの詳細については、「リファレンス ガイド」を参照してください。

  segrace ユーティリティを使用すると、ユーザに許可されている猶予ログインの残りの回数、ユーザの現在のパスワードが期限切れになるまでの残り日数、ユーザが最後にログインした日時とログインした端末などを表示できます。

  注: segrace コマンドの詳細については「リファレンス ガイド」を参照してください。

- ユーザは許可された端末からログインしているか。

  許可された端末からログインしている場合は、問題なく次のチェックに進みます。許可されていない端末からログインした場合は、ユーザはログインできません。

- 現在の時間帯と曜日で(事前定義された制限によって)ログインが許可されているか。

  許可されている場合、ログインは問題なく次のチェックに進みます。許可されていない場合、ユーザはログインできません。

- 事前定義された日数以上、このユーザ名が未使用の状態になっていないか。

  事前定義された日数より長く未使用の状態であった場合、アクセスは拒否されます(デフォルトは 90 日ですが、変更する場合は setoptions コマンドを使用します)。

時間帯と曜日に関するログイン ルールの定義

データが最も危険にさらされるのは、アクティビティが少ないときです。 深夜および週末は、監査レコードを監視できる人が少なくなるため、侵入者はシステムに侵入しやすくなります。 適切な端末許可ルールを設定すると、侵入者は保護されている場所にある端末を使用せざるを得なくなります。 また、曜日(DOW)と時間帯(TOD)のアクセス ルールを設定すると、侵入者はオフィスが開いている勤務時間中にしか侵入を試みることができなくなります。 これらのルールを組み合わせて、外部からの侵入を厳しく制限します。

ユーザがログインできる曜日と時間は、ユーザごとに制限できます。 ユーザの DOW ログイン制限および TOD ログイン制限を定義するには、以下のコマンドを使用します。

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

このコマンドでは、月曜日、火曜日、および水曜日の 8 時から 17 時の間にのみ、ユーザ USR1 のログインを許可することを指定しています。 USR1 は、指定された曜日の指定時間外、または指定された曜日以外にログインすることはできません。

days パラメータには、ANYDAY(週 7 日すべてログイン可能)および WEEKDAYS(月曜日から金曜日までログイン可能)の値を指定することもできます。 time パラメータには、ANYTIME(どの時間でもログイン可能)の値を指定することもできます。

注: DOW および TOD の制限は、データベースに定義されている多くのリソースに適用できます。 この機能は、端末および端末グループに対して使用可能な時間帯の制限を指定する場合に特に効果的です。

同時ログインの無効化

ほとんどの UNIX ベースのオペレーティング システムでは、同時ログインが可能です。しかし、複数の端末からログインすることをユーザに許可すると、ユーザ本人がログインしているときに、他のユーザが別の場所からそのユーザを装ってログインする危険性があります。

CA Access Control では、ログインした後に、ログインした本人が自分の同時ログイン権限を無効にすることができます。これによって、他のユーザが別の端末から本人を装ってログインすることはできなくなります。 ただし、使用している特定の端末からは、繰り返しログインできます。 secons コマンドで以下のスイッチを使用します。

```
# secons -d-          (同時ログインを無効にする)
# secons -d+          (同時ログインを有効にする)
```

-d オプションは任意のユーザが発行できます。（他のすべてのオプションは、ADMIN 属性または OPERATOR 属性を持つユーザにのみ許可されます）。 同時ログインを無効にするには、初期スクリプトでこのコマンドを使用します。 このコマンドを指定すると、必要な数だけウィンドウを開くことができますが、別の端末からログインすることはできません。

注: secons -d- コマンドを使用して同時ログインを防止する場合は、システムからロックアウトされないように、ログアウト前に secons -d+ を使用する必要があります。 同時ログインを元どおりにすることを忘れて再度ログインを試みた場合、同じユーザ ID によるプロセスがほかに実行されていなければログインが許可されます。

ユーザ単位の同時ログインの制限

CA Access Control では、以下の 2 つの方法で同時ログインの数を制御できます。

管理者レベル

データベースに、システム全体で 1 ユーザが保持できる同時セッション数の定義を設定します。 この値は、プロファイル グループまたは個々のユーザに対してグローバルに設定できます。

ユーザ レベル

個々のユーザが、自分に許可する同時ログイン数を制御します。 この方法では、ユーザはログインする時点で、自分の名前で他のログイン セッションが実行される可能性をブロックし、自分自身を保護することができます。

注: 同時ログインの数は、ユーザが特定の端末で実行しているセッションの数とは無関係です。 1 台の端末で複数のセッションが実行されていても、単一のログインとみなされます。 同時ログイン制限は、各ターミナルからのログイン数ではなく、1 つのユーザアカウントが同時にログインできる数を制限します。

グローバルな同時ログインの制限

selang で、以下のコマンドを入力します。

```
setoptions maxlogins(NumLogins)
```

個別の同時ログインの制限

selang で、以下のコマンドを入力します。

chusr username maxlogins(NumLogins)

ユーザに対して設定した同時ログイン制限は、システム全体の制限よりも優先されます。特定のユーザに対して同時ログインの制限を適用しないようにするには、そのユーザの同時ログイン制限を 0(ゼロ)に設定します (最大同時ログイン数を 1 に設定すると、selang を使用できなくなるので注意してください)。

ログイン イベントの認識

CA Access Control では、プロセスのユーザ ID を変更しようとするすべての試みがログイン イベントとして認識されるわけではありません。 通常、プログラムでは、setuid システム コールを使用してユーザ ID を変更します。 SURROGATE クラスがこれらのイベントを制御します。これらのイベントは、必ずしもログイン イベントとはみなされず、CA Access Control から見ると、必ずしもユーザ ID を変更するイベントとは認識されません。

CA Access Control では、ユーザが最初のログイン時に使用した、元のユーザ ID が常に保持されます。 通常の setuid システム コールを実行しても、CA Access Control ではユーザ ID の変更は登録されません。

CA Access Control が ID の変更を認識するには、このイベントをログイン イベントとして認識する必要があります。 CA Access Control は、以下のルールを使用してログイン イベントを認識します。

- ID の変更を試みるプログラムは、ログイン プログラムとして定義されます。 LOGINAPPL クラスのすべてのプログラムがログイン プログラムです。
- このプログラムは、LOGINAPPL クラスの定義に対応する一連のシステム コールを実行します。

管理セッションを(selang または CA Access Control エンドポイント管理で)開始すると、CA Access Control によってダミー ログイン イベントが実行されます。 これは実際のログインではなく、CA Access Control のログイン チェックと同様の一定の内部チェックが実行されます。

注: 詳細については、「selang リファレンス ガイド」で LOGINAPPL クラスの SEQUENCE プロパティの説明を参照してください。

管理セッションの開始時には、管理対象のマシンでユーザ名がチェックされます。 管理対象マシンにアクセスするには、セッションを実行する端末の WRITE アクセス権が必要です。

たとえば、ホスト Minerva にログインした状態でホスト Artemis の CA Access Control を管理するには、以下の 2 つの条件が満たされている必要があります。

- Minerva という名前(または適切な完全修飾名)の TERMINAL オブジェクトが Artemis のデータベース レコード内にあること
- 管理者がこのオブジェクトの ACL に WRITE アクセス権付きで登録されていること

これらの条件は、他のユーザ権限チェックよりも前にチェックされます。 データベースでは、管理者権限も必要です。

第 10 章: TCP/IP サービスの保護

機密データを格納しているファイル サーバで最も重要なことは、TCP/IP サービスを保護することです。 このようなファイル サーバは信頼されている端末のみにサービスを提供する必要があります。ホストで認証できないコンピュータや侵入者にサービスを提供してはなりません。

このセクションには、以下のトピックが含まれています。

TCP/IP サービスの制限

オープンなネットワーク環境では、すべての端末でネットワーク上の他のコンピュータにサービスを要求できます。 TCP/IP プロトコルを使用すると、多数のサービスを提供できます。 その中には、rlogin、rcp、rsh、ftp、telnet、rexec など、UNIX ベースのすべてのオペレーティング システムに共通するサービスもあります。 その他のサービスは、社内およびサードパーティ ソフトウェアによって提供されます。

CA Access Control は、ホスト コンピュータで TCP/IP の受信処理をインターセプトし、受信プログラムを通常どおり続行するか、または変更するかを判断します。 この判断は、ホストおよびサービスを制御する定義したアクセス ルールに基づいて行われます。 データベースで TCP/IP アクセス ルールを作成して、特定のコンピュータからファイル転送、リモート ログイン、リモート シェルなどのサービスを受信するコンピュータとネットワークを指定できます。

以下の例では、TCP/IP アクセス ルールを定義し、不正な部外者を効果的にブロックするための設定方法を示します。 データベースの作成が完了していない場合は、データベースに定義されていない端末が任意のサービスを受信するように設定できます。 その場合は、UACC クラスの HOST レコードを以下のように設定します。

```
chres UACC HOST defaccess(READ)
```

ローカル ホストから送信される TCP/IP サービスに関するアクセス ルールを設定する端末を、データベースの HOST クラスのレコードに定義します。 各端末に許可するサービスをこのレコードに指定します。 たとえば、以下の一連のコマンドでは、端末 ws5 のレコードを定義し、その端末がローカル ホストから TCP/IP サービスを受信できないようにします。

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

以下のコマンドでは、ws5 がローカル コンピュータに対して telnet を実行することを許可します。

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

これらの設定によって、ユーザは、telnet を使用してローカル コンピュータに接続できます。この場合、リモート ユーザは、ローカル システムを使用する前に、ユーザ名とパスワードを指定する必要があります。 service のキーワードにアスタリスクを使用すると、端末はローカル コンピュータからすべての TCP/IP サービスを受信することができます。たとえば、以下のコマンドを使用すると、端末 ws5 は、ローカル コンピュータから任意の TCP/IP サービスを呼び出すことができます。

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

サービスは、ポート番号による指定など複数の方法で指定できます。 ポート番号はサービスの識別番号です。 すべてのサービスにはポート番号があります。このポート番号は、/etc/services ファイルでサービスに割り当てられています。 サービスは以下の方法で指定できます。

- /etc/services ファイルに定義されている名前で指定します。
- ポート番号で指定します。
- ポート番号の範囲として指定します。
- /etc/rpc システム ファイルに設定されている RPC ポートとして指定します。

たとえば、以下のコマンドでは、ws5 に対して、ポート番号が 7045 から 7050 までの TCP/IP サービスの受信を許可しています。

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

多くの場合、ホストのグループを定義しこの許可を 1 回で設定する方が、個々のコンピュータに対して許可を与えるよりはるかに効率的です。 CA Access Control には GHOST クラスがあり、各 GHOST レコードでホストのグループが定義されます。GHOST レコードを定義してホストをそのメンバ リストに追加するには、以下のコマンドを入力します。

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

newres コマンドによって、メンバ ws2、ws3、および ws5 を含む gh1 というホスト グループが定義されます。 authorize コマンドによって、この 3 台の端末すべてに FTP (ファイル転送) サービスの受信が許可されます。

ホスト グループの管理は個々に端末を管理するより簡単ですが、柔軟性を持たせるために、CA Access Control では、ネットワーク アクセス ルールの定義もサポートされています。 ネットワークは HOSTNET クラスで定義します。 例として、以下のような一連のコマンドを考えてみましょう。

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

- 最初の行の newres コマンドでは、hn1 というネットワークを定義しています。 newres コマンドは、mask と match の値によって、IP アドレスの最初の 2 つの修飾子が 192.168 であるコンピュータが hn1 ネットワークに属していることを指定します。
- 2 行目と 3 行目の組み合わせでは、hn1 ネットワークに属するすべての端末に対して FTP を実行する許可が与えられます。ただし、ホスト コンピュータのその他のサービスは許可されません。

CA Access Control で TCP/IP アクセス ルールを定義するためのもう 1 つの方法は、名前パターン アクセス ルールです。 CA Access Control では、ワイルドカードを使用して HOSTNP クラス(ホスト名パターン)の包括的なレコードを定義できます。

注: CA Access Control で行われる文字列マッチングの詳細については、「selang リファレンス ガイド」を参照してください。

たとえば、以下の一連のコマンドでは、文字列「lin」で始まり、「.org.com」で終わる名前を持つすべてのホストに、ローカル ホスト上のすべての TCP/IP サービスの受信を許可しています。

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

注: NIS によって管理されるホストは、別名ではなく、NIS マップに示されている公式の名前で識別する必要があります。 以下のセクションのフローチャートに TCP/IP チェックの流れをまとめて示します。

TCP クラスの使用

もう 1 つの方法として、TCP クラスを使用すると、ホスト単位ではなくサービス単位で保護機能を指定できます。

注: TCP クラスの詳細については、「リファレンス ガイド」を参照してください。

受信サービスおよび送信サービスを制御するには、TCP クラスを使用します。

たとえば、以下のコマンドでは、ftp サービスのレコードを作成し、デフォルトのアクセスタイプを READ（サービスを使用できるという意味）に設定します。ただし、PUBLIC* という名前パターンと一致するホストは、そのサービスを受信できないようにします。

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

また、特定のユーザまたはグループのみに特定サービスの受信を許可するように指定することもできます。 たとえば、hermes というホストに対しては、すべてのユーザが FTP サービスを実行できるが、hermes に telnet でアクセスできるのは acctng というグループのメンバに限定する場合は、以下のコマンドを入力します。

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

注： defaccess(read) は送信サービスを無効にし、 defaccess(write) は受信サービスを無効にします。

HOST クラスがアクティブである（つまり、アクセスの基準として使用されている）場合、TCP クラスは実質上アクティブにできません。 setoptions class- HOST コマンドを使用すると、HOST クラスを無効にできます。その後、必要に応じて setoptions class+ TCP コマンドを使用して TCP クラスを有効にします。 HOST クラスを無効にすると、GHOST、HOSTNET、および HOSTNP も自動的に無効になります。

また、TCP クラスがアクティブである場合は、setoptions class- (CONNECT) コマンドを使用して、CONNECT クラスを無効にします。

ネットワーク インターセプト用のストリーム モジュール

デフォルトでは、TCP クラスはアクティブではありません。 TCP クラス、CONNECT クラス、または HOST クラスをアクティブにする前に、ストリーム モジュールが有効になっていることを確認します。

Solaris 上で CA Access Control ストリーム モジュールを読み込むには、以下の手順に従います。

1. CA Access Control を停止します。 以下のコマンドを入力します。

   ```
   secons -s
   ```

2. 以下のコマンドを入力します。

   ```
   SEOS_load -s
   ```

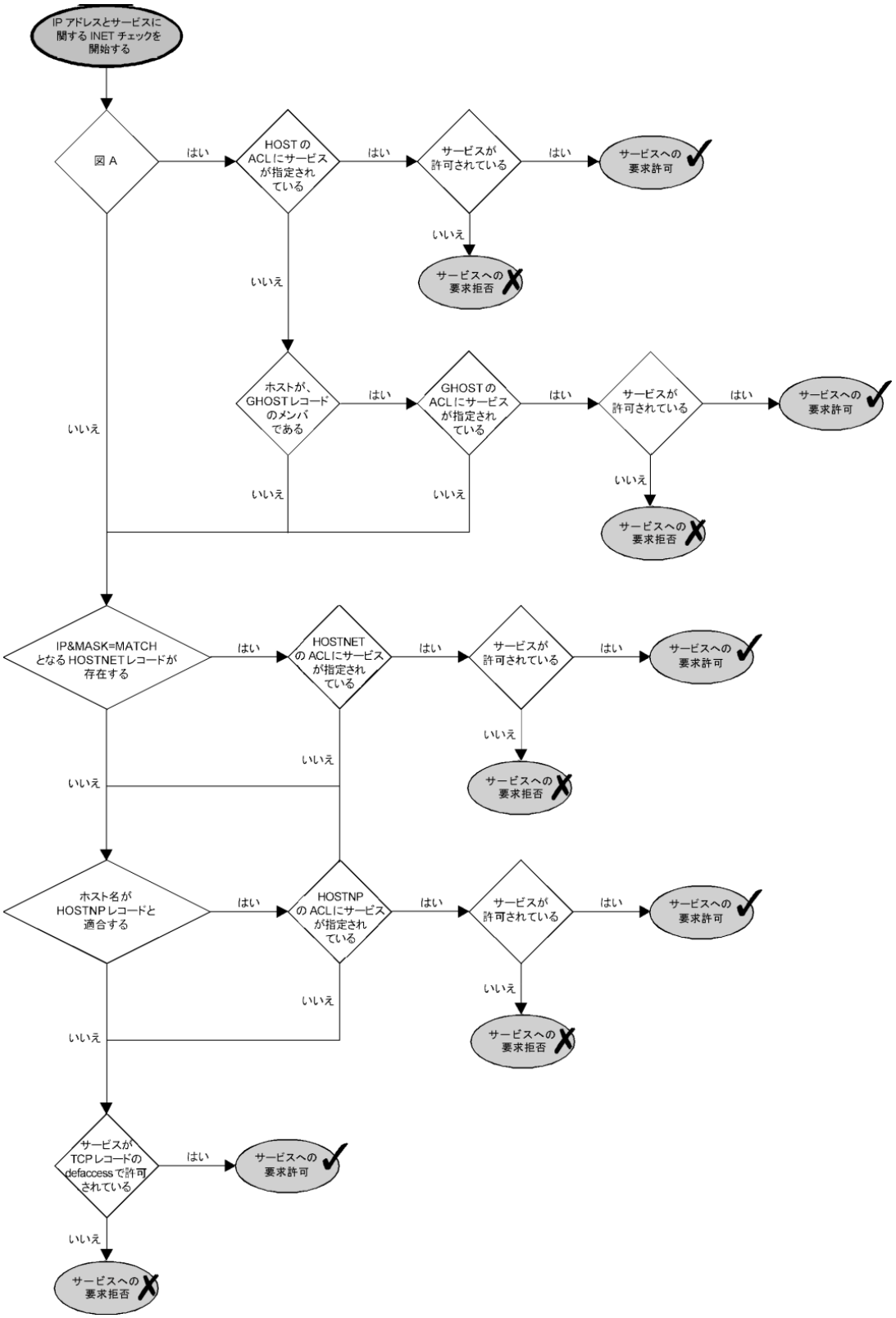
3. CA Access Control を起動します。 以下のコマンドを入力します。

```
seload
```

注: ストリーム モジュールがロードされていないときに TCP クラスをアクティブにしようとすると、以下のエラー メッセージが表示されます。

エラー: ストリームがロードされない場合に className クラスをアクティブにすることはできません。
SEOS_load -s を使用してストリームをロードしてください。

受信権限のアルゴリズムは、以下のとおりです。


IP アドレスとサービスに関する INET チェックを開始する
図 A
はい
HOST の ACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否
いいえ
ホストが、GHOST レコードのメンバである
はい
GHOST の ACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
いいえ
いいえ
サービスへの要求拒否
いいえ
IP&MASK=MATCH となる HOSTNET レコードが存在する
はい
HOSTNET の ACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
いいえ
サービスへの要求拒否
いいえ
ホスト名が HOSTNP レコードと適合する
はい
HOSTNP の ACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
いいえ
サービスへの要求拒否
いいえ
サービスが TCP レコードの defaccess で許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否

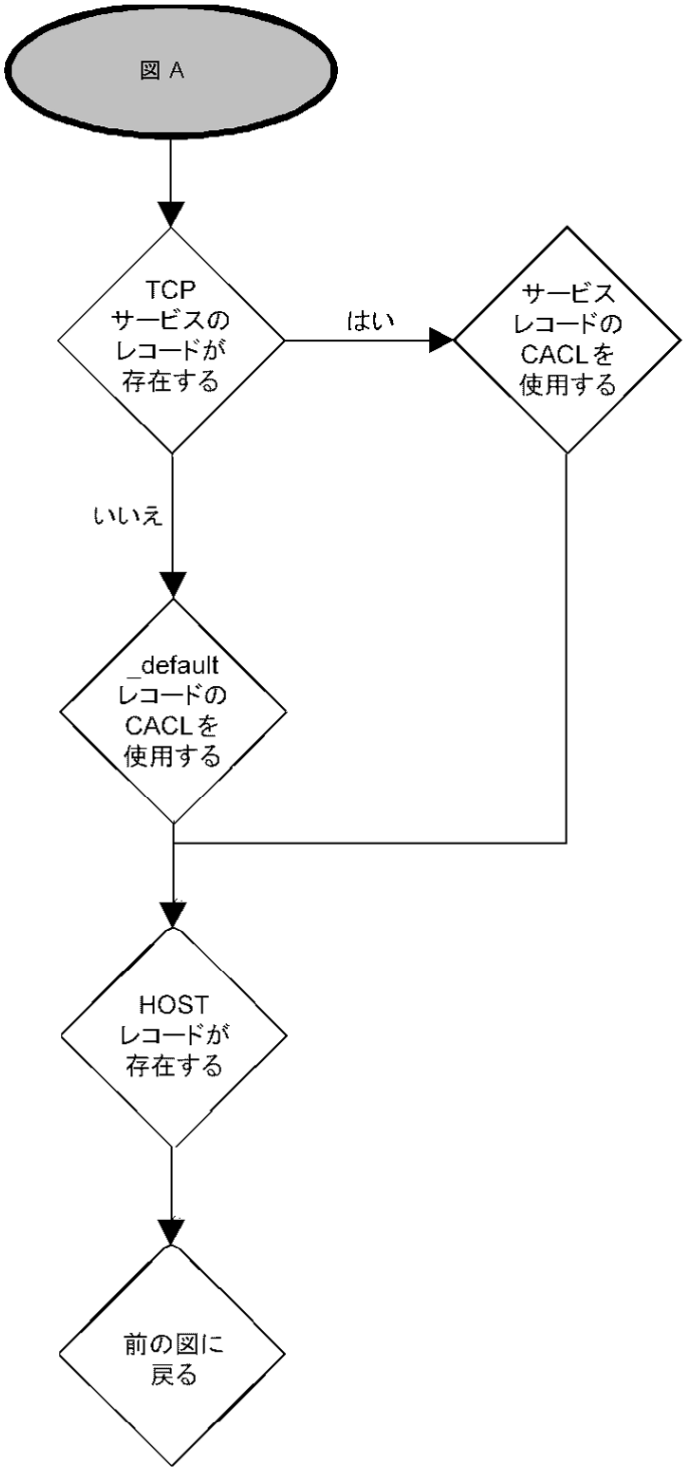
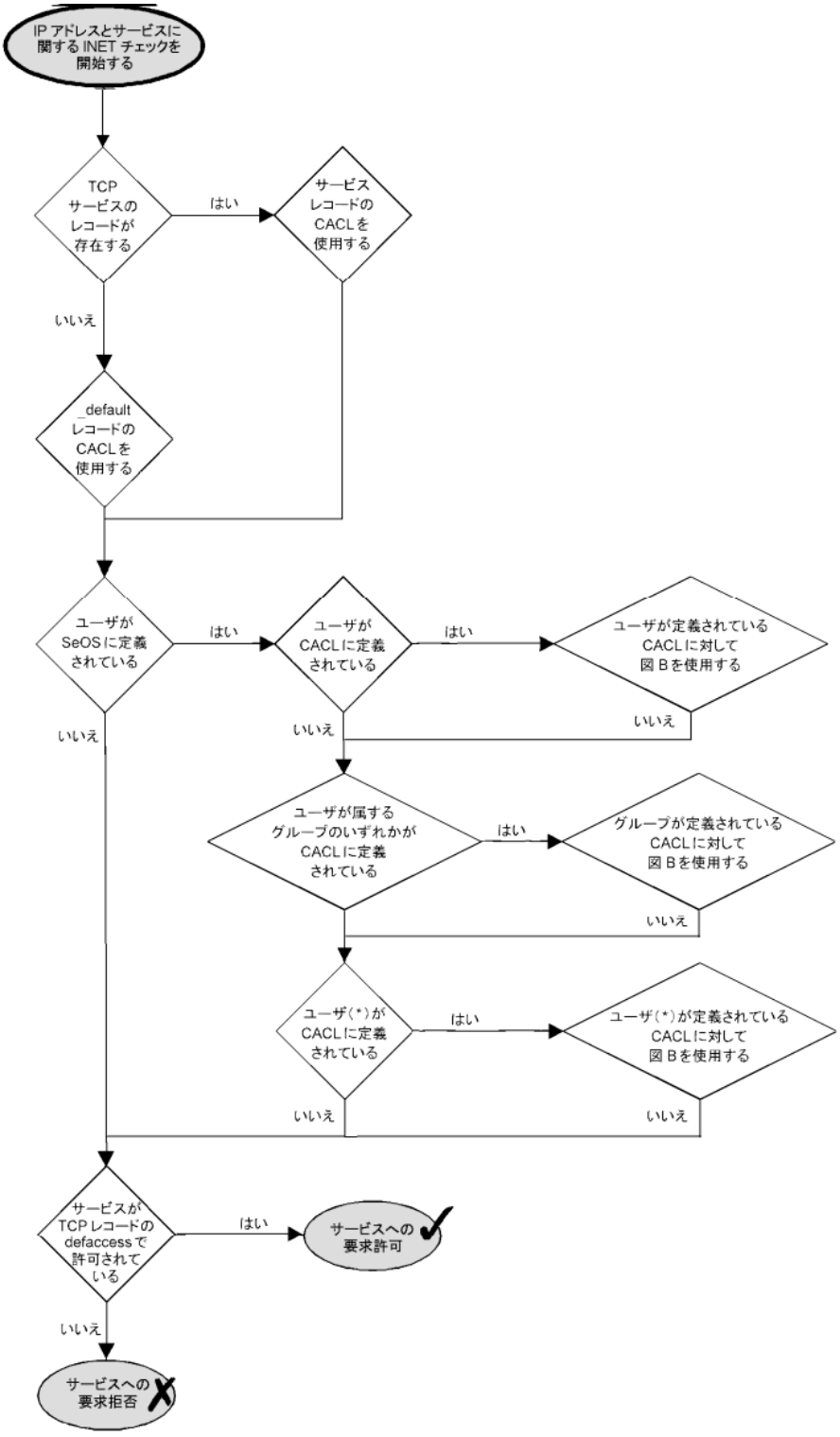


図 A
TCP
サービスの
レコードが
存在する
はい
サービス
レコードの
CACL を
使用する
いいえ
_default
レコードの
CACL を
使用する
HOST
レコードが
存在する
前の図に
戻る

送信権限のアルゴリズムは、以下のとおりです。



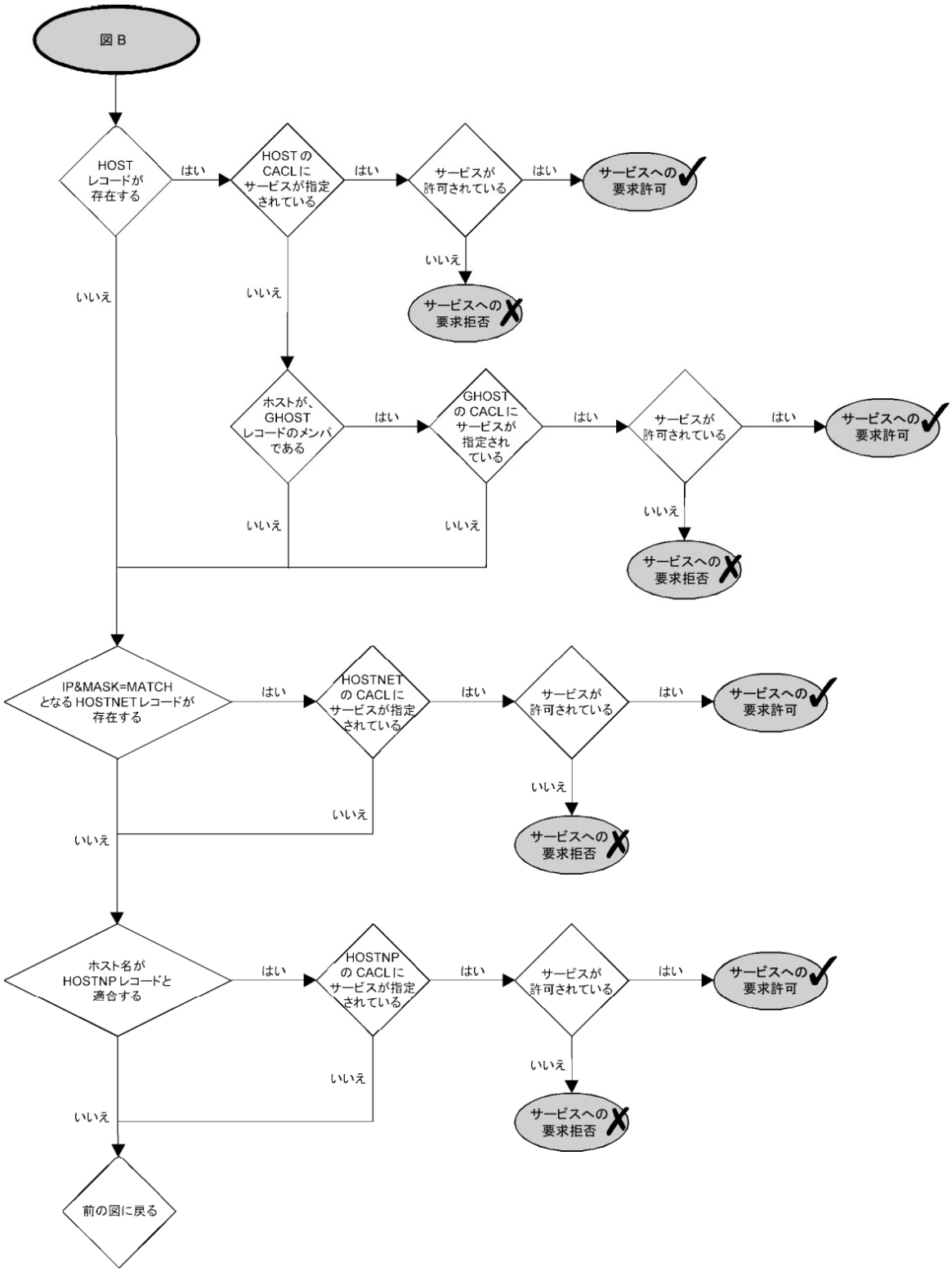


図 B
HOST レコードが存在する
はい
HOST の CACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否
いいえ
いいえ
ホストが、GHOST レコードのメンバである
はい
GHOST の CACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否
いいえ
いいえ
IP&MASK=MATCH となる HOSTNET レコードが存在する
はい
HOSTNET の CACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否
いいえ
いいえ
ホスト名が HOSTNP レコードと適合する
はい
HOSTNP の CACL にサービスが指定されている
はい
サービスが許可されている
はい
サービスへの要求許可
いいえ
サービスへの要求拒否
いいえ
いいえ
前の図に戻る

第 11 章: Policy Model の管理

このセクションには、以下のトピックが含まれています。

Policy Model データベース

何百、何千ものデータベースを個別に管理することは、現実的ではありません。 CA Access Control には、1 台の中央データベースから多数のデータベースを管理できるコンポーネントである Policy Model サービスが用意されています。 Policy Model サービスの使用は任意ですが、このサービスを使用すると、大規模なサイトでの管理を大幅に簡略化できます。

Policy Model (PMD) サービスは、Policy Model データベース (PMDB) を使用します。 PMDB には、他の CA Access Control データベースと同様に、ユーザ、グループ、保護されているリソース、およびリソースへのアクセスを管理するルールが保存されています。 PMDB にはこのほかに、サブスクライバ データベースのリストが含まれます。 各サブスクライバは、別々のコンピュータに存在する CA Access Control データベース、または同じコンピュータまたは別のコンピュータに存在する別の PMDB です。 サブスクライバを更新する PMDB をサブスクライバの親といいます。

PMDB は、同様の許可制約およびアクセス ルールが適用される多数のデータベースを管理するための便利なツールです。

注: PMDB の管理方法 (sepmd ユーティリティ) の詳細については、「リファレンス ガイド」を参照してください。 selang の使用によるリモートでの PMDB の管理方法の詳細については、「selang リファレンス ガイド」を参照してください。

ディスク上の PMDB の場所

すべての PMDB は、(コンピュータごとに 1 つある) 共通ディレクトリに格納されます。ディレクトリ名は、seos.ini ファイルの[pmd]セクションにある _pmd_directory_ トークンで指定します。 _pmd_directory_ のデフォルト値は ACInstallDir/policies です (ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です)。

各 PMDB は、共通ディレクトリ内のサブディレクトリに格納されます。 Policy Model の名前がそのままサブディレクトリの名前になります。 サブディレクトリ内のファイルには、pmd.ini ファイルなど、Policy Model を定義するために必要なすべてのデータが含まれています。

ローカル PMDB の管理

CA Access Control には、ローカル PMDB を管理するためのユーティリティがいくつか用意されています。

sepmd

以下を実行できる PMDB 管理ユーティリティ。

- サブスクライバの管理
- 更新ファイルの切り捨て
- デュアル コントロールの管理
- Policy Model のログ ファイルの管理
- その他の管理タスクの実行

sepmdadm

PMDB を作成し、階層をセットアップするために必要な設定で構成します。

注: Policy Model ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

リモート PMDB の管理

CA Access Control には、pmd 環境で使用できるさまざまな selang コマンドも用意されています。 これらのコマンドを使用して、PMDB をリモートで管理できます。

backuppmd

PMDB をバックアップします。

createpmd

PMDB を作成します。

deletepmd

PMDB を削除します。

findpmd

コンピュータ上のすべての PMDB の名前を表示します。

listpmd

PMDB に関する以下の情報を表示します。

- サブスクライバおよびそのステータス
- PMDB の説明およびそのステータス
- 更新ファイル内のコマンドおよび各コマンドのオフセット
- エラー ログの内容

pmd

以下の操作を実行できる PMDB 管理コマンドです。

- 使用不可のサブスクライバのリストからのサブスクライバの削除
- Policy Model のエラー ログの消去
- Policy Model のロックおよびロック解除
- Policy Model デーモンの開始および停止
- 更新ファイルの切り捨て
- 初期化ファイルの再ロード

restorepmd

バックアップ ファイルから PMDB をリストアします。

subs

以下の操作を実行できる PMDB サブスクリプション コマンドです。

- 親 PMDB への既存サブスクライバの追加
- 親 PMDB への新規サブスクライバの追加
- データベース(CA Access Control または別の PMDB)への親 PMDB の割り当て

subspmd

ローカル データベースに親 PMDB を割り当てます。

unsubs

PMDB からサブスクライバを削除します。

注: pmd 環境で使用できる selang コマンドの詳細については、「selang リファレンス ガイド」を参照してください。

アーキテクチャの依存関係

CA Access Control をデプロイするときは、環境の階層を考慮する必要があります。 多くのサイトで、ネットワークにはさまざまなアーキテクチャが採用されています。 trusted プログラムのリストなど、一部のポリシー ルールはアーキテクチャに依存します。 一方、ほとんどのルールは、システムのアーキテクチャに関係なく適用されます。

階層を使用すると、両方の種類のルールを適用できます。 アーキテクチャに依存しないルールをグローバル データベースで定義し、そのグローバル データベースのサブスクライバ PMDB で、アーキテクチャに依存するルールを定義できます。

注: ルート PMDB とそのすべてのサブスクライバは、環境の物理的ニーズに応じて、同じコンピュータ上に存在することも、別々のコンピュータ上に存在することも可能です。

例: 2 層のデプロイ階層

以下の UNIX の例は、少し変更して Windows アーキテクチャにも適用できます。

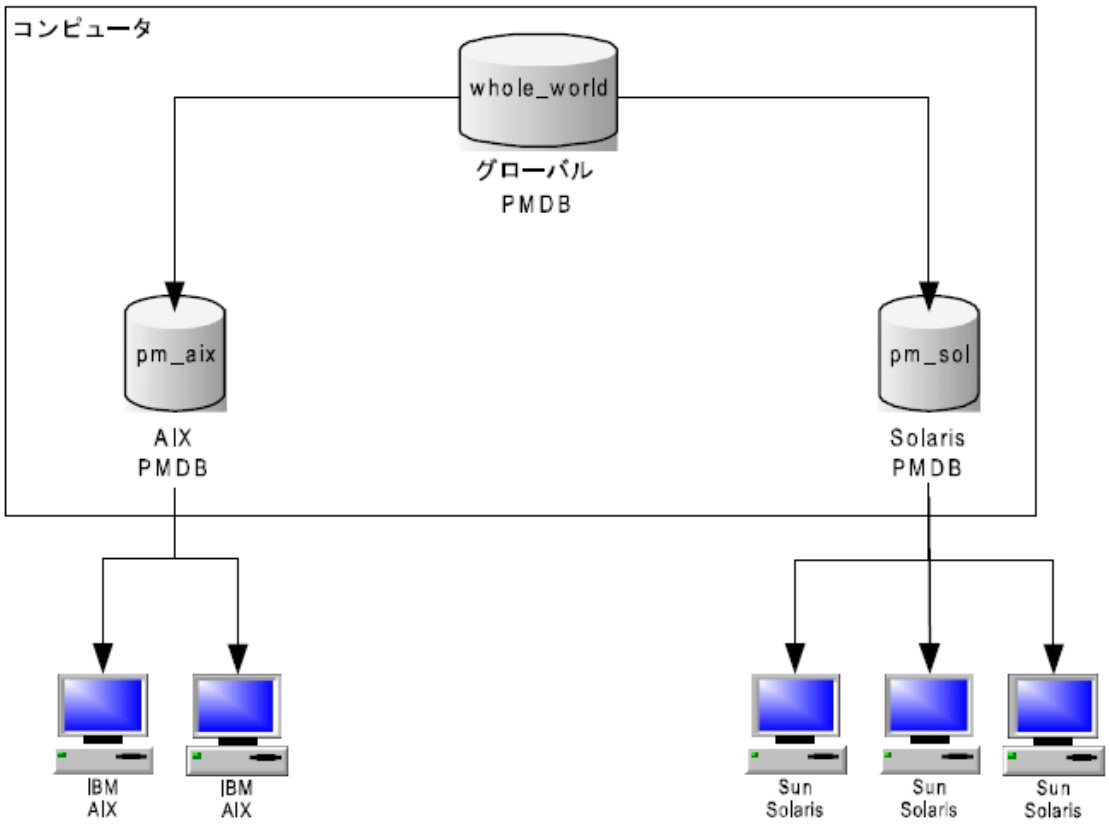
この例では、サイトは IBM AIX システムと Sun Solaris システムで構成されています。 IBM AIX の trusted プログラムのリストは Sun Solaris でのリストとは異なるため、アーキテクチャの依存関係を考慮した PMDB が必要です。

複数アーキテクチャに対応した PMDB をセットアップするには、PMDB を以下のようにセットアップします。

1. whole_world という PMDB を定義し、ユーザ、グループ、およびアーキテクチャに依存しないその他のすべてのポリシーを格納します。
2. pm_aix という PMDB を定義し、IBM AIX 固有のすべてのルールを格納します。

3. pm_sol という PMDB を定義し、Sun Solaris 固有のすべてのルールを格納します。

   pm_aix および pm_solaris という PMDB は、whole_world という PMDB のサブスクライバです。 サイト内のすべての IBM AIX コンピュータは pm_aix のサブスクライバです。 サイト内のすべての Sun Solaris コンピュータは pm_sol のサブスクライバです。 この概念を以下の図に示します。



4. ユーザの追加や SURROGATE ルールの設定など、プラットフォームに依存しないコマンドを whole_world に入力すると、サイト内のすべてのデータベースが自動的に更新されます。

5. trusted プログラムを pm_aix に追加すると、IBM AIX コンピュータのみが更新されます。Sun Solaris システムには影響はありません。

ポリシーの一元管理の方法

CA Access Control を使用すると、以下の 方法で 1 台のコンピュータから複数のデータベースを管理できます。

- 自動的なルール ベース ポリシー更新 – 中央のデータベース(PMDB)に定義した通常のルールは、設定された階層内のデータベースに自動的に伝達されます。

  注: デュアル コントロール(153 ページ)は、この方法でのみ使用できます。また、UNIX でのみ使用可能です。 自動的なルール ベース ポリシー更新のデュアル コントロールの詳細は、本書で説明しています。 また、自動的なルール ベース ポリシー更新の詳細は、「Windows エンドポイント管理ガイド」でも説明しています。

- 拡張ポリシー管理 – デプロイしたポリシー(ルールの集まり)は、ホストまたはホストグループの割り当てに基づいてすべてのデータベースに伝達されます。 また、ポリシーのデプロイ解除(削除)、デプロイのステータスやデプロイの偏差の表示を行うこともできます。 この機能を使用するには、追加のコンポーネントをインストールおよび設定する必要があります。

  注: 拡張ポリシー管理の詳細については、「エンタープライズ管理ガイド 」を参照してください。

自動的なルール ベース ポリシー更新

中央データベースで単一ルール ポリシー更新(標準の selang ルール)を行うと、サブスクライバ データベースに自動的に伝達されます。 複数のコンピュータを同じデータベースのサブスクライバとし、そのデータベースを別のデータベースのサブスクライバとすることによって、階層を作成できます。 インストール後に、自動的なルール ベース ポリシー更新を環境に設定します。

注: このポリシー管理方法は、単一ルール ポリシー更新を階層全体に伝達することだけに制限されます。 その他の機能を使用するには、拡張ポリシー管理およびレポートを実装する必要があります。

自動的なルール ベース ポリシー更新のしくみ

環境に自動的なルール ベース ポリシー更新を設定すると、中央データベースで定義した各ルールは、以下の方法ですべてのサブスクライバに自動的に伝達されます。

1. 少なくとも 1 つのサブスクライバを持つ任意の PMDB にルールを定義します。
2. PMDB がすべてのサブスクライバ データベースにコマンドを送信します。

3. 伝達されたコマンドをサブスクライバ データベースが適用します。

   a. サブスクライバ データベースから応答がない場合、PMDB はサブスクライバ データベースが更新されるまで、定期的に(デフォルトでは 30 分間隔)コマンドを送信し続けます。

      または、サブスクライバ データベースが使用可能になった時点でただちに更新することもできます。そのためには、サブスクライバ コンピュータで seos.ini ファイルの[pmd]セクションにある pull_option トークンを yes に設定します。

   b. サブスクライバ データベースから応答があっても、コマンドの適用が拒否された場合、PMDB はこのコマンドを Policy Model のエラー ログ(148 ページ)に記録します。

4. サブスクライバ データベースが別のサブスクライバの親である場合は、サブスクライバ データベースはそのサブスクライバにコマンドを送信します。

例: 階層内のすべてのコンピュータからユーザを削除する

rmusr コマンドによってユーザが PMDB から削除されると、同じ rmusr コマンドがすべてのサブスクライバ データベースに送信されます。 このように、rmusr コマンドを 1 回実行すれば、さまざまな種類のコンピュータ上にある多数のデータベースからユーザを削除できます。

PMDB を使用した設定の伝達方法

Policy Model の設定を編集すると、新しい設定値が Policy Model のサブスクライバに伝達されます。

以下プロセスでは、設定の更新を Policy Model のサブスクライバに伝達する方法について説明します。

1. Policy Model の 1 つ以上の設定値を編集します。
2. Policy Model は、新しい設定値を仮想環境設定ファイルに書き込みます。

   注: 仮想環境設定ファイルには、audit.cfg ファイルの値は含まれません。 Policy Model は、このファイルに加えた変更を仮想環境設定ファイルに書き込みません。
3. Policy Model は、そのサブスクライバに新しい設定値を伝達します。
4. selang コマンドは、新しい設定値で各サブスクライバを更新します。

仮想環境設定ファイル

各 Policy Model には、そのサブスクライバ用の設定値が含まれている仮想環境設定ファイルが存在します。 仮想環境設定ファイルは、cfg_configname という名前で PMD ディレクトリに置かれます(configname は Policy Model 設定の名前)。

仮想環境設定ファイルには、audit.cfg ファイルに保持されている設定値は含まれていません。

新しいサブスクライバの設定方法

Policy Model は、既存の設定値で個々の新しいサブスクライバを設定します。 既存の設定値は、仮想環境設定ファイルに格納されます。

注: 仮想環境設定ファイルには、audit.cfg ファイルの設定値は格納されません。 新しいサブスクライバを作成する前に audit.cfg ファイルに加えた変更は、新しいサブスクライバに伝達されません。

以下のプロセスでは、Policy Model が新しいサブスクライバを設定する方法について説明します。

1. Policy Model の新しいサブスクライバを作成します。
2. Policy Model は、その仮想環境設定ファイルの値を読み取ります。
3. Policy Model は、その仮想環境設定ファイルの設定値を updates.dat ファイルに追加します。 updates.dat ファイルには、ポリシーに対するアクセス ルールも含まれています。
4. Policy Model は、新しいサブスクライバに updates.dat ファイルを送ります。
5. selang コマンドは、updates.dat ファイルの値を使用して新しいサブスクライバを設定します。

階層のセットアップ方法

CA Access Control は、Policy Model サービスを使用して、設定された階層全体にルール ベース ポリシー更新を伝達します。 複数の CA Access Control コンピュータを同じ PMDB にサブスクライブし、ある PMDB を別の PMDB にサブスクライブすることによって、階層を作成します。

自動的なルール ベース ポリシー更新を有効にするには、以下の手順に従います。

1. マスタ PMDB を作成し、設定します(135 ページ)。
2. (オプション)サブスクライバ PMDB を作成し、設定します(137 ページ)。
3. サブスクライバ コンピュータに親 PMDB を定義します(139 ページ)。このサブスクライバ コンピュータはエンドポイントと呼ばれます。

注: 以下のセクションでは、PMDB 階層のセットアップ方法について説明します。 ほかにも、PMDB を作成して階層を設定する方法がいくつかあります。 Policy Model ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

マスタ PMDB の作成と設定

ポリシーを中央から一元管理するには、まずマスタ PMDB を作成して設定する必要があります。 ローカル ホスト上でこれを行うには、sepmdadm コマンドを使用します。

注: 以下の手順では、sepmdadm コマンドを対話形式で入力する方法を説明します。すべての入力に対するコマンド ライン パラメータの使い方については、「リファレンス ガイド」を参照してください。

マスタ PMDB を作成および設定するには、以下の手順に従います。

1. コマンド ラインで以下のコマンドを入力します。

   sepmdadm -i

   CA Access Control によって Policy Model データベース管理スクリプト(sepmdadm)が起動され、メニューが表示されます。ここでオプションを選択します。
2. 「1」を入力して、最初のオプション(マスタ PMDB を作成してサブスクライバを定義する)を選択します。

   スクリプトは、関連する質問をするように設定されています。
3. Enter キーを押して続行します。

   続いて、1 つ目の質問が表示されます。

   注: CA Access Control が実行中でない場合、CA Access Control を起動してからスクリプトを再実行するように警告が発行されます。
4. 作成する Policy Model の名前を入力します。

   Policy Model 名が登録され、次に進みます。
5. 指定する最初のサブスクライバ コンピュータの名前を入力します。

   最初のサブスクライバの名前が登録され、次のサブスクライバの名前を入力するように求められます。

6. 必要に応じてサブスクライバ名を入力したら、Enter キーを押します。

   すべてのサブスクライバ名が登録され、次に進みます。

   注: 各サブスクライバ コンピュータが親 PMDB を参照している必要があります。

7. NIS、NIS+、または DNS を実行している場合は、PMDB の変更で NIS/DNS テーブルを更新するかどうかを選択します。

   更新は PMDB 内のユーザおよびグループに対して行われます。 テーブルには、ユーザとユーザの特性に関する情報が保存されています。 yes を選択すると、Policy Model で更新された UNIX ユーザまたは UNIX グループは、NIS の passwd ファイルと group ファイルでも更新されます。

   a. NIS/DNS テーブルを更新する場合は、「y」と入力します。

      ここで、NIS の passwd ファイルと group ファイルの場所を尋ねられます。

      a. NIS パスワード ファイルの完全パスを入力します。

         完全パスが登録され、次に進みます。

      b. NIS グループ ファイルの完全パスを入力します。

         完全パスが登録され、次に進みます。

   b. 「n」と入力するか、NIS/DNS テーブルを更新する場合は Enter キーを押します。

      回答が登録され、次に進みます。

8. PMDB の特別な属性を付与するユーザを入力します。

   a. 必要に応じて CA Access Control 管理者名を入力したら、 Enter キーを押します。

      管理者には、PMDB のプロパティを変更する権限があります。

      注: PMDB には、管理者を最低 1 人は定義する必要があります(デフォルトは root)。

   b. 必要に応じてエンタープライズ ユーザ管理者名を入力したら、 Enter キーを押します。

   c. 必要に応じて CA Access Control 監査担当者名を入力したら、 Enter キーを押します。

      監査者には、PMDB の監査ログ ファイルを参照する権限があります。

   d. 必要に応じてエンタープライズ ユーザ監査担当者名を入力したら、Enter キーを押します。

e. 必要に応じて CA Access Control パスワード管理者名を入力したら、Enter キーを押します。

f. 必要に応じてエンタープライズ ユーザ パスワード管理者名を入力したら、Enter キーを押します。

パスワード管理者には、PMDB のパスワードを変更する権限があります。

回答が登録され、次に進みます。

9. 必要に応じて管理端末を入力したら、Enter キーを押します。

すべての管理端末が登録され、選択内容が表示されて確認を求められます。

10. Enter キーを押して選択内容を確定するか、「n」と入力してスクリプトを再実行し、新しい入力を指定します。

選択内容を確定すると、その情報をもとに新しい PMDB が作成されます。

詳細情報:

サブスクライバ PMDB の作成と設定(137 ページ)
サブスクライバ コンピュータの親 PMDB の定義(139 ページ)

サブスクライバ PMDB の作成と設定

マスタ PMDB を設定した後、階層を拡張する場合は、サブスクライバ PMDB を作成および設定する必要があります。 ローカル ホスト上でこれを行うには、sepmdadm コマンドを使用します。

注: 以下の手順では、sepmdadm コマンドを対話形式で入力する方法を説明します。すべての入力に対するコマンド ライン パラメータの使い方については、「リファレンス ガイド」を参照してください。

サブスクライバ PMDB を作成および設定するには、以下の手順に従います。

1. コマンド ラインで以下のコマンドを入力します。

```
sepmdadm -i
```

CA Access Control によって Policy Model データベース管理スクリプト (sepmdadm) が起動され、メニューが表示されます。ここでオプションを選択します。

2. 「2」を入力して、2 つ目のオプション(サブスクライバ PMDB を作成し、そのサブスクライバと親を定義する)を選択します。

スクリプトは、関連する質問をするように設定されています。

3. Enter キーを押して続行します。

続いて、1 つ目の質問が表示されます。

注: CA Access Control が実行中でない場合、CA Access Control を起動してからスクリプトを再実行するように警告が発行されます。

4. 作成する Policy Model の名前を入力します。

   Policy Model 名が登録され、次に進みます。

5. 指定する最初のサブスクライバ コンピュータの名前を入力します。

   最初のサブスクライバの名前が登録され、次のサブスクライバの名前を入力するように求められます。

6. 必要に応じてサブスクライバ名を入力したら、Enter キーを押します。

   すべてのサブスクライバ名が登録され、次に進みます。

   注: 各サブスクライバ コンピュータが親 PMDB を参照(139 ページ)している必要があります。

7. 親 PMDB の名前を入力します。

   親 PMDB 名が登録され、次に進みます。

   注: sepmdadm では、各サブスクライバ データベースに親を 1 つだけ入力できます。 ただし、本来はデータベースに複数の親を定義する事ができます。 そのためには、pmd.ini 環境設定ファイルの parent_pmd トークンを変更します。 このトークンの使用の詳細については、「リファレンス ガイド」を参照してください。

8. NIS、NIS+、または DNS を実行している場合は、PMDB の変更で NIS/DNS テーブルを更新するかどうかを選択します。

   更新は PMDB 内のユーザおよびグループに対して行われます。 テーブルには、ユーザとユーザの特性に関する情報が保存されています。 yes を選択すると、Policy Model で更新された UNIX ユーザまたは UNIX グループは、NIS の passwd ファイルと group ファイルでも更新されます。

   a. NIS/DNS テーブルを更新する場合は、「y」と入力します。

      ここで、NIS の passwd ファイルと group ファイルの場所を尋ねられます。

      a. NIS パスワード ファイルの完全パスを入力します。

         完全パスが登録され、次に進みます。

      b. NIS グループ ファイルの完全パスを入力します。

         完全パスが登録され、次に進みます。

   b. 「n」と入力するか、NIS/DNS テーブルを更新する場合は Enter キーを押します。

      回答が登録され、次に進みます。

9. PMDB の特別な属性を付与するユーザを入力します。

   a. 必要に応じて CA Access Control 管理者名を入力したら、 Enter キーを押します。

      管理者には、PMDB のプロパティを変更する権限があります。

      注: PMDB には、管理者を最低 1 人は定義する必要があります(デフォルトは root)。

   b. 必要に応じてエンタープライズ管理者名を入力したら、Enter キーを押します。

   c. 必要に応じて CA Access Control 監査担当者名を入力したら、 Enter キーを押します。

      監査者には、PMDB の監査ログ ファイルを参照する権限があります。

   d. 必要に応じてエンタープライズ ユーザ監査担当者名を入力したら、Enter キーを押します。

   e. 必要に応じて CA Access Control パスワード管理者名を入力したら、Enter キーを押します。

      パスワード管理者には、PMDB のパスワードを変更する権限があります。

   f. 必要に応じてエンタープライズ ユーザ パスワード管理者名を入力したら、Enter キーを押します。

   回答が登録され、次に進みます。

10. 必要に応じて管理端末を入力したら、Enter キーを押します。

    すべての管理端末が登録され、選択内容が表示されて確認を求められます。

11. Enter キーを押して選択内容を確定するか、「n」と入力してスクリプトを再実行し、新しい入力を指定します。

    選択内容を確定すると、その情報をもとに新しい PMDB が作成されます。

サブスクライバ コンピュータの親 PMDB の定義

エンドポイント コンピュータを PMDB のサブスクライバとして設定するには、サブスクライバの名前を PMDB に登録する以外にも操作が必要です。 サブスクライバ コンピュータでも操作を実行する必要があります。

サブスクライバ コンピュータに親 PMDB を定義するには、以下の手順に従います。

1. サブスクライバ コンピュータのコマンド ラインで、sepmdadm を対話モードで起動します。

   sepmdadm -i

   CA Access Control によって Policy Model データベース管理スクリプト（sepmdadm）が起動され、メニューが表示されます。ここでオプションを選択します。

2. 「3」を入力して、3 つ目のオプション（ローカル データベースの親 PMDB とパスワード PMDB を定義する）を選択します。

   スクリプトは、関連する質問をするように設定されています。

3. Enter キーを押して続行します。

   続いて、1 つ目の質問が表示されます。

   注: CA Access Control が実行中の場合、CA Access Control を停止してからスクリプトを再実行するように警告が発行されます。

4. 親 PMDB の名前を入力します。

   親 PMDB の名前が登録され、次に進みます。

5. 親パスワード PMDB の名前を入力します。

   親パスワード PMDB の名前が登録され、選択内容が表示されて確認を求められます。

6. Enter キーを押して選択内容を確定するか、「n」と入力してスクリプトを再実行し、新しい入力を指定します。

   選択内容を確定すると、サブスクライバ コンピュータはこれらの入力内容でセットアップされます。

注: sepmdadm では、各サブスクライバ データベースに親を 1 つだけ入力できます。ただし、本来はデータベースに複数の親を定義する事ができます。 そのためには、seos.ini 環境設定ファイルの parent_pmd トークンを変更します。 このトークンの使用の詳細については、「リファレンス ガイド」を参照してください。

UID と GID の同期

管理者として受け取るメッセージでは、UID がユーザを表し、GID がグループを表している場合があります。 UID と GID は、使用する場所に関係なく常に同じ意味を持っている必要があります。

PMDB のデフォルトでは、新規のユーザおよびグループがどこで使用されても、同じ UID と GID が使用されるように設定されていますが、開始時から必要な条件を指定すると、同期を確実に行うことができます。 そのためには、同じ passwd ファイルと同じ group ファイルを使用し、pmd.ini ファイルの synch_uid トークンを必ず yes に設定します。 ローカル データベースが PMDB のサブスクライバで、サブスクライバ データベースの新規ユーザおよび新規グループがその PMDB にのみ基づいて作成されている場合は、ローカル データベース、PMDB、および PMDB サブスクライバ間での UID と GID の互換性は信頼できます。

PMDB または他のサブスクライバ コンピュータですでに使用されている UID で新規ユーザを作成すると、そのサブスクライバ自体の更新は失敗しますが、このような競合がないすべてのサブスクライバ コンピュータで更新が正常に行われます。

passwd ファイルと group ファイルの同期は、各新規ユーザの UID および各新規グループの GID を明示的に指定して行うこともできます。

ユーザおよびグループの同期

さまざまなデータベース内でユーザとグループのリストが常に正しく対応するようにするには、同一リストの初期セットが必要です。 passwd ファイルおよび group ファイルは非常に重要であるため、ローカル ユーザおよびローカル グループに関する情報の蓄積が開始される前に、この 2 つのファイルの同期を行います。

ユーザおよびグループを同期するには、以下の手順に従います。

1. /etc/passwd ファイルおよび /etc/group ファイルを Policy Model ディレクトリにコピーすると、Policy Model ディレクトリ（128 ページ）にある既存の passwd ファイルおよび group ファイルは失われます。

   注: shadow ファイルを使用している場合にパスワードの同期を行うときは、secrepsw ユーティリティを使用することをお勧めします。 詳細については、「リファレンス ガイド」を参照してください。

2. 各サブスクライバ コンピュータに /etc/passwd ファイルと /etc/group ファイルをコピーし、ユーザ自身のコンピュータにある各ファイルと同一になるようにします。

3. PMDB が格納されているコンピュータで、pmd.ini ファイルの synch_uid トークンが yes に設定されていることを確認します。

   synch_uid トークンの値は、デフォルトでは yes です。 サブスクライバ データベースで独自のデフォルト UID とデフォルト GID を使用する場合（つまり、PMDB の UID と GID に一致させる必要がない場合）、synch_uid を no に設定できます。

UID の明示的な指定

同一の UID または GID を、PMDB とそのすべてのサブスクライバに送信するもう 1 つの方法は、新しいユーザを作成するときに明示的に設定することです。

UID を明示的に指定するには、各 newusr コマンドで userid または groupid パラメータを使用します。

例: 指定した UID を持つ新しいユーザを作成する

新規ユーザ terry_jones の UID として 1234 を明示的に設定するには(その UID がデータベース内の他のどのユーザにもまだ使用されていないと仮定した場合)、以下のコマンドを入力します。

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

指定した UID が PMDB ですでに使用されている場合、PMDB 自体は更新されませんが、他のサブスクライバ データベースへのコマンドの伝達は行われます。 他のサブスクライバ データベースでこの UID がすでに使用されている場合、そのサブスクライバ データベース自体の更新は失敗しますが、このような競合がないサブスクライバ データベースでは更新が正常に行われます。

Policy Model によるサブスクライバの更新方法

サブスクライバを更新すると、Policy Model は以下のアクションを実行します。

1. Policy Model からサブスクライバ名が追加または削除される場合、そのサブスクライバの名前を完全修飾しようとします。
2. PMDB デーモンである sepmdd は、_QD_timeout_ トークンで定義されている制限時間が経過するまで、サブスクライバ データベースの更新を試みます。

3. 制限時間が経過した時点でサブスクライバを更新できなかった場合、デーモンはそのサブスクライバの更新処理を省略して、サブスクライバ リストにある残りのサブスクライバの更新を試みます。
4. sepmdd は、サブスクライバ リストの 1 回目のスキャンが終了した後、2 回目のスキャンを実行します。2 回目のスキャンでは、1 回目のスキャンで更新できなかったサブスクライバの更新を試みます。 2 回目のスキャンでは、接続システム コールがタイムアウトになるまで(約 90 秒間)サブスクライバの更新を試みます。

注: _QD_timeout_ トークンが seos.ini ファイルおよび pmd.ini ファイルの両方に記述されている場合があります。 トークンが両方のファイルに存在する場合、sepmdd は pmd.ini ファイルの値を使用します。

注: サブスクライバへの更新情報の伝達時に PMDB でエラーが発生すると、sepmdd デーモンによって Policy Model のエラー ログ ファイル(148 ページ)にエントリが作成されます。 このファイル(ERROR_LOG)は、デフォルトでは PMDB ディレクトリ(128 ページ)に保存されます。

Policy Model データベースの更新

PMDB が格納されているコンピュータで操作を行っても、PMDB 自体は自動的に更新されません。 PMDB を更新するには、PMDB をターゲット データベースとして指定する必要があります。

ターゲット データベースを指定するには、selang のコマンド シェルで hosts コマンドを使用します。

```
hosts pmd_name@pmd_host
```

これで、指定した Policy Model データベースがすべての selang コマンドで更新されます。 次に、このコンピュータおよびすべてのサブスクライバ コンピュータ上のアクティブなデータベースにコマンドが自動的に伝達されます。

例: ターゲット PMDB を指定する

ターゲット データベースを myPMD_host の policy1 に設定するには、以下のコマンドを使用します。

```
hosts policy1@myPMD_host
```

ここで、newusr コマンドを入力すると、新規ユーザは policy1 データベースに追加される以外に、このコンピュータおよびすべてのサブスクライバ コンピュータ上のアクティブ データベースにも追加されます。

更新ファイルのクリーンアップ

sepmd ユーティリティは、受信した各更新情報を updates.dat ファイルに自動的に書き込みます。 このファイルのサイズが大きくなりすぎないように、処理済みの更新情報をファイルから定期的に削除することをお勧めします。

更新ファイルをクリーンアップするには、以下のコマンドを使用します。

```
sepmd -t pmdbName auto
```

sepmd は、まだ伝達されていない最初の更新エントリのオフセットを計算して、その前にあるすべての更新エントリを削除します

注: sepmd ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

更新ファイルの暗号化

PMDB の作成後、sepmdd を起動する前に、updates.dat ファイルに保存された情報を暗号化するように指定できます。

更新ファイルを暗号化するには、pmd.ini ファイルの[pmd]セクションにある UseEncryption トークンを yes に設定します。

updates.dat ファイルを復号化するには、-de スイッチを指定して sepmd ユーティリティを起動します。

注: sepmd の詳細については、「リファレンス ガイド」を参照してください。

サブスクライバの除外

サブスクライバを除外することにより、サブスクライバが親 PMDB から更新情報を受け取らないように設定できます。

ローカル ホストを除外するには、pmd.ini ファイルの exclude_localhost トークンを yes に設定します。

除外リストにサブスクライバを追加するには、exclude_file (name-of-file) トークンを設定します。

サブスクライバが更新情報を受け取るようにするには、除外リストからサブスクライバを削除します。

パスワードの伝達

ユーザが sepass ユーティリティを使用してパスワードを変更すると、通常は新規パスワードはコンピュータの親 PMDB に送信されます。 親 PMDB は、seos.ini ファイルの[seos]セクションにある parent_pmd トークンか passwd_pmd トークン、またはその両方で定義します。 ただし、ユーザが sepass ユーティリティを使用してパスワードを変更した場合は、ユーザの新規パスワードを別の PMDB に送信し、その PMDB から伝達されるように指定することもできます。

新規ユーザのパスワードを別の PMDB に送信するには、newusr、chusr、または editusr コマンドで pmdb パラメータを使用します。

例: 別の PMDB にパスワードの伝達を指定する

sepass を使用して作成された、Tony というユーザの新規パスワードを、別の PMDB pw_pmdb@name1.yourorg.com に送信し、その PMDB から伝達されるように指定するには、以下のコマンドを入力します。

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

サブスクライバの削除

更新情報が特定のサブスクライバに伝達されないようにする場合は、そのサブスクライバを削除する必要があります。 または、サブスクライバが更新情報を受け取らないように除外(144 ページ)することもできます。

サブスクライバを削除するには、以下の手順に従います。

1. コンピュータをサブスクライバ リストから削除します。

   ```
   sepmd -u PMDB_name computer_name
   ```

   コンピュータが Policy Model のサブスクライバ リストから削除されます。

2. サブスクライバ リストから削除したコンピュータで seosd を停止します。

   ```
   secons -s
   ```

   seosd デーモンが停止されます。

3. サブスクライバ リストから削除したコンピュータで seos.ini ファイルの[seos]セクションにある parent_pmd トークンの値を削除します。

   コンピュータは親 PMDB から更新情報を受け取らなくなります。

4. seosd を再起動します。

   サブスクライバ リストから削除したコンピュータ上のアクティブ データベースは、指定した PMDB のサブスクライバではなくなりました。

注: データベースが PMDB のサブスクライバから解除されると、PMDB はコマンドを送信しなくなります。

更新情報のフィルタ処理

1 つの PMDB を使用して、複数の異なるサブスクライバ データベースでデータのさまざまなサブセットを更新する場合は、サブスクライバ データベースにどのレコードを送信するかを定義する必要があります。

更新情報をフィルタ処理するには、以下の手順に従います。

1. サブスクライバのサブセットの親として PMDB を設定します(139 ページ)。
2. 親 PMDB の pmd.ini ファイルにある filter トークンを変更し、同じコンピュータで設定するフィルタ ファイルを参照するようにします。

   このように指定すると、フィルタ条件に該当するレコードのみがサブスクライバ データベースに更新情報として送信されます。

注: ネイティブ UNIX 環境で join または join- selang コマンドを実行すると、CA Access Control はコマンドを change group (cg)に変更します。 ネイティブ UNIX 環境で join または join- コマンドをフィルタリングするには、フィルタ ファイルの以下の行を使用します。

MODIFY UNIX GROUP GroupName USERS NOPASS

ネイティブ UNIX 環境では join または join- コマンドをユーザ名でフィルタリングすることはできません。 このルールはその他の環境の join または join- コマンドには適用されません。

Policy Model のフィルタ ファイル

フィルタ ファイルは、各行に 6 つのフィールドを持つ複数の行で構成されます。フィールドには以下の情報が含まれます。

- 許可または拒否されるアクセスの種類。

  例: READ または MODIFY
- 影響を受ける環境。

  例: AC またはネイティブ
- レコードのクラス。

  例: USER または TERMINAL
- ルールが適用される、クラスのオブジェクト。

  たとえば、User1、AuditGroup、または TTY1 になります。

- レコードによって許可または取り消されるプロパティ。

  たとえば、フィルタ行の OWNER および FULL_NAME は、これらのプロパティを持つコマンドはすべてフィルタ処理されることを意味します。 各プロパティは、「リファレンス ガイド」に記載されているとおりに、正確に入力する必要があります。

- 該当するレコードをサブスクライバ データベースに転送するかどうか。

  PASS または NOPASS

フィルタ ファイルの各行に以下のルールが適用されます。

- どのフィールドでも、アスタリスク(*)を使用して可能なすべての値を指定することができます。
- 同じレコードが複数の行に該当する場合は、最初の該当する行が使用されます。
- フィールドをスペースで区切ります。
- フィールドに複数の値がある場合は、値をセミコロンで区切ります。
- # で始まる行はコメント行とみなされます。
- 空白行は使用できません。

例: フィルタ ファイル

以下の例では、フィルタ ファイルの行について説明します。

CREATE	AC	USER	*	FULL_NAME;OBJ_TYPE	NOPASS
アクセス形式	環境	クラス	レコード名 (* = すべての名前)	プロパティ	処理方法

この例では、この行を指定したファイルの名前が TTY1_FILTER で、PMDB TTY1 の pmd.ini ファイルを編集してフィルタを /opt/CA/AccessControl/TTY1_FILTER と指定した場合、PMDB TTY1 は、FULL_NAME と OBJ_TYPE プロパティを指定してユーザを新規作成するレコードをサブスクライバに伝達しません。

Policy Model のエラー ログ ファイル

Policy Model のエラー ログ(発生順に書き込まれる)の例を以下に示します。

エラー テキスト	エラー カテゴリ
20 Nov 03 11:56:07 (pmdb1): fargo nu u5 0 Retry エラー: ログインできませんでした。(10068) エラー: 親ではない PMDB (pmdb1@name.company.com) からの更新を受け付けることはできません。(10104)	環境設定エラー
20 Nov 03 19:53:17 (pmdb1): fargo nu u5 0 Retry エラー: 接続できませんでした。(10071) ホストに接続できません。(12296)	接続エラー
20 Nov 03 11:57:06 (pmdb1): fargo nu u5 560 Cont エラー: USER u5 の作成に失敗しました。(10028) すでに存在しています (9) 20 Nov 03 11:57:06 (pmdb1): fargo nu u5 1120 Cont エラー: USER u5 の作成に失敗しました。(10028) すでに存在しています (9)	データベース更新エラー

Policy Model のエラー ログはバイナリ フォーマットであるため、以下のコマンドを入力することでのみ表示できます。

```
ACInstallDir/bin sepmd -e pmdname
```

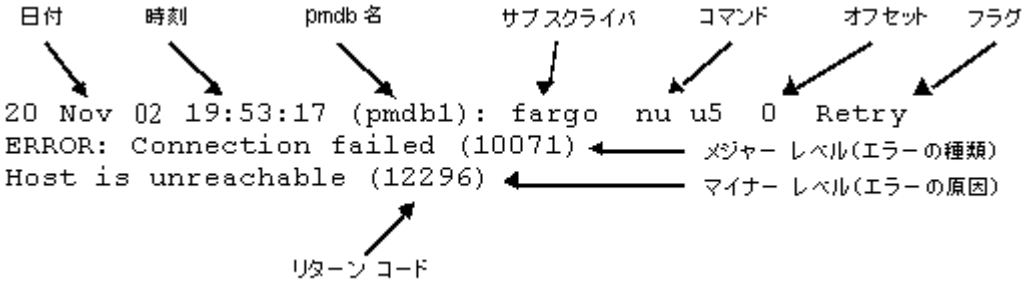
注: エラー ログは手動で削除しないでください(たとえば、UNIX の rm コマンドを使用した削除)。 ログを削除するには、以下のコマンドのみを使用してください。

```
ACInstallDir/bin sepmd -c pmdname
```

重要: CA Access Control r5.1 以降のバージョンでのエラー ログのフォーマットには、旧バージョンのフォーマットとの互換性はありません。 sepmd を使用して、旧バージョンのエラー ログを処理することはできません。 このバージョンのフォーマットにアップグレードする際に、旧エラー ログは ERROR_LOG.bak としてコピーされ、sepmdd を起動すると新しいログ ファイルが作成されます。

例: PMDB 更新のエラー メッセージ

以下の例は、標準的なエラー メッセージを示しています。



- 先頭行には必ず、日付、時刻、およびサブスクライバが表示されます。 次に、エラーを発生させたコマンドが表示され、その後に、更新ファイル内の失敗した更新の位置を示すオフセット(10 進数)が続きます。 最後のフラグは、PMDB が更新を自動的に再試行するか、または再試行せずに継続するかを示します。
- 2 行目は、メジャー レベル メッセージ(発生したエラーの種類)とリターン コードの例を示します。
- 3 行目は、マイナー レベル メッセージ(エラーの発生理由)とリターン コードの例を示します。

例: エラー メッセージ

1 つのコマンドによって、複数のエラーが生成および表示される場合があります。 また、エラーは、メジャー レベル メッセージ、マイナー レベル メッセージ、またはその両方で構成される場合があります。

以下のエラーには、メッセージ レベルが 1 つしかありません。

Fri Dec 29 10:30:43 2003 CIMV_PROD:リリースに失敗しました。 リターン コード = 9241

このメッセージは、すでに使用可能なサブスクライバのリリースを sepmd pull が試みた場合に表示されます。

ポリシー モデルのバックアップ

PMDB をバックアップする場合、Policy Model データベースのデータを別のディレクトリにコピーします。 これには以下のデータが含まれます。

- ポリシー情報
- Policy Model のサブスクライバのリスト
- 設定
- レジストリ エントリ
- updates.dat ファイル

別のプラットフォーム、オペレーティング システム、または CA Access Control バージョンを使用するバックアップ ファイルから PMDB をリストアすることはできません。 同じプラットフォーム、オペレーティング システム、および CA Access Control バージョンが動作するホストに Policy Model をバックアップしてください。

sepmd を使用した PMDB のバックアップ

PMDB のバックアップでは、Policy Model データベースのデータを指定のディレクトリにコピーします。 PMDB のバックアップ ファイルは、安全な場所、できれば CA Access Control アクセス ルールで保護された場所に保存してください。

PMDB をローカル ホストにバックアップする場合は、sepmd ユーティリティを使用します。 また、selang コマンドを使って PMDB をリモート ホストにバックアップすることもできます。

注: PMDB は再帰的にバックアップできます。 再帰的なバックアップでは、階層内のすべての PMDB が指定のホストにバックアップされ、バックアップがそのホストに移動してもサブスクリプションが引き続き機能するように PMDB サブスクライバが変更されます。 再帰的なバックアップは、マスタと子の PMDB が同じホスト上で展開される場合にのみ使用できます。

sepmd を使用して PMDB をバックアップする方法

1. 以下のコマンドを使って PMDB をロックします。

   ```
   sepmd -bl pmdb_name
   ```

   PMDB はロックされるため、サブスクライバにコマンドを送信できなくなります。

2. 以下のいずれかの操作を実行します。

   - 以下のコマンドを使って PMDB をバックアップします。

     ```
     sepmd -bh pmdb_name [destination_directory]
     ```

- 以下のコマンドを使って PMDB データベースを再帰的にバックアップします。

  sepmd -bh pmdb_name [destination_directory] [backup_host_name]

  注: ユーザが送信先ディレクトリを指定しない場合、バックアップは以下のディレクトリに保存されます。

  ACInstallDir/data/policies_backup/pmdb_name

3. 以下のコマンドを使って、PMDB のロックを解除します。

   sepmd -ul pmdb_name

   PMDB のロックが解除され、サブスクライバにコマンドを送信できるようになります。

selang を使用した PMDB のバックアップ

PMDB のバックアップでは、Policy Model データベースのデータを指定のディレクトリにコピーします。 PMDB のバックアップ ファイルは、安全な場所、できれば CA Access Control アクセス ルールで保護された場所に保存してください。

selang コマンドを使用して、PMDB をローカル ホスト、またはリモート ホストにバックアップできます。 ローカル ホストに PMDB をバックアップする場合は、sepmd ユーティリティも使用できます。

注: PMDB は再帰的にバックアップできます。 再帰的なバックアップでは、階層内のすべての PMDB が指定のホストにバックアップされ、バックアップがそのホストに移動してもサブスクリプションが引き続き機能するように PMDB サブスクライバが変更されます。 再帰的なバックアップは、マスタと子の PMDB が同じホスト上で展開される場合にのみ使用できます。

selang を使用して PMDB をバックアップする方法

1. （オプション） selang を使用してリモート ホストから PMDB に接続している場合は、以下のコマンドを使って PMDB ホストに接続します。

   host pmdb_host_name

2. 以下のコマンドを使って、PMD 環境に移動します。

   env pmd

3. 以下のコマンドを使って DMS をロックします。

   pmd pmdb_name lock

   PMDB はロックされるため、サブスクライバにコマンドを送信できなくなります。

4. 以下のコマンドを使って DMS データベースをバックアップします。

   backuppmd pmdb_name [destination(destination_directory)] [hir_host(host_name)]

   注: ユーザが送信先ディレクトリを指定しない場合、バックアップは以下のディレクトリに保存されます。

   ACInstallDir/data/policies_backup/pmdbName

5. 以下のコマンドを使って、PMDB のロックを解除します。

   pmd pmdb_name unlock

   PMDB のロックが解除され、サブスクライバにコマンドを送信できるようになります。

Policy Model のリストア

Policy Model をリストアする場合、CA Access Control は指定のディレクトリにバックアップ PMDB ファイルをコピーします。 元の PMDB ファイルのデータが、以下を含めてすべて新しい PMDB ディレクトリにコピーされます。

- ポリシー情報
- Policy Model のサブスクライバのリスト
- 設定
- レジストリ エントリ
- updates.dat ファイル

コピー先ディレクトリに既存の PMBD がある場合、CA Access Control は既存のファイルを削除してからリストア ファイルをそのディレクトリにコピーします。

別のプラットフォーム、オペレーティング システム、または CA Access Control バージョンを使用するバックアップ ファイルから PMDB をリストアすることはできません。 同じプラットフォーム、オペレーティング システム、および CA Access Control バージョンが動作するホストに Policy Model をバックアップしてください。

PMDB のリストア

任意の PMDB をリストアすると、CA Access Control はその PMDB のバックアップ ファイルから指定のディレクトリ上へデータをコピーします。 CA Access Control はリストア処理を行う端末上で実行されている必要があります。

注: PMDB を別の端末にバックアップおよびリストアする場合、PMDB はリストアされた PMDB データベースにあるターミナル リソースの更新を、自動的には行いません。 新しいターミナル リソースはリストアされた PMDB に追加する必要があります。 新しいターミナル リソースを追加するには、リストアされた PMDB を停止し、selang -p pmdb コマンドを実行して、さらにリストアされた PMDB を起動します。

任意の PMDB をリストアするには、以下のいずれかを PMDB をリストアする端末上で実行してください。

- sepmd –restore ユーティリティ
- selang restore pmd コマンド

注: sepmd ユーティリティの詳細については「リファレンス ガイド」を参照してください。 selang コマンドの詳細については、「selang リファレンス ガイド」を参照してください。

デュアル コントロール

デュアル コントロールは、PMDB の更新プロセスを 2 つの段階に分ける操作方法です。

- 1 つ以上のコマンドで構成されるトランザクションを作成します。

  ADMIN 属性を持つユーザである Maker(作成者)が、PMDB を更新する 1 つ以上のコマンドを入力します。 トランザクションには一意の ID 番号が割り当てられ、ファイルに格納されて、処理の実行を待ちます。

- トランザクションの実行を許可します。

  第 2 段階では、Checker(チェッカ)がトランザクション内のコマンドをロックし、チェックして、コマンドを許可または拒否します。 Checker(チェッカ)は、ADMIN 属性を持つ、Maker(作成者)以外のユーザです。 トランザクションが許可されると、そのコマンドは PMDB で実行されます。 トランザクションが拒否されると、そのトランザクションは削除され、PMDB は更新されません。 Checker(チェッカ)は、トランザクション内の一部のコマンドを許可して残りのコマンドを拒否することはできません。 トランザクションは全体で 1 つのものとして処理する必要があります。

  注: find コマンドおよび show コマンドには Checker(チェッカ)の許可は不要です。

sepmd ユーティリティでパラメータを使用すると、Maker(作成者)は、未処理のトランザクションを表示、検索、編集、または削除できます。Checker(チェッカ)は、トランザクションを許可または拒否するためにロックしたり、後で処理するため、または別の Checker (チェッカ)によって処理するためにトランザクションのロックを解除したりすることができます。

sepmdd デーモンは、start_transaction コマンドを受信すると、子プロセスに一意の番号を送信します。 子プロセスは、それ以降のすべてのコマンドにこの識別番号を付けます。その番号は新規トランザクションに追加され、sepmdd デーモンのメモリ内に保存されます。 sepmdd が end_transaction コマンドを受信すると、承認アルゴリズムが呼び出されます。 この承認アルゴリズムによって、トランザクションの Maker(作成者)に関係しているコマンドがトランザクション内のコマンドの中にないことがチェックされます。また、処理の実行を待機している別のトランザクションによってすでにロックされているオブジェクトがコマンド内のオブジェクトの中にないことがチェックされます。

オブジェクトを処理する前に、別のトランザクションで同じオブジェクトを使用することはできません。 チェックを通過すると、関連オブジェクトがロックされ、トランザクションに一意の連続番号が割り当てられて、データがファイルに保存されます。 各トランザクションは別々のファイルに保存されます。

注: sepmd ユーティリティまたは sepmdd デーモンの詳細については、「リファレンスガイド」を参照してください。

デュアル コントロールの有効化

デュアル コントロールでは、PMDB の更新の作業を Maker(作成者)と Checker (チェッカ)という 2 人のユーザに分けることができます。

デュアル コントロールを有効にするには、pmd.ini ファイル、および seos.ini ファイルの[pmd]セクションで is_maker_checker トークンを yes に設定します。

```
is_maker_checker=yes
```

注: これらのトークンの値を設定する前に、Policy Model の Maker(作成者)を作成してください。

トランザクションの作成または編集

デュアル コントロールが有効になっている場合、Maker(作成者)は、Checker(チェッカ)が処理する前にトランザクションを作成する必要があります。

トランザクションを作成するには、以下の手順に従います。

1. 以下の条件を満たしていることを確認します。
   - Maker（作成者）に ADMIN 権限があること。
   - 自分に関係するコマンドが含まれていないこと（自分自身を変更するコマンドは入力不可）。
   - コマンド内のオブジェクトの中に、Checker（チェッカ）がまだ処理していない別のトランザクションの構成要素であるオブジェクトがないこと。
   - コマンド内のすべてのオブジェクトが存在していること。
   - 別の Maker（作成者）が呼び出した既存のトランザクションを編集していないこと（編集できるのは自分自身のトランザクションのみ）。
2. maker という PMDB に接続します。

   ```
   hosts maker@
   ```

   この hosts コマンドによって、PMDB（maker）に接続されます。 デュアル コントロールが有効になっている場合、PMDB の名前は常に「maker」です。hosts コマンドの入力後、ホストへの接続が成功したかどうかを示すメッセージが表示されます。
3. トランザクションを開始します。

   ```
   start_transaction transactionName
   ```

   トランザクションを入力または更新する際の最初の手順として、start_transaction コマンドを使用します。 トランザクションの説明またはトランザクション名には、256 文字以内の英数字を使用できます。
4. トランザクションを入力します。

   これは、コマンドのリストです。 以下に例を示します。

   ```
   newusr mary owner(bob) audit(failure,loginfailure)
   chres TERMINAL tty30 defaccess(read) \
    restrictions(days(weekdays)time(0800:1800))
   ```
5. トランザクションを終了します。

   ```
   end_transaction
   ```

   これでトランザクションは完成です。トランザクションに割り当てられた一意の ID 番号が表示されます。 コマンドはファイルに格納され、Checker（チェッカ）が処理に備えてロックするまでは、アクセスおよび変更が可能です。

   注：後でトランザクションを編集する場合は、トランザクション ID 番号を控えておいてください。

トランザクションを編集するには、以下の手順に従います。

- end_transaction コマンドを入力すると、ID 番号が表示されます。 これは、トランザクションを識別する一意の番号です。 トランザクションを後で変更する場合、手順は新規トランザクションの作成と同じですが、ファイルのトランザクション名の後にトランザクションの ID 番号を追加します。 ファイルに必要な変更を入力することができます。 以下に例を示します。

  hosts maker@
  start_transaction transactionName transactionId

  この後、適切なコマンドを入力して、トランザクションを更新できます。

  chusr mary category (FINANCIAL)
  end_transaction

- 以下のパラメータを指定して、特定の未処理トランザクションを参照します。

  ACInstallDir/bin パス内で操作していることを確認してください(ここで、ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です)。

コマンドとパラメータ	説明
sepmd -m l	パラメータを呼び出したユーザの未処理トランザクションを一覧表示します。
sepmd -m la	すべての Maker(作成者)の処理待ちトランザクションをすべて一覧表示します。
sepmd -m lo	パラメータを呼び出したユーザのトランザクションを除く、すべての Maker(作成者)のトランザクションを一覧表示します。 リスト内の各トランザクションには、Maker(作成者)の名前、トランザクションの ID 番号、およびトランザクションの説明(Maker(作成者)が入力した場合)が含まれています。

- 以下のコマンドを使用して、特定のトランザクションを標準出力に取り出します。

  sepmd -m r transactionId

- 以下のコマンドを使用して、特定のトランザクションを削除します。

  sepmd -m d transactionId

トランザクションのチェックおよび処理

デュアル コントロールが有効になっている場合、Checker(チェッカ)は、Maker(作成者)が作成したトランザクションを処理する必要があります。

トランザクションをチェックするには、以下の手順に従います。

1. 以下の条件を満たしていることを確認します。
   - Checker（チェッカ）に ADMIN 権限があること。
   - 別の Checker（チェッカ）がトランザクションをロックしていないこと。
   - 自分に関係するコマンドが含まれていないこと（自分自身に関係するコマンドの処理は不可）。
2. ACInstallDir/bin パスに移動します。

   （ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です）。
3. 実行前の処理待ちトランザクションを表示します。

   sepmd -m la

   または、自分が作成したトランザクション以外のすべてのトランザクションを表示します。

   sepmd -m lo

   各トランザクションには、Maker（作成者）の名前、トランザクションの ID 番号、およびトランザクションの名前または説明が含まれます。
4. トランザクションを処理する前にロックします。

   sepmd -m r transactionId

   注: ロックされたトランザクションは変更できません。
5. トランザクションを処理します。

   sepmd -m p transactionId code

   code

   以下のいずれかを指定できます。
   - 0 - トランザクションを拒否します。

     この場合、トランザクション内のすべてのコマンドが削除され、PMDB では何も変更されません。

- 1 - トランザクションを許可します。

  トランザクション内のコマンドは PMDB でただちに実行されます。

- 2 - トランザクションのロックを解除します。

  トランザクションは待機トランザクションのキューに戻されるので、後で別の Checker（チェッカ）によって処理できます。

成功したコマンドと失敗したコマンドを示すメッセージが表示されます。

注: Maker（作成者）および Checker（チェッカ）の詳細については、「リファレンス ガイド」の sepmd ユーティリティの説明、および「selang リファレンス ガイド」の start_transaction コマンドの説明を参照してください。

seagent デーモンと sepmdd デーモンの使用法

seagent デーモンには、リモート コンピュータから要求を受け取り、その要求を PMDB に適用する役割があります。また、要求を seosd に送信する役割もあります。 sepmdd デーモンは、PMDB のデーモンです。 このセクションでは、これらのデーモンを PMDB 環境で組み合わせて使用する方法について説明します。

seagent デーモン

seagent デーモンは、seoslang および seoslang2 という TCP サービス（デフォルト値はそれぞれ 8890 および 8891）に対する接続を待機します。 接続要求を受け取ると、seagent は fork で子プロセスを作成し、接続から発生する通信を処理します。その後、引き続き新規の接続を待機します。

ユーザが selang の hosts コマンドを入力すると、seagent はユーザが接続しているマシン上に fork で子プロセスを作成します。 子プロセスはコマンド言語インタフェースからコマンドを受け取り、sepmdd デーモンに渡します。

sepmdd デーモン

sepmdd デーモンは以下の機能を実行します。

- PMDB の管理
- サブスクライバ データベースの管理
- PMDB からサブスクライバ データベースへの変更の伝達

sepmdd デーモンは、seagent が PMDB にアクセスする必要があるときに、seagent によって自動的に起動されます。 通常は sepmdd を明示的に実行する必要はありません。

注: sepmdd は、root ユーザではなく論理ユーザ _seagent によって、AC 環境で実行されます。 sepmdd によるリソースへのアクセスを許可または制限する(たとえば、PMDB ディレクトリへのアクセスを制限する)には、_seagent に対して適切なルールを作成します。

shadow ファイルの使用

通常、sepmdd はネイティブ環境の更新時に shadow ファイルを使用しません。 ただし、shadow ファイルの設定はできます。 これを行うには、pmd.ini ファイルの[pmd]セクションにある UseShadow トークンを yes に設定します。

UseShadow トークンが yes に設定されると、sepmdd は、PMDB と同じディレクトリにあるデフォルトの shadow ファイルを使用します。 shadow ファイルの場所を変更する場合は、pmd.ini の[pmd]セクションにある YpServerSecure トークンを使用して、新しい場所を指定します。

(YpServerSecure を使用して)shadow ファイルの場所をローカル ホストの shadow ファイル(たとえば、/etc/shadow)に変更する場合、sepmdd は UseSystemFiles トークンを yes に設定します。

重要: UseSystemFiles トークンそのものは変更しないでください。 このトークンは、sepmdd デーモンまたは seagent デーモンによって自動的に変更されます。

注: seagent デーモンまたは sepmdd デーモンの詳細については、「リファレンス ガイド」の seagent ユーティリティおよび sepmdd ユーティリティの説明を参照してください。

メインフレームのパスワード同期

CA Access Control では、CA Access Control を実行している Windows または UNIX マシンと、CA Top Secret、CA ACF2、または RACF セキュリティ製品(および CA Common Services CAICCI パッケージ)を実行している メインフレームとのパスワード同期(255 ページ)をサポートしています。 同期は、CA Access Control の標準のパスワード Policy Model 方式によって実現します。

メインフレームのユーザがパスワードを変更するたびに、パスワード Policy Model 階層内のすべてのマシンにその変更が伝達されます。

第 12 章: 包括的なセキュリティ機能

このセクションには、以下のトピックが含まれています。

アイドル状態の端末の保護

端末がオープンかつアクティブなままで放置されている場合、情報は極めて無防備な状態となります。 このような端末に(昼食時間などに)遭遇した侵入者は、サイトのすべての端末がすでにログイン済みで使用可能な状態であるため、パスワードの突破やネットワーク回線での盗聴を行うための複雑なツールを用意する必要がありません。 パスワードを入力しなければデスクトップを復元できないスクリーン セーバは役に立ちますが、セキュリティ管理者がすべてのユーザに安全なスクリーン セーバを確実に使用させることは不可能です。

CA Access Control には、selock というスクリーン ロック ユーティリティが用意されています。このユーティリティは、アイドル状態が指定した時間を超過したすべての端末をロックすることで、それらの端末を保護します。 作業に戻るときに、ユーザはパスワードを指定するように求められます。 1 分以内に適切なパスワードが入力されない場合、端末はロックされた状態になります。 selock ユーティリティは、画面のロックを解除する権限を持つユーザが selock の起動中にパスワードを変更していても、それらのユーザのパスワードを確認できます。

注: スクリーン ロック ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

要件に適合する selock オプションを使用する必要があります。

- 低いセキュリティ、高い利便性

  -timeout オプションを使用してタイムアウトを大きな値（10 分など）に設定し、-lock-timeout オプションを使用してロック タイムアウトをさらに大きな値（60 分など）に設定します。 この設定では、selock によるセーバモードへの移行による必要以上な作業の中断を回避します。 また、延長時間内に端末が操作されなかった場合のみ、画面がロックされます。

- 高いセキュリティ、低い利便性

  -timeout オプションを使用してタイムアウトを小さな値（1 分など）に設定し、-lock-timeout オプションを使用してロック タイムアウトを小さな値（0 ～2 分）に設定します。 この設定では、端末へのアクセスを終了するとただちに作業内容が非表示になり、アクセスを再開するためにはパスワードが必要です。 セーバ モードの開始後、端末を再びアクティブにするときに selock が常にパスワードの入力を要求するようにするには、-lock-timeout オプションでロック タイムアウトを 0 に設定します。

- selock コマンドは、X スタートアップ シェルの一部に設定できます。この場合、ユーザがシステムにログインするたびに、selock が自動的に起動します。 スクリプトは、root ID ではなく、ユーザ ID で実行する必要があります。 selock コマンドをスタートアップ スクリプトで指定する方法は、サイトの環境によって異なります。

  注: スタートアップ スクリプトの詳細については、ご使用の UNIX システムのマニュアルを参照してください。

保護モード

selock には、以下の 3 つの操作モードがあります。

モニタ モード

モニタ モードは selock の初期モードです。 このモードでは、キーボードおよびマウスの操作が監視されます。 タイムアウトの制限時間内にキーボードまたはマウスの操作が検出されず、transparent パラメータがオフになっている場合、selock は自動的にセーバ モードに切り替わります。 モニタ モードからセーバ モードに移行するためにパスワードを入力する必要はありません。

セーバ モード

セーバ モードの場合、画面全体が空白になり、位置が移動するシステム アイコンが表示されます。 空白の画面および移動するアイコンには、操作上、以下の 2 つの利点があります。

- 権限のないユーザに画面の内容を覗き見される可能性が少なくなります。
- 画面の焼け付きが少なくなります。

アイコンの外観および位置は、selock オプションを使用して調整できます。 キーボードまたはマウスの操作が検出されると、ただちにセーバ モードからモニタ モードに戻り、セーバ モードに移行する前に表示されていた内容が画面に再び表示されます。 モニタ モードからセーバ モードに移行するためにパスワードを入力する必要はありません。

セーバ モードの状態で lock-timeout パラメータに指定された時間が経過すると、自動的にロック モードに切り替わります。 このとき、セーバ モードからロック モードへの移行を示す画面上のメッセージは表示されません。

ロック モード

デフォルト設定のロック モードでは、移動するアイコンが黒い画面上に継続して表示されます。 selock がキーボードまたはマウスの操作を検出すると、ユーザのパスワード入力を求めるダイアログ ボックスが表示されます。

ユーザが正しいパスワードを入力すると、selock はモニタ モードに切り替わります。不正なパスワードを入力すると、パスワード入力のダイアログ ボックスが閉じられ、selock はロック モードの状態になります。

-transparent オプションを「on」に設定すると、画面はロックされますが、内容は引き続き表示され、実行中のプロセスは更新されます。 画面の背景は、画面がロックされていることを示すために変更されます。 ロック モードを使用している場合は、セーバ モードにはなりません。

アイドル状態の端末のロックの設定

selock ユーティリティではアイドル状態の端末をロックすることで、アイドル状態にある端末を不正なアクセスから保護することができます。

アイドル状態の端末をロックに設定するには、以下の手順に従います。

1. （オプション）DISPLAY 環境変数を設定します。

   selock コマンドを使用するには、DISPLAY 環境変数を設定する必要があります。ただし、selock コマンドでターゲット ディスプレイを直接指定することもできます。

2. ユーザのログイン スクリプト（.login ファイル）に selock コマンドを設定します。

   selock コマンドは、/etc/login ファイルや /etc/cshrc ファイルに設定することもできます。

注: いつでも画面のロックを解除できるユーザは 2 名います。 デフォルトでは、現在のユーザと root ユーザです。 ただし、seos.ini ファイルの[selock]セクションにある unlocking_user トークンで他のユーザ名を指定すると、root ユーザの代わりに他のユーザを割り当てることが可能です。 selock の実行時に -user オプションを使用すると、現在のユーザを他のユーザに変更できます。

例: スタートアップ ファイルに設定するアイドル状態の端末のロック コマンド

以下のステートメントは、X スタートアップ ファイルでの設定に適した一般的なスタートアップ コマンドです。

```
selock -display $DISPLAY -timeout 5
```

このコマンドは、端末がユーザ アクティビティのない状態になってから 5 分後に selock を起動します。

グローバル xstartup スクリプトに以下の行を追加することをお勧めします。 通常、xstartup スクリプトは /usr/lib/X11/xdm/Xstartup ディレクトリにあります。

```
selock -display $DISPLAY -user $USER -timeout 3 &
```

このステートメントにより、X 端末を使用しているすべてのユーザに対して、端末をロックするプログラムの使用が適用されます。

スクリーン ロック アイコンの変更

selock が使用するデフォルトのシステム アイコンは、CA Access Control のロゴで、ACInstallDir/data/admin/Selogo.xpm ファイルにあります。

このファイルを置き換えて、独自のアイコンを使用できます。

注: アイコン ファイルは、XPM バージョン 3.3 形式である必要があります。

API によるリソースの保護

CA Access Control にないリソース(社内リソース)を定義している場合は、 CA Access Control API を使用してそれらのリソースを保護することができます。 各 API には、以下の 2 つのレイヤがあります。

関数ライブラリ

プログラマが CA Access Control 認証エンジンを使用できるようにします。

user exit

CA Access Control の動作をシステム管理者がサイト要件に合わせて調整できるようにします。

注: CA Access Control API の詳細については、「SDK ガイド」を参照してください。

スタック オーバーフローの防止: STOP

スタック オーバーフローによって、ハッカーは、リモートまたはローカルのシステムに対して、root ユーザ(スーパーユーザ)としてあらゆるコマンドを何度でも実行できます。 ハッカーは、オペレーティング システムや他のプログラムのバグを利用して、スタック オーバーフローを発生させます。 これらのバグにより、ユーザがプログラム スタックを上書きできるようになるので、次に実行するコマンドに変更が加わります。

スタック オーバーフローは単なるバグではありません。戻りアドレスを意味のあるアドレスで上書きするブロックを作成できるので、(通常は同じブロックにある)未承認コードに制御が転送されることになります。

スタック オーバーフロー防止機能(STOP)は、ハッカーがスタック オーバーフローを発生させ、それを利用してシステムに侵入するのを防止する機能です。

STOP の開始と停止

STOP を初めてインストールする場合、デフォルトではスタック オーバーフロー防止機能が有効になっています。 これを無効にするには、seos.ini ファイルの[seos_syscall]セクションにあるトークンを変更して、CA Access Control を再起動する必要があります。これを行うには、seini コマンドを以下のように使用します。

```
seini -s SEOS_syscall.STOP_enabled 0
```

代わりに、seos.ini ファイルを手動で変更することもできます。

STOP を再度有効にするには、トークンの値を 1 に変更して CA Access Control を再起動します。

注: Sun Solaris 7 システム上で STOP が有効な場合、dbx プログラムは正常に機能しません。 STOP で保護されているシステムで dbx を使用する必要がある場合は、まず STOP を無効にする必要があります。

リソースに対する曜日と時間帯のアクセス ルールの定義

CA Access Control を使用して、リソース アクセスに対して時間帯と曜日の制限を指定できます。 この機能は、TERMINAL アクセス、SURROGATE 要求、およびユーザ定義リソースに対して使用できます。 たとえば、以下のルールでは、端末 ws3 の週末の使用および毎日 8 時から 19 時までの時間外の使用を完全に禁止しています。

```
chres TERMINAL ws3 restrictions(days(weekdays) time(0800:1900))
```

この端末からのログイン要求は、これらの時間帯以外は認められません。

CA Access Control では、勤務時間外における上位の権限ユーザ ID へのユーザ切り替えを防止できます。 たとえば、ユーザ AcctMgr は財務取引の実行を許可されている経理担当マネージャで、AcctMgr のログインは平日の勤務時間内のみに制限されているとします。 侵入者や権限のないユーザが、コマンド su AcctMgr を実行して AcctMgr のアカウントにアクセスしようとする場合があります。 指定時間外にユーザを AcctMgr に切り替えできないようにするには、以下のコマンドを使用します。

```
chres SURROGATE USER.AcctMgr restrictions(days(weekdays) time(0800:1900))
```

この回避策は、保護対象のあらゆるリソースに適用できます。社内アプリケーションの実装に使用するユーザ定義の抽象クラスにも適用できます。

B1 セキュリティ レベル認証

CA Access Control は、以下に示す「Orange Book」の B1 機能を備えています。

- セキュリティ カテゴリ
- セキュリティ ラベル
- セキュリティ レベル

セキュリティ レベル

セキュリティ レベルのチェックを有効にすると、CA Access Control は他の権限チェックに加えて、セキュリティ レベルのチェックを実行します。 セキュリティ レベルは、ユーザおよびリソースに割り当てることができる 1 から 255 までの正の整数です。 セキュリティ レベルが割り当てられているリソースに対してユーザがアクセスを要求すると、CA Access Control では、そのリソースのセキュリティ レベルとユーザのセキュリティ レベルが比較されます。 ユーザのセキュリティ レベルがリソースのセキュリティ レベルと同じか、それより上である場合、CA Access Control では他の権限チェックが続行されます。リソースのセキュリティ レベルより下の場合、リソースへのユーザのアクセスは拒否されます。

SECLABEL クラスがアクティブな場合は、リソースとユーザのセキュリティ ラベルに関連付けられているセキュリティ レベルが使用され、リソース レコードおよびユーザ レコードに明示的に設定されているセキュリティ レベルは無視されます。

セキュリティ レベルのチェックを使用してリソースを保護するには、セキュリティ レベルをリソースのレコードに割り当てます。 newres コマンドまたは chres コマンドの level パラメータによって、セキュリティ レベルをリソースに割り当てます。

セキュリティ レベルのチェックで保護されているリソースに対してユーザのアクセスを許可するには、セキュリティ レベルをユーザのレコードに割り当てます。 newusr コマンドまたは chusr コマンドの level パラメータによって、セキュリティ レベルをユーザに割り当てます。

セキュリティ レベル チェックの有効化

セキュリティ レベルのチェックを有効にするには、以下の setoptions コマンドを実行します。

```
setoptions class+ (SECLEVEL)
```

セキュリティ レベル チェックの無効化

セキュリティ レベルのチェックを無効にするには、以下の setoptions コマンドを実行します。

setoptions class- (SECLEVEL)

セキュリティ カテゴリ

セキュリティ カテゴリ チェックを有効にすると、CA Access Control では、他の権限チェックに加えて、セキュリティ カテゴリ チェックが実行されます。 1 つ以上のセキュリティ カテゴリが割り当てられているリソースに対してユーザがアクセスを要求すると、CA Access Control では、そのリソース レコードのセキュリティ カテゴリのリストとユーザ レコードのセキュリティ カテゴリのリストが比較されます。 リソースに割り当てられたすべてのカテゴリがユーザのカテゴリ リストに含まれている場合は、他の権限チェックが続行されます。含まれていない場合は、リソースに対するユーザのアクセスは拒否されます。

SECLABEL クラスがアクティブな場合は、リソースとユーザのセキュリティ ラベルに関連付けられているセキュリティ カテゴリのリストが使用され、ユーザ レコードおよびリソース レコード内のカテゴリのリストは無視されます。

セキュリティ カテゴリのチェックによってリソースを保護するには、1 つ以上のセキュリティ カテゴリをリソースのレコードに割り当てます。 newres コマンドまたは chres コマンドの category パラメータによって、セキュリティ カテゴリをリソースに割り当てます。

セキュリティ カテゴリのチェックで保護されているリソースに対して、ユーザのアクセスを許可するには、1 つ以上のセキュリティ カテゴリをユーザのレコードに割り当てます。 newusr コマンドまたは chusr コマンドの category パラメータによって、セキュリティ カテゴリをユーザに割り当てます。

セキュリティ カテゴリ チェックの有効化

セキュリティ カテゴリのチェックを有効にするには、以下の setoptions コマンドを実行します。

setoptions class+ (CATEGORY)

セキュリティ カテゴリ チェックの無効化

セキュリティ カテゴリのチェックを無効にするには、以下の setoptions コマンドを実行します。

setoptions class-(CATEGORY)

セキュリティ カテゴリの定義

セキュリティ カテゴリを定義するには、CATEGORY クラスでリソースを定義します。 セキュリティ カテゴリを定義するには、以下の newres コマンドを実行します。

newres CATEGORY name

name にはセキュリティ カテゴリの名前を指定します。

Sales というセキュリティ カテゴリを定義するには、以下のコマンドを入力します。

newres CATEGORY Sales

Sales および Accounts というセキュリティ カテゴリを定義するには、以下のコマンドを入力します。

newres CATEGORY (Sales,Accounts)

セキュリティ カテゴリの一覧表示

データベースで定義されているすべてのセキュリティ カテゴリのリストを表示するには、以下のような show コマンドを実行します。

find CATEGORY

セキュリティ カテゴリのリストが画面に表示されます。

セキュリティ カテゴリの削除

セキュリティ カテゴリを削除するには、CATEGORY クラスからそのレコードを削除します。 セキュリティ カテゴリを削除するには、以下の rmres コマンドを実行します。

rmres CATEGORY name

name にはセキュリティ カテゴリの名前を指定します。

「Sales」というセキュリティ カテゴリを削除するには、以下のコマンドを入力します。

rmres CATEGORY Sales

セキュリティ ラベル

セキュリティ ラベルは、特定のセキュリティ レベルと 0 個以上のセキュリティ カテゴリとの関係を表します。

セキュリティ ラベルのチェックを有効にすると、CA Access Control では他の権限チェックに加えて、セキュリティ ラベルのチェックが実行されます。 セキュリティ ラベルが割り当てられているリソースへのアクセスをユーザが要求すると、CA Access Control では、そのリソース レコードのセキュリティ ラベルに指定されているセキュリティ カテゴリのリストと、ユーザ レコードのセキュリティ ラベルに指定されているセキュリティ カテゴリのリストが比較されます。 リソースのセキュリティ ラベルに割り当てられたすべてのカテゴリがユーザのセキュリティ ラベルに含まれている場合、CA Access Control では、セキュリティ レベルのチェックが続行されます。含まれていない場合は、リソースに対するユーザのアクセスは拒否されます。 CA Access Control では、リソース レコードのセキュリティ ラベルに指定されているセキュリティ レベルと、ユーザ レコードのセキュリティ ラベルに指定されているセキュリティ レベルが比較されます。 ユーザのセキュリティ ラベルに割り当てられたセキュリティ レベルがリソースのセキュリティ ラベルに割り当てられたセキュリティ レベルと同じか、それより上である場合、CA Access Control では他の権限チェックが続行されます。リソースのセキュリティ レベルより下の場合は、リソースに対するユーザのアクセスは拒否されます。

セキュリティ ラベルのチェックが有効になっている場合、ユーザ レコードおよびリソース レコードに指定されているセキュリティ カテゴリとセキュリティ レベルは無視されます。セキュリティ ラベルの定義に指定されているセキュリティのレベルとカテゴリのみが使用されます。

セキュリティ ラベルのチェックによってリソースを保護するには、セキュリティ ラベルをリソースのレコードに割り当てます。 newres コマンドまたは chres コマンドの label パラメータによって、セキュリティ ラベルをリソースに割り当てます。

セキュリティ ラベルのチェックで保護されているリソースに対して、ユーザのアクセスを許可するには、セキュリティ ラベルをユーザのレコードに割り当てます。 newusr コマンドまたは chusr コマンドの label パラメータを使用して、セキュリティ ラベルをユーザに割り当てます。

セキュリティ ラベル チェックの有効化

セキュリティ ラベルのチェックを有効にするには、以下の setoptions コマンドを実行します。

```
setoptions class+(SECLABEL)
```

セキュリティ ラベル チェックの無効化

セキュリティ ラベルのチェックを無効にするには、以下の setoptions コマンドを実行します。

```
setoptions class-(SECLABEL)
```

セキュリティ ラベルの定義

セキュリティ ラベルを定義するには、SECLABEL クラスでリソースを定義します。 セキュリティ ラベルを定義するには、以下の newres コマンドを実行します。

```
newres SECLABEL name category(securityCategories) level(securityLevel)
```

各項目の説明:

name の端末から送信されたデータであるかのように、収集された監査ファイルに表示されます。

セキュリティ ラベルの名前を指定します。

securityCategories

セキュリティ カテゴリのリストを指定します。 複数のカテゴリを指定するには、カテゴリ名をスペースまたはカンマで区切ります。

securityLevel

セキュリティ レベルを指定します。 1 から 255 までの整数を使用します。

Managers というセキュリティ ラベルを定義し、Sales および Accounts というセキュリティ カテゴリとセキュリティ レベル 95 を指定するには、以下のコマンドを入力します。

```
newres SECLABEL Manager category(Sales,Accounts) level(95)
```

セキュリティ ラベルの一覧表示

データベースで定義されているすべてのセキュリティ ラベルのリストを表示するには、以下のような show コマンドを実行します。

```
find SECLABEL
```

セキュリティ ラベルのリストが画面に表示されます。

セキュリティ ラベルの削除

セキュリティ ラベルを削除するには、SECLABEL クラスからレコードを削除します。 セキュリティ ラベルを削除するには、以下の rmres コマンドを実行します。

rmres SECLABEL name

name にはセキュリティ ラベルの名前を指定します。

「Managers」というセキュリティ ラベルを削除するには、以下のコマンドを入力します。

rmres SECLABEL Manager

第 13 章: イベントの監査

このセクションには、以下のトピックが含まれています。

監査ルールの設定

CA Access Control では、セキュリティ監査のために、データベースに定義されている監査ルールに基づいて、アクセス拒否およびアクセス許可のイベントに関する監査レコードが保存されます。

すべてのアクセサおよびリソースに AUDIT プロパティがあり、このプロパティでは以下の 1 つ以上の値を設定できます。

FAIL

アクセサによるリソースへの失敗したアクセスをログに記録します。

SUCCESS

アクセサによるリソースへの成功したアクセスをログに記録します。

LOGINFAIL

アクセサによる失敗したすべてのログインをログに記録します。(この値はリソースには適用されません)。

LOGINSUCCESS

アクセサによる成功したすべてのログインをログに記録します。(この値はリソースには適用されません)。

ALL

アクセサの FAIL、SUCCESS、LOGINFAIL、および LOGINSUCCESS、またはリソースの FAIL および SUCCESS と同じ情報をログに記録します。

NONE

アクセサまたはリソースに関して、ログに何も記録しません。

TRACE

ALL と同じ情報およびすべてのシステム イベントをログに記録します。（この値はリソースには適用されません）。

データベースにアクセサまたはリソース レコードを作成または更新する場合はいつでも、AUDIT プロパティを指定できます。 また、ログに記録されたイベントを電子メールで通知するかどうか、通知する場合は誰に通知するかを指定することもできます。

監査ログのレコードは、これらの監査ルールに従って蓄積されます。 イベントをログに記録するかどうかは、以下のルールに基づいて決定されます。

- リソースまたはアクセサに AUDIT（ALL）が割り当てられている場合は、そのアクセサのすべてのログイン イベント、および CA Access Control によって保護されているリソースに関するすべてのイベントが、アクセスが失敗したか成功したかにかかわらず、ログに記録される。
- CA Access Control によって保護されているリソースへのアクセスが成功し、アクセサまたはリソースに AUDIT（SUCCESS）が割り当てられている場合は、イベントがログに記録される。
- CA Access Control によって保護されているリソースへのアクセスが失敗し、アクセサまたはリソースに AUDIT（FAIL）が割り当てられている場合は、イベントがログに記録される。

さらに、ユーザをトレース可能に設定した場合、そのユーザにトレース レコードが書き込まれるたびに、対応する監査レコードは監査ログに書き込まれます。

CA Access Control が監査ログに書き込む監査イベントの定義

CA Access Control では、アクセスの成功と失敗を監査ログに書き込みます。 CA Access Control が監査ログに書き込むアクセス イベントを定義するには、監査対象のリソースまたはアクセサの AUDIT プロパティの値を変更します。 また、CA Access Control では、この方法で、すべてのトレース イベントを監査ログに記録するように指定することもできます。

AUDIT プロパティを使用して CA Access Control が監査ログに書き込む監査イベントを指定します。 selang または CA Access Control エンドポイント管理 を使用して、以下のようにしてリソースおよびアクセサに AUDIT プロパティを設定します。

AUDIT の値	CA Access Control がログに記録する内容	適用可能なオブジェクト
FAIL	アクセスの失敗	ユーザおよびリソース
SUCCESS	アクセスの成功	ユーザおよびリソース
LOGINFAIL	ログインの失敗	ユーザ
LOGINSUCCESS	ログインの成功	ユーザ
ALL	FAIL、SUCCESS、LOGINFAIL、および LOGINSUCCESS に相当	ユーザおよびリソース
TRACE	ALL およびすべてのシステム イベントに相当	ユーザ
NONE	ログへの記録を行わない	ユーザおよびリソース

注: ユーザの監査プロパティが設定されていない場合は、グループまたはプロファイルグループの AUDIT 値は CA Access Control がユーザに対して使用する監査モードに影響する場合があります。

CA Access Control がユーザの監査モードを決定する方法

ユーザの監査モードでは、CA Access Control がそのユーザの監査ログに送信する監査イベントを指定します。 以下のプロセスでは、CA Access Control がユーザの監査モードを決定する方法について説明します。

1. CA Access Control は、USER クラスまたは XUSER クラスのユーザのレコードに AUDIT プロパティの値があるかどうかをチェックします。

   ユーザのレコードに AUDIT プロパティの値がある場合、CA Access Control はその値をユーザの監査モードとして使用します。

2. CA Access Control は、ユーザがプロファイル グループに割り当てられているかどうかをチェックします。 ユーザがプロファイル グループに割り当てられている場合、CA Access Control は GROUP クラスにある、そのプロファイル グループのレコードに AUDIT プロパティの値が含まれているかどうかをチェックします。

   ユーザがプロファイル グループに割り当てられていて、プロファイル グループのレコードに AUDIT プロパティの値がある場合、CA Access Control はその値をユーザの監査モードとして使用します。

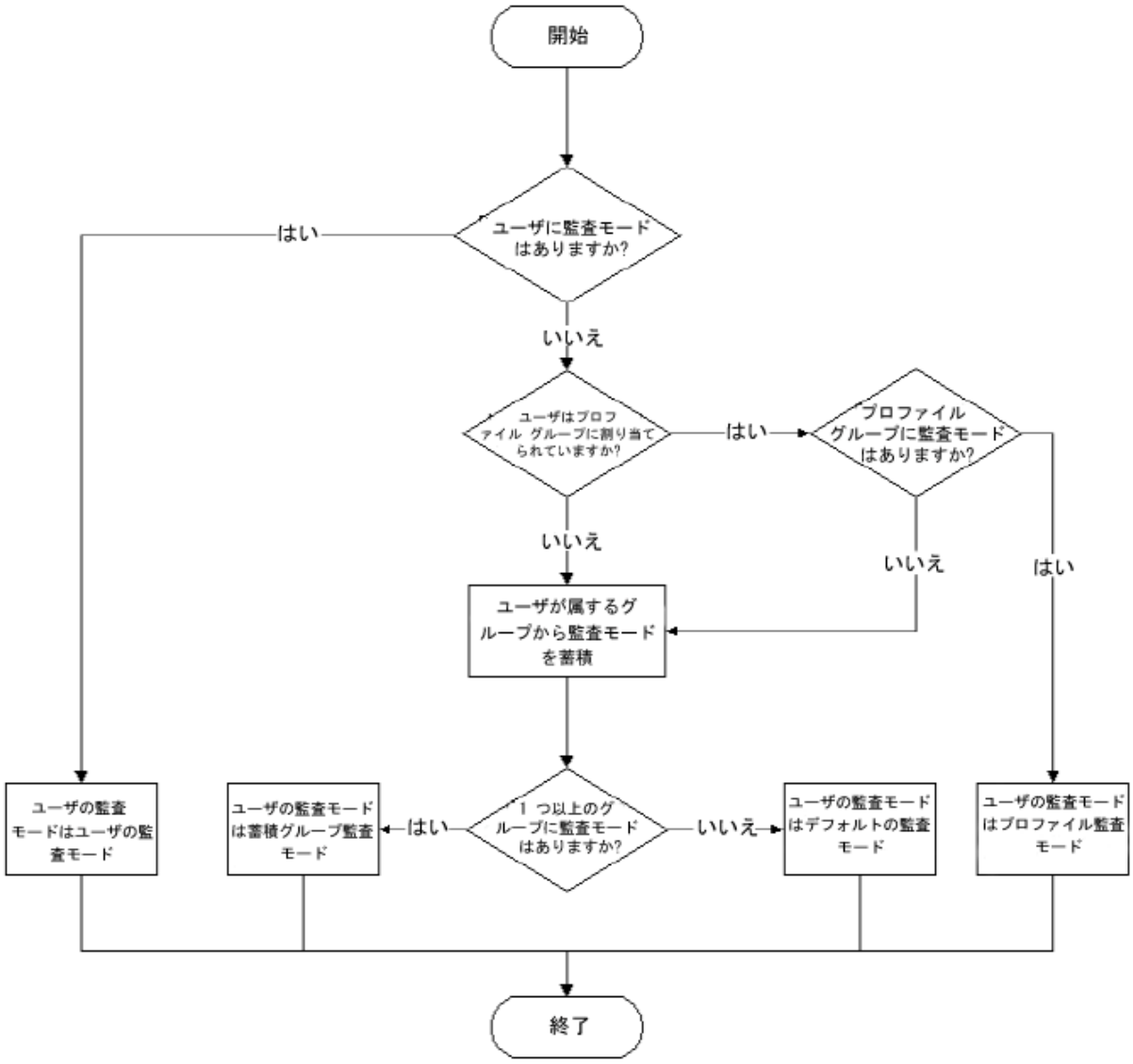
3. CA Access Control は、ユーザがグループのメンバであるかどうかを確認します。 ユーザがグループ メンバである場合、CA Access Control は GROUP または XGROUP クラスのグループのレコードに AUDIT プロパティの値があるかどうかをチェックします。

   ユーザがグループのメンバで、グループのレコードに AUDIT プロパティの値がある場合、CA Access Control はその値をユーザの監査モードとして使用します。 もしこのユーザがグループのメンバではないか、あるいはこのグループのレコードが AUDIT プロパティの値を持たない場合、 CA Access Control はシステム全体の監査モードをユーザに割り当てます。

   注: ユーザが複数のグループのメンバであり、グループごとに異なる監査モードがある場合、ユーザの監査モードは蓄積されます。 ユーザの監査モードは、メンバであるグループのすべての監査モードの合計です。

注: CA Access Control がグループの AUDIT プロパティの値を使用してユーザの監査モードを決定し、ユーザのログイン中にグループの監査モードを変更した場合は、ログイン中のユーザの監査モードも変更されます。 グループ監査モードの変更を有効にするためにユーザがログオフする必要はありません。

以下の図では、CA Access Control がユーザの監査モードを決定する方法について説明します。



例: グループ別監査

ユーザの Jan は、グループ A およびグループ B のメンバです。グループ A の監査モードは FAIL であり、グループ B の監査モードは SUCCESS です。Jan は両方のグループのメンバであるため、Jan には FAIL および SUCCESS の蓄積された監査モードがあります。

詳細情報:

CA Access Control がプロファイル グループを使用してユーザ プロパティを決定する方法(37 ページ)

ユーザおよびエンタープライズ ユーザのデフォルトの監査モード

ユーザ(USER オブジェクト)を作成すると、CA Access Control によってデフォルトの AUDIT_MODE がオブジェクトに割り当てられます。 AUDIT_MODE プロパティのデフォルト値は「Failure」、「SuccessLogin」、「SuccessFailure」です。

エンタープライズ ユーザ(XUSER オブジェクト)を作成すると、デフォルトで CA Access Control によってデフォルトの AUDIT_MODE 値がオブジェクトに割り当てられません。

注: (UNIX)USER オブジェクトの AUDIT_MODE プロパティのデフォルト値を変更するには、lang.ini ファイルの[newusr]セクションで、DefaultAudit の値を編集します。

警告モード

警告モードとは、リソースに適用できるプロパティであると同時に、クラスに適用できるオプションです。 警告モードがリソースまたはクラスに適用されている場合にアクセス ルールのアクセス違反が発生すると、CA Access Control は、リターン コード W を付けて監査ログにエントリを記録しますが、リソースへのアクセスは許可されます。 クラスが警告モードの場合は、そのクラス内のすべてのリソースが警告モードになります。

警告モードは、CA Access Control が Full Enforcement モードの場合のみ有効です。

注: Full Enforcement モードは、CA Access Control for UNIX がサポートする唯一のモードです。 CA Access Control for Windows では、Audit Only モードもサポートしています。

警告モードは、アクセス ポリシーを導入または変更する場合に使用できます。 警告モードを使用する場合は、ポリシーを有効にする前に、監査ログで対象となるポリシーの結果を事前に確認することができます。 監査ログを表示するには、seaudit コマンドを使用します。

クラスにプロパティ warning がある場合は、クラスを警告モードに設定できます。 リソース グループまたはクラスが警告モードの場合に、アクセス ルール違反が発生すると、CA Access Control は、アクセスを許可し、(リソース グループまたはクラスではなく)リソースを参照するエントリを監査ログに記録します。

リソースの警告モードの設定とクラスの警告モードの設定は独立しています。リソースを警告モードに設定した場合、そのリソースが属するクラスから警告モードを削除したとしても、そのリソースは警告モードのままとなります。

注: リソースまたはクラスを警告モードに設定できるのは、リソースまたはクラスにプロパティ warning がある場合だけです。必ずしもすべてのリソースまたはクラスにこのプロパティがあるわけではありません。

リソースの警告モードの設定

リソースを警告モードに設定することで、アクセス ルールを適用することなく、アクセス ルールの効果を監視できます。

注: 個々のリソースを警告モードに設定するだけでなく、クラスを警告モードに設定 (180 ページ)することもできます。

リソースを警告モードに設定するには、以下の手順に従います。

1. CA Access Control エンドポイント管理で、警告モードに設定するリソースを編集します。

   適切な[変更]ページが表示されます。

2. [監査]タブをクリックします。

   リソースに対する[監査モード]ページが表示されます。

3. [警告モード]を選択し、[保存]をクリックします。

   変更したリソースが警告モードになります。

注: 警告モードでは、アクセス ルール違反が発生した場合、アクセスは許可されますが、CA Access Control は必ず警告レコードを監査ログに記録します。このため、リソースの audit プロパティを設定する必要はありません。

sereport ユーティリティ(レポート番号 6)を使用すると、警告モードであるすべてのリソースが表示されます。

例: ファイルを警告モードに設定する

以下の selang の例では、ファイル c:¥myfile を警告モードに設定します。

```
chres FIlE c:¥myfile warning
```

例：ファイルから警告モードをクリアする

以下の selang の例では、ファイル c:¥myfile の警告モードを無効にします。

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

myfile の警告モードは無効になるので、CA Access Control は myfile に対するアクセス ルールを適用します。

例：端末を警告モードに設定する

以下の selang の例では、端末 myterminal を警告モードに設定します。

```
chres terminal myterminal warning
```

この場合、CA Access Control は権限のあるユーザによる端末 myterminal からのアクセスを許可しますが、その端末からのアクセスが通常拒否されるユーザについては監査レコードをログに記録します。

クラスを警告モードに設定する

個々のレコードを警告モードに設定するのではなく、クラス内のすべてのレコードを警告モードに設定することができます。 警告モードを使用することで、アクセス ルールを適用することなく、アクセス ルールの効果を監視できます。

クラスを警告モードに設定する方法

1. CA Access Control エンドポイント管理 で、以下の操作を実行します。
   a. ［設定］をクリックします。
   b. ［クラスのアクティブ化］をクリックします。

   ［クラスのアクティブ化］ページが表示されます。
2. ［警告］モードに設定するクラスの［警告］列のチェック ボックスをオンにします。
3. ［保存］をクリックします。

   確認メッセージが表示され、CA Access Control オプションが正常に更新されたことが通知されます。

警告モードが指定されたリソースの確認

CA Access Control を実装している場合は、警告モードを一時的な手段として使用する必要があります。 ユーザが必要とするリソースへの必要なアクセス権を持っていることを確認したら、警告モードをオフにします。CA Access Control は関連するルールの適用を開始します。

警告モードであるリソースを確認するために、警告モードであるすべてのリソースを示すレポートを作成できます。

レポートを作成するには、以下のコマンドを入力します。

```
sereport -r 6
```

CA Access Control によってレポートが作成されます。

注: sereport ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

警告モードであるクラスの確認

CA Access Control を実装している場合は、警告モードを一時的な手段として使用する必要があります。 ユーザが必要とするリソースへの必要なアクセス権を持っていることを確認したら、警告モードをオフにします。CA Access Control は関連するルールの適用を開始します。

警告モードであるクラスを確認するために、CA Access Control でこのデータを表示することができます。

このデータを表示するには、以下の selang コマンドを入力します。

```
setoptions cwarnlist
```

警告モードが指定されたクラスを示す表が表示されます。

注: setoptions の詳細については、「selang リファレンス ガイド」を参照してください。

監査ログ

監査レコードは、監査ログというファイルに格納されています。 監査ログの場所は、seos.ini ファイルで指定します。 seaudit ユーティリティまたは CA Access Control エンドポイント管理を使用して、監査ログに記録されたイベントを一覧表示したり、時間制限やイベント タイプなどでイベントをフィルタ処理したりすることができます。

注: seaudit の詳細については、「リファレンス ガイド」を参照してください。

監査ログはローカルに保存されていますが、CA Access Control のログ ルーティング機能を使用して監査情報を配布できます。 後でイベントを調査できるように、古い監査ログをテープにアーカイブすることをお勧めします。

デフォルトでは、認証デーモンである seosd によって、root 所有の監査ログが作成されます。これは、seosd プログラムがユーザ root で実行されるためです。 このため、作成される監査ログの読み取り/書き込み許可はユーザ root のみに与えられています。

root 以外のユーザが su コマンドで root にならなくても監査ログを参照できるように、CA Access Control の seos.ini ファイルには 2 つのエントリが用意されています。これらのエントリを使用して、ログ ファイルに割り当てるグループ所有者権限を指定します。

- 監査ログについて指定するエントリ。

  サイトの監査者全員が auditforce というグループのメンバであるとします。 ローカル監査ログ ファイルをこれらのユーザが表示できるようにする必要があります。 seos.ini ファイルを編集し、[logmgr]セクションの audit_group トークンを auditforce に設定します。 これにより、ローカル監査ログに対する読み取り許可が auditforce グループに与えられます。 この時点から、端末で作成されたローカル監査ログの所有者は auditforce グループになります。

  ログ ルーティング デーモンは、このトークンをクエリして、デーモンが作成して収集する監査ログに対するアクセス権を誰が持っているかを確認します。 監査ログは他のファイルと同じようにアクセス制御の対象であり、CA Access Control のルールによってユーザが監査ログにアクセスできない場合があることに注意してください。

- エラー ログ ファイルについて指定するエントリ。このエントリは、エラー ログ ファイルのグループ所有者権限を同様に指定します。

システム監査担当者

システム監査担当者は、AUDITOR 属性が割り当てられているユーザです。 システム監査担当者として定義されたユーザは、ユーザおよびリソースに割り当てられた監査属性の変更などの監査タスクを実行できます。

監査タスクは、中央から一括して実行できます。 監査担当者は、ログ ルーティング機能を使用して、1 台のホストからネットワーク上のさまざまな端末の監査情報を収集することができます。

ログ ルーティング機能のセットアップ

ログ ルーティングをセットアップするには、以下の手順に従います。

1. ログ ルーティング環境設定ファイルを作成します。

   seos.ini ファイルの RouteFile トークンで指定を変更しない限り、ACInstallDir/log/selogrd.cfg が CA Access Control のログ ルーティング環境設定ファイルとして使用されます。

   ここで、ACInstallDir は、CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です。

   ACInstallDir/samples/selogrd.init ディレクトリにサンプルのログ ルーティング環境設定ファイルがあります。 または、非常にシンプルなログ ルーティング環境設定ファイルとして、次の 3 行で構成されるファイルを作成できます。

   ```
   Rule
   host destination
   .
   ```

   destination には、監査レコードを受信するホスト名を入力します。 すべてのクラス、リソース、アクセサ、および結果がログに記録されます。

   注: 環境設定ファイルの構文の詳細については、「ユーティリティ ガイド」の selogrd ユーティリティに関する説明を参照してください。

2. 監査情報を転送するすべてのホストで送出デーモン(selogrd)を起動し、監査情報を収集するすべてのホストで収集デーモン(selogrcd)を実行します。

   注: これらのデーモンの使用の詳細については、「リファレンス ガイド」を参照してください。

ファイルの通知

ログ ルーティング機能を使用すると、ログを収集する以外に、ホストの表示画面、電子メール、またはその他の宛先に通知を送信することもできます。 端末自体の監査ログ、またはコレクタ デーモンによって収集され端末に転送されたログの情報に基づいて通知を送信できます。

このような通知を設定するには、ログ ルーティング環境設定ファイルおよび selang のコマンドを使用する必要があります。 たとえば、ユーザ root への setuid 要求が成功するたびに所有者のユーザ John に通知するとします。

1. 以下の selang コマンドを発行します。

   chres SURROGATE USER.root notify(John)

   この chres コマンドが指定するのは、ユーザが root になるための切り替え要求を誰かが実行するたびに、特別な監査ログ レコードが作成されて、seosd デーモンから John というユーザに通知を送ることです。 このデーモンは、通知レコードという特別な監査レコードも作成します。

2. 1 つ以上のリソースに対して通知を指定した後に、ログ ルーティング環境設定ファイルに以下の 3 行を追加できます。

   Rule2
   notify default
   .

   この行を追加すると、ログ ルーティング送出デーモンにより、通知監査レコードのメール メッセージが作成されます。

   注: 環境設定ファイルのフォーマットおよびログ ルーティング デーモンのセットアップの詳細については、「リファレンス ガイド」を参照してください。

ログ ルーティング

CA Access Control では、ログ ルーティング デーモンである selogrd を使用して、選択されたローカル監査ログ レコードを特定のホストに配布します。また、監査ログ レコードを電子メール メッセージ、ASCII ファイル、またはユーザ ウィンドウに再フォーマットし、監査済みイベントに基づいて通知メッセージを送信します。

監査レコードのルーティングを指定するには、selogrd は環境設定ファイル selogrd.cfg を使用します。 このファイルは、転送する(および転送しない)監査ログ レコードと、その転送先を指定したリストです。 このファイルの詳細については、「リファレンス ガイド」を参照してください。

ログ ルーティングの設定

seosd の起動時に selogrd または selogrcd を自動的に起動するには、[daemons]セクションにある seos.ini のトークン(selogrd または selogrcd)を yes に設定します。 これにより、seload を実行したときに、seload によってデーモンが起動されます。

たとえば、soes.ini の[daemons]セクションにある該当するトークンは以下のようになります。

selogrd = yes
selogrcd = yes

ログ ルーティング機能は RPC を使用して監査レコードを転送するため、ファイアウォールの内側にログ監査収集デーモンを配置すると、portmapper がサーバ デーモンに割り当てるポートを認識する手段が存在しなくなるため、UDP ポートを簡単に遮断できなくなります。 この問題を解決するために、ServicePort トークンを使用して、事前に定義されているポートをサーバ デーモンに割り当てることができます。

ネットワーク外部からのポート 111(portmapper のポート)への通信をファイアウォールで許可する場合は、サーバの seos.ini ファイルのみを変更してください。 保護されたネットワークで portmapper への通信をファイアウォールで許可しない場合は、クライアントおよびサーバの両方が同じ特定のポートを使用している必要があります。

これを確実に行うには、クライアントおよびサーバの両方の seos.ini ファイルの ServicePort トークンを同じ値に設定します。 数値(デーモンが指定されたポートにバインドされることを意味する)またはサービス名を指定できます。 サービス名を指定した場合は、クライアントおよびサーバの両方のサービス名解決が同じであることが必要です。 たとえば、サービス名 seoslogr を指定してから、クライアントおよびサーバの /etc/services ファイルに以下を追加する場合は、以下のようにします。

seoslogr 2022/udp # Audit log-routing

クライアントまたはサーバが、NIS を使用してサービス名を変換する場合は、NIS サービス マップを更新する必要があります。

監査ログ ルーティングの暗号化

監査ログ レコードを暗号化できます。 暗号化機能を使用すると、selogrd デーモンは、監査ログ レコードを暗号化してから収集デーモン(selogrcd または監査ログ ルータ)に送信します。 次に、収集デーモンは受信したレコードを復号化します。

CA Access Control には、selogrd に対して、CA Access Control 標準暗号化および adcipher による監査ログ暗号化という 2 種類の暗号化形式が用意されています。 暗号化では、seos.ini ファイルの[selogrd]セクションで指定されている共有ライブラリ オブジェクトの関数が使用されます。

標準暗号化では共有ライブラリ libcrypt が使用されますが、監査の暗号化では CipherName トークンに指定されているファイルの関数が使用されます。 デフォルトでは、このファイル名は adcipher です。これは、目的の共有ライブラリへのシンボリック リンクです。 CA Access Control のインストール時に、4 つの共有ライブラリ(lib1des、lib3des、libIDEA、および libblowfish)が CA Access Control の /lib ディレクトリに格納されます。

CA Access Control では共有ライブラリの標準暗号化鍵が保持されますが、監査の暗号化では KeyFile トークンに指定された個別のファイル(デフォルト値は adcipher.bin)が使用されます。

暗号化のタイプを決定するには、UseEncryption トークンを使用します。

- CA Access Control 標準暗号化を使用するには、UseEncryption=native を指定します。
- adcipher による監査ログ暗号化を使用するには、UseEncryption=eTrust を指定し、CipherName トークンおよび KeyFile トークンに適切な値を入力します。
- selogrd の暗号化を無効にするには、UseEncryption=no を指定します。

暗号化されていない監査を受け入れまたは拒否するには、RefuseUnencrypted トークンを使用します。 このトークンは UseEncryption トークンと一緒に使用されるので、UseEncryption が no に設定されている場合は重複します。

- 暗号化されていない監査を拒否するには、RefuseUnencrypted=yes を指定します。
- 暗号化されている監査と暗号化されていない監査の両方を受け入れるには、RefuseUnencrypted=no を指定します。

注: selogrcd デーモンは、seos.ini ファイルの同じトークンを使用します。

暗号化鍵を変更するには、この章で説明する sechkey ユーティリティを使用します。

重要: レコードを監査収集デーモンに送信する場合は、selogrd および収集デーモンの両方で同じ共有暗号化ファイルと暗号化鍵が使用されていることを確認してください。

電子メールによる監査ログ レコードの送信

selogrd を使用して、レコードを電子メールの宛先に直接送信することができます。 電子メールは、メーラ ユーティリティを使用して送信するか(従来の方法)、SMTP を使用してメール サーバに直接送信できます。

監査ログ レコードをメール サーバに直接送信するには、seos.ini ファイルの[selogrd]セクションに UseSmtpMail トークンを設定します。

以下の項目も指定できます。

- SmtpTimeLimit トークンを使用して、メール サーバが応答しない場合のタイムアウトを指定します。
- SmtpMailFrom トークンを使用して、「送信者:」メール ヘッダ フィールドを指定します。
- SmtpMailServer トークンを使用して、メール サーバのホスト アドレスを指定します。

注: 新しい方法では、UNIX メール ユーティリティを使用せず、メール サーバとの接続を直接確立し、SMTP プロトコルを使用してメールを送信します。

SNMP トラップの設定

インターネット ネットワーク管理プロトコルの SNMP (Simple Network Management Protocol) を使用するシステムの場合は、CA Access Control 監査レコードを使用して SNMP トラップを作成するように selogrd を設定できます。

SNMP トラップを実装するには、最初に CA Access Control ライブラリの SNMP 共有オブジェクトを検索し、次にこれらの共有オブジェクトを使用して selogrd を正しく設定します。

共有オブジェクト (通常、ACInstallDir/lib ディレクトリにあります) の名前は、snmp.xx および libsnmp.xx です。拡張子 xx はプラットフォームによって異なります。 有効な拡張子は、以下のとおりです。

- .o – AIX プラットフォーム
- .sl – HP プラットフォーム
- .so – その他のすべてのプラットフォーム

selogrd の SNMP 拡張機能を使用する場合、CA Access Control がデフォルトの場所にインストールされていなければ、selogrd を実行する前に以下の環境変数を設定する必要があります。

- AIX では、LIBPATH を ACInstallDir/lib に設定
- Solaris では、LD_LIBRARY_PATH を ACInstallDir/lib に設定
- Linux では、LD_LIBRARY_PATH を ACInstallDir/lib に設定
- HP では、SHLIB_PATH を ACInstallDir/lib に設定

ここで、ACInstallDir は、CA Access Control をインストールしたディレクトリです。

共有オブジェクトを使用するように selogrd を設定するには、以下の手順に従います。

1. ACInstallDir/etc/selogrd.ext というファイルを作成します。
2. ACInstallDir/etc/selogrd.ext ファイルに snmp.so の適切なパスを追加して、SNMP 共有オブジェクトの場所を定義します（この共有オブジェクトを指定するだけで、もう一方のオブジェクトは自動的にリンクされます）。以下に例を示します。

   ```
   snmp /opt/CA/AccessControl/lib/snmp.so
   ```

3. 最後に、selogrd.cfg ファイルを設定して、SNMP トラップのトリガとなるアクションのタイプ、および SNMP トラップのトリガ発生時に通知する宛先を指定する必要があります。 この設定は、他の監査通知の設定とよく似ていますが、snmp を送信システムとして指定します。

   たとえば、CA Access Control の起動時および停止時に SNMP トラップをアクティブにし、これらの SNMP トラップの通知を AuditPC に送信するとします。 これを行うには、以下のセクションを selogrd.cfg 環境設定ファイルに追加します。

   ```
   snmpRule
   snmp AuditPC
   include Class(START).
   include Class(SHUTDOWN)
   .
   ```

同様に、他のアクションまたはアクセスのタイプで SNMP トラップをアクティブにしたり、他の宛先に送信することもできます。

ユーザ トレース フィルタの移行

ユーザをトレース可能に設定した場合、そのユーザに関するトレース レコードが書き込まれるたびに、対応する監査レコードが seos.audit ファイルに書き込まれます。 CA Access Control の以前のリリースでは、これら監査レコードは trcfilter.init ファイルでフィルタされていました。 CA Access Control r12.0 SP1 以降から、ユーザ トレース レコードによって生成された監査レコードは audit.cfg ファイルによってフィルタされるようになりました。このフィルタは、他のすべての監査レコードをフィルタします。

監査レコード フィルタを trcfilter.init から audit.cfg に手動で移行する必要があります。フィルタを移行しない場合、ユーザ トレースによって生成された監査レコードはフィルタされなくなります。

注: トレース レコードは引き続き trcfilter.init でフィルタされます。 トレース フィルタは trcfilter.init から audit.cfg に移行しないでください。

ユーザ トレース フィルタを移行する方法

1. trcfilter.init で、移行する必要があるユーザ トレース フィルタを探します。

   このファイルの場所は、seos.ini ファイルの seosd セクションの trace_filter 設定でわかります。

2. audit.cfg に以下を入力します。usertracefilter は trcfilter.init から移行するユーザ トレース フィルタです。

   ```
   TRACE;*;*;*;*;*;usertracefilter
   ```

3. （オプション）移行する必要があるユーザ トレース フィルタごとに手順 1 ～ 2 を繰り返します。

例：ユーザ トレース フィルタの移行

この例では、以下のユーザ トレース フィルタが trcfilter.init ファイルにあります。

```
*ExampleFilter
```

このユーザ トレース フィルタを移行するには、audit.cfg ファイルの新しい行に以下を入力します。

```
TRACE;*;*;*;*;*;*ExampleFilter
```

第 14 章: 管理者権限の適用範囲

このセクションには、以下のトピックが含まれています。

グローバル権限属性

グローバル権限属性は、ユーザ レコードに設定します。 グローバル権限属性が設定されると、ユーザは特定の種類の機能を実行できます。 このセクションでは、各グローバル権限属性の機能および制限について説明します。

ADMIN 属性

ADMIN 属性により、ユーザは CA Access Control のほとんどすべてのコマンドを実行できます。 データベースで ADMIN 属性が割り当てられているユーザは、データベースのユーザ、グループ、およびリソースを定義および更新できます。 この属性は CA Access Control 内で最も強力な属性です。ただし、以下のような制限があります。

- データベース内で 1 ユーザのみに ADMIN 属性が割り当てられている場合は、そのユーザを削除できません。また、そのユーザのレコードから ADMIN 属性を削除することもできません。
- ADMIN 属性は割り当てられているが AUDITOR 属性は割り当てられていないユーザは、ユーザ、グループ、またはリソースに対して行われる監査の種類 (監査モード) を変更できません。 ADMIN 属性があり、ユーザ、グループ、またはリソースに関する監査特性を変更する必要がある場合は、AUDITOR 属性を自分自身に割り当てる必要があります。
- ADMIN 属性が割り当てられたユーザは、スーパーユーザ (UNIX の root アカウントまたは Windows の Administrator アカウント) を削除できません。ただし、スーパーユーザ を NOADMIN に設定することはできます。

AUDITOR 属性

AUDITOR 属性が割り当てられたユーザは、システムの使用状況を監視できます。 AUDITOR 属性が割り当てられたユーザの明示的な権限により、以下のことが可能です。

- データベース内の情報を表示できます。

  監査者は、selang のコマンド showusr、showgrp、showres、および showfile を実行できます。

- 既存のレコードに対して監査モードを設定できます。

  監査者は、selang のコマンド chusr、chgrp、chres、および chfile を実行できます。

OPERATOR 属性

OPERATOR 属性が割り当てられたユーザには、すべてのファイルに対する READ アクセス権があります。 このアクセス権により、データベース内のすべてのデータを一覧表示したり、バックアップ ジョブを実行できます。 オペレータは、showusr、showgrp、showres、showfile、および find の各コマンドを使用して、データベースのレコードを一覧表示できます。 OPERATOR 属性では、secons ユーティリティを使用することもできます。

注: secons ユーティリティの詳細については、「リファレンス ガイド」を参照してください。

PWMANAGER 属性

PWMANAGER 属性は、他のユーザのパスワードを変更するための chusr コマンドまたは sepass コマンドの使用権限を一般ユーザに付与します。

注: PWMANAGER によって ADMIN ユーザのパスワードを変更するには、setoptions コマンドの cng_adminpwd オプションを設定します。 詳細については、「selang リファレンス ガイド」を参照してください。

PWMANAGER 属性には、猶予ログイン回数、別のユーザのパスワード期間、または一般的なパスワード ルールを変更する権限は含まれていません。

PWMANAGER の権限には、showusr コマンドおよび find コマンドの使用権限も含まれます。

注: ユーザが nochngpass プロパティを yes に設定した場合、PWMANAGER ではそのユーザのパスワードを変更できません。

SERVER 属性

CA Access Control では、他の多くのセキュリティ モデルと同様に、一般ユーザによる「ユーザ A はリソース X にアクセスできるか」というクエリを許可していません。一般ユーザが可能な唯一のクエリは、「自分はリソース X にアクセスできるか」です。ただし、データベース サーバ サービスや社内アプリケーションのように、サービスを多数のユーザに提供するプロセスでは、他のユーザに代わって権限を照会することが許可されます。

SERVER 属性により、ユーザの権限をクエリするプロセスが許可されます。 SERVER 属性が割り当てられたユーザは、SEOSROUTE_VerifyCreate API を発行できます。

注: ERVER 属性および CA Access Control API の詳細については、「SDK 開発者ガイド」を参照してください。

IGN_HOL 属性

IGN_HOL 属性は、HOLIDAY レコードに定義されている期間中のログインをユーザに許可します。 HOLIDAY クラスの各レコードは、ログイン時に特別な許可が必要となる 1 つ以上の期間を定義します。 IGN_HOL 属性を持つユーザは、HOLIDAY レコードに定義されている期間に関係なく、いつでもログインすることができます。

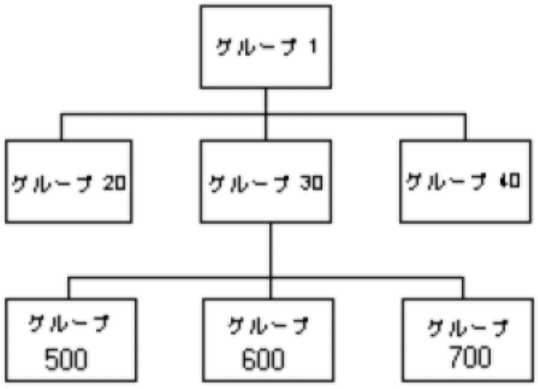
注: HOLIDAY クラスの詳細については、「リファレンス ガイド」を参照してください。

グループ権限

グループ権限属性を理解するには、親子関係の概念を理解しておく必要があります。

親子関係

下位グループと上位グループの概念は、親子関係ともいわれ、グループ管理者権限を説明する場合に重要です。 1 つのグループは、1 つ以上のグループの親（上位）になることができます。 1 つの子（つまり、下位グループ）に対して親として設定できるグループは 1 つのみです。 グループへの親の割り当ては、必要に応じて行います。 以下の図について考えてみましょう。



グループ 1 は、3 つのグループ（20、30、および 40）の親です。 グループ 30 も、3 つのグループ（500、600、および 700）の親です。 グループ 600 の親は 1 つのみ（グループ 30） です。 グループ 1 には親がありません。

グループ権限属性

リソース レコードおよびアクセサ レコードなど、すべてのレコードには所有者がいます。 レコードを「所有している」ということは、レコードを表示、編集、および削除する権限があることを意味します。

グループは、それぞれのレコードを所有できます。 ただし、レコードを所有するグループ内で、レコードを管理できるのは、特定の権限があるユーザのみです。 この特別なユーザには、グループ権限属性がそれぞれのユーザ レコードに設定されています。 グループ権限属性は、以下のとおりです。

- GROUP-ADMIN
- GROUP-AUDITOR
- GROUP-OPERATOR
- GROUP-PWMANAGER

これらの属性は、join コマンドによって設定されます。このコマンドは、正当な権限のあるユーザのみが発行できます。 join コマンドには、ユーザを 1 つのグループにまとめるという役割、およびユーザにグループ権限属性がある場合はその属性を指定するという役割があります。

グループのメンバを定義するユーザ レコードを管理する権限がグループの特権メンバに付与されるかどうかは、そのユーザ レコードの所有者によって決まります。

詳細情報:

所有者権限(197 ページ)

GROUP-ADMIN 属性

グループ管理者権限属性が割り当てられたユーザは、特定のレコードの集合を作成できます。 レコードを作成するために、グループ管理者はそのレコードの所有者を指定する必要があります。

レコードの所有者は、ユーザにグループ権限属性が設定されているグループである必要があります。 そのグループが他のグループの親である場合、所有者はその下位グループの 1 つでもかまいません。 これらのレコードの集合全体を「グループの有効範囲」といいます。 権限の例では、グループの有効範囲の概念を示します。

GROUP-ADMIN 属性が割り当てられたユーザには、グループの有効範囲内にあるレコードに対する以下のアクセス権限があります。

アクセス	説明	コマンド
読み取り	レコードのプロパティを表示します。	showusr、showgrp、showres、showfile
作成	データベースで新しいレコードを作成します。 所有者を指定する必要があります。	newusr、newgrp、newres、newfile
変更	レコードのプロパティを変更します。	chusr、chgrp、chres、chfile
削除	データベースからレコードを削除します。	rmusr、rmgrp、rmres、rmfile
接続	ユーザをグループに追加またはグループから分離します。	join、join

GROUP-ADMIN 属性には、以下のような制限事項もあります。

- GROUP-ADMIN ユーザは、自分に対してリソースをアクセス不可に設定できません。したがって、以下の制限を受けます。
  - GROUP-ADMIN ユーザは、自分のセキュリティ レベルより高いセキュリティ レベルを割り当てることはできません。
  - GROUP-ADMIN ユーザは、自分が所有していないセキュリティ カテゴリまたはセキュリティ ラベルを割り当てることはできません。
- GROUP-ADMIN ユーザは、データベースからスーパーユーザ(UNIX の root アカウントまたは Windows の Administrator アカウント)を削除できません。
- 以下に示すいくつかの制限は、この章の「グローバル権限属性」で説明しているグローバル権限属性に関連があります。
  - GROUP-ADMIN ユーザは、データベース内で唯一の ADMIN ユーザ レコードを削除できません。
  - GROUP-ADMIN ユーザは、データベース内の最後の ADMIN ユーザのレコードから ADMIN 属性を削除できません。
  - AUDITOR 属性のない GROUP-ADMIN ユーザは、監査モードを更新できません。 監査モードを更新できるのは、AUDITOR 属性が割り当てられた GROUP-ADMIN ユーザのみです。
  - GROUP-ADMIN ユーザは、どのユーザに対しても、グローバル権限属性(ADMIN、AUDITOR、OPERATOR、PWMANAGER、および SERVER)を設定できません。

GROUP-AUDITOR 属性

GROUP-AUDITOR 属性が割り当てられたユーザは、グループの適用範囲内にあるすべてのレコードのプロパティを一覧表示できます。 グループ監査者は、グループの適用範囲内のレコードに対して、監査モードを設定することもできます。

GROUP-OPERATOR 属性

GROUP-OPERATOR 属性が割り当てられたユーザは、グループの適用範囲内にあるすべてのレコードのプロパティを一覧表示できます。

GROUP-PWMANAGER 属性

GROUP-PWMANAGER 属性が割り当てられたユーザは、グループの有効範囲内にレコードがあるユーザのパスワードを変更できます。

所有者権限

データベースのすべてのレコード(アクセサ レコードおよびリソース レコードの両方)には、所有者が存在します。 レコードをデータベースに追加する場合は、owner パラメータを使用してレコードの所有者を明示的に割り当てるか、または CA Access Control によって、レコードを定義したユーザをレコードの所有者として割り当てることができます。

以下のいずれかが true の場合、アクセサがレコードを所有します。

- これらはレコードの所有者として定義されます。
- これらはレコードの所有者として定義されたグループのメンバであり、なおかつ GROUP-ADMIN 属性でグループのメンバとして追加されています。
- これらは、リソースがメンバとなっているリソース グループ レコードの所有者です。

レコードを所有するユーザまたはグループをデータベースから削除すると、そのレコードは所有者のないレコードになります。

レコードを所有するユーザには、所有するレコードに対して以下のアクセス権限があります。

Access	説明	コマンド
Read	レコードのプロパティを表示します。	showusr、showgrp、showres、showfile
Modify	レコードのプロパティを変更します。	chusr、chgrp、chres、chfile
削除	データベースからレコードを削除します。	rmusr、rmgrp、rmres、rmfile
接続	ユーザをグループに追加またはグループから分離します。	join、join-

ユーザまたはグループに特定レコードに対する所有者権限を与えない場合は、レコードおよびそのレコードがメンバとなっているすべてのリソース グループ レコードの所有者に対して nobody 強調を指定します。

所有者権限に関する制限事項は、以下のとおりです。

- データベース内の最後の ADMIN ユーザの所有者は、そのユーザ レコードを削除できません。
- AUDITOR 属性のない所有者は、監査モードを更新できません。 監査モードを更新できるのは、AUDITOR 属性が割り当てられた所有者のみです。
- スーパーユーザ(UNIX の root アカウントまたは Windows の Administrator アカウント)の所有者は、データベースから スーパーユーザを削除できません。

- 所有者は、自分が所有するユーザに対して、グローバル権限属性(ADMIN、AUDITOR、OPERATOR、および PWMANAGER)を設定できません。
- 所有者は、自分に対してリソースをアクセス不可に設定できません。従って、以下の制限を受けます。
  - 所有者は、自分のセキュリティ レベルより高いセキュリティ レベルを割り当てることはできません。
  - 所有者は、自分が所有していないセキュリティ カテゴリまたはセキュリティ ラベルを割り当てることはできません。

ファイルの所有者権限

CA Access Control では、FILE クラスにレコードを定義することによって、ファイルの所有者がファイルを保護することを許可します。 ファイルの所有者には、そのファイルのレコードに関するすべての権限があります。そのため、所有者はそのファイルを保護するレコードについて、newfile、chfile、showfile、authorize、および authorize- の各コマンドとすべてのパラメータを使用できます。

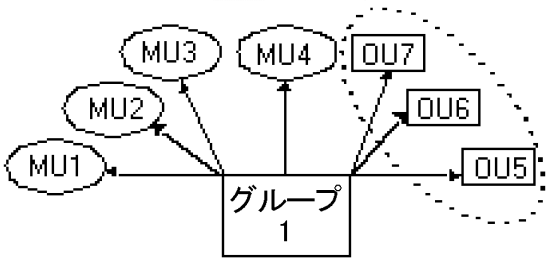
UNIX では、ユーザがファイルを作成すると、そのユーザがファイルの所有者に指定されます。 CA Access Control では、この機能が明示的に無効にされない限り、UNIX のファイル所有者による FILE レコードの定義が許可されます。 ファイル所有者による FILE レコードの定義を許可しない場合は、seos.ini ファイルの[seos]セクションにある use_unix_file_owner トークンを no に設定します(これはデフォルトの設定です)。

権限の例

グループ権限属性、親子関係、所有者権限、メンバシップ、およびグループの適用範囲の概念をこの後の図に示します。 これらの図にはユーザおよびグループしか含まれていませんが、所有者権限の概念はリソースおよびファイル レコードにも適用されます。

単一グループの権限

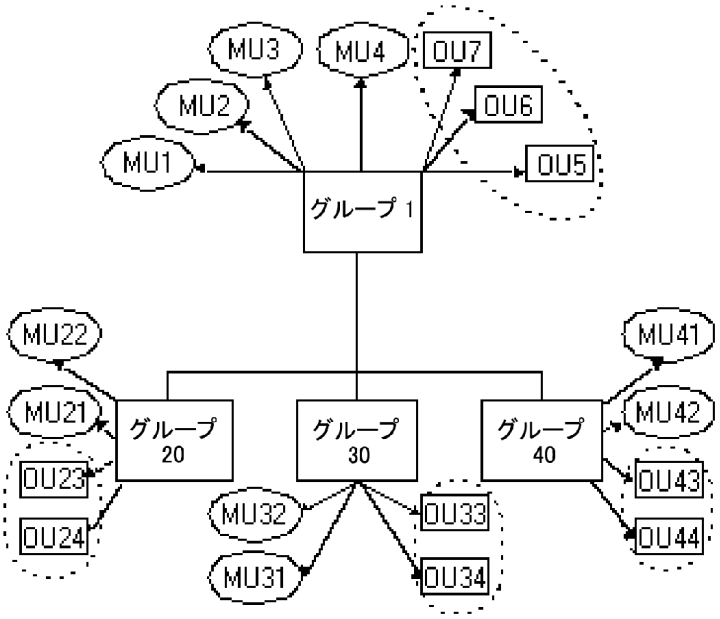
以下の図では、4 人のユーザ（MU1、MU2、MU3、および MU4）が グループ 1 のメンバです。 グループ 1 は、3 人のユーザ（OU5、OU6、および OU7）も所有しています。 メンバ MU4 には GROUP-ADMIN 属性が設定されています。



点線で囲まれた部分は、ユーザ MU4 が実行するコマンドの影響を受けるグループの適用範囲を示します。 この適用範囲には、グループ 1 が所有するすべてのユーザ（OU5、OU6、および OU7）が含まれます。

親グループおよび子グループ

以下の図では、4 人のユーザ(MU1、MU2、MU3、および MU4)が グループ 1 のメンバです。 グループ 1 は、3 人のユーザ(OU5、OU6、および OU7)も所有しています。 メンバ MU4 のレコードには、GROUP-ADMIN 属性が設定されています。



グループ 1 は、3 つのグループ(20、30、および 40)の親でもあります。 これらの下位グループにはそれぞれ、そのグループのメンバである 2 人のユーザと、そのグループが所有する 2 人のユーザがいます。

点線で囲まれた 4 つの部分は、ユーザ MU4 が実行するコマンドの影響を受けるグループの適用範囲を示します。 この適用範囲には、グループ 1 が所有するすべてのユーザと、グループ 1 の下位グループが所有するすべてのユーザが含まれます。 MU4 のグループの適用範囲に含まれるユーザは、OU5、OU6、OU7、OU23、OU24、OU33、OU34、OU43、および OU44 です。

仮に グループ 20、30、または 40 にも下位グループがあり、ユーザ、グループ、またはリソースを所有していた場合は、これらの下位グループが所有するレコードも、ユーザ MU4 が実行するコマンドの影響を受けるグループの適用範囲に含まれます。

サブ管理

セキュリティ管理者（ADMIN 属性が割り当てられたユーザ）は、一般ユーザに特定の管理者権限を与えることができます。 このような一般ユーザをサブ管理者といいます。 サブ管理者には、指定した CA Access Control のクラスまたはオブジェクトのみを管理する権限が与えられます。 たとえば、サブ管理者に、ユーザ オブジェクトとグループ オブジェクトのみを管理する権限を与えることができます。 また、クラスの特定のオブジェクトの管理者権限をサブ管理者ユーザに与えることによって、より高いレベルのサブ管理者を設定できます。

ユーザ、グループ、およびリソースのサブ管理者は、selang を使用して、これらのリソースに関連する管理タスクを実行できます。

特定の管理権限を一般ユーザに付与する方法

管理者、つまり ADMIN 属性が割り当てられたユーザは、CA Access Control のほぼすべてのアクションを実行できるため、特定の管理タスクをサブ管理者に委任したい場合があります。 この場合は、以下のように、CA Access Control データベースで、ユーザが実行する必要がある特定の管理タスクを制御するクラスに対する権限をそのユーザに付与する必要があります。

1. 委任するタスクを制御する 1 つ以上のクラスを識別します。

   たとえば、CA Access Control は、USER クラスと GROUP クラスを使用して、アクセサ リソースを作成します。 アクセサ管理を委任する場合は、ADMIN クラスの USER レコードと GROUP レコードを使用する必要があります。

2. 1 人以上のサブ管理者に、ADMIN クラスの該当リソースに対する権限を付与します。

   たとえば、サブ管理者がユーザ レコードを表示および変更できる権限を与えるには、そのユーザに、ADMIN クラスの USER レコードに対する読み取りアクセス権と変更アクセス権を付与します。

ADMIN クラス

ADMIN クラスのレコードのアクセス制御リスト（ACL）に指定されているユーザには、ADMIN 属性が割り当てられたユーザと同じ権限があります。 ただし、ADMIN クラスのレコードの ACL に指定されているユーザの権限は、そのレコードが示す特定のクラスに制限されます。 たとえば、ADMIN クラスの SURROGATE レコードでは、SURROGATE クラスのレコードを管理できるユーザが決定されます。

注: CA Access Control クラスの詳細については、「リファレンス ガイド」を参照してください。

ADMIN クラスにある特定レコードの ACL に指定されているユーザは、以下のコマンドを実行できます。

アクセス	説明	コマンド
読み取り	クラスのレコードのプロパティを表示します。	showusr、showgrp、showres、showfile、find
作成	クラスで新しいデータベース レコードを作成します。	newusr、newgrp、newres、newfile
変更	クラスのプロパティを変更します。	chusr、chgrp、chres、chfile
削除	データベースから既存のクラス レコードを削除します。	rmusr、rmgrp、rmres、rmfile
接続	ユーザをグループに追加したり、グループから除外したりします。 このアクセス権限は GROUP レコードの ACL でのみ有効です。	join、join
パスワード	データベース内の全ユーザのパスワードとその属性を管理します。 PWMANAGER 属性が割り当てられたユーザに許可されているアクセス権限と同じ権限が与えられます。 このアクセス権限は、USER レコードの ACL でのみ有効です。	chusr

ADMIN クラス権限を持つユーザには、以下の制限事項があります。

- ADMIN クラスにある USER レコードの ACL に定義されているユーザは、データベース内の最後の ADMIN ユーザを削除できません。
- ADMIN クラス ユーザは、自分が所有するユーザに対して、グローバル権限属性（ADMIN、AUDITOR、OPERATOR、および PWMANAGER）を設定できません。
- すべての ADMIN クラス ユーザが、監査モードを更新できるわけではありません。監査モードを更新できるのは、AUDITOR 属性が割り当てられた ADMIN クラス ユーザのみです。
- ADMIN クラスのユーザは、スーパーユーザ（UNIX の root アカウントまたは Windows の Administrator アカウント）を削除できません。ただし、スーパーユーザを NOADMIN に設定することはできます。
- ADMIN クラス ユーザは、自分に対してリソースをアクセス不可に設定できません。したがって、以下の制限を受けます。
  - ADMIN クラス ユーザは、自分のセキュリティ レベルより高いセキュリティ レベルをリソースに割り当てることはできません。
  - ADMIN クラス ユーザは、自分が所有していないセキュリティ カテゴリまたはセキュリティ ラベルを割り当てることはできません。

これらの制限は、B1 セキュリティ レベル認証の一部です。

環境に関する考慮事項

データベース内の情報を更新できるかどうかを制御する要因の 1 つとして、該当する環境でユーザが占めるポジションが挙げられます。

リモート管理の制限

管理者は、ネットワーク上のリモート端末にアクセスし、その端末のデータベースを更新できます。 リモート端末のデータベースを更新するには、管理者自身と管理者の端末の両方に許可が必要です。

- 管理者は、リモート端末のデータベースでユーザとして明示的に定義されている必要があります。 実行するコマンドの種類に関係なく、リモート端末のデータベースにある自分のユーザ レコードに、適切な属性が設定されている必要があります。
- リモート端末にアクセスするための WRITE 権限を与えるルールの中に、自分のローカル端末のニーズを明示的に記述する必要があります。記述がない場合、リモート端末での CA Access Control 管理を実行することはできません。

  デフォルトのアクセス フィールド(_default)または UACC クラスに WRITE 権限が設定されている場合は、リモート端末で selang のコマンド シェルを入力できます。 ただし、selang のコマンドを実行することはできません。また、リモート データベースにアクセスすることもできません。 READ 権限が設定されている場合、リモート端末にログインすることはできますが、その端末での CA Access Control 管理を実行することはできません。

  この WRITE 権限と READ 権限の違いの例を以下に示します。

  1. 新しい端末をデフォルトのアクセス権限に READ を使用して指定するには、以下のコマンドを発行します。この権限では、管理者はその端末からログインすることはできますが、データベースを操作することはできません。

     ```
     newres TERMINAL tty13 defacc(read)
     ```

  2. 新しい端末からデータベースを操作する権限を ADMIN1 というユーザに与える(つまり、WRITE 権限と READ 権限の両方を与える)には、以下のコマンドを発行します。

     ```
     authorize TERMINAL tty13 uid(ADMIN1) access(r,w)
     ```

UNIX 環境

UNIX でユーザおよびグループを管理する場合、CA Access Control でグローバル権限属性またはグループ権限属性が割り当てられたユーザには、CA Access Control の場合と同じ権限と制限が UNIX でも適用されます。

インストール時など、seosd デーモンが実行されて いない状態で selang を使用する場合は、以下のルールに従う必要があります。

- selang のコマンドに必ず -l オプションを指定すること。
- selang のユーザは root であること。 この排他的な root 権限は、UNIX の一般的な制限事項に準拠しています。

Windows 環境

Windows でユーザおよびグループを管理するため、CA Access Control でグローバル権限属性またはグループ権限属性が割り当てられたユーザには、CA Access Control の場合と同じ権限と制限が Windows でも適用されます。

インストール時など、seosd デーモンが実行されて いない状態で selang を使用する場合は、以下のルールに従う必要があります。

- selang のコマンドに必ず -l オプションを指定すること。
- selang のユーザは、Administrators グループに属する Administrator であること。

第 15 章: パフォーマンスの向上

このセクションには、以下のトピックが含まれています。

Global Access Check の使用

頻繁に開かれる保護されたファイルがあり、そのファイルのアクセス ルールがほとんど変更されない場合は、Global Access Check 機能 (GAC) を使用すると、ファイルへのアクセス速度が向上します。

GAC を使用すると、CA Access Control の管理者は read、write、chown、chmod、rename、unlink、utimes、chattr、link、chdir、create、および all のルールをキャッシュすることができるため、seosd に制御を渡すことなく、ファイルに対する適切なアクセスが許可されます。 デフォルトは all です。 ただし、実行要求は、セキュリティ ホールになる可能性があるため、GAC による処理には適していません。

GAC を使用しない場合は、ユーザまたはプログラムが保護対象ファイルにアクセスを試みるたびに、CA Access Control による完全なセキュリティ チェックが実行されます。 アクセス頻度の高いファイルの場合は、アクセス権限を確認するために詳細なチェックが何度も必要になります。

GAC を使用すると、CA Access Control の管理者は、アクセス頻度の高い特定の保護対象ファイルに対して簡略化されたセキュリティ チェックを許可できます。 CA Access Control の管理者は、簡略化されたチェックに適するファイルを選択できます。 簡略化されたセキュリティ チェックを許可する前に、設定したルールに基づいてファイルの完全なセキュリティ チェックを実行する必要があります。 ルール自体は、一般的なファイル名およびアクセス権のリストで構成されます。 ルールは、ユーザに応じてキャッシュされます。

特定のファイルだけを対象に行う簡略化されたセキュリティ チェックの信頼性は十分です。これは、保護対象ファイルに関するルールが変更されると、GAC 機能によって簡略化されたセキュリティ チェックのテーブルがフラッシュされ、元の完全なセキュリティ チェックが実行されるからです。

注: GAC の制限事項により、この機能は root 以外のすべてのユーザに適用されます。

GAC の機能

指定したファイルへのアクセスは、CA Access Control によって監視され、許可されたアクセスのテーブルが実行時に作成されます。 これらのファイルは、GAC のルールを設定するためにあらかじめ指定するファイルです。

CA Access Control では、特定のファイルに対する特定レベルのアクセスをユーザに許可することが決定すると、さらに次の 2 つの条件が満たされているかどうかがチェックされます。

- 許可されたアクセスが無条件であること(つまり、日時、実行するプログラム、その他同様の条件に依存しないこと)。
- あらかじめ選択されたファイル マスクの 1 つとファイルが一致すること。

注: ファイル ルールは、ファイルへのアクセスの許可を定義します。

これらの条件が満たされると、CA Access Control によって UID、ファイル ルール、およびアクセス権の 3 つの情報が 1 組にまとめられ、これらの情報項目で構成されるテーブルに保存されます。 このテーブルは、他のデータベース アクセス ルールが解釈される前にチェックされます。 ユーザがファイルへのアクセスを試みるたびに、フィルタ処理メカニズムとしてこのテーブルがクエリされます。

このテーブルには、一度認識された後はアクセス許可チェックが不要なファイル マスクのリストが格納されています。そのため、このテーブルは、do-not-call-me テーブルといいます。 また、アクセスがファイル マスク リスト内で指定されたファイルに常に付与されるので、always-grant テーブルともいいます。

ユーザがファイルへのアクセスを試みるたびに、このテーブルがクエリされます。 ファイルがテーブル内の 3 つの情報項目の 1 つと一致した場合は、seosd に制御を渡すことなく、適切なアクセスが許可されます。 これにより、アクセス ルールの分析は省略されます。 その後は、テーブルに格納されている 3 つの情報項目に基づいて、このパターンに一致するファイルへのアクセスはすべて許可されます。アクセス ルール データベースはクエリされません。

データベースに新しいアクセス ルールが追加されるたびに、テーブル全体がフラッシュされ、新しいルールに関する学習プロセスが最初から開始されます。

GAC の実装

GAC をセットアップするには、頻繁にアクセスされるファイルのマスクを選択し、選択したファイル マスクを格納する GAC ファイルをセットアップする必要があります。その後でキャッシュ プロセスを開始します。

GAC ルールの設定

注: データベース内のファイル ルールは、FILE クラスのパラメータおよびファイル マスクを使用して作成します。 ルールは、ファイル マスクに一致するすべてのファイルに適用されます。 FILE クラスのアクセス タイプには、all、chdir、control、create、delete、execute、none、read、rename、sec、update、utime、write があります。

データベースに定義されているファイル ルールから、キャッシュするファイル マスクを選択します。 ACInstallDir/etc/GAC.init ファイルにファイル マスクのリストを入力します（ここで、ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です）。データベースに格納されている形式と同じ形式で入力します。

各マスクは別々の行に指定する必要があります。 たとえば、データベースに格納されている /tmp/mydir/* というファイル マスクをキャッシュする場合は、ACInstallDir/etc/GAC.init ファイルに以下の行を追加します。

```
/tmp/mydir/*
```

注: GAC.init ファイル内に特定のファイル名を指定することはできません。 指定できるのは、ファイル マスクのみです。

GAC の開始

CA Access Control の現在のバージョンを GAC 互換バージョンにするには、キャッシュ対象のファイル マスクが指定された ACInstallDir/etc/GAC.init ファイルを用意します。 このファイルに指定できるのはファイル マスクのみです。

ACInstallDir/etc/にある 1 行のみの GAC.init というファイルの例を示します。

```
/IBBS/REL63/*
```

GAC の制限事項

ファイルへのアクセス件数が毎秒数百にも及ぶ場合は、GAC を実装するとパフォーマンスが向上します。このことはすでに立証されていますが、以下のような制限事項があります。

- デフォルトでは、GAC のルールは root ユーザ(通常は ADMIN)に適用されません。 root ユーザにルールを適用できるようにするには、seos.ini ファイルの[SEOS_syscall]セクションにある以下のトークンを設定します。

  ```
  GAC_root=1
  ```

  このトークンのデフォルト値は 0(ゼロ)です。 デフォルトに戻すには、トークンを 0(ゼロ)に設定するか、このトークンを削除します。

- 条件付きで保護されるファイル ルール(日付や時間帯の制限、Program Pathing など)をテーブルに含めることはできません。 GAC.init ファイルにこのようなファイル ルールを指定しても、日付や時間の制限およびその他の制限は適用されません。

- audit(ALL)属性または audit(success)属性を指定したファイル ルールを GAC.init ファイルに含めることはできません。 GAC.init ファイルにこのようなファイル ルールを指定した場合、成功したアクセスの監査は記録されません。

- フィルタ処理プロセスでは、実際の（現在の）UID（つまり、実行時のプロセスに関連付けられた UID）が使用されます。 これは、現在の UID ではなく元の UID（ユーザがログイン時に使用した UID）を CA Access Control で追跡する場合にループホールとなります（CA Access Control では、UID 使用状況の追跡を実行して、責任の所在をいっそう明確にするセキュリティを実現しています）。

  ここでは例を挙げて、このループホールがどのように悪用されているかを説明します。ユーザ Tony には、Accounts/tmp ファイルへのアクセス権がありません。 そのため Tony は、Accounts/tmp へのアクセスを許可されているユーザ Sandra の代理になります（/bin/su を使用します）。 Sandra がすでに Accounts/tmp ファイルにアクセスしている場合、このファイルは Sandra の UID と共に do-not-call-me テーブルに追加されています。 したがって、Sandra の UID を使用している Tony にもファイルへのアクセスが許可されます。 これは、カーネル コードに UID の履歴が保持されないためです。

  一方、Sandra がそれまでにファイルにアクセスしていない場合は、seosd を使用して通常どおりアクセス許可のチェックが行われ、Tony のファイルへのアクセスは拒否されます。 このようなループホールを防止するために、ADMIN ユーザはデータベースの SURROGATE オブジェクトを保護する必要があります。 この例の場合、ADMIN ユーザは以下のルールをデータベースに追加できます。

  ```
  newres SURROGATE USER.Sandra default(N) owner(nobody)
  ```

  このコマンドを実行すると、Tony が su コマンドを使用して Sandra のアクセス権限を取得することはできなくなります。

- アクセサが root の場合、キャッシュ メカニズムは影響しません。 これは、データベースをクエリしなければ root にアクセス許可が与えられないためです。

GAC のトラブルシューティング

GAC が機能しているかどうかを確認するには、以下の手順で GAC をテストします。

1. トレースを有効にします（secons -t+）。
2. GAC.init に指定したファイル マスクの 1 つに対応するファイルにアクセスします。最初のアクセスはトレースで報告されます。
3. そのファイルに再度アクセスを試みます。 2 度目のファイル アクセスはトレースに記録されません

   記録された場合は、GAC が機能していません。 GAC.init をチェックして、フォーマットが適切なことを確認します。

リソース キャッシュの使用

CA Access Control には、リソース キャッシュ機能（ファイル キャッシュ）というパフォーマンス向上ツールも用意されています。

キャッシュにより、FILE クラスのリソースについての承認要求に対する以前の応答（許可または拒否）が「記憶」されます。 結果は、ファイル名、ユーザ名、および承認応答（アクセス モード、プログラム名、結果）と共に保存されます。 同じ承認が要求されると、その要求はキャッシュ メモリ テーブル内に格納された前回の応答を使用して回答されます。 これにより、CA Access Control では要求を再評価する必要がなくなるため、時間が節約されます。つまり、CA Access Control はすばやく応答を返すことができます。ルールが変更されると、キャッシュは自動的かつ速やかに同期されます。

キャッシュはランタイム テーブルです。 管理者は、以下の 2 つの方法でこのテーブルを構成できます。

- seos.ini ファイルで初期設定パラメータを設定します。
- 実行時にキャッシュを ON または OFF に切り替えてパラメータを変更します。

セキュリティ管理者は、seos.ini ファイルのトークンを使用して、テーブルのサイズ、テーブルを消去する間隔、他の内部テーブル パラメータを定義できます。

管理者権限を持つユーザは、キャッシュ テーブルを ON または OFF に切り替えて、キャッシュ パラメータを変更し、標準出力にキャッシュ テーブルを書き込むことができます。

注: secons ユーティリティまたは seos.ini 初期設定ファイルの[seosd]セクションの詳細については、「リファレンス ガイド」を参照してください。

チューニングの推奨事項

これらの推奨事項を実行すると、パフォーマンスがさらに向上します。

- 3 つのテーブル（プール）の 1 つにレコードの最大数が設定されていて、別のテーブルには設定されていない場合、テーブル全体のサイズを拡張します。

  注: 3 つのテーブルとは、ファイル、ユーザ、および権限のテーブルです。

  プールが低い値に設定されている場合、設定値を増やしてプールを拡張します。
- 必要な場合を除き、最大サイズのトークンを設定しないようにします。 テーブルが大きくなると、レコードのスキャンにより多くの時間がかかります。

ネットワーク キャッシュの使用

ネットワーク キャッシュまたは IP キャッシュ機能により、受け取った TCP 着信要求はキャッシュされ、データベースには送信されません。代わりに、これらの要求は syscall 関数で自動的に許可されます。 この機能により、多くの TCP 着信接続を起動するホストのパフォーマンスが向上します。

IP キャッシュ機能を有効にするには、seos.ini ファイルの[seosd]セクションにある以下のトークンを変更して、CA Access Control を再起動します。

network_cache_timeout

キャッシュ テーブルを消去する頻度を定義します。 このトークンは、受け入れ要求の時間制限を設定する場合に重要です。

UseNetworkCache

このトークンを yes に設定して、IP キャッシュ機能を有効にします。

キャッシュ機能が有効になると、受け取ったすべての TCP 接続がカーネル テーブルに保存されます。 レコードは、ピア IP アドレス、ピア ポート、ローカル ポートで構成されます。 新しい接続はすべて、このキャッシュ内で検索されます。 IP アドレス、IP ポート、およびローカル ポートが一致するデータのセットが存在する場合、接続はすぐに許可されます。 これにより、接続を確立する時間が短縮されます。

実在パス キャッシュの使用

ファイルの名前解決は、CA Access Control がファイル システムの情報を使用するため、処理時間が長くなります。 CA Access Control のカーネルは、適切なイベントがインターセプトされたときに、ノード番号を完全なファイル名に変換します。 実在パス キャッシュ機能は、ファイル名を内部テーブルに保存します。

この機能を有効にするには、seos.ini ファイルの[SEOS_syscall]セクションにあるトークン cache_enabled を 1 に設定します。 ファイル名は、データのペア(i-node 番号とデバイス番号)と共にテーブル内にキャッシュされます。

注: seos.ini 初期設定ファイルの詳細については、「リファレンス ガイド」を参照してください。

fork 同期の使用

seos.ini ファイルの[SEOS_syscall]セクションにある fork 同期トークン (synchronize_fork)は、新規プロセス作成時の fork イベントの動作を管理します。 このトークンの値を小さくすると、fork イベントが頻繁に発生するため、パフォーマンスが向上します。

注: seos.ini 初期設定ファイルの詳細については、「リファレンス ガイド」を参照してください。

高優先順位の使用

CA Access Control には、一部のプラットフォーム上で seosd デーモンにリアルタイムの優先順位を設定するオプションが用意されています。 この機能を有効にするには、seos.ini ファイルの[seosd]セクションにある rt_priority トークンを yes に設定します。 リアルタイムで実行すると、システム パフォーマンスが向上します。

注: seos.ini 初期設定ファイルの詳細については、「リファレンス ガイド」を参照してください。

プロセス ファイル システムの省略

システムの負荷を軽減するために、ファイルがプロセス ファイル システム(/proc)に属している場合に CA Access Control でファイル アクセスをチェックする必要があるかどうかを指定できます。

この機能を有効にするには、seos.ini ファイルの[SEOS_syscall]セクションにある proc_bypass トークンを使用します。 このトークンには、CA Access Control がプロセス ファイル システムにアクセスするたびに省略されるアクセス情報が格納されます。

注: seos.ini ファイルのトークンの詳細については、「リファレンス ガイド」を参照してください。

このトークンは、アクセスの総数として設定できます。 アクセス値は次のとおりです。

- 1 – read
- 2 – write
- 4 – chown
- 8 – chmod
- 16 – rename

- 32 – unlink
- 64 – utimes
- 128 – chattr
- 256 – link
- 512 – chdir
- 1024 – create

たとえば、proc_bypass=513 は、read(1)および chdir(512)へのアクセスの試みをすべて確認しないことを指定します(1 + 512 = 513)。

実在パスの省略

ファイルの相対パスではなく、ファイルの絶対パスでファイルを検索すると、システムの負荷が高くなります。ただし、この検索を省略することにより、ファイル イベントが高速になります。

この省略を有効にするには、seos.ini ファイルの[SEOS_syscall]セクションにあるトークン bypass_realpath を 1 に設定します このトークンを有効にした場合、CA Access Control は、実際のファイル名を取得しません。これは、たとえば、シンボリック リンクになる場合があります。

注: seos.ini ファイルのトークンの詳細については、「リファレンス ガイド」を参照してください。

重要: この機能は、セキュリティに影響を与えるため、慎重に使用する必要があります。相対パスを使用してファイルにアクセスする場合、包括的なルールは機能しません。

trusted プロセス承認の省略

CA Access Control では、プログラムを trusted プログラムとして定義できます。 trusted プログラムとその子プログラムは 1 つのテーブルに格納されます。 trusted プロセス(およびその trusted プロセスに対応するポート)に関連するすべてのイベント(受信イベントおよび送信イベント)は、完全なネットワークの省略の一部として承認なしで許可されます。

これらのプログラムを指定するには、SPECIALPGM クラスを以下のように使用します。

- 指定したプログラムのファイル イベントおよびネットワーク イベントを省略するには、PGMTYPE プロパティを使用して pbf 値および pbn 値を指定します。
- 指定したプログラムの setuid イベントおよび setgid イベントを省略するには、PGMTYPE プロパティを使用して surrogate 値を指定します。
- 指定したプログラムから呼び出されるすべてのプログラムに省略を伝達するには、PGMTYPE プロパティを使用して propagate 値を指定します。

  注: セキュリティ権限の伝達は、PBF、PBN、DCM、および SURROGATE 権限の場合にのみ有効です。

ネットワーク アクティビティ ポートのバイパス

CA Access Control による認証を行わずに特定の TCP/IP ポートに関連するすべての接続イベント(受信および送信)を確立できることを指定するには、これらのポートのバイパスを指定します。 これらのポートをバイパスすると、システム負荷が軽減され、イベント処理が高速化されます。 バイパスされた接続イベントは、監査ログおよびトレース ログに記録されません。

注: CA Access Control では、ネットワーク接続イベントのみをバイパスできます。ネットワーク接続を使用するそれ以降のイベント(ファイルのオープンなど)はバイパスできません。

trusted 受信接続は、送信接続とは別に指定されます。

- 受信接続をバイパスするには、seos.ini ファイルの[seosd]セクションにある bypass_TCPIP 設定を変更します。
- 送信接続をバイパスするには、seos.ini ファイルの[seosd]セクションにある bypass_outgoing_TCPIP 設定を変更します。

注: 初期設定ファイル、トークンの更新、および影響を及ぼす変更事項の詳細については、「リファレンス ガイド」を参照してください 。

例: 受信 Telnet イベントの省略

bypass_TCPIP 設定を 23(Telnet ポート)に設定すると、ワークステーションへの Telnet 接続時に、監査ファイルおよびトレース ファイルがネットワーク イベントをログに記録しないようになります。 ssh、login、および FTP などのほかのサービスに関連するイベント、およびネットワーク接続を使用する以降のイベント(ファイルのオープンなど)は、引き続きログに記録されます。

例: 送信 FTP イベントのバイパス

bypass_outgoing_TCPIP 設定を 21 (FTP ポート) に設定すると、ワークステーションからの FTP 接続時に、監査ファイルおよびトレース ファイルがネットワーク イベントをログに記録しないようになります。 ssh、login、および Telnet などのほかのサービスに関連するイベント、およびネットワーク接続を使用する以降のイベント (ファイルのオープンなど) は、引き続きログに記録されます。

監査およびトレースの負荷の軽減

CA Access Control では、ファイル システムを使用して、監査データおよびトレース データを保持します。 システム内のほとんどのプロセスは、CA Access Control がこのファイル システムに書き込んでいる間、ブロックされます。 このファイル システムへのアクセス時間を短縮するには、以下の操作を行います。

- 必要なリソースおよびアクセスのみに監査モードを設定します。
- 必要な場合にのみ、トレースを開きます。
- 処理速度が最も速いファイル システムに、監査ファイル、トレース ファイル、CA Access Control データベース ファイルを格納します。
- 処理速度が速いファイル システムに、lookaside データベース ディレクトリを格納します。

データベースの負荷の軽減

データベースにルールを定義する方法は、システム パフォーマンスに影響を与えます。

- 多くの検証は、一般的に使用されるディレクトリに対する包括的なルールに基づいて行われます。その結果としてシステム負荷が高くなります。

  たとえば、/usr/lib/* を保護すると、システム内のすべての操作が CA Access Control によってチェックされます。 パフォーマンスを向上させるには、頻繁に使用するファイルに対して包括的なルールを適用しないようにします。
- ユーザおよびリソースの階層が深い場合、すべての依存関係を取得およびチェックするにはシステム負荷がかかります。 パフォーマンスを向上させるには、データベース内で深い階層を使用しないようにします。

PMDB 更新の改善

Policy Model は、そのサブスクライバに対して 1 つのループ内で 1 つずつコマンドを送信します。 Policy Model が各ループ内で各サブスクライバに送信するコマンドの最大数を制御するには、updates_in_chunk トークンを使用します。このトークンについては、付録「pmd.ini ファイル」の[pmd]セクションを参照してください。

このトークンの値を大きくすると、Policy Model でコマンドを送信するために使用されるサイクルが少なくなります。 ループが終了するたびに、Policy Model は新しい要求をチェックします。 トークンの値を大きく設定すると、Policy Model が新しい要求をチェックする頻度は減少します。

たとえば、(sepmd -n オプションを使用して)新しいサブスクライバを Policy Model に追加する場合、Policy Model が送信するコマンドは他のサブスクライバがすでに受け取っているため、トークンの値を大きく設定します。 Policy Model では、他のサブスクライバへのコマンドの送信時間は短くなり、新しいサブスクライバへのコマンドの送信時間は長くなることにより、サブスクライバの追加にかかる時間が短縮します。

注: このトークンには 100 より大きい値を設定しないでください。

Watchdog のパフォーマンスの向上

システム負荷を軽減するには、保護対象ファイルを常にスキャンするのではなく、定期的にスキャンするように Watchdog デーモン(seoswd)を設定します。 システム負荷が小さいときにスキャンを実行するように Watchdog を設定できます。

この機能を有効にするには、seos.ini ファイルの[seoswd]セクションにある IgnoreScanInterval トークンを使用し、追加のトークンでスキャンの間隔と開始時刻を設定します。

注: これらのトークンの詳細については、「リファレンス ガイド」の seos.ini 初期設定ファイルの説明を参照してください。

Class パラメータの機能向上

CA Access Control でクラスのアクティブ化機能およびクラスの権限付与機能を使用すると、パフォーマンスがさらに向上します。

クラスのアクティブ化

CA Access Control には、CLASS がデータベース内でアクティブまたは非アクティブのいずれであるかに関する情報が格納されます。 CA Access Control を起動すると、アクティブなクラスのリストが SEOS_syscall に渡されます。したがって、CA Access Control が常にこれらのクラスをインターセプトする必要はありません。 CA Access Control がクラスをインターセプトするのは、ユーザがクラスのアクティビティ ステータスを変更した場合のみです。 クラスがアクティブでない場合、リソースへのアクセスはインターセプトされません。

FILE、HOST、TCP、CONNECT、および PROCESS クラスについては、アクティブでないクラスのインターセプトを省略できます。

クラスの権限付与

リソース クラス SEOS は、CA Access Control 権限付与システムの動作を制御します。SEOS クラスには、クラスがアクティブかどうかを指定する変更可能なプロパティがあります。（setoptions コマンドを使用して）未使用のクラスを無効にし、権限付与に要する時間を短縮できます。

名前の解決

seos.ini ファイルの［seosd］セクションにある複数のトークン（GroupidResolution、HostResolution、ServiceResolution、UseridResolution など）は、CA Access Control による名前解決の実行方法を制御します。 これらのトークンを適切に設定すると、パフォーマンスが向上します。

または、（システムの名前解決を実行する代わりに）lookaside データベースを作成できます。 パフォーマンスを向上させるには、lookaside データベース オプションを選択します。 この機能に関するトークンには、lookaside_path や use_lookaside などがあります。

注: これらのトークンの詳細については、「リファレンス ガイド」の seos.ini 初期設定ファイルの説明を参照してください。

UID をユーザ名に、GID をグループ名に、IP アドレスをホスト名に、およびポート番号をサービス名にそれぞれ変換する場合は、CA Access Control のパフォーマンスに影響が出ることがあります。 CA Access Control がこれらの変換を実行する方法は、seos.ini ファイルのトークンの値によって決まります。特に関係するトークンは、under_NIS_server、use_lookaside、GroupidResolution、HostResolution、ServiceResolution、UseridResolution、および resolve_timeout です。

ネイティブ オペレーティング システムのメカニズムを使用して変換を実行する場合は、システム パフォーマンスへの影響は比較的少なくなります。 IP アドレスをホスト名に変換する場合は、DNS などの外部メカニズムを使用して変換を実行する必要があります。この場合は、システム パフォーマンスが大幅に低下することがあります。 パフォーマンスが大幅に低下する理由は、seosd がホスト名の受け取りを待機している間、CA Access Control がインターセプトした他のすべてのプロセスも、seosd が処理を完了するまで待機する必要があるためです。

- under_NIS_server トークンの値を no に設定すると、seosd は、以下のソースからデータを取得して、UID、GID、IP アドレス、およびポート番号の変換を UNIX に許可します。

端末の種類		ソース
スタンドアロン		seosd は、以下のファイルを変換に使用します。 ■ UID をユーザ名に変換する場合は、/etc/passwd ■ GID をグループ名に変換する場合は、/etc/group ■ IP アドレスをホスト名に変換する場合は、/etc/hosts ■ サービス ポートをサービス名に変換する場合は、/etc/services
	NIS クライアント	情報のソースは、オペレーティング システムおよびそのバージョン番号によって異なります。 通常は、/etc ファイルおよび NIS サーバから情報を取得します。 ただし、一部のシステムでは、/etc のファイルはソースではなく、変換が行われる順序はシステムの環境設定の際に変更されます。 たとえば、Solaris 2.x システムの場合、変換順序は /etc/nsswitch.conf ファイルによって決定されます。
DNS クライアント		ユーザ、グループ、およびサービスの変換は、/etc のファイルを使用して実行されます。 ホスト名は、DNS サーバを呼び出して変換されます。一部のシステムでは、/etc/hosts ファイルの読み込みも行われます。
NIS クライアントおよび DNS クライアント		IP アドレスからホスト名への変換は、DNS で実行されます。 ユーザ、グループ、およびサービスの変換については、NIS クライアントの場合と同様の変換方法で実行されます。

- under_NIS_server トークンの値を yes に設定すると、seosd は独自に変換を実行します。 seosd で変換用データをキャッシュする場合、データのソースは以下のとおりです。

端末の種類	ソース
NIS サーバ	通常、サーバ コンピュータはサーバおよびクライアントとして動作し、変換を行うために NIS サーバ デーモンをクエリします。 NIS 名前解決マップのソースが含まれるファイルは、通常 /var/yp にあります。ただし、サイトの構成およびオペレーティング システムの種類とバージョンによって、ファイルの場

端末の種類	ソース
	所は異なる場合があります。
DNS サーバ	変換に使用される情報のソースは、サイトの構成によって異なります。 DNS には、名前解決データベースをスキャンするオプションがありません。したがって、CA Access Control はキャッシュを使用できないため、lookaside データベースを使用する必要があります。 sebuildla ユーティリティがホスト リスト ファイルを使用できるように、lookaside データベースを設定する必要があります。 詳細については、この章の sebuildla ユーティリティの説明を参照してください。
その他すべて	DNS サーバと同様です。

バージョン 2 以降の CA Access Control の seosd では、変換プロセスを制御するために、GroupidResolution、HostResolution、ServiceResolution、UseridResolution、および resolve_timeout の各トークンも使用できます。 これらのトークンの詳細については、「リファレンス ガイド」を参照してください。

第 16 章: UNIX exit の使用

このセクションには、以下のトピックが含まれています。

UNIX exit

UNIX exit は、定義された別の CA Access Control アクティビティが行われた場合に自動的に実行されるように指定されたプログラム(シェル スクリプトまたは実行可能ファイル)です。 CA Access Control では、CA Access Control カーネル モジュールをロードまたはアンロードするとき、または特定の selang コマンドを発行するときに、UNIX exit を使用できます。 たとえば、新たに追加した各ユーザの初期化プロセスを実行できます。

UNIX exit は、以下の 1 つまたは複数の状況で実行できます。

- pre-update exit として、ユーザまたはグループのレコードを更新する各 selang コマンドの前
- post-update exit として、ユーザまたはグループのレコードを更新する各 selang コマンドの後
- pre-load exit として、SEOS_load が CA Access Control カーネルをロードする前
- post-load exit として、SEOS_load が CA Access Control カーネルをロードした後
- pre-unload exit として、SEOS_load -u が CA Access Control カーネルをアンロードする前
- post-unload exit として、SEOS_load -u が CA Access Control カーネルをアンロードした後

ユーザ レコードまたはグループ レコード更新の exit

UNIX exit は、ユーザ レコードまたはグループ レコードを更新する selang のコマンドが UNIX 環境で実行されるたびに呼び出されます。コマンドライン インタフェース (selang) または GUI (CA Access Control エンドポイント管理 など) のいずれのツールを使用しても結果は同じです。

更新という用語は、ユーザ レコードまたはグループ レコードの作成、変更、または削除を意味します。 ユーザまたはグループのクエリを実行しても UNIX exit は実行されません。 以下のコマンドによって UNIX exit が実行されます。

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

UNIX 側からは、各 exit プロセスが root プロセスとして実行されるように見えますが、CA Access Control 側からは、このプロセスが _seagent という Agent ID で実行されているように見えます。

用意されている selang exit スクリプトのしくみ

CA Access Control には、スクリプトが用意されています。これをマスタ スクリプトとして使用し、現在の selang のコマンドの種類およびステータスに応じて他のプログラムを呼び出すことができます。 CA Access Control の一部として用意されている exit スクリプトは ACInstallDir/exits/lang_exit.sh です (ここで、 ACInstallDir は CA Access Control のインストール ディレクトリです)。このスクリプトは以下のように実行されます。

1. CA Access Control では、スクリプトの 3 つのパラメータに値が自動的に設定されます。

パラメータ	指定可能な値
CLASS	USER | GROUP
ACTION	CREATE | MODIFY | DELETE
STAGE	PRE | POST

これらのパラメータが示す内容は、CA Access Control の処理対象がユーザまたはグループのいずれであるか、ユーザまたはグループに対して行われる処理が作成、削除、または変更のいずれであるか、selang のコマンドが実行前（PRE）または実行後（POST）のいずれであるかです。

スクリプトは、呼び出すプログラムにパラメータ値を渡すことができます。

パラメータ	指定可能な値
EXEC_RV	UNIX コマンドの戻り値を受け取ります。戻り値は、コマンドが成功したか失敗したかの判断に使用されます。 PRE コマンドの場合、戻り値は常に 0 になります。 POST コマンドの場合、戻り値を使用して exit を実行するか省略するかを決定できます。 このパラメータの使用例については、ACInstallDir/samples/exits_src ファイルを参照してください。

2. CLASS パラメータおよび STAGE パラメータを使用して、適切なディレクトリにあるプログラムが検索されます。

   ACInstallDir/exits/USER_PRE/
   ACInstallDir/exits/USER_POST/
   ACInstallDir/exits/GROUP_PRE/
   ACInstallDir/exits/GROUP_POST/

3. ファイル名が大文字の S で始まる以下の形式で、適切なアクションを参照するすべてのプログラムが、適切なディレクトリから選択されます。

   Snnaction_string

   ここで、nn はプログラムの実行順序を定義する 2 桁の 10 進数、action は CREATE、MODIFY、または DELETE のいずれか、string は説明文字列です。

4. 該当するすべてのプログラムが、名前の 2 文字目および 3 文字目にある数値の順序に従って実行されます。

例：UNIX exit スクリプト

ユーザを削除します。ACInstallDir/exits/USER_PRE/ ディレクトリには以下のファイルがあります。

- S10CREATE_precustom.sh
- S10DELETE_precustom.sh
- S99DELETE_prermusrdir.sh

ユーザを削除するコマンドを発行した場合、ユーザを作成するのではなく削除するので、1 つ目のプログラムは実行されません。 最初の S の後の 2 桁の数字に基づいて、2 つ目と 3 つ目のプログラムが順に実行されます。

selang exit に渡すことができる引数

exit を記述する場合、前述した 3 つのパラメータ(CLASS、ACTION、および STAGE)のほかに、CA Access Control のすべての標準データ(名前やアクセス許可など)を利用できます。 またこれ以外にも、exit スクリプト専用に使用するユーザ データまたはグループ データも指定できます。 ユーザまたはグループに関するこのような補足データを格納するには、newusr、chusr、newgrp、または chgrp コマンドで、データを一重引用符で囲み、ユーザまたはグループの UNIX APPL プロパティの値として定義します。 以下に例を示します。

```
chusr JONESY unix APPL('HIRED=MAY93,CLEARANCE=2')
```

一重引用符で囲まれたデータが、exit プログラムで処理可能であることが前提となります。

実行する selang exit プログラムの指定

実行する exit プログラムを CA Access Control に指示するには、seos.ini ファイルの[lang]セクションを変更します。 CA Access Control には、preuser、postuser、pre-group、post-group exit 用の lang_exit.sh スクリプトが用意されています。 また、exit を指定しないことや独自の exit を作成することもできます。

独自の selang exit を指定するには、必要に応じて seos.ini の[lang]セクションに任意の設定またはすべての設定を設定します。

注: exit が呼び出されるのは、exit トークンの値として完全パス名が指定されている場合のみです。

例: selang exit を指定する

以下の例に示す seos.ini ファイルのトークンの設定では、グループ操作の前に groupcheck というプログラムを実行し、グループ操作の後に flag_exceptions というプログラムを実行します。また、lang_exit.sh というプログラムをユーザ操作の後に実行します。ユーザ操作の前に実行される exit プログラムはありません。 seos.ini ファイルのトークンは以下のように設定されています。

```
[lang]
pre_group_exit = /opt/CA/AccessControl/exits/groupcheck
post_group_exit = /opt/CA/AccessControl/exits/flag_exceptions
post_user_exit = /opt/CA/AccessControl/exits/lang_exit.sh
```

タイムアウトおよびその他のエラー

seos.ini ファイルの exit_timeout 変数で特に指定されていない限り、exit の実行は 15 秒後にタイムアウトします。 ゼロ以外の戻り値はエラーを示します。

- pre-update exit がタイムアウトしたか 16 以上のリターン コードを返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示して更新コマンドの実行を中止します。 これ以外の正数のリターン コードが返された場合、コマンドの実行は中止されません。
- post-update がタイムアウトしたかゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示します。 CA Access Control の更新コマンドはすでに実行されているため、コマンドの効力は保持されたままになります。

selang exit のサンプル

推奨されるスクリプト作成テクニックを習得するためには、以下のディレクトリにあるスクリプトを参照してください。

ACInstallDir/samples/exits-src
ACInstallDir/samples/sample_exits

CA Access Control カーネル ローダ exit

CA Access Control カーネルがロードまたはアンロード(SEOS_load)されると必ず UNIX exit が呼び出されます。 これにより、CA Access Control カーネルをロードまたはアンロードしたときのオペレーティング システムとサード パーティ製プログラムの処理方法を定義できます。 たとえば、カーネル アンロードの UNIX exit を使用して、SEOS_load -u の実行時に CA Access Control のアンロードを妨げるプロセスを自動的に停止し、後で再起動することができます。

一部のオペレーティング システムでは、CA Access Control にカーネル ロードの exit、カーネル アンロードの exit、またはその両方が用意されており、すぐに使用できます。

注: CA Access Control カーネルのアンロードを妨げるプロセスの特定の詳細については、「リファレンス ガイド」を参照してください。

カーネル ロードの exit のしくみ

オペレーティング システムとサード パーティ製プロセスを制御するために、CA Access Control では、CA Access Control のカーネル拡張機能をロードするときに UNIX exit を自動的に呼び出すことができます。

SEOS_load を実行すると、CA Access Control は以下のアクションを行います。

1. 以下のディレクトリ内でプログラムを検索します。

   ACInstallDir/exits/LOAD

2. 以下の形式のファイル名を持つすべてのプログラムを選択します。

   SEOS_load_string.always

   ここで、string は任意の説明文字列です。

3. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

   SEOS_load_string.always -pre

   -pre パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する exit を記述して、カーネルをロードする前に必要なアクションを実行できます。

   注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示してカーネルのロードを中止します。

4. カーネル(SEOS_syscall)をロードします。

5. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

   SEOS_load_string.always -post

   -post パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する exit を記述して、カーネルをロードした後に必要なアクションを実行できます。

   注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示します。 CA Access Control カーネルはすでにロードされているため、ロードされたままになります。

カーネル アンロードの exit のしくみ

オペレーティング システムとサード パーティ製プロセスを制御するために、CA Access Control では、CA Access Control のカーネル拡張機能をアンロードするときに UNIX exit を自動的に呼び出すことができます。

SEOS_load -u を実行すると、CA Access Control は以下のアクションを行います。

1. 以下のディレクトリ内でプログラムを検索します。

   ACInstallDir/exits/LOAD

2. 以下の形式のファイル名を持つすべてのプログラムを選択します。

   SEOS_unload_string.always

   ここで、string は任意の説明文字列です。

3. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

   SEOS_load_string.always -pre

   -pre パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する exit を記述して、カーネルをアンロードする前に必要なアクションを実行できます。

   注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示してカーネルのアンロードを中止します。

4. カーネルのアンロードを試行します。

   カーネルがアンロードされない場合は、以下の手順に従います。

   a. 以下の形式のファイル名を持つすべてのプログラムを選択します。

      SEOS_unload_string.opt

   b. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

      SEOS_unload_string.opt -pre

      -pre パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する条件付きの exit を記述して、カーネルをアンロードする前に必要な追加のオプションのアクションを実行できます。

      注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示してカーネルのアンロードを中止します。

c. カーネルをアンロードします。

d. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

SEOS_unload_string.opt -post

-post パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する条件付きの exit を記述して、カーネルをアンロードする前に必要な追加のオプションのアクションを実行できます。

注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示します。 CA Access Control カーネルはすでにアンロードされているため、アンロードされたままになります。

5. ACInstallDir/exits/LOAD ディレクトリで見つかった各ファイルを辞書式順序で実行します。

SEOS_unload_string.always -post

-post パラメータを指定して各ファイルを実行します。これにより、このパラメータを検出する exit を記述して、カーネルをロードした後に必要なアクションを実行できます。

注: exit がゼロ以外の値を返した場合、CA Access Control は exit プロセスを強制終了(kill)し、エラー メッセージを表示します。 CA Access Control カーネルはすでにアンロードされているため、アンロードされたままになります。

第 17 章: LDAP の操作

このセクションには、以下のトピックが含まれています。

ユーザ名の転送

CA Access Control と LDAP の両方を使用している場合は、独自に作成したスクリプトを使用して、両者の間でユーザ名を転送できます。サンプル スクリプトは、3 つ用意されています。

重要: sebuildla および必要な LDAP 設定をセットアップするには、LDAP をよく理解していること、および ldapsearch コマンドを実行できることが必要です。 ldap(1)、ldapsearch(1)についての man ページ、および LDAP クライアント用のマニュアルでセットアップの説明を参照することをお勧めします。

用意されたスクリプトのうち 2 つのスクリプト(ldap2seos および seos2ldap)では、ユーザの集合全体を CA Access Control と LDAP サーバの間で相互にエクスポートおよびインポートします。

もう 1 つのサンプル スクリプト S50CREATE_u_LdapE.Sh では、新規 UNIX ユーザ名の作成時にそのユーザ名を CA Access Control から LDAP に自動的に転送します。

サンプル スクリプトでは、Language Client API(LCA)ライブラリ拡張である tcllca.so を使用するため、TCL シェル環境にアクセスする必要があります。

注: LCA および TCL 拡張の詳細については、「SDK 開発者ガイド」の第 5 章「Language Client API」および付録 A「LCA 拡張機能」を参照してください。

TCL がない場合は、comp.lang.t_c_l に毎月掲示される Larry Virden による FAQ を参照してください。この FAQ は MIT Web サイトおよび Terafirm Web サイトで参照できます。

また、TCL に関するニュース、ドキュメント、およびリソースについても、Sun の Web サイトで参照できます。

S50CREATE_u_LdapE

S50CREATE_u_LdapE.sh は、新規 UNIX ユーザが作成されると、そのユーザを LDAP にアップロードします。

CA Access Control には、新規 UNIX ユーザを LDAP サーバに自動的にインポートするサンプル シェル スクリプトが用意されています。 実際に必要なスクリプトは、サンプルとは異なる場合があります。

サンプル シェル スクリプトを使用するには、用意されている exit スクリプトをすでに使用していることを前提として、以下の手順に従います。

1. S50CREATE_u_LdapE.sh ファイルをディレクトリ ACInstallDir/exits/USER_POST にコピーします。 このディレクトリでは、スクリプトが post-user exit になります。
2. seos.ini ファイルの[ldap]で、base_entry トークンに LDAP 基本エントリを設定します。

   たとえば、カナダにある ServerWorld という組織の場合、基本エントリは o=ServerWorld, c=CA となります。
3. 同じセクションで、ホスト名として LDAP サーバのホスト名を設定します。 LDAP 基本ディレクトリのパスを設定します（サンプル スクリプトにより、そのディレクトリの下の bin ディレクトリでライン コマンド ユーティリティが検索されます）。

Common Name（cn）はユーザのフルネームから取得されます。 たとえば、CA Access Control データベースにユーザの名前と姓のみが格納されている場合、Common Name はユーザの名前と姓で構成されます。 基本的にはユーザは Common Name にロックされます。したがって、Common Name はユーザ名を基準にしないことをお勧めします。

その後で selang を使用して UNIX に追加される各ユーザは、自動的に LDAP サーバにアップロードされます。 ユーザがすでに LDAP に存在する場合は、エラー メッセージが表示されます。

このスクリプトを使用してユーザを追加した場合、関連する LDAP の応答および警告があると、それらは /tmp/add_User2Ldap.tcl.log ファイルに収集されます。 このファイルにエラーがあるかどうかは、vi またはその他 UNIX の標準エディタを使用して確認できます。 このファイルは、新規ユーザを追加するたびに、新しい応答および警告によって上書きされます。

第 18 章: Unicenter セキュリティの移行と統合

このセクションには、以下のトピックが含まれています。

Unicenter セキュリティ統合ツール

CA Access Control は、Unicenter エンタープライズ管理環境に完全に統合されます。

重要: Unicenter TNG を CA Access Control と統合するには、Unicenter TNG を CA Access Control と同じマシンにインストールする必要があります。

注: インストール手順の詳細については、「実装ガイド」を参照してください。

Unicenter セキュリティの統合機能

以下のセクションでは、CA Access Control を Unicenter TNG と統合する方法について説明します。

SSF/EMSec API サポート

UNIX のセキュリティ API はすべてメッセージ キューに切り替わります。 新しいユーティリティでは、メッセージ キューを使用して送信された API 要求を処理し、再フォーマットおよび再転送されたこれらの要求を転送します。 その後、ユーティリティは CA Access Control のリターン コードを Unicenter TNG の対応するリターン コードに変換します。 このアプローチにより、EMSec API を現在使用している既存のアプリケーションの整合性が保護されます。

UNIX では、このユーティリティを sessfgate といいます。 このゲートウェイは、Unicenter 統合のセットアップ手順が完了すると有効になります。 実際には、Unicenter NSM Event Notification Facility（ENF）が実行されている場合、CA Access Control のサービスが（seload を使用して）開始または停止されると必ずゲートウェイは自動的に起動または停止します。 ENF が実行されていない場合は、CA Access Control の seload プログラムは sessfgate デーモンを起動しません。 Unicenter 統合のセットアップでは、sessfgate プログラムが ACInstallDir/tng/bin ディレクトリにインストールされます。 Unicenter セキュリティが停止されて CA Access Control が起動された後に、sessfgate は SSF の代わりに API 要求を受け取ることができます。

sessfgate を以下のように実行します。

```
# sessfgate [-i|-s|-l] -t
```

-l

ゲートウェイの起動を指定します。

-s

ゲートウェイの停止を指定します。

-l

ステータスを指定します。

-t

トレース ファイルの有効または無効を切り替えます（ログ ファイルは ACInstallDir/log/sessftrace.log です）。

注: Unicenter TNG を実行する前に seload を実行する場合は、以下のコマンドを使用して、sessfagte を手動で起動する必要があります。

```
ACInstallDir/tng/bin/sessfagte -I
```

この場合、ACInstallDir は CA Access Control がインストールされているディレクトリです。

CA Access Control から Unicenter セキュリティへの同期ユーティリティ

Unicenter セキュリティと CA Access Control は共に、全体的な移行が行われる前の企業の IT 環境を管理します。

さまざまな製品ツールを使用して管理作業を行う複雑さを軽減するために、CA Access Control には同期デーモンが用意されています。 このデーモンを seostngd といいます。 CA Access Control は、CA Common Communication Interface (CAICCI) を使用して、Policy Model データベース (PMDB) の更新情報を seostngd に送信します。 seostngd デーモンは CAICCI で更新情報を待機し、その後、このグローバル データで Unicenter Security データベースを更新するために、メッセージを等価な cautil コマンドに変換します。

現在の Unicenter TNG 処理は、引き続きその他の Unicenter TNG クライアント インストールを更新できます。 Unicenter セキュリティデータベースが格納されているコンピュータ (通常は Unicenter TNG マスタ コンピュータといいます) で seostngd を実行する必要があります。また、CA Access Control も同じコンピュータで実行する必要があります。

注: このデーモンは、移行プロセスの際にのみ使用します。 移行プロセスの完了後は、このデーモンを使用しないでください。 このデーモンは、IT 環境での長期的なソリューションではありません。

seostngd の主要なタスクは、CA Access Control で加えた変更 (GUI または selang のコマンドのいずれかを使用して) を受け取り、その変更を Unicenter セキュリティデータベースに適用することです。 変更内容を Unicenter セキュリティデータベースに正しく適用できた場合は、CA Access Control で加えた変更と同じ結果を得ることができます。

たとえば、CA Access Control で USER オブジェクトを作成すると、(必要なフィールドが存在する限り) Unicenter TNG に同じ USER オブジェクトが作成されます。

サブスクライバ デーモンの起動と停止

サブスクライバ デーモンを起動するには、以下のように入力します。

```
seostngd
```

デーモンを停止するには、以下のいずれかを入力します。

```
seostngd -shut
```

または

```
seostngd -stop
```

重要: 複数のコマンドを記述している場合は、移行中に selang -c を使用しないでください。 代わりに、selang -f input_file_name を使用してください。

注:

- デーモンを起動するためには、Unicenter TNG SSF_AUTH ユーザ リストの root ユーザである必要があります。
- サブスクライバ デーモンを手動で操作する必要があります。 たとえば、コンピュータを再起動する場合は、このユーティリティが CA Access Control の起動プログラムによって制御されないので、seostngd コマンドを使用して再起動します。
- これは、Unicenter TNG の統合インストール手順の実行前には使用できません。セットアップ手順の実行中に、ACInstallDir/data/tng/assettypes.txt という環境設定ファイルが生成されます。 このユーティリティを実行するには、この環境設定ファイルが必要です。
- $CAIGLBL0000/bin が PATH 環境に存在することを確認してください。存在すれば、Unicenter コマンド ライン ユーティリティの cautil を実行できます。 これを行うには、(ksh または sh #から)スクリプト ファイル $CAIGLBL0000/scripts/envset を実行します。
- Unicenter セキュリティデーモン(sdm)を実行する必要があります。実行していない場合、Unicenter TNG サブスクライバ デーモンは、Unicenter セキュリティデータベースに変更を適用できません。

SEOSTNGD の制限事項

CA Access Control と Unicenter セキュリティの最大フィールド長が異なると、データ値が切り捨てられる場合があります。 以下の表に、サポートされているフィールド長が大幅に異なるフィールドを示します。

	Unicenter TNG	CA Access Control
ユーザ ID	20 文字	256 文字
パスワード	8 文字	14 文字
ユーザ グループ ID	8 文字	254 文字
アセット グループ ID	8 文字	255 文字

- Unicenter TNG は、ユーザ オブジェクトまたはアセット オブジェクトの名前の変更をサポートしないため、seostngd デーモンはユーザおよびアセットに対する CA Access Control の rename コマンドを無視します。
- Unicenter TNG ユーザ グループとアセット グループは、最低 1 つのメンバを保持する必要があります。 ユーザ グループまたはアセット グループを作成するためのメンバが CA Access Control コマンドで指定されていない場合は、最低 1 つのメンバが追加されるまで、対応する Unicenter TNG ユーザ グループまたはアセット グループは作成されません。

- CA Access Control のユーザ グループまたはアセット グループから最後のメンバが削除されると、そのユーザ グループまたはアセット グループが Unicenter TNG から削除されます。
- Unicenter TNG のアセットには最低 1 つの定義済みアクセサが必要です。そのため、そのアセットに対して最低 1 回は CA Access Control の「authorize」コマンドが実行されるまで、Unicenter TNG の新しいアセットを作成できません。
- CA Access Control では、オブジェクトが削除されると、オブジェクトの関連ルールもすべて削除されます。 ただし、Unicenter セキュリティはこの処理を実行しません。
- CA Access Control のユーザには ADMIN 属性が割り当てられますが、この属性は、Unicenter セキュリティ オプションの SSF_AUTH ユーザ リストのメンバである Unicenter TNG のユーザに割り当てられる属性と類似しています。 ただし、Unicenter TNG ではリモート セキュリティ オプションを自動的に操作できないため、SSF_AUTH ユーザ リストを手動で変更する必要があります。
- 新しい Unicenter TNG ユーザを作成するには、パスワード フィールドに指定された値を使用して、CA Access Control の新規ユーザをネイティブ環境に作成する必要があります。 Unicenter TNG にはデフォルトのパスワード制限があり、2 文字の英字と 1 文字の数字を含む最低 6 文字のパスワードが必要です。
- CA Access Control の authorize コマンドはアクセサ ID としてアスタリスク(*)をサポートしますが、Unicenter TNG ではサポートされません。 seostngd デーモンは、このような CA Access Control の authorize コマンドを無視します。
- CA Access Control の authorize コマンドは via パラメータを使用する条件付きアクセス ルールをサポートしますが、Unicenter TNG ではサポートされません。 seostngd デーモンは、このような CA Access Control の authorize コマンドを無視します。
- CA Access Control の authorize コマンドで access パラメータが指定されなかった場合、seostngd デーモンはすべての UNIX-FILE または Unicenter TNG のアセット グループに READ 権限を与え、Unicenter TNG のその他すべての事前定義アセット タイプにすべての権限を与えます。

Unicenter セキュリティ データの移行機能

以下のセクションでは、Unicenter セキュリティデータを CA Access Control に移行する方法について説明します。

Unicenter セキュリティオプションの移行

CA Access Control には、migopts というプログラムが付属しています。このプログラムを使用すると、選択した Unicenter セキュリティのオプションを抽出し、そのオプションに基づいて、対象のデータベースをカスタマイズできます。 この機能を有効にするには、Unicenter セキュリティ データの移行のセットアップ手順を使用して、Unicenter 統合を実行する必要があります。 このセットアップ手順では、migopts が自動的に実行されます。

重要: この機能は Unicenter セキュリティ データの移行の一部であり、cautil コマンドプロセッサ、イベント管理、および負荷管理との統合のみに Unicenter セキュリティを使用するユーザを対象としています。 Unicenter セキュリティと CA Access Control のアーキテクチャには違いがあるため、Unicenter セキュリティ データの移行は、ファイルシステムの保護に Unicenter セキュリティを使用するユーザを対象としていません。

注: 以下の Unicenter セキュリティオプションは、CA Access Control 環境に完全に移行できます。

- CREDAUTHEXIT
- DEFSESID
- PASSWORD_ALPHA
- PSWDVALEXIT
- PWDQUEUESIZE
- SSF_MAXPWDVIO
- SSF_MINPWDLEN
- SSF_NUMSUBP
- SSF_SECPWEXCL
- SSO_APPLNAME
- USER_PWDCHANGE
- USER_PWDCHGMAXDAYS
- USER_PWDCHGMINDAYS

また、exporttngdb により、SSF_AUTH Unicenter セキュリティオプションのメンバである Unicenter TNG ユーザが環境に移行されます。その際、ユーザを CA Access Control に追加する前に、そのユーザに「admin」属性が設定されます。

注: migopts ユーティリティは、移行スクリプト ACInstallDir/tng/bin/uni_migrate_master.sh および ACInstallDir/tng/bin/uni_migrate_node.sh によって実行されます。 詳細については、「リファレンス ガイド」を参照してください。

Unicenter セキュリティデータベースの移行

CA Access Control には、exporttngdb というプログラムが付属しています。このプログラムを使用すると、Unicenter セキュリティデータベースからデータを抽出し、そのデータを CA Access Control のコマンドに変換して CA Access Control データベースに格納できます。 exporttngdb は、以下の項目を移行します。

- Unicenter セキュリティユーザ
- Unicenter セキュリティユーザ グループ
- Unicenter セキュリティルール

注:

1. Unicenter の統合および移行インストールの実行後に、Unicenter TNG のログイン インターセプトを実行しないことをお勧めします。 Unicenter の統合および移行インストールが正常に完了した後に、Unicenter TNG ログイン インターセプトが無効になっていることを確認します。
2. Unicenter TNG のデータ スコーピング ルールとキーワード スコーピング ルール (-DT サフィックスまたは -KW サフィックスが付いた Unicenter TNG のアセット タイプを対象とするルール) は、CA Access Control の移行プロセスではサポートされていません。 移行プロセスでは、このタイプのルールは無視されます。
3. Unicenter セキュリティは現在使用されていないため、Unicenter セキュリティのアセット タイプ (CA-USER、CA-ACCESS、CA-USERGROUP、CAASSETGROUP、CA-ASSETTYPE、および CA-UPSNODE) に対して実装された Unicenter セキュリティ ルールはどれも使用されていません。 このようなアセット タイプまたはそれらから派生したタイプを対象とするルールは、移行プロセスではすべて無視されます。
4. exporttngdb ユーティリティは、移行スクリプト ACInstallDir/tng/bin/uni_migrate_master.sh および ACInstallDir/tng/bin/uni_migrate_node.sh によって実行されます。

   注: exporttngdb ユーティリティの詳細については、「ユ ーティリティ ガイド」を参照してください。

exporttngdb を有効にするには、Unicenter セキュリティ データの移行のセットアップ手順を使用して、Unicenter 統合を実行する必要があります。 このセットアップ手順では、Unicenter セキュリティ データの移行プロセスが自動的に実行されます。

重要: この機能は Unicenter セキュリティ データの移行の一部であり、cautil コマンドプロセッサ、イベント管理、および負荷管理との統合のみに Unicenter セキュリティを使用するユーザを対象としています。 Unicenter セキュリティと CA Access Control のアーキテクチャには違いがあるため、Unicenter セキュリティ データの移行は、ファイルシステムの保護に Unicenter セキュリティを使用するユーザを対象としていません。

注: すべての Unicenter TNG オブジェクトの作成および変更に関する統計情報は、移行プロセスで失われます。

Unicenter TNG と CA Access Control の製品の相違により、Unicenter セキュリティ ユーザの以下の属性は CA Access Control に移行されません。

統計情報

以下のユーザ統計情報は CA Access Control ではサポートされていません。

- 最終ログイン統計情報(最終ログインの日時とノード)
- パスワード変更統計情報(日時、ノード、最後にパスワードを変更したユーザ、およびパスワードの有効期限)
- パスワード違反統計情報(最後に失敗したログインの日時とノード、および最後にログインが成功した後に失敗したログインの数)
- アクセス違反統計情報(最後のアクセス違反の日時とノード、およびアクセス違反の数)
- 停止統計情報(停止日時)

PWDCHANGE VALUE (RANDOM)

無作為なパスワードの生成。

UPSSTATGROUP

UPS 端末のグループ。 CA Access Control ではサポートされていません。

VIOLMODE

違反モード(FAIL、MONITOR、WARN、QUIET)。 CA Access Control では、FAIL モードのみがサポートされています。

VIOLACTION

違反アクション(CANUSER、CANU&LOG、CANU&LOG&SUS)。 CA Access Control では、CANUSER アクションのみがサポートされています。

Unicenter TNG と CA Access Control の製品の相違により、Unicenter セキュリティ ユーザの以下の属性は CA Access Control に移行されません。

EXPIRES

ルールの有効期限は、CA Access Control ではサポートされていません。

Unicenter TNG user exit のサポート

CA Access Control への移行を行っている現在の Unicenter TNG ユーザをサポートするために、CA Access Control では、既存の Unicenter セキュリティの user exit を、CA Access Control 環境でも変更することなく実行できるようになっています。 移行の一環としてすべての user exit を記述し直す必要はありません。

Unicenter セキュリティおよび CA Access Control で既存の user exit インタフェースのみを使用すると、インストールした各コンポーネントが標準の CA Access Control user exit として登録されます。これにより、対応する Unicenter セキュリティの exit が起動します。

この機能を開始するには、Unicenter セキュリティ データの移行のセットアップ手順を使用して、Unicenter 統合を実行する必要があります。 セットアップ手順を完了すると、この機能がアクティブになります。

重要: この機能は Unicenter セキュリティ データの移行の一部であり、cautil コマンド プロセッサ、イベント管理、および負荷管理との統合のみに Unicenter セキュリティを使用するユーザを対象としています。 Unicenter セキュリティと CA Access Control のアーキテクチャには違いがあるため、Unicenter セキュリティ データの移行は、ファイル システムの保護に Unicenter セキュリティを使用するユーザを対象としていません。

注: Unicenter TNG と CA Access Control のアーキテクチャの違いにより、Unicenter セキュリティと CA Access Control の間で比較可能な exit ポイントおよびデータ項目のみがサポートされています。 以下の Unicenter セキュリティ exit ポイントがサポートされています。

EmSec_CredExit()

Unicenter Credential Authentication exit に対する入力 EmSec_CredExit() は、EMSECSIGNON によってマップされます。 CA Access Control では、この構造体内のユーザおよびノード メンバのデータのみが意味を持ちます。 ユーザ メンバは認証されたユーザ名に設定され、ノード メンバは現在のローカル ノード名に設定されます。 EMSECSIGNON 構造の他のすべてのメンバは、バイナリ ゼロに設定されます。 他のパラメータ、詳細なリターン コード、Unicenter リソース チェック exit から返されたメッセージは無視されます。

EmSec_PwExit()

Password Validation exit である EmSec_PwExit() は、完全にサポートされます。

注: exit は、/usr/local/Calib/ または ACInstallDir/Calib/ にある libemsec2.xx に含まれています(ここで、ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です)。

Unicenter 統合のセットアップが完了すると、この機能が有効になります。

PMDB の使用による Unicenter セキュリティオブジェクトの保護

PMDB を Unicenter TNG オブジェクトで使用して、ルールを作成することができます。このルールでは、さまざまな Unicenter TNG コンポーネント(cautil コマンド プロセッサ、イベント管理、負荷管理など)によって Unicenter TNG オブジェクトが操作されないようにセキュリティで保護します。 この統合は手動で実行する必要があります。

Unicenter TNG オブジェクトに PMDB を使用するには、以下の手順に従います。

1. PMDB を作成します。
2. 以下のコマンドを使用して、Unicenter セキュリティオプションを PMDB に移行します。

   migopts -d pmdb-name

   pmdb-name には PMDB の名前を指定します。

   重要: この手順は、Unicenter セキュリティを使用し、かつ Security Data Migration のために Unicenter 統合のインストール スクリプトを実行した場合にのみ実行する必要があります。 Unicenter セキュリティを使用しなかった場合は、セキュリティ オプションを設定していないので、PMDB に移行する必要のあるオプションはありません。

   注: 移行と統合は 2 つの異なる処理です。
3. 以下のコマンドを使用して、ユーザ定義の Unicenter TNG のアセット タイプに関するクラスを作成します。

   defclass.sh.pmdb-name

   pmdb-name には PMDB の名前を指定します。

   重要: この手順は、Unicenter Security を使用し、かつユーザ定義のアセット タイプを作成した場合にのみ実行する必要があります。 CA Access Control のインストール時に Unicenter 統合を選択した場合、Unicenter TNG のアセット タイプは新しい PMDB を作成するたびに自動的に定義されます。

Unicenter TNG カレンダ

Unicenter TNG にはカレンダ機能が提供されており、ユーザ、グループ、リソースに対して時間制限を設定できます。 カレンダは、15 分間隔で ON または OFF に設定できます。 カレンダの間隔を OFF に設定すると、リソースへのアクセスが拒否されます。 ON に設定すると、リソースの承認が続行されます。

UNIX では、セキュリティの起動前のみ、管理者がカレンダの使用を設定できます。

CA Access Control で Unicenter TNG カレンダを使用するには、以下の手順に従います。

注: Unicenter TNG はローカル マシンにインストールする必要があります。 CA Access Control では、ローカル Unicenter TNG サービスを使用して、カレンダの設定を取得します。

1. CA Access Control のセキュリティを停止します。 以下を入力します。

   ```
   secons -s
   ```

2. seos.ini ファイルの[seauxd]セクションにある TNG_calendars トークンを yes に設定します。

   ```
   TNG_calendars=yes
   ```

3. CA Access Control のセキュリティを起動します。 以下を入力します。

   ```
   seosd
   ```

4. 補助デーモン seauxd が実行されていることをチェックします。 以下を入力します。

   ```
   issec
   ```

seos.ini ファイルにある以下のトークンを変更することもできます。

TNG_refresh_interval

CA Access Control カレンダを更新する間隔を分単位で指定します。 デフォルトは 10 分です。

TNG_lib_path

Unicenter TNG ライブラリへの完全パスを指定します。 デフォルトは /usr/local/Calib です。

TNG_cal_lib

Unicenter TNG カレンダ ライブラリの名前を指定します。 デフォルトは libcalendar です。

CA Access Control リソースをカレンダとリンクするには、以下のデータベース コマンドを発行する必要があります。

注: 発行されるカレンダ名では、大文字と小文字の組み合わせを Unicenter TNG カレンダ名と同じにする必要があります。

```
nr CALENDAR calendar_name
nr file /tmp/test calendar (calendar_name) defaccess (a)
```

Unicenter TNG カレンダ アクセス制御リスト(ACL)は、高度なセキュリティ制限機能です。 Unicenter TNG カレンダの標準プロパティでは、適切な Unicenter TNG カレンダ ステータスに基づいて現在のリソースが制限されます。 Unicenter TNG カレンダ ACL のプロパティにより、Unicenter TNG カレンダ ステータスに基づいて、現在のリソースに対する特定のユーザまたはグループのアクセス権が制限されます(またはアクセス権が与えられます)。

ACL Unicenter TNG カレンダ プロパティには、標準プロパティと制限プロパティの 2 種類があります。

- カレンダ ACL の標準プロパティでは、ACL アクセスに基づいて、ユーザまたはグループにリソースへのアクセスが許可されます。
- カレンダ ACL の制限(拒否)プロパティでは、ACL に基づいて、ユーザまたはグループのリソースへのアクセスが拒否されます。

標準のカレンダ ACL(CALACL)にユーザまたはグループを追加するには、以下の selang のコマンドを入力します。

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

以下に例を示します。

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

拒否のカレンダ ACL にユーザまたはグループを追加するには、以下の selang のコマンドを入力します。

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

以下に例を示します。

```
auth file file2 uid(george) calendar(holidays) deniedaccess(rw)
```

標準プロパティおよび制限プロパティの両方を同じリソース(カレンダや uid など)に対して使用できます。 以下のコマンドは、読み取りアクセス権を持つ George という名前のユーザを file1 の拒否カレンダ ACL に追加します。

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

Unicenter TNG カレンダ ACL プロパティからユーザまたはグループを削除するには、auth- コマンドを使用します。

```
auth- file file2 uid(george) calendar(holidays)
```

特定のリソースに割り当てられたすべての Unicenter TNG カレンダ ACL を参照するには、Show Resource(sr)コマンドを使用します。

```
sr file file1
```

Unicenter TNG および Unicenter NSM との動作確認

以下の機能は、Unicenter TNG 2.2 SP1、Unicenter TNG 2.4、および Unicenter NSM 3.0 に準拠しています。

- 「イベント」の送信
- メインフレーム パスワードの同期
- Unicenter TNG カレンダの使用

監査イベントの統合

Unicenter TNG との統合は、インストール時に設定します。

監査データを Unicenter TNG に送信するかどうかを選択できます。 Unicenter TNG に渡される監査イベントは、[Unicenter エンタープライズ管理]-[エンタープライズ マネージャ]-[Windows NT]-[イベント]ウィンドウのコンソール ログに表示されます。

監査イベント	表示色	重大度
成功	青	S
拒否	オレンジ色	F
失敗	オレンジ色	F
警告	青	W
CA Access Control の停止(監査終了)	青	I

監査イベント	表示色	重大度
CA Access Control の開始(監査開始)	青	I

2 番目のオプションを選択すると、Unicenter の[Worldview]メニューから CA Access Control を起動できます。CA Access Control を起動するには、[管理対象オブジェクト]ウィンドウで、TCP/IP ネットワークを表すアイコンをポイントして右クリックし、表示されたメニューから[CA Access Control]を選択します。

また、イベントに関する以下の情報も送信されます。

- 製品名(CA Access Control + バージョン番号)
- ユーザ名
- 端末名
- クラス名
- リソース名
- プロセス名
- イベントの時刻
- CA Access Control 監査形式の完全な監査メッセージ

[ユーザ名]、[端末名]、[クラス名]、[リソース名]、および[プロセス名]の各フィールドは、イベント タイプによっては送信されないこともあります。

第 19 章: 設定

CA Access Control では、CA Access Control エンドポイントの設定をリモートで管理できます。 この場合は、CA Access Control エンドポイント管理または selang インタフェースを使用できます。

このセクションには、以下のトピックが含まれています。

設定

CA Access Control は、使用しているエンドポイントと Policy Model の設定を以下に保存します。

- Windows コンピュータ: Windows レジストリ
- UNIX コンピュータ: 初期設定 (.ini) ファイル

注: 実行できる設定およびその設定の意味の詳細については、「リファレンス ガイド」を参照してください。

設定の変更

CA Access Control と Policy Model の動作を制御するには、設定を変更する必要があります。

設定を変更するには、以下の手順に従います。

1. CA Access Control エンドポイント管理 内で、以下の操作を実行します。

   [設定]をクリックします。

2. [リモート設定]をクリックします。

   [リモート設定]ページが表示されます。

3. 左側の[リモート設定]セクション ペインで、必要に応じて[設定]ツリーを展開して変更する設定が含まれているセクションを表示し、そのセクションをクリックします。

   [セクション: セクション名 システム トークン]ページが表示され、そのセクションに含まれるすべての設定が表示されます。

4. 必要に応じて設定を検索して編集し、[トークンの保存]をクリックします。

変更した設定が保存されます。

監査設定の変更

CA Access Control が監査レコードを生成し、格納する方法を変更するには、監査設定ファイルの設定を変更する必要があります。 監査設定ファイルの設定を変更するには、selang コマンドを使用します。

監査設定を変更するには、以下の手順に従います。

1. （オプション）selang を使ってリモート ホストに接続するには、以下のコマンドを使用します。

   host host_name

2. 以下のコマンドを使って、config 環境に移動します。

   env config

3. editres config コマンドは、必要に応じて環境設定の変更に使用します。

   監査設定は変更されました。

例：監査設定ファイルの変更

以下の例では、監査設定ファイルに 1 行追加します。

er CONFIG audit.cfg line+("FILE;*;Administrator;*;R;P")

付録 A: NIS の環境設定

このセクションには、以下のトピックが含まれています。

インストール上の注意事項

注: ここに示す情報は、インストール スクリプトで説明されている内容を補完するものです。 この付録は、読者が Network Information Systems (NIS)、Domain Name Services (DNS)、および UNIX の名前解決の概念を理解していることを前提にしています。

CA Access Control のインストール時には、2 つのオプションのどちらか 1 つを指定して、ユーザ ID をユーザ名に、グループ ID をグループ名に、ホストの IP アドレスをホスト名に、サービス ポートをサービス名にそれぞれ変換できます。

- システム関数を使用します。この関数では、システム上のネットワーク キャッシング デーモンの省略を定義します。
  - Digital DEC UNIX を使用し、これが NIS サーバではない場合は、デフォルトで名前解決にシステム関数が使用されます。
  - Digital DEC UNIX を使用し、これが NIS サーバであるときは、lookaside データベースを使用するかシステム関数を使用するかの選択を促すメッセージがインストール時に表示されます。これにより、ネットワーク キャッシング デーモンの省略が定義されます。
- lookaside データベースを使用します。このデータベースは、sebuildla ユーティリティで作成されます。
  - NIS サーバ上で実行するように設定された CA Access Control を使用している場合は、lookaside データベースを使用します。
  - インストール時のデフォルトでは、HP-UX 11.0 以上、Sun Solaris 2.6 以上、IBM AIX 5.1L 以上、およびサポート対象のすべての Linux プラットフォーム上で、lookaside データベースが使用されます。

注: IBM AIX プラットフォーム上では、lookaside データベースを使用する必要があります。システム関数を使用するためのオプションはありません。

名前解決

CA Access Control ではシステム リソースへのアクセス要求をインターセプトし、要求の許可または拒否を決定します。 これは、データベースに定義されたアクセス ルールとポリシーに基づいて決定されます。 システム リソースへのアクセス要求のインターセプトは、カーネル レベルで行われます。

ホスト、グループ、ユーザ、およびサービスを制御するために、カーネルおよび関連するシステム コールでは、名前ではなくコードまたは数値(IP アドレス、グループ ID、ユーザ ID、およびサービス番号など)が使用されます。 CA Access Control では、名前に基づいてアクセス ルールが定義されます。 名前は、CA Access Control によってカーネルが認識できるコードに変換されます。 このプロセスを名前解決といいます。

スタンドアロンの端末では、名前解決は、ローカルのユーザ、グループ、およびホスト ファイル(/etc/passwd、/etc/group、および /etc/hosts)を使用して直接実行されます。ただし、Sun Solaris 2.5 以上を実行している端末を除きます。 CA Access Control で名前を解決する必要がある場合は、関連するファイルを次に読み込むシステム関数が呼び出されます。

ただし、大規模ネットワークでは、この情報がローカルに保存されることはほとんどありません。 NIS、DNS、またはその両方を使用する場合、名前を解決する際に参照できるローカル ファイルはありません。 必要な情報は、ネットワーク経由でサーバに要求し、サーバから受け取ります。

NIS/DNS クライアントでの名前解決

CA Access Control では、クライアント専用(サーバを兼ねていない)の NIS 端末または DNS 端末で、以下のようにして名前解決が実行されます。

1. CA Access Control により、関連サーバへの接続を求めるネットワーク要求が生成されます。
2. CA Access Control のカーネル拡張機能により、要求がインターセプトされます。
3. カーネル拡張機能により、要求が許可されます。これは、CA Access Control のプロセスによって要求が内部で生成されたことを識別できたためです。
4. NIS サーバまたは DNS サーバへの接続が確立され、名前解決に必要な情報が取得されます。
5. 名前が解決されると、CA Access Control では、元のアクセス要求に対する許可または拒否を決定する処理が続行されます。

CA Access Control の標準環境設定でも、クライアント サーバで簡単に名前解決が実行されます。

サーバでの名前解決: デッドロック

CA Access Control では、以下のように、クライアントを兼ねるサーバ上で名前解決を実行します。

1. CA Access Control により、関連サーバへの接続を求めるネットワーク要求が生成されます。
2. カーネル拡張機能により、要求がインターセプトされます。
3. カーネル拡張機能により、要求が許可されます。これは、CA Access Control のプロセスによって要求が内部で生成されたことを識別できたためです。
4. NIS サーバまたは DNS サーバ(同一端末上)によって、ネットワーク接続の受け入れ要求が生成されます。
5. カーネル拡張機能により、要求がインターセプトされます。
6. カーネル拡張機能により、この要求が CA Access Control プロセスによって生成されたものではないことが識別されます。 この要求は seosd の判断を待機する要求のキューに格納されます。
7. seosd デーモンがデッドロック状態になります。 seosd デーモンは名前解決に必要な応答を待機しますが、この応答を提供するプロセスでは、seosd デーモンからネットワーク接続の受け入れ許可を受け取るまで処理を進めることができないため、こうした状態が発生します。 最初の要求によって、次の要求が生成され、デッドロックが作成されます。

Sun Solaris での名前解決: デッドロック

Sun Solaris の名前解決では、nscd キャッシュにアクセスする必要があります。 nscd は、一般的なネーム サービス要求のキャッシュを提供するプロセスであり、 passwd、group、および hosts データベースのキャッシュを提供します。

このキャッシュは永続的ではありません。 passwd、group、および hosts データベースが変更されるか、存続期間スタンプの期限が過ぎると無効になります。

Sun Solaris の設定では、前のセクションで説明したように、デッドロックが発生する可能性があります。 以下の例では、CA Access Control と nscd プロセスが相互作用することにより、デッドロックが発生します。

1. 名前解決では、CA Access Control が nscd キャッシュにアクセスします。
2. nscd プロセスによって、そのキャッシュが古すぎると判断されることがあります。 この場合、nscd は、(ローカルまたはサーバ上の)passwd、group、および hosts データベースにアクセスして情報を更新しようとします。
3. これらのデータベースへのアクセス要求は、カーネル拡張機能によってインターセプトされます。 CA Access Control のプロセスでは要求が生成されないため、要求は seosd の決定を待つキューに格納されます。 ただし、seosd はまだその前の要求を処理中であるため、要求を許可するかどうかを決定できません。 最初の要求によって、次の要求が生成され、デッドロックが作成されます。

デッドロックの回避：lookaside データベース

seos.ini 環境設定ファイルの under_NIS_server トークンは、デフォルトでは、デッドロックを回避するように yes に設定されています。 これにより、NIS、DNS、または nscd キャッシュではなく、内部の名前解決テーブルが使用されます。 特に指定がない限り、これらのテーブルはメモリに格納されます。

CA Access Control 内部の名前解決は、NIS の名前解決よりもはるかに速く、/etc ファイルを使用するよりも高速に処理されます。つまり、内部の名前解決を使用すると、デッドロックが発生する危険性のない環境でもパフォーマンスが向上します。

注：lookaside データベースには、内部名前解決テーブル用のキャッシュはありません。CA Access Control はオープン ファイル ハンドルを使ってテーブルからデータを読み取ります。

名前解決テーブルのディスクへの保存

CA Access Control の名前解決テーブルは、CA Access Control の起動時に生成されます。 このテーブルは、メモリではなくディスクに格納する必要があります。メモリへの格納は、メモリの過負荷につながる可能性があるためです。 また、情報はメモリに読み込まれると更新されません。 このため、CA Access Control では、ユーザ、グループ、またはホストに関する情報の変更が認識されません。 メモリ内のテーブルを更新するには、CA Access Control を再起動する必要があります。

最新データを維持するために、CA Access Control には内部の名前解決テーブルを確実にディスクに格納する lookaside データベースが用意されています。

注：lookaside データベースを実装するには、seos.ini の設定を使用する必要があります。 seos.ini の設定の詳細については、「リファレンス ガイド」を参照してください。

lookaside データベースの設定

lookaside データベースは、userdb.la、groupdb.la、hostdb.la、および servdb.la という 4 つのテーブルで構成されます。 この 4 つのテーブルは、ユーザ、グループ、ホスト、およびサービスの名前解決要求を処理します。 これらのテーブルは、seos.ini ファイルの lookaside_path トークンで指定されるディレクトリに格納されます。このディレクトリは、デフォルトでは /opt/CA/AccessControl/ladb です。

4 つのテーブルを含む lookaside データベース

4 つのテーブルを含む lookaside データベースを設定するには、以下のいずれかの操作を行います。

- CA Access Control をインストールする場合に、lookaside データベースを作成するかどうかを確認するメッセージが表示されたら、yes を指定します。
- CA Access Control がすでにインストールされている場合は、以下の手順に従います。
  a. seos.ini の[seosd]セクションで、以下のトークンを yes に変更します。
     - under_NIS_server
     - use_lookaside
  b. sebuildla -a を実行して、4 つのテーブルをすべて作成します。

3 つ以下のテーブルを含む lookaside データベース

作成するテーブルは、3 つ以下でもかまいません。 たとえば、lookaside データベースを使用して、ホストのみを解決する場合は、以下の手順に従います。

1. CA Access Control をインストールした後、seos.ini ファイルの[seosd]セクションで以下のトークンを変更します。
   - under_NIS_server を空に設定します。
   - HostResolution を ladb に設定します。
2. sebuildla -h を実行して、ローカル ホストと DNS ホストを含むすべてのホストで構成されるテーブルを作成します。

   または

   sebuildla -e を実行して、(/etc/hosts で定義された)ローカル ホストのみのテーブルを作成します。

他のテーブルを含む lookaside データベースを作成するには、seos.ini ファイルの適切なトークンを使用して、sebuildla で適切なオプションを実行します。

注: これらのトークンの詳細については、「リファレンス ガイド」の seos.ini 初期設定ファイルの説明を参照してください。 sebuildla の詳細については、「ユーティリティ ガイド」を参照してください。

重要: ホストを追加する場合は、常に sebuildla を実行してください。

lookaside データベースの機能

lookaside データベースの 4 つのテーブル(groupdb.la、hostdb.la、servdb.la、userdb.la)には、グループ、ホスト、サービス、およびホストの名前を解決するための情報が格納されています。 これらのテーブルは、seos.ini ファイルの lookaside_path トークンで指定されるディレクトリに格納されます。このディレクトリは、デフォルトでは /opt/CA/AccessControl/ ladb です。

CA Access Control 内部の名前解決は、NIS の名前解決よりもはるかに速く、/etc ファイルを参照するよりも高速に処理されます。

lookaside データベースの実装

注: ここで扱う問題とその解決法はあくまで参考用です。 インストール時に適切に設定されているため、ほとんどのユーザは設定を変更する必要はありません。

CA Access Control での lookaside データベースの実装方法の概要は、以下のとおりです。

- seos.ini ファイルの関連するトークンが設定されます。
- /opt/CA/AccessControl/exits ディレクトリ内の関連するシンボリック リンクが定義されます。
- lookaside データベースを作成するために、/opt/CA/AccessControl/bin/sebuildla -a コマンドが発行されます。

sebuildla ユーティリティは、E4703 ファイルや NIS などのネイティブの解決メカニズムを利用して、look-aside データベースを作成します。

機密情報(パスワード、ホーム ディレクトリの場所、gecos など)は lookaside テーブルに格納されます。 lookaside データベース テーブルには、ID 番号の数値と名前のみが格納されます。

lookaside データベースが作成されたら、sebuildla ユーティリティを使用してこのデータベースを更新します。 CA Access Control を再起動する必要はありません。

ホストの lookaside テーブルの更新

ホストの lookaside テーブルは更新する必要があります。 これを行うには、定期的に sebuildla -h ユーティリティを実行します(間隔はサイトによって異なります)。 この処理には、cron ジョブを使用します。

selang のコマンドを使用して UNIX のユーザまたはグループのデータベースを変更するたびに、sebuildla ユーティリティを実行する必要があります。 CA Access Control にはこの目的のための exit スクリプトが用意されています。このスクリプトは、適切なパラメータを使用して sebuildla ユーティリティを実行します。

付録 B: メインフレームとのパスワード同期

このセクションには、以下のトピックが含まれています。

パスワードの同期のサポート

CA Access Control では、CA Access Control を実行している Windows または UNIX マシンと、CA Top Secret、CA ACF2、または RACF セキュリティ製品(および CA Common Services CAICCI パッケージ)を実行しているメインフレームとのパスワード同期をサポートしています。 同期は、CA Access Control の標準のパスワード Policy Model 方式によって実現します。

パスワード Policy Model 方式

ネットワークに存在するメインフレームとのパスワード同期を実装するには、CA Access Control を実行している Windows マシンをメインフレームの親として選択し、メインフレームのパスワード同期オプションがインストールされていることを確認します。 次に、CA Access Control に対してメインフレームを定義し、親となる Windows マシンからパスワード Policy Model にメインフレームをサブスクライブします。 このような設定を行うと、メインフレームのユーザがパスワードを変更するたびに、パスワード Policy Model 階層内のすべてのマシンにその変更が伝達されます。

メインフレーム管理者にパスワードを変更するための CA Access Control の権限を与えた場合、メインフレームでその管理者が実行したユーザ パスワードの変更、ユーザ アクションの一時停止または再開は、メインフレームからパスワード Policy Model の階層全体に伝達されます。 同様に、管理者がパスワード Policy Model 階層で実行したパスワード変更、ユーザ アクションの一時停止または再開も、メインフレームに伝達されます。

パスワード同期のインストール

メインフレーム側のインストール要件

パスワード Policy Model の階層に追加する各メインフレームに、CA Common Services をインストールする必要があります。 このインストールの手順、およびメインフレームにパスワード同期を設定する方法については、以下を参照してください。

- CA ACF2 Security については、「CA ACF2 Administrator Guide」を参照してください。
- CA Top Secret については、「CA Top Secret Administrator Guide」を参照してください。
- RACF については、CA Access Control Endpoint Components for UNIX DVD のメインフレーム ディレクトリを参照してください。

注: UNIX 側から CAICCI 接続を確立することをお勧めします。 これを行うには、CCIRMTD ファイルにメインフレームへの接続を追加します。メインフレームの CAICCI パラメータには NODE ステートメントも CONNECT ステートメントも追加しないでください。

UNIX 側のインストール要件

パスワード同期用にメインフレームの親として使用する各 UNIX マシンに、[メインフレームのパスワード同期]オプションを指定して CA Access Control をインストールする必要があります。

注: CA Access Control をすでにインストールしている場合は、インストール スクリプトを再実行し、[メインフレームのパスワード同期]オプションを選択します。 再インストールを実行しても、現在の CA Access Control データベースや設定が変更されることはありません。

インストールを始める前に、このマシンから Policy Model にサブスクライブする各メインフレームのホスト名、SYSID、および管理者名を調べておきます。 ただし、インストールの時点でこの情報が不明な場合は、この部分を省略し、後でメインフレームをサブスクライブしてもかまいません。

「install_base mfsd」を使用して、MFSD パッケージをインストールします。SERVER パッケージは MFSD パッケージをインストールする前にインストールする必要があります。MFSD パッケージと共に、CAICCI というパッケージも自動的にインストールされます。このパッケージは、CA Common Communication Interface のスタンドアロン バージョンです。

すでに Unicenter TNG がインストールされている場合、mfsd デーモンは CAICCI Communication パッケージを使用します。 この場合、CAICCI はインストールされません。 Unicenter TNG が実行されている場合は、代わりに Unicenter TNG がシャットダウンされ、Unicenter TNG のセキュリティ オプション ファイルである UniDir/secopts がアップグレードされます(ここで、UniDir は Unicenter TNG のインストール ディレクトリです)。CA Access Control をインストールした後、必ず Unicenter TNG を再起動してください。 Unicenter TNG がインストールされていない場合は、CA Access Control の */uni/cci ディレクトリに CAICCI パッケージがインストールされます。

インストールにより、Policy Model にホストをサブスクライブできます。 メインフレームのホスト名と SYSID がわかっている場合は、ここでサブスクライブできます。 インストールの時点でこの情報が不明な場合は、この手順を省略して、後でホストをサブスクライブすることができます。 ホスト名を後で入力する場合は、その際に以下の行を入力して ACInstallDir/uni/cci/config/hostname/ccirmtd.prf を更新する必要があります。

"REMOTE = mainframeName mainframeSysid 1024 startup port=1721"

インストールの確認

1. CA Access Control の */lbin ディレクトリに、mfsd ファイルおよび unixcpfd ファイルがインストールされます。

   Unicenter TNG がインストールされていない場合は、CA Access Control の */uni/cci ディレクトリに CAICCI パッケージがインストールされ、/usr/local/CAlib ディレクトリに共有ライブラリ セットが追加されます。

2. サービスの開始と停止は、CA Access Control の */bin ディレクトリにある mfscntrl スクリプトで行われます。

3. Unicenter TNG がインストールされていない場合は、プロファイル ファイルが更新され、CAIGLBL0000 環境変数が CA Access Control の */seos/uni ディレクトリに設定されます。 この変数を手動で定義する必要がある場合は、以下のコマンドを入力します。

   setenv CAIGLBL0000 ACInstallDir/uni

   ACInstallDir

   CA Access Control のインストール ディレクトリを定義します。デフォルトでは、/opt/CA/AccessControl です。

mfsd の設定

mfsd を機能させるには、いくつかのファイルを設定する必要があります。

1. CAICCI 環境設定ファイルを設定します。
2. mfsd を実行する UNIX マシンに CA Access Control データベースを設定します。
3. メインフレーム更新プログラムを配布する Policy Model を設定します。

スクリプトを使用して msfd を設定します。 このスクリプトは ACInstallDir/bin/mfsdconf.sh にあります(ここで、ACInstallDir は CA Access Control のインストール ディレクトリです)。 以下のセクションの説明に従い、すべての設定を手動で実行することもできます。

このスクリプトを実行する前に、以下のものを用意してください。

- 伝達にメインフレーム更新プログラムを使用する Policy Model。
- Policy Model に更新プログラムを渡すメインフレームの SYSID とメインフレーム名、および管理者のユーザ。

変換ファイルの設定

ACInstallDir/data ディレクトリにある変換ファイル trans_mfsd.txt の詳細については、以下の事項を参照してください。

- 最初の行は、mfsd で使用される文字 sub を含むパターンです(sub は selang のコマンドに挿入される語を識別するための文字です)。 このパターンは、メインフレームのコマンドを同じ文字で selang のコマンドにコピーしたもの(そのままのコピー)を表しています。

  このパターンは変更できます。ただしその場合は、ファイル内に記述されているこのパターンをすべて変更する必要があります。 例はファイルで確認してください。
- メインフレームのコマンドと、それを selang のコマンドに変換したものとのペアを書き留めます。
- RACF セクションで、かっこを含むコマンドとかっこを含まないコマンドを変換する場合、かっこを含まない行の前に、1 単語をかっこで囲んだコマンドを記述する必要があります。 変換ファイルでは大文字と小文字が区別されます。
- コマンドでは、スペースが非常に重要です。
- コマンドでは以下の擬似的な正規表現を使用できます。

文字	機能
-	0 文字以上の文字に一致します。
?	1 文字に一致します。
[ ]	範囲または値リストを区切ります。 範囲は以下のように指定できます。 ■ 全範囲 [az] ■ 指定した範囲を除外 [^az] ■ プレフィクス不完全 [z] ■ プレフィクス不完全の否定 [^z] ■ フィックス不完全 [a] ■ フィックス不完全の否定 [^a] 値は以下のように指定できます。 ■ 指定した文字を含める [abf] ■ 指定した文字を除外する [^abf]

- 変換後の新しいコマンドをファイルに同様の形式で追加できます。
  - MF はメインフレームのコマンドです。
  - PM は変換後の selang のコマンドです。

変換ファイルの設定例 1

trans_mfsd.txt を以下のように設定した場合

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

MFSD による変換結果は以下のとおりです。 以下のメインフレームのコマンドが、その後の selang のコマンドに変換されます。

```
TSS REPLACE(username) PASSWORD(123)
env seos; cu username password(123)
```

変換ファイルの設定例 2

trans_mfsd.txt を以下のように設定した場合

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

MFSD により、RACF コマンドの ALT username PASSWORD(123) および ALT (username) PASSWORD(123) はいずれも次の selang のコマンドに変換されます。

```
env seos; cu username password(123)
```

trans_mfsd.txt ファイルの編集後、mfsd デーモンを再起動する必要があります。

exit 関数の定義

メインフレームのコマンドから変換した selang のコマンドを実行（mfsd 出力）前に編集する場合は、exit 関数を定義する必要があります。

exit 関数内部では、password フィールド以外の selang コマンド部分をすべて編集できます（セキュリティ上の理由により password フィールドは編集できません）。

API パッケージをインストールすると、このサンプルを確認できます。 このサンプルでは、username フィールドに文字列「isrdv」が含まれるコマンドを検索し、その部分を「ISRDV」に変更します。 たとえば、以下の最初のコマンドは、exit 関数の実行後、2 番目のコマンドのように変更されます。

```
cu isrdv01 password(*)
cu ISRDV01 password(*)
```

Policy Model の環境設定

パスワード同期に必要な環境設定を完了するには、メインフレームおよび親である UNIX システムの両方に適切なソフトウェアをインストールした後、UNIX システムで以下の手順を実行する必要があります。

1. mfsd を実行するローカル CA Access Control データベースで以下の変更を行います。

   a. mfsd を実行するユーザは root なので、ユーザ root を CA Access Control 管理者として定義し、ローカル端末に対する読み取りおよび書き込みアクセス権を設定する必要があります。

   ```
   editusr root admin
   auth terminal name.companyname.com uid(root) acc(r w)
   ```

   b. root を管理者として指定しない場合は、SPECIALPGM クラスを使用できます。

   SPECIALPGM クラスを使用すると、CA Access Control で UNIX ユーザまたは CA Access Control ユーザによって実行されるプログラムの機能が有効になります。 プログラムに UNIX ユーザおよび論理ユーザ(CA Access Control ユーザ)を割り当てます。 割り当てたユーザがプログラムを呼び出すと、CA Access Control は論理ユーザの権限を使用します。

   ```
   editusr mfs_adm admin
   auth terminal name.companyname.com uid(mfs_adm) acc(r w)
   newres specialpgm /opt/CA/AccessControl/lbin/mfsd ¥
   owner(nobody) unix(root) seosuid(mfs_adm)
   ```

   c. PMDB を作成します。次にこのデータベースを参照するように seos.ini の password_pmd トークンを設定し、Policy Model に接続します(host はローカル ホストまたはリモート ホストのどちらでもかまいません)。

   ```
   host pmd@
   ```

2. ユーザ root または mfs_adm を Policy Model の管理者として定義します。

   ```
   editusr mfs_adm admin
   ```

   a. Policy Model 内に TERMINAL レコードを作成します。 端末に対する読み取り権限と書き込み権限を mfs_adm に与えます(localhost は mfsd デーモンを実行するホストの名前です)。

   ```
   newres terminal name.companyname.com owner(nobody)
   auth terminal name.companyname.com uid(mfs_adm) acc(r w)
   ```

   b. Policy Model にサブスクライブする予定の各メインフレーム ホストの PMDB に、MFTERMINAL レコードを作成します(mfSYSID はメインフレームの SYSID です)。

   ```
   newres mfterminal mfSYSID defaccess(none) owner(nobody)
   ```

3. パスワード変更コマンドを発行する各メインフレーム管理者の USER クラスのレコードを PMDB に作成します。これらのユーザがパスワードの変更権限を持っていることを CA Access Control が認識できるように、ユーザには管理者権限を与えます。

   newusr mfAdmin admin

   ローカルの CA Access Control データベースにではなく、PMDB にレコードを作成すると、このレコードは Policy Model 階層内のすべてのホストに伝達されます。

4. 再び PMDB で、パスワード変更コマンドをどのメインフレームで発行しても MFTERMINAL レコードを読み取ることのできるアクセス権限をメインフレームの管理者に与えます（mfSYSID はメインフレームの SYSID、mfAdmin はメインフレーム管理者のユーザ名です）。

   authorize MFTERMINAL mfSYSID uid(mfAdmin) access(read)

5. メインフレームを Policy Model にサブスクライブします（インストール時にサブスクライブしなかった場合）。

   sepmd -sm pmdb unixhost.ca.com TSS sys1 user1

注: 上記の手順は、特定の順序で実行する必要はありません（ただし、当然ながら、管理者ユーザ レコードおよびメインフレーム MFTERMINAL レコードの両方を作成してからでなければ、メインフレーム管理者に MFTERMINAL レコードに対するアクセス許可を与えることはできません）。

メインフレーム同期の開始

同期サービスを実行するには、以下の 3 つのコンポーネントを起動する必要があります。

- CAICCI
- unixcpfd
- mfsd デーモン

Unicenter TNG を使用する場合は、Unicenter TNG も起動する必要があります。 プロセス ステータスをチェックして、3 つの CAICCI プロセス（caiccid、ccirmtd、および cciclnd）が実行中であることを確認します。

Unicenter TNG をインストールしてある場合は、UniDir/cci/bin/cciito ユーティリティを実行して、CAICCI が正常に動作しているかどうかをテストします。 ccirmtd.prf ファイルが正しく設定されている場合は、メインフレーム ホストの名前がモニタに表示されます。

Unicenter TNG がインストールされていない場合は、CA Access Control の install_base によって、スタンドアロンの CAICCI パッケージが追加されます。この場合は、CA Access Control ディレクトリから CAICCI デーモンを起動する必要があります。CAICCI を起動するには、mfscntrl ユーティリティを使用します。

```
unixhost:bin> mfscntrl start cci
cci デーモンを起動しています
```

接続が確立されたことを確認するには、以下のユーティリティを実行します。

```
unixhost:bin> /opt/CA/AccessControl/uni/cci/bin/ccii
```

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

CA Access Control のインストール時に CA Access Control ディレクトリの /uni/cci/config/hostname/ccirmtd.prf が更新されなかった場合は、CAICCI デーモンを起動する前に ccirmtd.prf を設定する必要があります。

Command Propagation Facility デーモン(unixcpfd)を起動します。 mfscntrl コマンドによって、CA Access Control の /lbin/ ディレクトリにある unixcpfd プログラムが実行されます。 プロセス ステータス レポートには、unixcpfd という名前のデーモンが 4 つ表示されます。 最初のデーモンは親プロセスです。2 番目のデーモンは CA-Top Secret メインフレーム システムからの更新を、3 番目のデーモンは RACF メインフレームからの更新を、4 番目のデーモンは CA ACF2 メインフレームからの更新を、それぞれ待機しています。

```
unixhost:bin> mfscntrl start unixcpfd
unixcpfd デーモンを起動しています。
```

unixcpfd が CAICCI サービスからメッセージを受け取ると、デーモンは /opt/CA/AccessControl/log/unixcpfd.log ファイルを作成します。 CA_CAIDEBUG 環境変数を yes に設定して、このシェル プロンプトから unixcpfd を実行すると、unixcpfd をトレース モードで実行できます。

メインフレーム同期デーモンを起動するには、以下のコマンドを使用します。 mfsd を実行する前に CA_CAIDEBUG 環境変数を yes に設定した場合は、トレース ファイル /opt/CA/AccessControl/log/mfsd.trace が作成されます。

```
unixhost:bin> mfscntrl start mfsd
unixhost:bin> Starting mfsd. PID = 5983
```

CAICCI 環境設定ファイル

CA Access Control では、インストール時、および CA Access Control Policy Model にメインフレームをサブスクライブするたびに、CAICCI 環境設定ファイルが自動的に更新されます。

何らかの理由で環境設定ファイルを手動で更新する必要がある場合は、以下の手順に従います。

1. CAICCI 環境設定ファイルを開きます。
   - Unicenter TNG を使用している場合は、以下のコマンドを入力します。

     UniDir/cci/config/hostname/ccirmtd.prf

     UniDir は Unicnter TNG のインストール ディレクトリです。
   - スタンドアロンの CAICCI を使用している場合は、以下のコマンドを入力します。

     ACInstallDir/uni/cci/config/hostname/ccirmtd.prf

     (ACInstallDir は CA Access Control のインストール ディレクトリで、デフォルトでは /opt/CA/AccessControl です)。

2. 以下の行を追加します。

   REMOTE = mainframeName mfSysid 1024 startup port=1721

   mfSysid はメインフレームの SYSID です。

3. ファイルを保存します。

4. リモート CAICCI サービスを停止します。

   - Unicenter TNG を使用している場合は、以下のコマンドを実行して Unicenter TNG サービスを停止します。

     unicntrl stop all

   - スタンドアロンの CAICCI を使用している場合は、以下のコマンドを使用します。

     ACInstallDir/uni/cci/bin/ccicntrl shutdown

5. リモート CAICCI サービスを再起動します。

   - Unicenter TNG を使用している場合は、以下のコマンドを実行して Unicenter TNG サービスを起動します。

     unicntrl start all

   - スタンドアロンの CAICCI を使用している場合は、以下のコマンドを使用します。

     ACInstallDir/bin/mfscntrl start cci

付録 C: CA Audit の統合

このセクションには、以下のトピックが含まれています。

CA Audit の統合

CA Access Control と CA Audit を統合するには、CA Access Control iRecorder を使用することをお勧めします。

注: CA Access Control iRecorder を使用して CA Audit と統合する方法の詳細については、「iRecorder Integration Guide for CA Access Control」を参照してください (http://www.ca.com/jp/support/)。

この附録では、CA Access Control iRecorder を使用しないで CA Access Control と CA Audit を統合する概要と、ログを CA Audit に送信するように CA Access Control を設定する方法について説明しています。

CA Access Control はホスト ベースであるため、その監査ログは各ホスト上でローカルに保存されます。 これらのログは、UNIX の場合、CA Access Control のネイティブ ツールを使用して中央に収集できます。ただし、Windows の場合、この機能は CA Access Control では使用できません。 また、重要なレポートを取得したりイベントの対比を行うには、中央に収集したログ ファイルに対してネイティブ UNIX ツール (awk や sed など) を使用する必要がありますが、この操作は煩雑で時間もかかります。

CA Audit は、Windows のイベント ログや UNIX の syslog などのさまざまなリソースから監査ログを収集して、そのログを 1 つの ODBC データベースに保存します。 CA Audit には、レポートおよびイベント相関のためのツールが用意されています。データはテキスト ファイルではなく、データベースに格納されているので、操作がより簡単です。

情報をデータベースに保存することによって得られるあらゆる利点を活用するために、CA Access Control でログを CA Audit に送信するように設定できます。

CA Access Control 監査ログ

デフォルトでは、CA Access Control は以下のイベントを監査ログというファイルに記録します。

- データベースのルールへのすべての変更
- すべてのアクセスの失敗

他のイベントを監査するように定義するには、リソース ACL を変更します。

CA Access Control エンドポイント管理 を使用して監査ログを読み取ります。 seaudit ユーティリティを使用することもできます。 監査ログ ファイルはバイナリ ファイルなので、テキスト エディタで読み取ることができません。

CA Access Control はすべての CA Access Control サーバ上に監査ログを作成します。監査ログの完全ファイル名は以下のとおりです。

- ACDir/log/seos.audit（UNIX システムの場合）
- ACDir¥log¥seos.audit（WIndows システムの場合）

監査設定に基づき、CA Access Control は監査ログ ファイルが特定のサイズまたは時間に達するたびにバックアップ ファイルを作成し（ファイルの既存のバックアップ コピーが上書きされる可能性があります）、新しい監査ログを開始します 古い監査ログについては、上書きされる前にアーカイブすることを考慮する必要があります。

CA Access Control が監査ログ ファイルを作成および変更する方法を変更するには、以下の方法に従います。

- ファイルが特定のサイズに達したときにバックアップの監査ログ ファイルが自動的に作成されるように、監査設定を変更します。

  注： CA Access Control ログ ルータは、定期的に作成されるログ ファイルを読み取ることができません。 ログ ファイルを定期的にバックアップする場合は、CA Access Control iRecorder を使用して CA Access Control と CA Audit を統合します。

- CA Access Control が監査ログを自動的にバックアップするサイズを変更します。

このような変更を行う方法は、CA Access Control が UNIX システムまたは Windows システムのどちらに存在するかによって異なります。

CA Access Control の設定方法

CA Access Control のログを CA Audit に収集するには、以下の手順に従います。

UNIX の場合

CA Access Control がすでに UNIX サーバにインストールされて実行され、ユーザは root としてログインし、root は CA Access Control の Security Administrator として定義されていることを前提とします。

1. CA Access Control を停止します。

   ```
   # ACDir/bin/secons -s
   ```

2. CA Access Control を起動するたびにそのログ ルーティング デーモン selogrd を自動的に起動させるには、以下のコマンドを入力します。

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

3. selogrd.cfg という名前の新規ファイルを作成します。

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

4. このファイルに次の 3 行を入力します。

   ```
   Hostrule
   hostname_or_IP_address_of_CA_Audit_Router
   .
   ```

   注: selogrd.cfg ファイルの最後の行は、ピリオドで終了する必要があります。

   作成した単純な設定により、すべての監査レコードが CA Access Control から CA Audit に送信されます。

   CA Audit ルータ サーバが正しく識別されたことを確認するには、以下の手順に従います。

   - システムで DNS が使用されている場合、サーバの DNS 完全修飾名を入力します。 以下に例を示します。

     ```
     name.companyname.com
     ```

   - システムで検索にローカルの /etc/hosts ファイルを使用している場合、/etc/hosts ファイルに最初に表示されているサーバの名前を入力します。 たとえば、/etc/hosts ファイルに以下のように表示されている場合は、「name01」と入力します。

     ```
     141.202.243.10      name01      name01.companyname.com
     ```

     サーバの名前を確認できない場合は、サーバの IP アドレスを入力します。

     システムで使用されているホスト名の解決方法を確認するには、/etc/nsswitch.conf ファイルをチェックして、hosts トークンを検索します。このトークンは、解決の順番を指定します。

5. CA Access Control のデーモンを起動するには、次のコマンドを入力します。

   ```
   # ACDir/bin/seload
   ```

   4 つのデーモンが起動します。 最後のデーモンは selogrd デーモンです。

CA Audit の環境設定

CA Access Control のログを CA Audit データベースで受け取るためには、CA Audit ルータが実行されている場所を判別する必要があります。 これは、CA Audit の実装によって異なることがあります。 一般的な事例のいくつかを以下で説明します。

Audit への CA Access Control for UNIX ログの収集

(ネイティブ UNIX の syslog や sulog ではなく)CA Access Control for UNIX のログのみを Audit に収集することが目的の場合、CA Audit コレクタのあるサーバと同じサーバ上で CA Audit ルータを設定することができます。

この事例、すなわち CA Access Control を実行する 100 台の UNIX サーバに実装する場合、どの UNIX サーバにも CA Audit エージェントを追加インストールする必要はありません。

CA Audit コレクタおよび CA Audit ルータは同じ Windows サーバにインストールできます。そのため、 CA Access Control を実行するすべての UNIX サーバでは、前のセクションで説明した selogrd.cfg ファイルでこのサーバを参照できます。

以下の操作を実行するには、CA Audit のマニュアルを参照してください。

- CA Audit で監査ノードを作成します。
- CA Audit ポリシー マネージャで CA Access Control のポリシーを作成します。
- ポリシーを有効にして配布します。

CA Audit への CA Access Control for UNIX ログ、syslog、および sulog の収集

CA Access Control for UNIX のログに加えて、ネイティブ UNIX の syslog および sulog も収集することが目的の場合は、CA Access Control を実行している UNIX サーバに CA Audit クライアントをインストールする必要があります。

この事例では、CA Access Control が実行されている UNIX サーバと同じサーバ上で CA Audit ルータが実行されています。 selogrd.cfg ファイルのエントリは、localhost に設定する必要があります。

以下の操作を実行するには、CA Audit のマニュアルを参照してください。

- CA Audit で 2 つの監査ノードを作成します。
- CA Audit ポリシー マネージャで CA Access Control と UNIX の 2 つのポリシーを作成します。
- ポリシーを有効にして配布します。

CA Audit への CA Access Control for Windows ログの収集

CA Access Control を Windows サーバで実行していてログを CA Audit に集中させるには、CA Access Control for Windows を実行しているすべてのサーバに CA Audit クライアントをインストールする必要があります。

以下の操作を実行するには、CA Audit のマニュアルを参照してください。

- CA Audit で監査ノードを作成します。
- CA Audit ポリシー マネージャで CA Access Control のポリシーを作成します。
- ポリシーを有効にして配布します。