AIX バージョン 7.2

4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4

IBM

AIX バージョン 7.2

4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4

IBM

お願い

本書および本書で紹介する製品をご使用になる前に、 73 ページの『特記事項』に記載されている情報をお読みください。

本書は AIX バージョン 7.2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： AIX Version 7.2
4765 PCIe Cryptographic Coprocessor
AIX CCA Support Program Installation
4.4

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

このインストール・ガイドでは、IBM 4765 PCIe 暗号化コプロセッサー用の IBM® Common Cryptographic Architecture (CCA) サポート・プログラム (以下、サポート・プログラムという) のリリース 4.4 について説明します。このサポート・プログラムにはデバイス・ドライバー、ユーティリティー、および CCA コプロセッサー・コードが含まれています。

本書は、以下のタスクに役立てるために使用してください。
- インターネットからサポート・プログラムを取得する
- ソフトウェアをホスト・コンピューターおよびコプロセッサーにロードする
- サポート・プログラムで提供されているユーティリティーを使用して、以下の作業を行う。
  - コプロセッサーの機能制御ベクトル (FCV) をロードする
  - 1 つ以上のコプロセッサーを初期化する
  - アクセス制御データを作成および管理する
  - マスター・キーと 1 次鍵暗号鍵 (KEK) を作成する
  - 暗号化ノードで鍵ストアを管理する
  - ほかの暗号化ノードをセットアップおよび構成するためのノード初期化ファイル・リストを作成する
- アプリケーション・ソフトウェアを CCA ライブラリーにリンクする
- アプリケーション開発および操作慣習でのセキュリティーに関する考慮事項のガイダンスを取得する

対象読者

本書の対象読者としては、以下のユーザーが挙げられます。
- ソフトウェアをインストールするシステム管理者
- コプロセッサーのアクセス制御システムの責任を負うセキュリティー担当者
- ソフトウェアの使用方法を決定するシステム・プログラマーおよびアプリケーション・プログラマー

強調表示

本書では、以下の強調表示規則を使用しています。

太字	名前がシステムによって事前定義されているコマンド、サブルーチン、キーワード、ファイル、構造体、ディレクトリー、およびその他の項目を示します。さらに、ユーザーが選択するボタン、ラベル、 およびアイコンなどのグラフィカル・オブジェクトも示します。
イタリック	ユーザーが実際の名前や値を指定するパラメーターを示します。
モノスペース	具体的なデータ値の例、画面に表示されるテキストの例、プログラマーとしてユーザーが記述するプログラム・コード部分の例、システムからのメッセージ、またはユーザーが実際に入力する情報を示します。

AIX® における大/小文字の区別

AIX オペレーティング・システムでは、すべてケース・センシティブとなっています。これは、英大文字と小文字を区別するということです。例えば、**ls** コマンドを使用するとファイルをリストできます。`LS` と入力すると、システムではそのコマンドが「`not found`」(見つからない) と応答します。同様に、`FILEA`、`FiLea`、および `filea` は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

関連資料

PCIe 暗号化コプロセッサーおよび一般的な市販の暗号アプリケーションの資料は、次のとおりです。

暗号ハードウェアの資料は、*CryptoCards* Web サイト (http://www.ibm.com/security/cryptocards) で入手可能です。

- *IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and the IBM 4764 PCI-X Cryptographic Coprocessors*

4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4

この情報を効果的に使用するためには、コマンド、システム呼び出し、サブルーチン、ファイル・フォーマット、および特殊ファイルを熟知している必要があります。

4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4 の新機能

「4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4」のトピック集の新規または著しく変更された情報についてお読みください。

新規情報または変更情報の参照方法

この PDF ファイルでは、左マージンに新規および変更情報を示すリビジョン・バー (|) が表示される場合があります。

2015 年 12 月

以下の説明は、このトピック集に加えられた更新の要約です。

- IBM PCIe Cryptographic Coprocessor は、以下のトピックで IBM PCIe Cryptographic Coprocessor バージョン 4.4.55 に更新されました。
  - 『4765 PCIe Cryptographic Coprocessor AIX CCA サポート・プログラムのインストール・ガイド 4.4』
  - 4 ページの『サポート・プログラム・ベース リリース 4.4 のインストール』
  - 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』

  コプロセッサー・ソフトウェアを IBM PCIe Cryptographic Coprocessor http://www-03.ibm.com/security/cryptocards/pciecc/release4455.shtml Web サイトからダウンロードできます。

サポート・プログラムのインストール・プロセスの概要

AIX CCA のこの概要では、IBM 暗号化コプロセッサー・サポート・プログラムをホスト・コンピューターにインストールするためのインストールと操作の手順について説明しています。

関連情報:

2 ページの『サポート・プログラムのインストール』
このセクションでは、IBM Common Cryptographic Architecture (CCA) サポート・プログラムをコプロセッサー・ホスト・コンピューターにインストールする手順について説明します。

コプロセッサー・ハードウェアおよびソフトウェアの取得

このトピックは、コプロセッサー・ハードウェアの選択、インストール、および注文に関する情報を見つけ、ソフトウェアをダウンロードするのに役立ちます。

以下のセクションでは、次の実行方法について説明します。

- コプロセッサーの発注
- IBM 4765 コプロセッサーの注文方法
- IBM 4765 ハードウェアのインストール
- コプロセッサー・ソフトウェアの取得

コプロセッサーの注文

IBM 4765-001 は、IBM にマシン・タイプおよびモデルとして注文できます。コプロセッサーには、2/3 の長さの PCIe アダプターを受け入れる PCIe スロットが必要です。

ソフトウェアは、1 つのシステムにつき (使用可能な PCIe スロットの数に応じて) 最大 8 台のコプロセッサーをサポートします

IBM 4765 コプロセッサーの注文方法

コプロセッサー・ハードウェアを注文するには、お客様の地域の IBM 担当員または IBM ビジネス・パートナーに連絡して、選択したモデルとフィーチャーを注文してください。

米国内のお客様は、IBM Direct (1-800-IBM-CALL) にご連絡いただくこともできます。具体的に *IBM 4765* の名前を述べてください。それにより、お客様の注文が IBM 4765 の注文を処理するグループに指図されます。

IBM 4765 ハードウェアのインストール

IBM 4765 は、他の PCIe アダプターと同様の方法でインストールされます。詳細については、「*IBM 4765 PCIe Cryptographic Coprocessor Installation 4.4*」に記載されたプロセスに従ってください。

コプロセッサー・ソフトウェアの取得

このソフトウェアは、以下の Web サイトからダウンロードして取得できます。http://www.ibm.com/security/cryptocards/pciecc/ordersoftware.shtml

サポート・プログラムのインストール

このセクションでは、IBM Common Cryptographic Architecture (CCA) サポート・プログラムをコプロセッサー・ホスト・コンピューターにインストールする手順について説明します。

IBM Common Cryptographic Architecture (CCA) サポート・プログラムは、次の複数のコンポーネントで構成されています。

- PCIe 暗号化コプロセッサー・ハードウェア用デバイス・ドライバーおよびオペレーティング・システム
- IBM Common Cryptographic Architecture (CCA) アプリケーション・プログラム・インターフェース (API)
- 機能制御ベクトル (FCV)

注**:** FCV は、IBM によって提供されている符号付きの値です。コプロセッサー内の CCA アプリケーションが、適用される暗号輸出入規制に合致した暗号サービスの水準を提供することを可能にします。

- ホスト・マシンで実行されるコプロセッサーがインストールされなければならないユーティリティー・アプリケーション。

IBM Common Cryptographic Architecture (CCA) サポート・プログラムをインストールし、構成するには、以下の手順を実行します。

1. セットアップに適したプラットフォーム・サポート・パッケージを選択します。

   AIX 6.1 以降。

   詳しくは、 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。
2. IBM または IBM ビジネス・パートナーにハードウェアを発注します。IBM にコプロセッサー・ハードウェアを発注し、受け取る方法を説明している 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。
3. ご使用のオペレーティング・システム用のサポート・プログラムをダウンロードします。組み込みオペレーティング・システムおよび CCA アプリケーション・プログラムを PCIe Cryptographic Coprocessor にインストールする方法を説明している 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。
4. サポート・プログラムをコプロセッサー・ホスト・コンピューターにインストールします。
5. コプロセッサー・ハードウェアを取り付けます。詳しくは、 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。
6. コプロセッサー・ソフトウェアをロードします。詳しくは、 8 ページの『コプロセッサーへのソフトウェアのロードとアンロード』を参照してください。
7. CCA テスト・ノードをセットアップします。サポート・プログラムで提供されているユーティリティーを使用するか、ご使用のアプリケーション・プログラムを CCA API にリンクすることにより、CCA 暗号化ノードを確立することができます。IBM 4765 での使用を計画しているアプリケーション・ソフトウェアによって課せられたアクセス制御およびその他のセットアップ要件の妥当性の検証も行ってください。 18 ページの『CNM および CNI ユーティリティーを使用した暗号化ノードの管理』で説明されている CCA Node Management (CNM) ユーティリティーには、以下を実行するために必要なセットアップ機能と管理機能が含まれています。
   - FCV のロード
   - アクセス制御データの作成および編集
   - コプロセッサー・マスター・キーの管理
   - 1 次鍵暗号鍵 (KEK) の管理
   - データ・キーのストレージの管理
   - CCA Node Initialization (CNI) ユーティリティー用のリスト (scripts) の作成
8. CCA ライブラリーを使用するテスト・プログラムを実行します。詳しくは、 44 ページの『CCA API で使用するためのアプリケーションの作成』を参照してください。

関連情報:

2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』
このトピックは、コプロセッサー・ハードウェアの選択、インストール、および注文に関する情報を見つけ、ソフトウェアをダウンロードするのに役立ちます。

8 ページの『コプロセッサーへのソフトウェアのロードとアンロード』
ホスト・コンピューターに IBM Common Cryptographic Architecture (CCA) サポート・プログラムをインストールした後、Coprocessor Load Utility (CLU) を使用してコプロセッサー・オペレーティング・システムと CCA アプリケーションをコプロセッサーにロードします。

18 ページの『CNM および CNI ユーティリティーを使用した暗号化ノードの管理』
鍵の生成およびデジタル署名サポートなどの暗号化サービスを提供するコンピューターは、ここでは暗号化ノードと定義されています。

サポート・プログラム・ベース リリース 4.4 のインストール

このセクションでは、サポート・プログラムをコプロセッサー・ホスト・コンピューターにインストールする手順を説明しています。

前提条件

インストールを開始する前に、セットアップに適したプラットフォーム・サポート・パッケージを選択してください。 AIX のソフトウェア要件とハードウェア要件について詳しくは、 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。

注**:** 初めてこのプログラムをインストールする場合以外は、鍵ストレージ・ファイルをバックアップしてください。

サポート・プログラムをインストールするには、以下の手順に従ってください。

1. **smitty install_all** コマンドを入力します。
2. 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』の『コプロセッサー・ソフトウェアの取得』セクションに記載された手順を使用して取得したインストール・イメージのロケーションを入力します。Enter キーを押します。
3. 「ソフトウェア・インストール **(SOFTWARE install)**」フィールドに「`csufx.4765.cca csufx.4765.man`」と入力するか、F4 (Display) を押してリストから選択します。「前提条件ソフトウェアを自動インストールする **(AUTOMATICALLY install requisite software)**」が「はい (yes)」、「新規ご使用条件を受諾する **(ACCEPT new license agreements)**」が「はい (yes)」に設定されていることを確認します。タブ・キーを使用して切り替えるか、F4 (Display) キーを使用してリストします。Enter キーを押し、「本当によろしいですか (ARE YOU SURE)」のプロンプトが出されたら、もう一度 Enter キーを押して続行します。
4. F10 (Exit) キーを使用して **smitty** を終了します。
5. `/usr/lpp/csufx.4765/README` ファイルを読んでください。このファイルには、サポート・プログラム製品についての最新情報が記載されています。
6. 『サポート・プログラムの構成』の説明に従い、構成ユーティリティーを使用してソフトウェアを構成します。

サポート・プログラムの構成

このセクションでは、CCA 暗号化コプロセッサー・サポート・プログラム・ソフトウェアの構成に使用されるユーティリティーとシステム・コマンドについて説明します。

csufadmin

> csufkeys、csufappl、csufclu (Coprocessor Load Utility)、csufcnm (Cryptographic Node Management)、および csufcni (Cryptographic Node Initialization) ユーティリティーに関連付けられたシステム・アクセス権を指定します。

デフォルト・アクセス権はこれらのユーティリティーの使用を、root ユーザーとシステム・グループ内のユーザーに制限しています。csufadmin ユーティリティーを使用してこれらのアクセス権を変更します。

csufappl

CCA ライブラリーに関連付けられたシステム・アクセス権を指定します。

デフォルト・アクセス権は CCA ライブラリーの使用を、root ユーザーとシステム・グループ内のメンバーに制限しています。CCA API で提供されているサービスの使用を他のグループに許可するには、csufappl ユーティリティーを使用します。

csufkeys

暗号鍵と鍵リストが保管されているロケーションのファイル名とディレクトリー名を作成および識別します。インストール・プログラムは、AIX オブジェクト・データ・マネージャー (ODM) に、以下のデフォルト・ディレクトリーを定義します。

- AES 鍵レコード・リスト・ディレクトリー: `/usr/lpp/csufx.4765/csufkeys/aeslist`
- AES 鍵ストレージ・ファイル: `/usr/lpp/csufx.4765/csufkeys/aes.keys`
- DES 鍵レコード・リスト・ディレクトリー: `/usr/lpp/csufx.4765/csufkeys/deslist`
- DES 鍵ストレージ・ファイル: `/usr/lpp/csufx.4765/csufkeys/des.keys`
- PKA 鍵レコード・リスト・ディレクトリー: `/usr/lpp/csufx.4765/csufkeys/pkalist`
- PKA 鍵ストレージ・ファイル: `/usr/lpp/csufx.4765/csufkeys/pka.keys`

ストレージ・ロケーションを変更するには、csufkeys ユーティリティーを使用します。

注**:** Cryptographic Node Management ユーティリティーを使用して鍵ストレージを初期化する場合は、必ずこのユーティリティーによって定義された ODM ディレクトリーを指定してください。

odmget

odmget システム・コマンドを使用して、鍵ストレージ・ファイル名を検証します。**odmget csufodm** コマンドを入力することにより、CCA サポート・プログラムで使用される鍵ストレージ名を検証することができます。4 個のパラメーター名属性は以下の値を指定します。

- **csuaesds**: AES 鍵レコードを含むファイル
- **csuaesld**: AES 鍵レコード・リスト・ファイルを含むディレクトリー
- **csudesds**: DES 鍵レコードを含むファイル
- **csudesld**: DES 鍵レコード・リスト・ファイルを含むディレクトリー
- **csupkads**: PKA 鍵レコードを含むファイル
- **csupkald**: PKA 鍵レコード・リスト・ファイルを含むディレクトリー

CNM ユーティリティーまたは csnbksi CCA verb によって CCA 鍵ストレージを初期化する場合は、ODM から返されたファイル名を使用する必要があります。これらのファイル名を変更するには、csufkeys ユーティリティーを使用します。

DES_Key_Record_List verb、PKA_Key_Record_List verb、および AES_Key_Record_List verb は、`/usr/lpp/csufx.4765/csufkeys/deslist`、`/usr/lpp/csufx.4765/csufkeys/pkalist`、および `/usr/lpp/csufx.4765/csufkeys/aeslist` ディレクトリーにそれぞれリスト・ファイルを作成します。これらはデフォルト・ディレクトリー名です。これらのディレクトリー名は、ソフトウェアをインストールする

ときに変更できます。リスト・サービスを要求すると、所有権の下にリスト・ファイルが作成されます。ファイルが、インストール環境によって要求されたグループ ID の下に作成されていることを確認してください。これは、これら 3 つのディレクトリーで set-group-id-on-execution ビットをオンに設定しても実行できます。詳しくは、**chmod** コマンドの g+s フラグを参照してください。この手順に従わないと、key-record-list verb でエラーが戻されます。

デフォルトの CCA コプロセッサーを割り当てるには、EXPORT コマンドを使用して環境変数 **CSU_DEFAULT_ADAPTER** を **CRP0***n* に設定します。この場合、どのインストール済み CCA コプロセッサーをデフォルトに指定するかによって、*n* は 1、2、3、4、5、6、7、または 8 になります。プロセスの最初の CCA verb が呼び出されたときにこの環境変数が設定されていないと、CCA ソフトウェアはコプロセッサー CRP01 をデフォルトとして使用します。この環境変数が無効な値に設定されている場合は、環境変数が有効な値に設定されるまでエラーを受け取ります。

関連情報:

41 ページの『鍵ラベルの作成』

CCA サポート・プログラムと AIX のファイル許可

CCA サポート・プログラムは、正しく機能するために、グループ・レベルのファイル許可に依存しています。

このプログラムが完全に機能し、エラーなしで実行されるためには、サポート・プログラムのユーザーと管理者が、CCA 共用ライブラリー、ユーティリティー、鍵ストレージ・ファイル、およびディレクトリーに対する正しいグループ・ファイル許可を持っている必要があります。

注**:** 鍵ストレージのファイルおよびディレクトリーは、鍵ストレージ・ディレクトリーに含まれているファイルおよびディレクトリーとして定義されます。このディレクトリーには、最上位の鍵ストレージ・ディレクトリー、すなわち、デフォルト構成で、`/usr/lpp/csufx.4765/csufkeys/deslist` ディレクトリーの下のすべてのファイルおよびディレクトリーと `/usr/lpp/csufx.4765/csufkeys` ディレクトリー自体が含まれます。

正しく機能するためには、鍵ストレージのファイルおよびディレクトリーが、アプリケーション・ユーザー・グループのグループ ID を持っている必要があります。これは、`csufappl` ユーティリティーの実行時に使用された groupname パラメーターです。

また、原則としてすべての鍵ストレージ・ディレクトリーは 2770 (drwxrws---) のファイル許可を持ち、root によって所有されている必要があります。すべての鍵ストレージ・ファイルは、660 (-rw-rw----) のファイル許可を持っている必要があります。

ライブラリーおよび ODM データベースでの競合のため、4765 CCA ソフトウェアおよび鍵ストアが、4764 CCA ソフトウェアおよび鍵ストアと同時に存在することはできません。

コプロセッサー・ハードウェア・エラーの検査

IBM Power Systems™ コプロセッサー・ハードウェアで発生したエラーは、AIX エラー・ログに記録されます。

このログを処理および表示するためには、以下のコマンドを入力してください。

```
errpt -a -N Cryptn,libxcrypt.a | more
```

この場合の *n* は、表示する CCA コプロセッサー・ログに応じて、0、1、2、3、4、5、6、または 7 (例えば、Crypt 0) になります。

関連情報:

8 ページの『コプロセッサーへのソフトウェアのロードとアンロード』

ホスト・コンピューターに IBM Common Cryptographic Architecture (CCA) サポート・プログラムをインストールした後、Coprocessor Load Utility (CLU) を使用してコプロセッサー・オペレーティング・システムと CCA アプリケーションをコプロセッサーにロードします。

サポート・プログラムの除去

鍵ストレージ・ファイルがデフォルト・ディレクトリーにある場合は、IBM Cryptographic Coprocessor (CCA) サポート・プログラムを除去する前にそれらをバックアップまたは保存してください。ソフトウェアを除去すると、デフォルト・ディレクトリーにある鍵ストレージ・ファイルは削除されます。

IBM Cryptographic Coprocessor サポート・プログラムを除去するには、以下の手順を実行します。

1. `root` としてログオンします。
2. **rmdev -dl Crypt0** コマンドを入力します。コプロセッサーのデバイス・ドライバーおよびその他の関連情報が除去されます。除去または再配置を計画している CCA コプロセッサーごとにこのコマンドを使用できます。
3. **smitty install_remove** コマンドを入力します。

   注**:** プロンプトが出されたら、`csufx.4765.com` および `devices.pciex.14107a0314107b03.rte` 製品名を入力します。
4. **REMOVE dependent software** 値が NO になっていることを確認します。また、**Preview Only** 値も NO になっていることを確認してください。
5. **Enter** キーを押します。

AIX ハードウェア要件およびソフトウェア要件

このトピックでは、CCA のインストールに必要な前提条件について説明します。

ハードウェア

4765 PCIe 暗号化コプロセッサーを搭載可能な IBM Power Systems サーバーをインストールします。

ソフトウェアのインストール中、ドライバーはコプロセッサーと対話して、割り込み設定、DMA チャネル、およびその他のシステム・リソースの調整を行います。コプロセッサー・ハードウェアおよびデバイス・ドライバーに関するインストール手順については、 2 ページの『コプロセッサー・ハードウェアおよびソフトウェアの取得』を参照してください。

ソフトウェア

1. IBM AIX 6.1 以降。
2. CCA Node Management (CNM) ユーティリティーを実行するために必要な Java ランタイム環境 (JRE) 1.6.0 以降。
3. http://www.ibm.com/security/cryptocards/pcixcc/ordersoftware.shtml の Web サイトからダウンロードしたソフトウェア・パッケージ **csufx.4765**。このソフトウェア・パッケージには、次のファイルセットが入っています。
   - **csufx.4765.com** - 4765 CCA サポート・プログラム

- **csufx.4765.cca** - 4765 サポート・プログラム - 共通ユーティリティー
- **csufx.4765.man** - サポート・プログラム マニュアル・ページ

ファイル許可

CCA Node Management (CNM) ユーティリティーによってファイル許可を管理します。

CCA Node Management (CNM) ユーティリティーは、アクセス制御点を管理する方法を提供します。CNM ユーティリティーの実行可能ファイルが誤って破損、または意図的に破壊されるのを防ぐために、`CNM.jar` ファイルのファイル許可は、読み取りおよび実行専用に設定します。同様に、アクセス制御点のデータ・ファイルを保護するために、`csuap.def` ファイルのファイル許可は読み取り専用に設定してください。

コプロセッサーへのソフトウェアのロードとアンロード

ホスト・コンピューターに IBM Common Cryptographic Architecture (CCA) サポート・プログラムをインストールした後、Coprocessor Load Utility (CLU) を使用してコプロセッサー・オペレーティング・システムと CCA アプリケーションをコプロセッサーにロードします。

サポート・プログラムの更新を取得した場合は、CLU を使用して必要なプログラム・セグメントを再ロードします。また、CLU を使用してベンダー・ソフトウェアをロードすることもできます。

このセクションには、以下の事項が含まれています。

- CLU を使用して、インストールされているコプロセッサーとその状況を理解し、コプロセッサー内で実行されているソフトウェアをインストールおよびアンインストールするための説明。
- 以下の事項について説明する参照セクション。
  - コプロセッサーのメモリー・セグメント
  - コプロセッサー状況の検証
  - CLU ユーティリティーの開始に使用される構文
  - CLU の戻りコード

コード・ロード制御をより深く理解し、コプロセッサーによって実装されるセキュリティーに関する考慮事項を確認するためには、研究論文「*Building a High-Performance, Programmable Secure Coprocessor*」を参照してください。この論文は、製品 Web サイトのライブラリー・ページ (http://www.ibm.com/security/cryptocards) から入手できます。

注**:**

1. このセクションで参照するファイル・ロケーションは、デフォルト・ディレクトリー・パスです。
2. コプロセッサーのデバイス・ドライバーによって戻されるエラー・コードは、16 進数の形式で提示されます (X'8040xxxx' など)。特に初めて CLU ユーティリティーを使用してこの製品と手順をあまり熟知していない場合に、これらのエラーを検出する可能性があります。
3. コプロセッサーの機能制御ベクトル (FCV) は、CCA Node Management (CNM) ユーティリティーによってロードされます。

関連情報:

51 ページの『デバイス・ドライバー・エラー・コード』
コプロセッサーのデバイス・ドライバーは、コプロセッサーおよびコプロセッサー・ハードウェア状況レジスターとの通信の状況をモニターします。

18 ページの『CNM および CNI ユーティリティーを使用した暗号化ノードの管理』
鍵の生成およびデジタル署名サポートなどの暗号化サービスを提供するコンピューターは、ここでは暗号化ノードと定義されています。

コプロセッサー・ソフトウェアのロード

ソフトウェアをコプロセッサーにロードするには、このセクションの手順を使用してください。

特定の .clu ファイル名については、インストールしているソフトウェア配布に添付されている README ファイルを参照してください。 README ファイルには、これらの一般手順を拡張または変更する追加情報も記載されている場合があります。

以下のサブトピックを使用し、以下のタスクの順に従ってください。

1. コマンド・プロンプトで、Coprocessor Load Utility (CLU) ファイルの入ったディレクトリーに変更し、CLU を実行します。
2. 現在コプロセッサー内に常駐するソフトウェアを判別します。
3. 必要に応じて、ソフトウェア・セグメント 1、2、および 3 の内容を変更します。
4. ソフトウェア・セグメントの最終的な内容の妥当性を検査します。

デフォルト・ディレクトリーの変更と CLU の実行

デフォルト・ディレクトリーを変更するには、コプロセッサー・コード・ファイル (*.clu) および Coprocessor Load Utility (CLU) の入ったディレクトリーを見つける必要があります。

デフォルト・ディレクトリーの変更

コマンド・プロンプトで、コード・ファイルにアクセスするために、デフォルト・ディレクトリーのコプロセッサー・コード・ディレクトリー /usr/lpp/csufx.4765/clu に変更します。CLU がデフォルト・ディレクトリーに入っていない場合は、必ずご使用のオペレーティング・システムが CLU の場所を見つけられることを確認してください。

CLU の実行

注: CLU を使用している場合は、CCA を使用するアプリケーションを実行しないでください。

CLU ユーティリティーを実行するには、コマンド・プロンプトに **csufclu** プログラム名を入力します。

パラメーターは対話式に CLU ユーティリティーに提供するか、コマンド行に組み込むことができます。CLU を使用するたびに、ログ・ファイル名を指定する必要があります。これは 1 番目のパラメーターであり、コマンド行に組み込むことができます。一般に特定のコプロセッサーを扱う場合は、コプロセッサーのシリアル番号をログ・ファイル名として使用するのが最善です。シリアル番号は、コプロセッサーの端にあるブラケット上のラベルから取得できます。

CLU は 2 個のログ・ファイルに情報を追加します。ログ・ファイルがない場合は、作成されます。1 つのログ・ファイルには、通常コンソール上に表示されるのと同じ情報が入ります。もう一方のログ・ファイル (CLU がファイル名拡張子として MRL を割り当てるログ・ファイル) には、機械可読ログが入ります。MRL ファイルは分析ユーティリティーに使用されます。

注: このセクションの後続の説明では、対話式に CLU を使用することが想定されています。コプロセッサーのコード・ファイルの入ったディレクトリーに変更します。 ご使用のオペレーティング・システムに適切な名前を使用して CLU を開始します。要求に従ってプロンプトに応答します。

CLU はデバイス・ドライバーからインストール済みコプロセッサーの番号を取得します。複数のインストール済みコプロセッサーがある場合は、CLU は、対話したいコプロセッサーの番号を要求します。これらの番号 (coprocessor_*number*) は、0 から 2 になります。これらの番号を特定のコプロセッサーと相互に関連付けるには、System Status (SS) コマンドを使用して、各インストール済みコプロセッサーの番号を確認してください。(出力の例については、Coprocessor Load Utility コマンドのトピックにある 18 ページの図 2 を参照してください。)

注**:** CLU ユーティリティーは、コプロセッサーの排他制御を取得すると、そのコプロセッサーで作動することができます。CCA verb 呼び出しを実行した他のアプリケーション (スレッドなど) が実行されていると、CCA のロードされたコプロセッサーは「ビジー」になるため、CLU は使用できません。

関連情報:

15 ページの『Coprocessor Load Utility の構文』

コプロセッサー・ソフトウェアのセグメント内容の判別

コプロセッサーには、3 個の「セグメント」(セグメント 1、セグメント 2、およびセグメント 3) があります。各セグメントには状況があり、ソフトウェアと検証公開鍵、および所有者の ID (セグメント 1 を除く) を保持します。

コプロセッサーのセグメントについては、表 1を参照してください。

表 *1.* ソフトウェア・セグメントの内容

セグメント	内容
1	「Miniboot」、診断およびコード・ロード制御を含む
2	組み込み制御プログラム
3	CCA または別のアプリケーション

コプロセッサー・セグメントの現在の内容と状況を判別するには、**ST** コマンドを使用します。 11 ページの図 1 は、標準の ST 応答を示しています。

```
====================================================================
CSUFCLU V4.1.1 st.log ST    begun Tue Sep 13 09:30:25 2011
*********** Command ST started. ---- Tue Sep 13 09:30:25 2011

 *** VPD data;  PartNum = 45D5117
 *** VPD data;  EC Num = 0G43192
 *** VPD data;  Ser Num = 99000543
 *** VPD data;  Description = IBM 4765-001 PCI-e Cryptographic Coprocessor
 *** VPD data;  Mfg. Loc. = 91
 *** ROM Status; POST0 Version 1, Release 27
 *** ROM Status; MiniBoot0 Version 1, Release 20
 *** ROM Status; INIT: INITIALIZED
 *** ROM Status; SEG2: RUNNABLE , OWNER2: 2
 *** ROM Status; SEG3: RUNNABLE , OWNER3: 2
 *** Page 1 Certified: YES
 *** Segment 1 Image: S0103 P1v0607 M1v011B P2v0706 F5180 201104151205401A000022000000000000
 *** Segment 1 Revision: 40105
 *** Segment 1 Hash: 177C AF13 C601 2276 90AA 8E20 D3BB BA58 79A6 7EBA 6C2A D68B 0A34 33E0 802C 4EA7
 *** Segment 1 Hash: 177C AF13
 *** Segment 2 Image: 4.1.7    y4_12-lnx-2011-03-04-16    201108111338401A000000000100010900
 *** Segment 2 Revision: 40107
 *** Segment 2 Hash: 698A 29DC EF8A 44D8 A025 3117 491B C552 45DA EC6F 0D0C 6671 BABE 7ABF 41E7 2FF5
 *** Segment 2 Hash: 698A 29DC
 *** Segment 3 Image: 4.1.7    CCA                         201108121155401A000000000000000000
 *** Segment 3 Revision: 40107
 *** Segment 3 Hash: EC02 B93A 309F 882A D859 031D 1F22 839D 2233 4D6A C58D D93C E43F 4A4C 1234 9F48
 *** Segment 3 Hash: EC02 B93A
 *** Query Adapter Status successful ***
Obtain Status ended successfully!
*********** Command ST ended. ---- Tue Sep 13 09:31:26 2011

  ...finishing up...
*********** Command ST exited. ---- Tue Sep 13 09:31:46 2011
```

図 *1.* 標準の *CLU* 状況応答

ST 応答のフィールドの定義は、次のとおりです。

フィールド
: **Description**

PartNum
: コプロセッサーの部品番号 (P/N)。

EC Num
: コプロセッサーの技術変更番号。

Ser Num
: コプロセッサーの製造元のシリアル番号。これは、保証の確認とダウンロードの許可に使用される IBM トラッキング・シリアル番号ではありません。

Description
: 一般用語でコプロセッサーのタイプを記述する文。監査員は、適切なコプロセッサーが使用されていることを監査するために、この記述と他の状況情報を検討する必要があります。

ROM Status
: コプロセッサーは常に、INITIALIZED 状態になっていなければなりません。状況が ZEROIZED の場合、コプロセッサーは潜在的な改ざんイベントを検出しており、リカバリー不能な、機能しない状態になっています。(コプロセッサーが正しく扱われないと、意図しない改ざんイベントが生じます。バッテリーは、推奨されたバッテリー交換手順に従っている場合にのみ取り外し、コプロセッサーを安全な温度範囲に保ち、指示に従ってください。

ROM Status SEG2 / SEG3

以下の状況を含め、セグメント 2 およびセグメント 3 のいくつかの状況条件が存在します。

- UNOWNED: 現在使用されていません。内容はありません。
- RUNNABLE: コードを含んでおり、使用可能な状態です。

所有者 ID も表示されます。標準の CCA サポート・プログラムでは、セグメント 2 とセグメント 3 の両方に ID 2 が割り当てられます。「それ以外の所有者 **ID**」は、そのソフトウェアが標準の IBM CCA 製品コードでないことを示しています。いずれの場合も、必ず正しいソフトウェアがコプロセッサーにロードされていることを確認してください。無許可または不明のソフトウェアは、インストール済み環境へのセキュリティー・リスクを示している場合があります。

Segment 1 Image

セグメント 1 のソフトウェア内容の名前と記述。工場から出荷されたコプロセッサーの場合、名前に **Factory** が含まれています。このイメージおよび関連付けられた検証キーを変更する必要があります。

以前にロードされたコプロセッサーの場合、セグメント 1 の名前は CCA を含んでいる可能性があります。必ず改訂レベルに注意してください 。

Segment 2 and Segment 3 Images

これらのセグメントが Owned 状況になっている場合は、イメージ名と改訂レベルに注意してください。IBM では、イメージが CCA サポート・プログラムの一部として提供されていることを示すために、イメージ名に CCA を含みます。必ず改訂レベルに注意してください。

Segment Hash values

各セグメントのハッシュ値は、 11 ページの図 1 に示された値と一致している必要があります。

ソフトウェア・セグメントの内容の変更

一般にコプロセッサー内のソフトウェアは、ホスティング・システム内の CCA ソフトウェアと同じリリース・レベルでなければなりません。

IBM から特に指示があった場合を除き、さまざまな異なるリリース・レベルを使用しないようにしてください。

Coprocessor Load Utility (CLU) を開始し、各パラメーターを対話式に入力してください。その方法については、 9 ページの『デフォルト・ディレクトリーの変更と CLU の実行』を参照してください。

1. ログ・ファイル名 (*nnnnnnnn*.**LOG**、ここで *nnnnnnnn* はコプロセッサーのシリアル番号) を入力します。
2. コマンド **PL** を入力します。
3. 複数のコプロセッサーがある場合は、コプロセッサー番号を入力します。
4. README ファイルに示されている CLU ファイル名を入力します。

セグメント 1、2、および 3 に適切なソフトウェアがロードされるように、必要に応じて繰り返してください。

コプロセッサー・セグメントの内容の検証

このセクションでは、コプロセッサー・セグメントの内容を検証するために従うべき手順について説明します。

セグメント 1、2、および 3 のコードをロードまたは置換した後、**CLU VA** コマンドを使用してセグメントの内容を確認し、コプロセッサーによって作成された応答のデジタル署名の妥当性を検査します。

使用されている IBM 4765 コプロセッサー (PartNum) に応じて、[1]次のコマンドを発行して、データ・ファイル名を、表 2 のクラス鍵 (class-key) 証明書ファイル名に置き換えます。データ・ファイル名 `v.clu` は、コプロセッサーの部品番号に追加される (すべて小文字) ことに注意してください。

csuxclu *nnnnnnn*`.log` **VA** [coprocessor_*n*] *datafile*

部品番号は、Coprocessor Load Utility (CLU) の ST コマンドを使用して取得できます。

表 *2. CLU VA* コマンドに使用するクラス鍵 *(class-key)* ファイル

PartNum	クラス鍵 **(class-key)** 証明書ファイル
12R8565	12r8565v.clu
41U0441	41u0441v.clu

[coprocessor_*n*] パラメーターは特定のコプロセッサーのオプション指定子で、デフォルトでゼロになります。

コプロセッサー・ソフトウェアのアンロードと CCA ノードのゼロ化

このトピックでは、コプロセッサー・ソフトウェアをアンロードし、CCA ノードをゼロ化して、セグメントの所有権を放棄する手順について説明します。

Coprocessor Load Utility (CLU) を使用してセグメント 2 の所有権を放棄するファイルを処理すると、セグメント 2 と従属セグメント 3 の両方がクリアされ、コードが除去されます。そのセグメントの検証公開鍵がクリアされ、コプロセッサー内に保持されていたそのセグメントのセキュリティー関連データ項目がゼロ化されます。所有者 ID がクリアされ、セグメントの状況が UNOWNED に設定されます。

セグメント 2 と 3 の所有権の放棄に使用される特定の `.clu` ファイル名については、ご使用のソフトウェア配布に添付された README ファイルを参照してください。README ファイルには、この一般手順を詳述または変更する追加情報も記載されている可能性があります。

以下のアクションを実行してください。

- CLU ファイルが入ったディレクトリーに変更します。
- CLU ユーティリティーを開始します。
- プロンプトに応答して、ログ・ファイル名内のコプロセッサーのシリアル番号を使用します。
- ご使用プラットフォーム用の README ファイルの指示に従って、**PL** コマンドを使用してセグメント 2 を放棄します。

注**:**

1. また、CCA 再初期化プロセスを使用することにより、ソフトウェアを除去せずに、CCA をゼロ化することもできます。
2. IBM では通常、コプロセッサーを工場出荷時の製品と同様の状態にするための、セグメント 1 の工場検証キーを復元するためのファイルを利用可能にしません。セグメント 1 の変更可能回数は制限されています。この回数を超えると、使用可能なデバイス・キーの証明書スペースが使い尽くされ、コプロセッサーが使用不可になる可能性があります。セグメント 1 の検証キーの復元機能が必要で、コプロセッサーがロック状態になる可能性にさらされてもかまわないという場合は、製品 Web サイト (http://www.ibm.com/security/cryptocards) の Support Form を使用して照会を送信することにより、IBM から必要なファイルを入手できます。証明書スペースは、更新不可リソースですので注意してください。使用した後は、リカバリーできません。

1. コプロセッサーの整合性を検査するための手順については、IBM 製品 Web サイト (http://www.ibm.com/security/cryptocards) の FAQ セクションを参照することができます。そのトピックには、クラス鍵 (class-key) 証明書ファイルの現行リストがあります。

関連情報:

28 ページの『ノードの初期化』
CCA ノードを初期化して初期状態にする手順。

Coprocessor Load Utility (CLU) の参照

このセクションでは、ソフトウェアをロードするコプロセッサーのメモリー・セグメントについて説明しています。コプロセッサーがソフトウェア・ロードの検証に使用する方法、CLU の開始に使用される構文、および CLU 戻りコードについて説明します。

このセクションに記載されている詳細情報が必要ない場合は、 18 ページの『CNM および CNI ユーティリティーを使用した暗号化ノードの管理』セクションにスキップしてください。

コプロセッサーのメモリー・セグメント

コプロセッサーのメモリー・セグメントは、複数の異なるセグメントに編成されます。

メモリー・セグメントの編成とその機能は、以下のとおりです。

表 3. メモリー・セグメントの編成

セグメント	説明
0	基本コード 基本コードは、コプロセッサーの初期化とハードウェア・コンポーネント・インターフェースを管理します。このコードは、コプロセッサーが工場から出荷された後は変更できません。
1	ソフトウェア管理と暗号ルーチン このセグメントのソフトウェアは、次のようになります。 • すでにセグメント 1 にロードされているソフトウェアの置換を管理します。 • セグメント 2 と 3 へのデータおよびソフトウェアのロードを管理します。 • 工場でロードされますが、CLU ユーティリティーを使用して置換できます。
2	組み込みオペレーティング・システム コプロセッサーのサポート・プログラムにはオペレーティング・システムが含まれています。このオペレーティング・システムは、セグメント 3 にロードされるアプリケーションをサポートします。コプロセッサーの工場出荷時、セグメント 2 は空です。
3	アプリケーション・ソフトウェア コプロセッサーのサポート・プログラムには、セグメント 3 にインストールできる CCA アプリケーション・プログラムが含まれています。このアプリケーションは、IBM CCA に従って機能し、アクセス制御、鍵管理、および暗号操作を実行します。コプロセッサーの工場出荷時、セグメント 3 は空です。

コプロセッサー・ソフトウェア・ロードの検証

工場からの出荷時、コプロセッサーの内部には、セグメント 1 の置換ソフトウェアの検証に必要な公開鍵が入っています。

コプロセッサーのセグメント 2 とセグメント 3 へコードをロードするには、各セグメントごとに次の手順に従います。

1. 「**Establish Owner**」コマンドを使用して、セグメントの所有者を識別します。所有者 ID は、この ID に関連付けられたデジタル署名が、すぐ下のセグメントに常駐する公開鍵によって検証できる場合にのみ受け入れられます。所有権は、一度確立されると、コプロセッサーによって **Surrender Owner** コマンドが処理されるまで有効なままとなります。

2. コードをセグメントにロードします。2 個の異なるコマンドを使用できます。
   a. 最初に、**Load** コマンドを使用します。**Load** コマンド・データには、次の下位セグメントに存在する公開鍵によって検証する必要のある公開鍵証明書が含まれます。以下のいずれかの条件が満たされた場合、コプロセッサーはこのコードを受け入れて、そのセグメントの検証された公開鍵を保存します。
      - 証明書が検証されている。
      - **Load** コマンド内の所有者 ID のデータが、そのセグメントのコプロセッサーによって保持されている現行の所有権と一致している。
      - **Load** コマンド内の完全なデータは、検証に使用された証明書の公開鍵によって検証できる。
   b. セグメントにすでに公開鍵がある場合は、**Reload** コマンドを使用して、セグメント内のコードを置換することができます。コプロセッサーのアクションは、組み込まれている証明書を、次に低位のセグメントに関連付けられた鍵ではなく、ターゲット・セグメントに関連付けられた公開鍵によって検証する必要がある点を除くと、**Load** コマンドと同じです。

組み込みオペレーティング・システムは、コプロセッサー・ハードウェアと協働して、そのオペレーティング・システム自体とセグメント 3 のアプリケーションのために、セキュリティー関連データ項目 (SRDI) を保管することができます。SRDI は、改ざん、セグメント・ソフトウェアのロード、またはセグメントの **Surrender Owner** コマンドの処理を検出するとゼロ化されます。セグメントの SRDI は、**Reload** コマンドを使用しているときにはゼロ化されません。 CCA アプリケーションは、マスター・キー、機能制御ベクトル (FCV)、アクセス制御テーブル、および保存 RSA 秘密鍵を、セグメント 3 に関連付けられた SRDI 情報として保管します。

IBM は、IBM 作成のソフトウェアに署名します。別のベンダーがコプロセッサー用のソフトウェアを提供しようとした場合は、そのベンダーの **Establish Owner** コマンドとコード署名の公開鍵証明書が、適切な契約の下に IBM によって署名されている必要があります。これらの制限により、以下の条件を確実に満たします。

- 許可されたコードのみをコプロセッサーにロードできる。
- 暗号実装のインポートとエクスポートに関連する政府の制約事項に従う。

Coprocessor Load Utility の構文

このセクションでは、Coprocessor Load Utility (CLU) を開始するために使用される構文、および CLU の各機能について説明します。

CLU は、以下の機能で使用する必要があります。

- コプロセッサーを使用しているアプリケーションをすべて終了することにより、そのコプロセッサーがビジーになっていないことを確認します。例えば、CCA API を使用するすべてのアプリケーションを終了します。
- コプロセッサーのメモリー・セグメントにインストールされているソフトウェアのリリース・レベルと状況を取得します。
- コプロセッサーから戻されたデジタル署名付きメッセージの妥当性を確認します。
- コプロセッサー・ソフトウェアの一部をロードおよび再ロードします。
- コプロセッサーをリセットします。

---

2. 本書では、ロード と再ロード という用語を使用しています。他の資料では、これらの操作を、*emergency burn* (EmBurn)、および *regular burn* または *remote burn* (RemBurn) と呼ぶ場合があります。

このユーティリティーを開始するには、以下の手順に従ってください。

1. オペレーティング・システムの要件に応じてログオンします。
2. コマンド行で、CLU ファイルが入ったディレクトリーに変更します。デフォルト・ディレクトリーは `/usr/lpp/csufx.4765/clu` です。
3. **csufclu** ユーティリティー名の後に、該当するパラメーターを続けて入力します。

必要なパラメーターを提供しないと、情報が必要な場合、ユーティリティーは入力を求めるプロンプトを出します。オプション・パラメーターは、大括弧で囲まれます。ユーティリティー名の後のパラメーターの構文は、以下のとおりです。

[*log_filecmd*[*coprocessor _#*][*data_file*][*-Q*]]

この場合:

log_file
: ログ・ファイル名を識別します。ユーティリティーは、要求された操作を実行するときに、この ASCII テキスト・ファイルにエントリーを追加します。2 番目の機械可読ログ・ファイル (logfile_name.MRL というファイル名) も作成されます。このログ・ファイルはプログラムによって処理でき、コプロセッサーからの 2 進化エンコード応答を含みます。

cmd
: 実行するローダー・コマンドを表す 2 文字の省略語を指定します。

coprocessor *_number*
: デバイス・ドライバーによって確立されたコプロセッサー番号を提供します。このパラメーターはデフォルトで 0 です。コプロセッサーは、番号 0、1、および 2 としてデバイス・ドライバーに指定されます。**ST** または **VA** コマンドによって取得したシリアル番号情報、および特定のコプロセッサーを coprocessor*_number* に相互に関連付けるためにコプロセッサーの終端ブラケットに印刷されたシリアル番号を使用できます。このユーティリティーは、システム 1 台につき最大 8 台のコプロセッサーをサポートします。

data_file
: 要求された操作に使用されるデータ・ファイル (ドライブ、ディレクトリー、およびファイル名) を識別します。*data_file* 名を識別するには、次のいずれかの方法を使用してください。
  - ソフトウェアのロードおよび再ロードの場合、*data_file* 名は、コプロセッサーにロードするソフトウェア・イメージのファイル名です。サポート・プログラムの README ファイルに *data_file* 名が記載されています。
  - コプロセッサーの場合、コプロセッサー状況は **VA** コマンドによって取得されます。 *data_file* 名は、コプロセッサー応答の検証に使用されるクラス鍵 (class-key) 証明書ファイル名です。製品 Web サイト (http://www.ibm.com/security/cryptocards) の FAQ セクションには、コプロセッサーとそのコードを検証するための手順の説明があります。この説明には、現行クラス鍵 (class-key) 証明書ファイル名のリストも含まれています。必要な証明書ファイルはこの Web サイトからダウンロードできます。

-Q
: 標準出力デバイスへの CLU プログラム出力を抑止 (静止) します。この場合も、状況情報はログ・ファイルに追加されます。

例: コプロセッサー状況を取得してその結果をログ・ファイルに保存するには、以下のように入力します。

csufclu nnnnnnnn.log va *datafile_name***.clu**

nnnnnnnn は、コプロセッサーのシリアル番号にすることをお勧めします。シリアル番号の使用は必須ではなく、各固有コプロセッサーに対して行われたすべてのソフトウェア変更のヒストリーを保持するためにシリアル番号が使用されます。

関連情報:

51 ページの『機械可読ログの内容』
CLU ユーティリティーは 2 つのログ・ファイルを作成します。1 つのログ・ファイルは読み取り用で、もう一方はプログラムへの入力用の可能性があるログ・ファイルです。

『Coprocessor Load Utility コマンド』
Coprocessor Load Utility (CLU) は、複数のローダー・コマンドをサポートしています。

Coprocessor Load Utility コマンド**:**

Coprocessor Load Utility (CLU) は、複数のローダー・コマンドをサポートしています。

CLU によってサポートされるローダー・コマンドとその機能は、以下のとおりです。

表 *4. CLU* ローダー・コマンド

ローダー・コマンド	説明
PL: マイクロコードをコプロセッサーにロードする コマンド R1、E2、L2、R2、S2、E3、L3、R3、および S3 は、PL コマンドに使用するデータ・ファイルに含まれている情報から推論されます。単一の「PL」ファイルには、複数の所有権とロードするコマンドに関する情報を組み込むことができます。	データ・ファイルの内容の指示に従って一連のコマンドを処理してセグメントの所有権を確立し、セグメント・ソフトウェアをロードまたは再ロードします。
RS: コプロセッサーをリセットする	コプロセッサーをリセットします。一般にこのコマンドは使用されません。このコマンドを実行すると、コプロセッサーはパワーオン・リセットを実行します。このコマンドは、コプロセッサーとホスト・システム・ソフトウェアが同期を失った場合に役立つ可能性があります。このコマンドを実行して完全な暗号化サブシステムをリセット状態にする前に、コプロセッサーで動作しているすべてのホスト・システム・ソフトウェア・プロセスを終了する必要があります。
SS: システム状況を取得する	インストールされている各コプロセッサーのパーツ・ナンバー、シリアル番号、およびセグメント 3 ソフトウェア・イメージ名の一部を取得します (これらが、CCA などの一部のアプリケーションによって使用されていない場合)。 18 ページの図 2 を参照してください。
ST: コプロセッサーの状況を取得する	ロードされているソフトウェアの状況とほかのコンポーネントのリリース・レベルを取得します。この状況は、ログ・ファイルに追加されます。
VA: コプロセッサーの状況を検証する	ロードされているソフトウェアの状況とほかのコンポーネントのリリース・レベルを取得します。データは、コプロセッサーのデバイス・キーによって署名されたメッセージで伝送され、その後ユーティリティーのログ・ファイルに保管されます。 ユーティリティーは内蔵の公開鍵を使用して、*data_file* 名前パラメーターに含まれた 1 つ以上のクラス鍵 (class-key) 証明書を検証します。これらの証明書の 1 つが、コプロセッサーから取得した公開鍵、または一連の公開鍵を検証して、コプロセッサーが改ざんされていないことを確認する必要があります。

一般に、ユーティリティーはスクリプト・ファイルまたはコマンド・ファイルによって呼び出すことができます。不在システム上でユーティリティーを開始するためのスクリプト・ファイルまたはコマンド・ファイルを作成する場合は、「quiet」構文、**-q** (または **-Q**、**/q** あるいは **/Q**) パラメーターを追加して、ディスプレイに出力が送信されないよう要求します。デフォルトにより、ユーティリティーはディスプレイにプロンプトとメッセージを戻します。

標準の *CLU* システム状況応答 の図は、CLU システムの応答を示しています。

```
======================================================================
CSUFCLU V4.00 ss.log SS    begun Tue Sep 28 10:49:36 2010
*********** Command SS started. ---- Tue Sep 28 10:49:36 2010

 Card #    P/N        S/N         Segment 3 Description
 ------    -------    --------    -----------------------------------
   0       45D6045    99000627    4.1.0    CCA
 *** Query System Status successful ***
System Status ended successfully!
*********** Command SS ended. ---- Tue Sep 28 10:50:37 2010

  ...finishing up...
*********** Command SS exited. ---- Tue Sep 28 10:50:57 2010
```

図 *2.* 標準の *CLU* システム状況応答

Coprocessor Load Utility の戻りコード

このセクションでは、CLU から戻されたコード値を指定します。

CLU は処理を終了すると、スクリプト・ファイルまたはコマンド・ファイルでテストできる値を戻します。戻された値にはそれぞれ意味があります。

0 OK。これは、CLU が処理を適切に完了したことを意味します。

1 コマンド行パラメーターが無効です。

2 コプロセッサーにアクセスできません。この場合は、コプロセッサーとドライバーが正しくインストールされていることを確認してください。

3 異常条件報告が記載されていないか、ユーティリティーのログ・ファイルを確認してください。

4 コプロセッサーがインストールされていません。この場合は、コプロセッサーとドライバーが正しくインストールされていることを確認してください。

5 無効なコプロセッサー番号が指定されています。

6 このコマンドにはデータ・ファイルが必要です。

7 このコマンドによって指定されたデータ・ファイルが間違いか無効です。

CNM および CNI ユーティリティーを使用した暗号化ノードの管理

鍵の生成およびデジタル署名サポートなどの暗号化サービスを提供するコンピューターは、ここでは暗号化ノードと定義されています。

サポート・プログラムで提供されている CCA Node Management (CNM) ユーティリティーと CCA Node Initialization (CNI) ユーティリティーは、ノードによって提供される CCA 暗号化サービスをセットアップおよび管理するためのツールです。

このセクションには、以下の事項が含まれています。

- ユーティリティーの内容および開始方法の説明
- 検討が必要な、ユーティリティーを使用するためのサンプル・シナリオ
- CNM ユーティリティー管理機能の使用方法: トピック 21 ページの『シナリオ: テスト・ノードの作成』をひととおり実行した後に、この資料を検討してください。

- アクセス制御データの作成および管理方法: CNM ユーティリティーのアクセス制御部分に関する詳細情報をお読みください。
- 暗号鍵の管理方法: CNM ユーティリティーによって実行できるいくつかの鍵管理タスクについてお読みください。
- CNI ユーティリティーを使用した他のノードの確立方法: カプセル化されたプロシージャーを使用して CNM ユーティリティーの使用を自動化できます。

これらのユーティリティーは Java™ で作成されているため、Java ランタイム環境 (JRE) の使用が必要です。また、Java Development Kit (JDK) も使用できます。

CNM および CNI の概要

CCA Node Management (CNM) ユーティリティーと CCA Node Initialization (CNI) ユーティリティーの典型的なユーザーは、セキュリティー管理担当者、アプリケーション開発者、およびシステム管理者で、時には実動モード・オペレーターも含まれます。

注**:**

1. CNM ユーティリティーには、限られた CCA API サービスのセットが装備されています。このユーティリティーを熟知した後は、それが要件を満たしているかどうか、またはより包括的な管理制御および鍵管理を実行するためにカスタム・アプリケーションが必要かどうかを判別することができます。
2. CNM ユーティリティーを使用して作成するファイルは、Java ランタイム環境 (JRE) のリリースに依存する場合があります。使用する Java ランタイム環境 (JRE) のリリースを変更すると、CNM ユーティリティーによって作成したファイルが新規リリースで正しく機能しなくなる場合があります。
3. CNM ユーティリティーは、マウスで使用するように設計されています。一貫した結果を得るためには、**Enter** キーではなくマウスを使用してください。
4. このユーティリティーの「マスター・キーの複製」部分については、ヘルプ・パネルは提供されていません。
5. これらのユーティリティーは IBM Common Cryptographic Architecture (CCA) サポート・プログラム API を使用して、コプロセッサーからのサービスを要求します。「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルには、CCA API によって提供されている verb (呼び出し可能サービスまたはプロシージャー呼び出しとも呼ばれる) の包括的リストが含まれています。どのコマンドに、このセクションで説明する手順を使用して定義されるさまざまなロールの権限が必要な可能性があるかを理解するためには、この資料、およびそこで説明されている個別のサービスを参照してください。

CCA ノード管理ユーティリティーの概要

CCA Node Management ユーティリティーは、IBM 4765 CCA 暗号化ノードのセットアップと構成に使用するグラフィカル・インターフェースを提供する Java アプリケーションです。このユーティリティーの主な機能は、ノードのセットアップ、アクセス制御データの作成と管理、および暗号化ノードの管理に必要な CCA マスター・キーの管理です。

データ・オブジェクトは、直接コプロセッサーにロードすることも、ディスクに保存することもできます。データ・オブジェクトは、同じオペレーティング・システムと、互換性のある Java アプリケーションのレベルを使用する、ほかの IBM 4765 CCA ノードで使用できます。

注**:** CCA Node Management ユーティリティーの開始: CCA Node Management ユーティリティーを開始するには、**csufcnm** コマンドを入力します。CNM ユーティリティーのロゴが表示され、次にメインウィンドウが表示されます。

CCA ノード初期化ユーティリティーの概要

CCA Node Initialization ユーティリティーは、CNM ユーティリティー内で *CNI* エディター を使用して作成されたスクリプトを実行します。これらのスクリプトは、*CNI* リスト と呼ばれます。 CNI ユーティリティーは、ノードのセットアップに必要な CNM ユーティリティー機能を実行することができます。例えば、アクセス制御ロールおよびプロファイルをロードするために使用できます。

CNI リストを作成する際に、CNI ユーティリティーがターゲット・ノードにロードするデータ・オブジェクトのディスク・ロケーションを指定します。CNI リストを作成した後は、その CNI リストと (ロール、プロファイルなどの) すべての付属データ・ファイルを、CNI ユーティリティーが自動セットアップのために使用されるノードに配布できます。ソース・ノードと、配布された CNI リストを実行するすべてのノードは、同じオペレーティング・システムと、互換性のある Java アプリケーションのレベルを使用する必要があります。

注**:** CCA Node Management ユーティリティーの開始: CCA Node Management ユーティリティーを開始するには、**csufcnm** コマンドを入力します。CNM ユーティリティーのロゴが表示され、次にメインウィンドウが表示されます。

関連情報:

24 ページの『シナリオ: DES または PKA マスター・キーの複製』
このセクションでは、あるコプロセッサーから別のコプロセッサーへのデータ暗号化規格 (DES) または公開鍵アルゴリズム (PKA) マスター・キーの複製に関連した手順について説明します。

43 ページの『CNI ユーティリティーを使用した他のノードの作成』
CCA Node Initialization (CNI) ユーティリティー用の CNI リストを作成することにより、ターゲット・ノード上で CNM ユーティリティーを実行せずに、ディスクに保管されている鍵とアクセス制御データを他の暗号化ノードにロードすることができます。

シナリオ**: CNM** および **CNI** ユーティリティーの使用

このセクションでは、CCA Node Management (CNM) ユーティリティーおよび CCA Node Initialization (CNI) ユーティリティーを使用して、ノードを作成し、そのノードを別のコプロセッサーに複製する方法について説明します。

これらのユーティリティーの使用は、以下のシナリオで説明されています。

1. アプリケーションの開発に使用されるテスト・ノードを作成するか、CNM ユーティリティーを使用するための手順を確立します。初めてのユーザーはこの手順に従って、ユーティリティーとコプロセッサーの試用を開始してください。
2. 鍵パーツを使用して実稼働環境用ノードを作成します。このシナリオでは、CNI リストを使用してターゲット実動ノードの確立を自動化します。
3. 1 つのコプロセッサーから別のコプロセッサーにマスター・キーを複製します。これは、複数のコプロセッサーを用いる、ハイ・セキュリティーのインストール環境に重要な手順です。

これらのシナリオの目的は、ここで説明する手順をどのように使用できるかを示すことです。シナリオは必要に応じて、より詳しい情報の記載された、このトピック集の他のセクションを参照します。

コプロセッサーの CCA アクセス制御システムを熟知していない場合は、 30 ページの『アクセス制御の概要』および 31 ページの『アクセス制御システムの初期状態』を参照してください。ここには、ロール、初期 *DEFAULT* ロール、およびユーザー・プロファイル などの用語説明があります。これらのシナリオでは、アクセス制御システムが初期状態であることが想定されています。

注**:** これらのシナリオは、説明目的のためのみです。ユーザーはユーザー固有の環境に最も適した手順を判別することをお勧めします。「IBM CCA Basic Services Reference and Guide for the *IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」で、セキュア操作に関する付録を参照してください。

シナリオ**:** テスト・ノードの作成

このシナリオでは、暗号サービスへの無制限アクセスを可能にするために、一人のプログラマーがノードをセットアップします。

重要**:** このシナリオでは多くの機密コマンドが無制限の使用を許可されているため、結果の暗号化ノードはセキュアと見なすべきではありません。

前提条件**:** 適切なレベルの Java ランタイム環境 (JRE) または Java Development Kit (JDK) をインストールしていることが必要です。

テスト・ノードを作成するには、以下の手順に従ってください。

1. サポート・プログラムのインストールの説明に従って、コプロセッサーと IBM 暗号化コプロセッサー・サポート・プログラムをインストールします。
2. **csufcnm** コマンドを入力して、CCA Node Management ユーティリティーを開始します。CNM ユーティリティーのロゴとメインパネルが表示されます。
3. CCA のインストールされたコプロセッサーが複数ある場合は、CNM ユーティリティーにどのコプロセッサーを使用するか指定してください。「暗号化ノード **(Crypto Node)**」メニューから「アダプターの選択 **(Select Adapter)**」を選択します。使用可能なアダプター番号 (1 から 8) のリストが表示されます。リストからアダプター (コプロセッサー) を選択します。「アダプターの選択 **(Select Adapter)**」リストを使用してアダプターを選択しない場合は、デフォルト・アダプター (コプロセッサー) が使用されます。
4. コプロセッサーとホスト・コンピューター内のクロックを同期化します。「暗号化ノード **(Crypto Node)**」メニューから「時間 **(Time)**」をクリックします。結果のサブメニューから「設定 **(Set)**」をクリックします。クロックが同期します。
5. CNM ユーティリティーを使用して、DEFAULT ロールにすべてのコマンドを許可します。
   a. 「アクセス制御 **(Access Control)**」メニューから「ロール **(Roles)**」をクリックします。
   b. 「**DEFAULT**」エントリーを強調表示して、「編集 **(Edit)**」をクリックします。DEFAULT ロールで使用可能なコマンドと使用可能でないコマンドが、ウィンドウに表示されます。
   c. 「すべて許可 **(Permit All)**」をクリックします。
   d. 「ロード **(Load)**」をクリックし、「**OK**」を選択して、変更されたロールをコプロセッサーに再びロードします。
   e. 「保存 **(Save)**」ボタンをクリックしてロールのコピーを保存し、ロールに名前を指定します。
6. 機能制御ベクトル (FCV) をコプロセッサーにロードします。「暗号化ノード **(Crypto Node)**」メニューから「権限 **(Authorization)**」をクリックします。結果のサブメニューから「ロード **(Load)**」をクリックし、FCV を指定してロードします。

   FCV ファイルは、インストール・プロセス中にサーバーに配置された FCV ファイルです。通常、FCV のファイル名は `fcv_td4kECC521.crt` などで、オペレーティング・システムにあるファイル検索ユーティリティーを使用して検索します。
7. 「マスター・キー **(Master Key)**」メニューからマスター・キーをインストールして、「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」または「**AES** マスター・キー **(AES Master Keys)**」のいずれかをクリックし、「はい **(Yes)**」をクリックします。コプロセッサーがランダム・マスター・キーを作成して設定します。

「自動設定 **(Auto Set)**」オプションによってインストールされたマスター・キーは、鍵パーツとしてシステム・プロセッサーのメイン・メモリーを通して実際に渡されます。実動目的のためには、ランダム生成や、2 人以上の個人によって入力される既知の鍵パーツをインストールするなど、もっと安全な方法を使用してマスター・キーを確立してください。これらのオプションは、前述のメニューからもアクセスされます。

8. 鍵ストレージ・ファイルを初期化します。鍵ストレージ・ファイルの初期化については、 40 ページの『鍵ストレージの作成または初期化』を参照してください。

   鍵ストレージ は、サポート・プログラムが、ユーザー (またはアプリケーション) の定義した名前でデータ暗号化規格 (DES)、Rivest-Shamir-Adleman (RSA) アルゴリズム、および Advanced Encryption Standard (AES) 暗号鍵を保管できる場所を記述する CCA 用語です。鍵ストレージを使用したい場合は、使用している鍵のタイプ (DES、RSA (PKA)、または AES) に一致した鍵ストレージ・ファイルを初期化する必要があります。例えば、DES 鍵のみを使用したい場合は、DES 鍵ストレージ・ファイルの初期化が必要ですが、他のファイルを初期化する必要はありません。DES 鍵と PKA 鍵を使用したい場合は、DES 鍵ストレージ・ファイルと PKA 鍵ストレージ・ファイルの初期化が必要ですが、AES 鍵ストレージ・ファイルの初期化は必要ありません。3 つすべてを使用したい場合は、3 つすべてを初期化する必要があります。

関連リンク: 31 ページの『ロールの作成』

38 ページの『マスター・キーの自動ロード』

シナリオ: 実稼働環境でのノードの作成

このシナリオでは、暗号化ノードを作成する責任が、3 人の個人 (1 人のアクセス制御管理者と 2 人の鍵管理担当者) に分担されます。

管理者はノードとアクセス制御システムをセットアップします。鍵管理担当者はマスター・キーと必要なすべての鍵暗号鍵 (KEK) をロードします。 KEK は、ノード間でほかの鍵を移送するためのトランスポート鍵として使用できます。

このシナリオでは、マスター・キーと高水準の、ノード間の DES (データ暗号化規格) KEK を鍵パーツからインストールすることを中心に扱います。 CCA の実装環境では、ランダム・マスター・キーの生成や RSA (Rivest-Shamir-Adleman) 公開鍵テクノロジーに基づく技法を使用した DES 鍵の配布など、鍵パーツ技法の代替方法がサポートされます。鍵パーツ技法では、タスクの実行と鍵パーツ情報を共有しないことについて信頼できる 2 人の鍵管理担当者 がいることが想定されています。このテクノロジーにより、知識分割 ポリシーが実装されます。アクセス制御システムは、第 1 担当者と第 2 担当者のタスクを分離することにより、二重管理を実施するようにセットアップされます。

このシナリオでは、アクセス制御管理者は CNM (暗号化ノード管理) ユーティリティーを使用して、ターゲット・ノード用の CNI (Coprocessor Node Initialization) リストを準備します。CNI リストにより、ターゲット・ノードで CNM ユーティリティーを使用するプロセスが自動化されます。管理者は、ターゲット・ノードのアクセス制御管理者および 2 人鍵管理担当者によって実行されるタスク用の CNI リストを準備します。管理者は、別の条件下では、コマンドにターゲット・ノードの権限が必要なことを認識している必要があります。以下のような条件があります。

- 標準、制限された操作 (デフォルト・ロールが使用される場合)
- アクセス制御管理者タスクが実行される場合
- 各鍵管理担当者タスクが実行される場合
- 追加ロールとプロファイルを使用するそれ以外のすべての特殊環境の場合

注**:** CNM ユーティリティーと CNI ユーティリティーは、ノードによって提供される CCA 暗号化サービスのセットアップおよび管理に使用されるツールです。

管理者は、さまざまなロールでコマンドを許可することにより、必ず必要なコマンドのみが使用可能になるようにします。第 2 鍵パーツのロードや後続の鍵パーツのロードといった重要なコマンドは、これらのコマンドを使用する責任と権限のあるユーザーのロールでのみ使用可能になります。コプロセッサーのアクセス制御システムが、知識分割や二重管理などのポリシーを実施できるように、責任を分割することが重要です。

関連情報:

30 ページの『アクセス制御データの作成および管理』

シナリオ**:** ターゲット・ノード用の **CNI** リストの準備**:** このタスクでは、アクセス制御管理者は CCA Node Management (CNM) ユーティリティーを使用して、ターゲット・ノード用の CCA Node Initialization (CNI) リストを準備します。

ノードをセットアップしてアクセス制御データを作成するために、アクセス制御管理者は以下を実行できます。

1. 確立されたノードで、CNM ユーティリティーを開始します。
2. 以下を含む、ターゲット・ノード用アクセス制御データを作成してディスクに保存します。
   - アクセス制御管理者および鍵管理担当者の監視ロールとユーザー・プロファイル
   - 初期デフォルト・ロールを置換するデフォルト・ロール

   a. CNI リストを作成して、コプロセッサーとホスト・コンピューター内のクロックおよびカレンダーを同期化します。
      1) アクセス制御データをロードする。
      2) アクセス制御管理者としてログオンする。
      3) 置換デフォルト・ロールをロードする。
      4) 機能制御ベクトル (FCV) をロードする。
      5) ログオフする。

   b. 第 1 鍵管理担当者のための CNI リストを作成します。
      1) 第 1 鍵管理担当者としてログオンする。
      2) 鍵パーツの第 1 マスター・キーをロードする。
      3) 第 1 パーツの鍵暗号鍵情報をロードする。
      4) ログオフする。

   c. 第 2 鍵管理担当者のための CNI リストを作成します。
      1) 第 2 鍵管理担当者としてログオンする。
      2) 鍵パーツの第 2 マスター・キーをロードする。
      3) 第 2 パーツの鍵暗号鍵情報をロードする。
      4) ログオフする。
3. コプロセッサーと IBM Common Cryptographic Architecture (CCA) サポート・プログラムをターゲット・ノードにインストールします。
4. アクセス制御データと CNI リストに指定された FCV をターゲット・ノードに移送します。
5. 鍵管理担当者を含めて、各ターゲット・ノードで、ステップ 2a、2b、および 2c で作成した CNI リストを実行します。

これにより、ターゲット・ノードで暗号化サービスを提供する準備ができました。

関連情報:

30 ページの『アクセス制御データの作成および管理』

43 ページの『CNI ユーティリティーを使用した他のノードの作成』

CCA Node Initialization (CNI) ユーティリティー用の CNI リストを作成することにより、ターゲット・ノード上で CNM ユーティリティーを実行せずに、ディスクに保管されている鍵とアクセス制御データを他の暗号化ノードにロードすることができます。

シナリオ: 鍵パーツの作成とロード:

このセクションでは、鍵パーツを作成、ロード、および移送する手順について説明します。

鍵管理担当者は、ターゲット・ノードで使用するための鍵パーツを作成し、ターゲット・ノードにロードします。

生成地点からインストール地点への鍵パーツの移送方法を決定します。以下のいくつかの方法が考えられます。

- 鍵パーツを中央で生成して、ディスケットで転送する
- 鍵パーツを中央で生成して、紙のフォームで転送する
- 鍵パーツを (最初の) インストール時にインストール地点で生成する。インストール後に (再ロードするか別のノードと共有するために) この鍵パーツが必要になった場合は、鍵パーツの転送方法を決定しなければなりません。

CNM ユーティリティーを使用して作業することによって、このユーティリティーの特定の機能を検討してください。その後で、選択する特定の方法を検討し、アクセス制御管理者と協力して準備された CCA Node Initialization ユーティリティー (CNI) リストをテストしてください。

シナリオ: DES または PKA マスター・キーの複製

このセクションでは、あるコプロセッサーから別のコプロセッサーへのデータ暗号化規格 (DES) または公開鍵アルゴリズム (PKA) マスター・キーの複製に関連した手順について説明します。

マスター・キーは、コプロセッサー間の移送のために複数の Share に分割されるため、コピーではなく「複製」という用語が使用されます。この手法については、「IBM CCA Basic Services Reference and Guide for the *IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」のトピック『Understanding and managing master keys』で説明されています。 52 ページの『マスター・キーの複製』のセクションでは、ユーザーのためのステップバイステップ手順を提供しています。この手順を変更できるバックグラウンド情報は、このセクションに記載されています。

注: AES マスター・キーの複製はサポートされていません。

マスター・キーの複製には、次のうち 2 台または 3 台のノードが必要です。

- マスター・キー・ソース・ノード
- マスター・キー・ターゲット・ノード
- Share 管理 (SA) ノード。SA ノードはソース・ノードまたはターゲット・ノードのどちらでもかまいません。

CNM ユーティリティーは、このプロセスに関連したさまざまなデータ項目をデータベースに保管できます。それらは (ディスケットで) 持ち運んだり、異なるノード間で転送 (FTP) することができます。1 つ

のデータベースはデフォルトで sa.db と呼ばれ、認証された SA 鍵 (複数可) に関する情報を含みます。マスター・キーが複製されるターゲット・ノードにも、デフォルトで csr.db と呼ばれるデータベースがあります。

CNM ユーティリティーを使用して、以下のタスクを実行できます。

1. **csufcnm** コマンドを入力して、CCA Node Management ユーティリティーを開始します。CNM ユーティリティーのロゴが表示され、メイン・ウィンドウが表示されます。
2. アクセス制御ロール、ユーザー・プロファイル、およびマスター・キーとともに、安全な方法でノードをセットアップします。

   Share を取得または保管するユーザーごとに、ソース・ノードとターゲット・ノードで 1 つのロールおよび 1 つ以上のユーザー・プロファイルが必要です。Share の処理は別個のコマンドによって行われるため、必要な場合はロールによって、異なる Share の取得とインストールに、独立した個人が関係するよう保証することができます。

   ランダム・マスター・キー生成の使用、および二重管理セキュリティー・ポリシーを実施するロールを検討してください。例えば、1 人の個人または 1 つのロールにハッシュの登録を許可し、別の個人またはロールに公開鍵の登録を許可します。マスター・キーの個別の Share の取得とインストールについては、別の個人またはロールを選択します。

   Master_Key_Process verb と Master_Key_Distribute verb の説明については、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルのガイダンスのセクションを参照してください。
3. 選択した固有の 1 バイトから 16 バイトの環境 ID (EID) を各ノードにインストールします。

   「暗号化ノード **(Crypto Node)**」メニューから「環境 ID の設定 (Set Environment ID)」をクリックし、ID を入力し、「ロード (Load)」をクリックします。EID には、A - Z、a - z、0 - 9 と @ (X'40')、スペース文字 (X'20')、& (X'26')、および = (X'3D') の文字のみを使用してください。

   完全な 16 文字の ID を入力してください。短い ID の場合は、スペース文字を使用してエントリーを完了してください。
4. ソース・ノードとターゲット・ノードでマスター・キーの Share の m 値と n 値を初期設定します。これらの値は、ソース・ノードとターゲット・ノードで同じでなければなりません。値 n は、ターゲット・ノードにマスター・キーを再構築するためにインストールされるのに必要な Share の最大数で、m はその最小数です。

   「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「**Share** 数の設定 **(Set number of shares)**」をクリックし、これらの値を入力して、「ロード (Load)」をクリックします。
5. さまざまなノードでこれらの鍵を生成し、それぞれの公開鍵を SA 鍵で認証します。ユーティリティーの sa.db データベースを使用して、これらの鍵と証明書を移送することができます。

   Share 管理 **(SA)**

   この鍵は、この鍵自体と後続の鍵を認証するために使用されます。SA 公開鍵ハッシュと、公開鍵自体を SA、ソース・ノード、およびターゲット・ノードに登録する必要があります。

   SA 鍵が作成されると、ユーティリティーは、SA 鍵ハッシュの一部である 8 バイトまたは 16 桁の 16 進文字値を提供します。必ず、この値のコピーを保存しておいてください。 この値は、ソース・ノードとターゲット・ノードに SA 公開鍵を登録するためにデータベースに記録されたハッシュ値を確認するのに必要になります。

Coprocessor Share Signing (CSS)
この鍵は、ソース・ノードから配布された Share に署名するために使用されます。秘密鍵は、ソース・ノード内に保存されます。

Coprocessor Share Receiving (CSR)
この鍵は、Share 暗号化鍵をターゲット・ノードに受信するために使用されます。SA 認証済みの公開 CSR 鍵は、ソース・ノードで、各 Share ごとに固有の Share 暗号化鍵をラップ (暗号化) するために使用されます。秘密鍵は、ターゲット・ノード内に保存されます。

鍵ペアの生成**: SA**、**CSS**、および **CSR**
「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「鍵の作成 **(Create Keys)**」をクリックします。「**Share** 管理鍵 **(Share Administration Keys)**」、「**CSS** 鍵 **(CSS key)**」、または「**CSR** 鍵 **(CSR key)**」をクリックします。「作成 **(Create)**」をクリックします。

ソース・ノードとターゲット・ノードに保存されている CSS 鍵と CSR 鍵の鍵ラベルを提供する必要があります (例えば、`IBM4765.CLONING.CSS.KEY` と `IBM4765.CLONING.CSR.KEY`)。使用するラベルは、ご使用のアプリケーションで使用されている他の鍵ラベルと競合しないようにする必要があります。

Share 受信ノードで CSR 鍵を生成する場合は、コプロセッサーのシリアル番号を取得する必要があります。「暗号化ノード **(Crypto Node)**」から「状況 **(Status)**」をクリックします。CSR 鍵を認証するときには、シリアル番号を入力する必要があります。

6. SA ノード、ソース・ノード、ターゲット・ノードにコプロセッサーの SA 公開鍵を登録します。これは、二重管理セキュリティー・ポリシーの下で実行する必要のある、2 段階プロセスです。

   1 人の個人が SA 公開鍵ハッシュをインストールします。「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「**Share** 管理の登録 **(Register Share Administration)**」をクリックして、「SA 鍵ハッシュ (SA Key hash)」をクリックします。 SA 鍵の作成中に取得したハッシュ値を入力する必要があります。

   もう一人の個人は、実際の SA 公開鍵をインストールします。「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「**Share** 管理の登録 **(Register Share Administration)**」をクリックして、「SA 鍵 (SA Key)」をクリックします。デフォルトで、公開鍵情報は `sa.db` ファイルに入っています。
7. CSS 鍵と CSR 鍵を SA ノードに移送して、これらの鍵を認証します。

   「暗号化ノード **(Crypto Node)**」ドロップダウン・メニューから「**Share** 管理鍵 **(Share Administration Keys)**」、「鍵の認証 **(Certify Keys)**」「**CSS** 鍵 **(CSS key)**」、または「**CSR** 鍵 **(CSR key)**」を選択します。

   CSR 鍵の場合は、適切な鍵が認証されていることの手順検査として、ターゲット・コプロセッサーのシリアル番号を提供する必要があります。手順には、信頼できる方法でこの情報を伝えることを含める必要があります。
8. ソース・ノードで、権限のある個人が、各個人に Share の取得を許可するロールにサインオンする必要があります。少なくとも m 個の Share を取得する必要があります。これらが、現行マスター・キーの Share になります。

   「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「**Share** の取得 **(Get Share)**」をクリックし、取得する Share 数を入力します。「シリアル番号とデータベース **ID** の監視 **(Observe the serial numbers and database identifiers)**」。これらが正しい場合は、

「Share の取得 (Get Share)」をクリックします。 Share 情報はデフォルトで csr.db ファイルに入れられる必要があり、デフォルトで sa.db ファイルから CSR 鍵証明書を取得します。

後でターゲット・ノードで使用するために、current-master-key の検証情報を取得します。「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」 > 「検証 **(Verify)**」をクリックします。「現行 **(Current)**」をクリックします。

9. ターゲット・ノードで、権限のある個人が、各個人に Share のインストールを許可するロールにサインオンする必要があります。新規マスター・キー・レジスターにマスター・キーを再設定するために、少なくとも m 個の Share をインストールする必要があります。

   「暗号化ノード **(Crypto Node)**」メニューから「**Share** 管理 **(Share Administration)**」 > 「**Share** のロード **(Load Share)**」をクリックし、インストールする Share 数を選択します。シリアル番号とデータベース ID が正しいことを確認して、「シリアル番号とデータベース **ID** の監視 **(Observe the serial numbers and database identifiers)**」をクリックします。これらの Share が正しい場合は、「Share の取得 (Get Share)」をクリックします。ターゲット・ノードで、権限のある個人が、各個人に Share のインストールを許可するロールにサインオンする必要があります。 Share 情報はデフォルトで csr.db ファイルから取得され、CSS 鍵証明書はデフォルトで sa.db ファイルから取得されます。ご使用のサーバーに CCA のロードされた暗号化コプロセッサーが複数ある場合、鍵ストレージが正しく機能するためには、それらのコプロセッサーに全く同じマスター・キーがインストールされている必要があります。

   m 個の Share がロードされたら、新規マスター・キー・レジスターの鍵が、Share を取得したときのソース・ノード内の現行のマスター・キーと同じであることを確認してください。ターゲット・ノードで、「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」 > 「新規 **(New)**」をクリックします。

10. マスター・キーの検証により、マスター・キーが複製されていることが確認されると、権限のある個人がマスター・キーを設定できます。このアクションにより古いマスター・キーはすべて削除され、現行のマスター・キーが古いマスター・キー・レジスターに移動されます。マスター・キーによって暗号化された鍵を使用するアプリケーション・プログラムは、この変更の影響を受ける可能性があります。したがって、マスター・キーを設定する場合は、必ずアプリケーション・プログラムの必要性と調整して行うようにしてください。

11. 「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」 > 「設定 **(Set)**」をクリックします。

CNM ユーティリティー機能の使用

このセクションでは、CNM ユーティリティーの各種機能を使用する手順について説明します。

特定のコプロセッサーの選択

システムで選択可能な複数のコプロセッサーの中から 1 つのコプロセッサーを選択する手順。

使用するシステムに、CCA コードのロードされた複数のコプロセッサーがある場合は、作業に使用する特定のコプロセッサーを選択する必要があります。選択を行わない場合は、デフォルトのコプロセッサーで操作することになります。コプロセッサーを選択すると、その選択は現行のユーティリティー・セッションの間、またはそのユーティリティー・セッションの中で別の選択を行うまで有効なままとなります。

コプロセッサーを選択するには、「暗号化ノード **(Crypto Node)**」メニューから「アダプターの選択 **(Select Adapter)**」をクリックします。アダプターを選択しないと、デフォルト・アダプターが使用されます。

注**:**

1. CLU ユーティリティーの使用中、コプロセッサーは 0、1、および 2 と呼ばれます。どの特定のコプロセッサーも、CCA アプリケーションがインストールされている場合もあれば、されていない場合もあります。CNM ユーティリティー (および CCA API を使用するほかのアプリケーション) では、CCA アプリケーションがロードされたコプロセッサーは 1、2、および 3 として指定されます。これらの新規 ID は、CCA が CCA アプリケーションがロードされているコプロセッサーについて、インストールされているすべてのコプロセッサーをスキャンしたときに、CCA によって割り当てられます。
2. CCA アプリケーションをコーディングするときには、コプロセッサーを割り振るために、キーワード **CRP01**、**CRP02**、および **CRP03** が使用されます。これらは、CNM ユーティリティー・メニューで使用されている番号 1、2、および 3 に一致します。

ノードの初期化

CCA ノードを初期化して初期状態にする手順。

操作に使用しているロール (デフォルト・ロールまたはログオン・ロール) で **Reinitialize Device** コマンド (オフセット X'0111') の使用が許可されている場合は、CCA ノードを初期状態にリストアすることができます。

Reinitialize Device コマンドを使用すると、次のアクションが行われます。

- マスター・キー・レジスターのクリア
- 保持されている公開鍵アルゴリズム (PKA) 公開鍵および登録されている PKA 公開鍵のクリア
- ロールとプロファイルのクリア、および初期状態へのアクセス制御のリストア

CCA ノードを初期化するには、「暗号化ノード (Crypto Node)」メニューから「初期化 **(Initialize)**」を選択します。アクションを確認するよう要求されます。

関連情報:

31 ページの『アクセス制御システムの初期状態』
初期状態には、初期デフォルト・ロールがあります。

ノードのログオンとログオフ

ユーザーは、ユーザー・プロファイルと関連ロールをアクティブにするために、コプロセッサーにログオンする必要があります。これは、デフォルト・ロール以外のロールを使用する唯一の方法です。

ログオンするには、「ファイル **(File)**」メニューから「パスフレーズ・ログオン **(Passphrase Logon)**」を選択します。

ログオフするには、「ファイル **(File)**」メニューから「ログオフ **(Logoff)**」を選択します。

注**:** DEFAULT ロールを除き、コプロセッサーへのアクセスはパスフレーズの認証によって制限されます。

機能制御ベクトルのロード

コプロセッサー FCV をロードする手順。

機能制御ベクトル (FCV) は、コプロセッサー内で CCA アプリケーションを使用可能にして、適用される輸出入規制に合致した暗号サービス水準を提供するために、IBM が提供する符号付き値です。現在の規則では、すべてのユーザーは同一水準の暗号機能を使用する権利があります。したがって、IBM では現在、IBM Common Cryptographic Architecture (CCA) サポート・プログラムに単一の FCV を提供しています。

CNM ユーティリティーを使用して FCV をコプロセッサーにロードします。FCV ファイルの名前は fcv_td4kECC521.crt です。

FCV をロードするには、以下の手順を実行してください。

1. 「暗号化ノード **(Crypto Node)**」メニューから「権限 **(Authorization)**」を選択します。
2. 結果のサブメニューから「ロード **(Load)**」をクリックして、ディスク上の FCV ファイルを指定します。ファイル名を指定して、「更新 **(Update)**」をクリックします。ユーティリティーが FCV をロードします。
3. 「**OK**」をクリックします。

CCA Node Management ユーティリティーの構成

CNM ユーティリティーのデフォルト値を構成する手順。

CNM ユーティリティーの構成パネルにより、ユーティリティーを使用して作成するファイルのディレクトリー・パスを指定することができます。ただし、一般にこのユーティリティーは、構成パネルに保管したパスを使用しません。代わりに、デフォルト・パスが Windows 環境変数に保管されます。構成パネルは、さまざまなクラスのデータ項目を保持したい場所を記録するのに有用な場所として使用できます。

クロックおよびカレンダーの同期化

コプロセッサーとホスト・コンピューター内のクロックおよびカレンダーを同期化する手順。

コプロセッサーは、日時を記録したり、パスフレーズ・ベースのプロファイル認証でのリプレイ攻撃を回避したりするために、自身のクロックおよびカレンダーを使用します。コプロセッサーをインストールしたら、そのクロックおよびカレンダーをホスト・システムのクロック・カレンダーと同期化してください。

クロックおよびカレンダーを同期化するには、以下の手順に従ってください。

1. 「暗号化ノード **(Crypto Node)**」メニューから「時間 **(Time)**」をクリックします。
2. 結果のサブメニューから「設定 **(Set)**」をクリックします。
3. クロックおよびカレンダーをホストと同期化するために、「はい **(Yes)**」と入力します。
4. 「**OK**」をクリックします。

CCA アプリケーションの状況情報の取得

CNM ユーティリティー・コプロセッサーを使用して、CCA アプリケーションの状況を取得することができます。

CNM ユーティリティー・コプロセッサーでサポートされる状況表示パネルは次のとおりです。

CCA アプリケーション**:**
: アプリケーションのバージョンとビルド日を表示します。また、マスター・キー・レジスターの状況も表示します。

アダプター**:**
: コプロセッサーのシリアル番号、ID、およびハードウェア・レベルを表示します。

コマンド・ヒストリー**:**
: コプロセッサーに送信された最新 5 個のコマンドおよびサブコマンドを表示します。

診断**:** コプロセッサーのいずれかの改ざんセンサーが起動されたかどうか、およびログに記録されたエラーがあるかどうかを示し、コプロセッサーのバッテリーの状況を反映します。

エクスポート制御**:**

コプロセッサー内に常駐する機能制御ベクトル (FCV) によって定義された、ノードが使用する暗号鍵の最大強度を表示します。

状況表示パネルを表示するには、以下に手順に従ってください。

1. 「暗号化ノード **(Crypto Node)**」メニューから「状況 **(Status)**」をクリックします。CCA アプリケーションの状況が表示されます。
2. ほかの状況情報を選択するには、下部のボタンを使用します。
3. 「取消 **(Cancel)**」をクリックします。

関連情報:

37 ページの『マスター・キーの管理』
マスター・キーはコプロセッサーの外に格納された状態 で、ローカル・ノードで機能する鍵を暗号化するために使用されます。

アクセス制御データの作成および管理

IBM CCA 暗号化コプロセッサー・サポート・プログラムのアクセス制御システムは、コプロセッサーを使用できる環境を定義します。アクセス制御システムは、CCA コマンドの使用を制限することによって、それを行います。

これらの CCA コマンドのリストについては、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」を参照してください。また、各 verb の記述の最後にある『Required commands』セクションも参照してください。

管理者はユーザーに異なる権限を付与することができます。それにより一部のユーザーは、他のユーザーが使用できない CCA サービスを使用することができます。このセクションには、アクセス制御システムの概要とアクセス制御データを管理するための説明が含まれています。必要なコマンドと、そのコマンドがどのような環境で必要かを知る必要があります。一部のコマンドは、信頼できる個人、または特定の時間に動作する特定のプログラムにのみ許可すべきであるということも検討します。一般には、ユーザーのインストール環境のセキュリティーを弱めるために使用される可能性のある機能を誤って使用可能にしないように、必要なコマンドのみを許可するようにしてください。

コマンドの使用に関する情報は、サポートしたいアプリケーションの資料から取得します。追加のガイダンスについては、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」を参照してください。

アクセス制御の概要

アクセス制御システムは、ロールとユーザー・プロファイルに基づいてコマンドの使用を制限または許可します。

CNM ユーティリティーを使用して、割り当て済みユーザーの必要性と特権に一致するロールを作成します。

ロールに割り当てられた特権 (デフォルト・ロールに許可されていない特権) にアクセスするためには、ユーザーは固有のユーザー・プロファイルを使用してコプロセッサーにログオンする必要があります。それぞれのユーザー・プロファイルは、1 つのロールに関連付けられており、複数のプロファイルが同じロールを使用できます。コプロセッサーは、そのユーザーを識別するプロファイルに関連付けられたパスフレーズを使用してログオンを認証します。

注**:** 「ユーザー」という用語は、人間とプログラムの両方に適用されます。

コプロセッサーには必ず、少なくとも 1 つのロール (デフォルト・ロール) があります。デフォルト・ロールの使用には、ユーザー・プロファイルは必要ありません。ユーザーは誰でも、コプロセッサーにログオンせずに、またはコプロセッサーの認証を受けずに、デフォルト・ロールで許可されているサービスを使用することができます。

例えば、基本システムには以下のロールが含まれている可能性があります。

- アクセス制御管理者: 新規ユーザー・プロファイルを作成し、現行ユーザーのアクセス権を変更できます。
- 鍵管理担当者: 暗号鍵を変更できます。この責任は、第 1 または後続の鍵パーツを入力する権利を使用する 2 人以上の個人によって共有するのが最良です。
- 一般ユーザー: 自分の作業を保護するために暗号化サービスを使用できますが、管理特権はありません。セキュリティー計画で一般ユーザーに対するログオン認証が必要ない場合は、デフォルト・ロールの中でその要件に対応してください。

注**:** 鍵管理担当者またはアクセス制御管理者のロールに割り当てられる個人の数は多くありません。一般に、ログオンしないユーザー数の方が多いため、デフォルト・ロールで付与される権限をもつユーザー数の方が多くなります。

アクセス制御システムの初期状態

初期状態には、初期デフォルト・ロールがあります。

CCA ソフトウェア・サポートをコプロセッサーのセグメント 3 にロードした後、またはアクセス制御システムを初期化した後は、非認証ユーザーによるアクセス制御データの作成とロードを許可する初期デフォルト・ロール用を除き、アクセス制御データは存在しません。

ユーザーの環境に必要なロールとプロファイル (アクセス制御データのロードと暗号鍵の管理に必要な監視ロールを含む) を作成した後は、デフォルト・ロールに割り当てられているすべてのアクセス権を除去します。その後に、非認証ユーザーに付与するアクセス権のみを追加します。

重要**:** 暗号化ノードとそのノードが保護するデータは、デフォルト・ロールがアクセス制御データのロードを許可されている間はセキュアではありません。

関連情報:

50 ページの『初期デフォルト・ロール・コマンド』
このトピックでは、コプロセッサーの初期化後にほかにアクセス制御データが存在しない場合のデフォルト・ロールの特性について説明します。 また、使用可能になっているアクセス制御コマンドもリストします。

ロールの作成

ロールは、そのロールに割り当てられたユーザーのアクセス権とその他の特性を定義します。

ロールを作成するには、以下の手順を実行します。

1. 「アクセス制御 **(Access Control)**」メニューから「ロール **(Roles)**」をクリックします。現在定義されているロールのリストが表示されます。
2. 「新規 **(New)**」を選択して、「ロール管理 (Role Management)」ウィンドウを表示します。このプロセス中いつでも「リスト **(List)**」をクリックすると、現在定義されているロールのリストに戻ります。

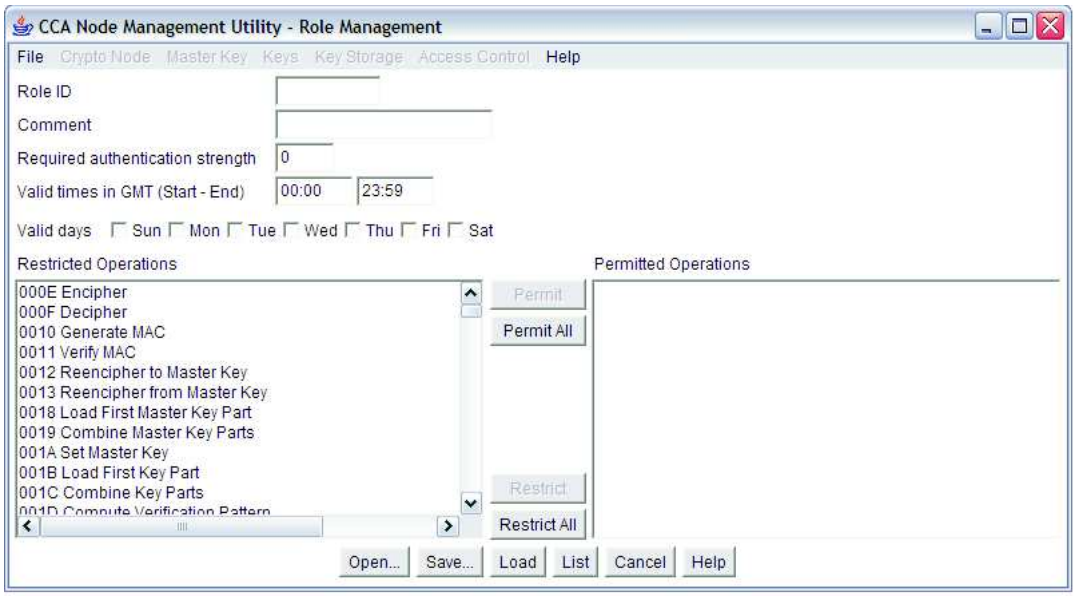

図 3. 「ロール管理 *(Role Management)*」ウィンドウ

3. 次のパラメーターを使用して、ロールを定義します。

   ロール **ID (Role ID)**
   ロールの名前を定義する文字ストリング。この名前は、このロールと関連付けられた各ユーザー・プロファイルに含まれています。

   コメント **(Comment)**
   ロールを記述するためのオプションの文字ストリング。

   必要な認証強度 **(Required authentication strength)**
   ユーザーがログオンすると、提供された認証強度が、そのロールに必要な強度レベルと比較されます。その認証強度が、要求されている認証強度に満たない場合、ユーザーはログオンできません。現在、パスフレーズ認証方式のみがサポートされています。50 の強度を使用します。

   有効時刻と有効曜日 **(Valid times and valid days)**
   いつユーザーがログオンできるか。これらの時間は協定世界時ですので注意してください。まだアクセス制御システムを熟知していない場合は、「IBM CCA Basic Services Reference and Guide for the *IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルのアクセス制御システムに関する章を参照してください。

   制限される操作と許可される操作 **(Restricted operations and permitted operations)**
   ロールが使用を許可されるコマンドを定義するリスト。

   各 CCA API verb には、コプロセッサーからのサービスを得るために、1 つ以上のコマンドが必要な場合があります。サービスを要求するユーザーは、verb を実行するために必要なこれらのコマンドを許可するロールに割り当てられている必要があります。

   CCA verb 呼び出しとコマンドについて詳しくは、「IBM CCA Basic Services Reference and Guide for the *IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルを参照してください。

4. 「保存 **(Save)**」をクリックして、ロールをディスクに保存します。
5. 「ロード **(Load)**」をクリックして、コプロセッサーにロールをロードします。

既存ロールの変更

CNM ユーティリティーを使用すると、ディスクに保管されたロールとコプロセッサーに保管されたロールの編集、およびコプロセッサーに保管されたロールの削除を行うことができます。

注**:** 既存のロールはすべて、新規ロールを作成するためのテンプレートとして使用できます。保存されているロールを開くと、「ロール定義 (Role Definition)」ウィンドウに既存の情報が表示されます。新規ロールに固有の情報のみを変更または入力する必要があります。その後、新規ロール ID を指定してから、ロードまたは保存してください。

ディスクに保管されたロールの編集**:**

このセクションでは、ディスクに保管されている既存のロールを編集する手順について説明します。

ディスクに保管されたロールを編集するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「ロール **(Roles)**」をクリックします。現在定義されているロールのリストが表示されます。
2. 「開く **(Open)**」をクリックします。ファイルの選択を求めるプロンプトが出されます。
3. ファイルを開きます。「ロール定義 (Role Definition)」ウィンドウにデータが表示されます。
4. ロールを編集します。
5. 「保存 **(Save)**」をクリックして、ロールをディスクに保存します。
6. オプション: 「ロード **(Load)**」をクリックして、ロールをコプロセッサーにロードします。

コプロセッサーに保管されたロールの編集**:**

このセクションでは、CCA コプロセッサーに保管されているロールを編集する手順について説明します。

コプロセッサーに保管されたロールを編集するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「ロール **(Roles)**」をクリックします。現在定義されているロールのリストが表示されます。
2. 編集するロールを強調表示します。
3. 「編集 **(Edit)**」をクリックします。「ロール定義 (Role Definition)」パネルにデータが表示されます。
4. ロールを編集します。
5. 「保存 **(Save)**」をクリックします。ロールをディスクに保存します。
6. オプション:「ロード **(Load)**」をクリックします。コプロセッサーにロールをロードします。

コプロセッサーに保管されたロールの削除**:**

このセクションでは、CCA コプロセッサーからロールを削除する手順について説明します。

重要: ロールを削除した場合、CNM ユーティリティーはそのロールに関連付けられたユーザー・プロファイルを自動的に削除または再割り当てしません。ロールを削除する前に、ロールに関連付けられているユーザー・プロファイルを削除または再割り当てすることが必要です。

コプロセッサーに保管されたロールを削除するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「ロール **(Roles)**」をクリックします。現在定義されているロールのリストが表示されます。
2. 削除するロールを強調表示します。
3. 「削除 **(Delete)**」をクリックします。ロールが削除されます。

ユーザー・プロファイルの作成

ユーザー・プロファイルは、特定のユーザーをコプロセッサーに対して識別します。

ユーザー・プロファイルを作成するには、以下の手順に従ってください。

1. 「アクセス制御 **(Access Control)**」メニューから「プロファイル **(Profiles)**」をクリックします。現在定義されているプロファイルのリストが表示されます。
2. 「新規 **(New)**」を選択して、「プロファイル管理 (Profile Management)」ウィンドウを表示します。「プロファイル管理 (Profile Management)」ウィンドウのフィールドを確認するには、図 4 を参照してください。

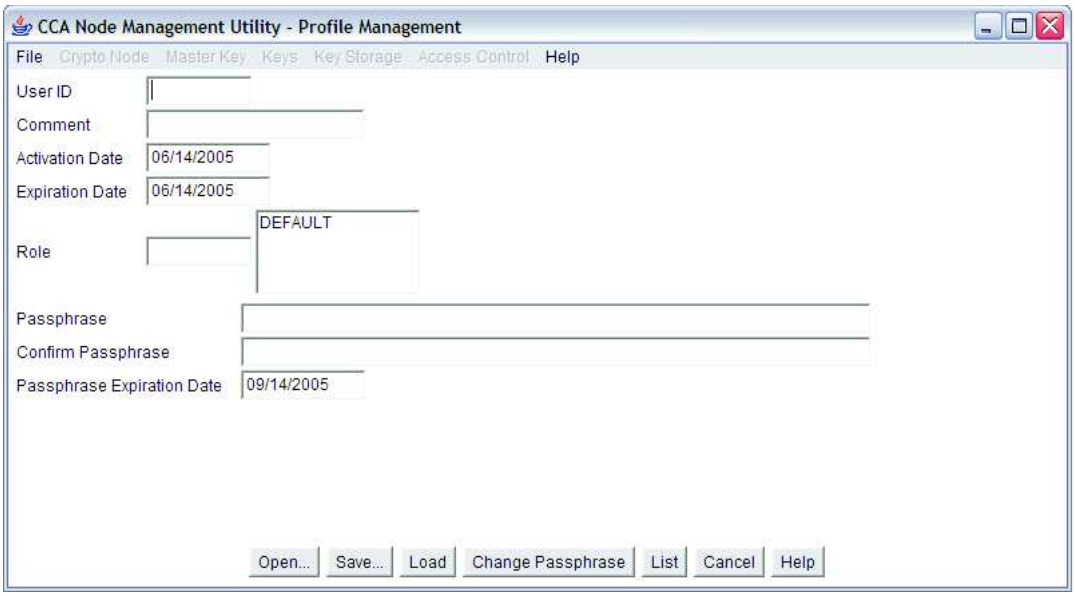

図 *4.* 「プロファイル管理 *(Profile Management)*」パネル

3. ユーザー・プロファイルを定義します。

   ユーザー・プロファイルのフィールドは以下のとおりです。

   ユーザー **ID (User ID)**
   暗号化コプロセッサーのユーザー・プロファイルに指定された名前。

   コメント **(Comment)**
   ユーザー・プロファイルを記述するためのオプションの文字ストリング。

   活動化日と満了日 **(Activation Date and Expiration Date)**
   ユーザーがユーザー・プロファイルにログオンできる最初の日と最後の日。

   ロール **(Role)**
   ユーザー・プロファイルに付与されるアクセス権を定義するロールの名前。

   パスフレーズと確認パスフレーズ **(Passphrase and Confirm Passphrase)**
   ユーザーが暗号化ノードにアクセスするために入力しなければならない文字ストリング。

   パスフレーズの満了日 **(Passphrase Expiration Date)**
   パスフレーズの満了日。ユーティリティーはデフォルトで、これを現在日付から 90 日に設定します。満了日は変更可能です。 すべてのパスフレーズには満了日があって、そのパスフレーズの存続期間を定義します。これは、プロファイル自体の満了日とは異なります。

4. 「保存 **(Save)**」をクリックして、プロファイルをディスクに保存します。
5. オプション: 「ロード **(Load)**」をクリックして、プロファイルをコプロセッサーにロードします。

既存のプロファイルの変更

CNM ユーティリティーを使用すると、ディスクに保管されたプロファイルとコプロセッサーに保管されたプロファイルの編集、およびコプロセッサーに保管されたプロファイルの削除を行うことができます。

注**:** 既存のプロファイルはすべて、新規プロファイルを作成するためのテンプレートとして使用できます。保存されているプロファイルを開くと、「プロファイル定義 (Profile Definition)」ウィンドウに既存の情報が表示されます。新規プロファイルに固有の情報のみを変更または入力する必要があります。その後、新規プロファイル ID を指定してから、ロードまたは保存してください。

ディスクに保管されたユーザー・プロファイルの編集**:**

このセクションでは、ディスクに保管されたユーザー・プロファイルを編集する手順について説明します。

ディスクに保管されたユーザー・プロファイルを編集するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「プロファイル **(Profiles)**」を選択します。現在定義されているプロファイルのリストが表示されます。
2. 「開く **(Open)**」をクリックします。ファイルの選択を求めるプロンプトが出されます。
3. ファイルを開きます。「ユーザー・プロファイル定義 (User Profile Definition)」ウィンドウにデータが表示されます。
4. プロファイルを編集します。
5. 「保存 **(Save)**」をクリックして、プロファイルをディスクに保存します。
6. オプション: 「ロード **(Load)**」をクリックして、プロファイルをコプロセッサーにロードします。

コプロセッサーに保管されたユーザー・プロファイルの編集**:**

このセクションでは、CCA コプロセッサー内のユーザー・プロファイルを編集する手順について説明します。

コプロセッサーに保管されたユーザー・プロファイルを編集するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「プロファイル **(Profiles)**」をクリックします。現在定義されているプロファイルのリストが表示されます。
2. 編集するユーザー・プロファイルを強調表示します。
3. 「編集 **(Edit)**」をクリックします。「プロファイル定義 (Profile Definition)」ウィンドウにデータが表示されます。
4. ユーザー・プロファイルを編集します。
5. 「保存 **(Save)**」をクリックします。プロファイルをディスクに保存します。
6. オプション:「ロード **(Load)**」をクリックします。プロファイルをコプロセッサーにロードします。

コプロセッサーに保管されたユーザー・プロファイルの削除**:**

このセクションでは、CCA コプロセッサーに保管されているユーザー・プロファイルを削除する手順について説明します。

コプロセッサーに保管されたプロファイルを削除するには、以下の手順を実行してください。

1. 「アクセス制御 **(Access Control)**」メニューから「プロファイル **(Profiles)**」をクリックします。現在定義されているユーザー・プロファイルのリストが表示されます。
2. 削除するユーザー・プロファイルを強調表示します。
3. 「削除 **(Delete)**」をクリックします。ユーザー・プロファイルが削除されます。

ユーザー・プロファイルの失敗カウントのリセット**:** 無許可ログオンを防ぐために、アクセス制御システムは、各ユーザー・プロファイルごとにログオン試行の失敗カウントを維持しています。ユーザー・プロファイルの失敗試行回数が、プロファイルに定義された制限を超えると、問題のプロファイルは使用不可になります。

失敗カウントをリセットするには、以下の手順を実行します。

1. 「アクセス制御 **(Access Control)**」メニューから「プロファイル **(Profiles)**」をクリックします。現在定義されているユーザー・プロファイルのリストが表示されます。
2. ユーザー・プロファイルを強調表示します。
3. 「**FC** をリセット」をクリックします。確認ウィンドウが表示されます。
4. 確認して「はい **(Yes)**」をクリックします。ログオン試行の失敗カウントが 0 に設定されます。

アクセス制御システムの初期化

アクセス制御システムを初期化すると、CNM ユーティリティーは、コプロセッサー内のアクセス制御データをクリアし、アクセス制御データのロードに必要なコマンドをデフォルト・ロールに提供します。

重要**:** 暗号化ノードとそのノードが保護するデータは、デフォルト・ロールがアクセス制御データのロードを許可されている間はセキュアではありません。

このアクションを正常に実行すると、インストールされているアクセス制御と鍵が除去されます。したがってこのアクションは、ノードが実動では動作しないようにする可能性のある、重要な操作です。インストール済み環境の中には、この機能の権限をコプロセッサーのロールから除去するよう選択するものがあります。この場合は、CCA 暗号化ノードを初期化するためには、コプロセッサーから CCA ソフトウェアを除去して、CCA ソフトウェアを再インストールする必要があります。

アクセス制御システムを初期化するには、以下の手順を実行します。

1. 「アクセス制御 **(Access Control)**」メニューから「初期化 **(Initialize)**」をクリックします。確認ウィンドウが表示されます。
2. 確認して「はい **(Yes)**」を選択します。ユーティリティーがアクセス制御システムを初期化します。

注**:** CCA Node Management ユーティリティーを開始するには、**csufcnm** コマンドを入力します。CNM ユーティリティーのロゴが表示され、メイン・ウィンドウが表示されます。

暗号鍵の管理

CNM ユーティリティーを使用すると、マスター・キーの管理、1 次鍵暗号鍵 (KEK) の管理、ならびにデータ暗号化規格 (DES)、公開鍵アルゴリズム (PKA)、および Advanced Encryption Standard (AES) 鍵ストアのリセットと管理を行うことができます。鍵タイプは、以下のように定義されます。

マスター・キーは、平文テキスト (暗号化されない) で格納される 特殊 KEK で、コプロセッサーのセキュア・モジュール内に保持されます。3 種類のマスター・キー (DES、PKA、および AES) がサポートされています。これらは、他の鍵をセキュア・モジュールの外に保管できるように、それらの鍵をラップするために使用されます。DES および PKA マスター・キーは、3 個の 56 ビット DES 鍵から形成される 168 ビットの PKA マスター・キーです。AES マスター・キーは 256 ビットの鍵です。

1 次 **KEK** は、暗号化ノードによって共有される DES 鍵で、トランスポート鍵と呼ばれることもあります。これらは、ノードによって共有されるほかの鍵を暗号化するために使用されます。1 次 KEK は、マスター・キーと同様に、鍵パーツからインストールされます。鍵パーツの知識は、知識分割、二重管理セキュリティー・ポリシーを達成するために、2 人の人物によって部分的に共有することができます。

ほかの **DES** 鍵、**PKA** 鍵、および **AES** 鍵は、メディア・アクセス制御 (MAC) 鍵、DATA 鍵および秘密 PKA 鍵などの暗号化サービスを提供するために使用される暗号化された鍵です。

注**:** 鍵パーツの管理は各製造元および各暗号製品の間で異なるため、暗号化されていない鍵パーツを交換する場合は、交換されたデータがどのように使用されるかについて両者が理解していることを必ず確認してください。

マスター・キーの管理

マスター・キーはコプロセッサーの外に格納された状態 で、ローカル・ノードで機能する鍵を暗号化するために使用されます。

CCA では 3 個のマスター・キー・レジスターを定義しています。

- **current-master-key** レジスターには、ローカル・キーを暗号化および復号するために現在コプロセッサーによって使用されているマスター・キーが保管されます。
- **old-master-key** レジスターには前のマスター・キーが保管され、そのマスター・キーによって暗号化された鍵の復号に使用されます。
- **new-master-key** レジスターは、新規マスター・キーの形成のために累積されたマスター・キー情報を保管するために使用される一時ロケーションです。

IBM Common Cryptographic Architecture (CCA) サポート・プログラムでは 3 セットのマスター・キー・レジスターを使用しています。1 セットは DES (対称) 鍵を暗号化するためのもの、1 セットは PKA 秘密 (非対称) 鍵を暗号化するためのもの、そして 1 セットは AES (対称) 鍵を暗号化するためのものです。

注**:**

1. Master_Key_Distribution の master-key-administration verb は、AES マスター・キーをサポートしていません。CCA Master_Key_Process および Master_Key_Distribution を使用するプログラムの場合、master-key-administration verb は、ASYM-MK キーワードを使用して操作を PKA 非対称マスター・キー・レジスターに向けたり、SYM-MK キーワードを使用して DES 対称マスター・キー・レジスターに向けたり、両方のキーワードを使用して DES 対称マスター・キー・レジスターおよび PKA 非対称マスター・キー・レジスター両方のセットに向けたりすることができます。CNM ユーティリティーは **BOTH** オプションを使用します。別のプログラムを使用してマスター・キーをロードし、このプログラムが SYM-MK または ASYM-MK マスター・キー・レジスターのいずれかに限定して操作を行う場合は、一般に CNM ユーティリティーを使用してこれらのマスター・キーを使用することはできなくなります。AES マスター・キーは、DES および PKA マスター・キーとは関係なく機能することに注意してください。
2. インストール済み環境に、CCA のロードされた暗号化コプロセッサーが複数ある場合は、それぞれのコプロセッサーで個別にマスター・キーを管理する必要があります。
3. インストール済み環境に、CCA のロードされた暗号化コプロセッサーを複数もつサーバーがある場合は、これらのコプロセッサーに全く同じマスター・キーをインストールする必要があります。

関連情報:

29 ページの『CCA アプリケーションの状況情報の取得』
CNM ユーティリティー・コプロセッサーを使用して、CCA アプリケーションの状況を取得することができます。

既存マスター・キーの検証:

CNM ユーティリティーは、マスター・キー・レジスターに保管されている各マスター・キーの検証番号を生成します。この番号は鍵を識別しますが、実際のキー値に関する情報を明らかにしません。

マスター・キーの検証番号を表示するには、以下の手順に従ってください。

1. 「マスター・キーのロード (Load Master Key)」ウィンドウから、「マスター・キー **(Master Key)**」をクリックします。
2. 「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」または「**AES** マスター・キー **(AES Master Key)**」のいずれかを選択し、「検証 **(Verify)**」をクリックします。サブメニューが表示されます。
3. 結果のサブメニューからマスター・キー・レジスターを選択します。そのレジスターに保管されている鍵の検証番号が表示されます。

マスター・キーの自動ロード:

CNM ユーティリティーは、コプロセッサーでマスター・キーを自動的に設定することができます。マスター・キーの値は、ユーティリティーからは表示できません。

重要: 不明の値のマスター・キーを失くした場合は、それに付加された鍵を復号できません。

マスター・キーを自動的にロードするには、以下の手順に従ってください。

1. 「マスター・キーのロード (Load Master Key)」ウィンドウから、「マスター・キー **(Master Key)**」をクリックします。
2. 「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」または「**AES** マスター・キー **(AES Master Key)**」のいずれかを選択します。
3. 「自動設定 **(Auto Set)**」または「ランダム **(Random)**」を選択します。コマンドの確認を求めるプロンプトが出されます。
4. 「はい **(Yes)**」をクリックします。コプロセッサーがマスター・キーを作成して設定します。

注:

1. 「自動設定 **(Auto Set)**」オプションは、ホスト・システム・メモリーから暗号化されていない鍵パーツを渡すので、「ランダム **(Random)**」オプションをお勧めします。
2. マスター・キーを設定または自動設定する場合は、前の鍵で暗号化された鍵はすべて再暗号化する必要があります。

関連情報:

41 ページの『保管された鍵の再暗号化』

鍵パーツからの新規マスター・キーのロード:

新規マスター・キーをコプロセッサーに設定するには、新規マスター・キー・レジスターに鍵の任意の部分を入力して、新規マスター・キーを設定します。

新規マスター・キーを設定するには、以下の手順に従ってください。

1. 「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」または「**AES** マスター・キー **(AES Master Key)**」のいずれかを選択し、「パーツ **(Parts)**」をクリックします。「マスター・キーのロード (Load Master Key)」ウィンドウが表示されます (図 5 を参照)。

図 *5.* 「マスター・キーのロード *(Load Master Key)*」ウィンドウ

2. 編集中の鍵パーツ (最初のパーツ **(First Part)**、中間のパーツ **(Middle Part)**、または最後のパーツ **(Last Part)**) のラジオ・ボタンを選択します。
3. 次のいずれかのアクションを実行してデータを入力します。
   - 「新規 **(New)**」をクリックして、入力されてエラーになっているデータをクリアする。
   - 「開く **(Open)**」をクリックして、前からあるデータを検索する。
   - 「生成 **(Generate)**」をクリックして、コプロセッサーの生成した乱数をフィールドに入力する。
   - 手動で「マスター・キー・パーツ **(Master Key Part)**」フィールドにデータを入力します。各フィールドは、4 桁の 16 進数を受け入れます。
4. 「ロード **(Load)**」をクリックして、新規マスター・キー・レジスターに鍵パーツをロードします。
5. 「保存 **(Save)**」をクリックして、鍵パーツをディスクに保存します。

   重要**:** ディスクに保存される鍵パーツは暗号化されません。鍵パーツの入ったディスクは、金庫または貴重品保管室に入れておくことを検討してください。

   注**:** パーツから鍵を作成する場合は、最初のパーツと最後パーツの両方が必要です。中間のパーツはオプションです。
6. 前述のステップを繰り返して、残りの鍵パーツを新規マスター・キー・レジスターにロードします。

   注**:** 知識分割セキュリティー・ポリシーの場合は、別々の人が別々の鍵パーツを入力する必要があります。二重管理セキュリティー・ポリシーを実施するためには、アクセス制御システムは 1 番目の鍵を入力する権限を 1 つのロールに割り当て、その後の鍵パーツを入力する権限を別のロールに割り当てる必要があります。その後、許可ユーザーがログオンして、それぞれの鍵パーツを入力できます。

7. 「マスター・キー **(Master Key)**」メニューから「**DES/PKA** マスター・キー **(DES/PKA Master Keys)**」または「**AES** マスター・キー **(AES Master Key)**」のいずれかを選択します。
8. 「設定 **(Set)**」をクリックします。以下のようにユーティリティーがデータを転送します。
   a. 現行マスター・キー・レジスターから、古いマスター・キー・レジスターへ。次に古いマスター・キーを削除します。
   b. 新規マスター・キー・レジスターから、現行マスター・キー・レジスターへ。

新規マスター・キーを設定した後、現在ストレージに入っている鍵を再暗号化します。

関連リンク*:* 41 ページの『保管された鍵の再暗号化』

鍵ストレージの管理

CNM ユーティリティーは、鍵の基本的な鍵ストレージ管理機能を使用可能にします。これらのユーティリティー機能は、包括的な鍵管理システムを形成していません。

アプリケーション・プログラムは、反復的な鍵管理タスクを実行するのに適しています。

鍵ストレージは、ユーザーまたはユーザーのアプリケーションが定義したラベルを使用して、鍵ラベルによってアクセスする鍵のリポジトリーです。DES (データ暗号化規格) 鍵、PKA (公開鍵アルゴリズム) RSA (Rivest-Shamir-Adleman) 鍵、および AES (Advanced Encryption Standard) 鍵は、別々のストレージ・システムに保持されます。また、鍵ストレージのもつ PKA 鍵用の内部ストレージは限られています。コプロセッサーに保管された鍵は、この議論では鍵ストレージの一部とは見なされません。

注**:**

1. ご使用のサーバーに CCA のロードされた暗号化コプロセッサーが複数ある場合は、鍵ストレージが正しく機能するためには、それらのコプロセッサーに全く同じマスター・キーがインストールされている必要があります。
2. CNM ユーティリティーは最大 1,000 個の鍵ラベルを表示します。1,000 個を超える鍵ラベルが鍵ストレージにある場合は、アプリケーション・プログラムを使用して管理してください。

鍵ストレージの作成または初期化**:** データ暗号化規格 (DES) 鍵、公開鍵アルゴリズム (PKA) または Advanced Encryption Standard (AES) 鍵の鍵ストレージを作成または初期化するには、以下の手順を実行します。

1. 「鍵ストレージ **(Key Storage)**」メニューから「**DES** 鍵ストレージ **(DES Key Storage)**」、「**PKA** 鍵ストレージ **(PKA Key Storage)**」、または「**AES** 鍵ストレージ **(AES Key Storage)**」を選択します。
2. 結果のサブメニューから「初期化 **(Initialize)**」をクリックします。「DES 鍵ストレージの初期化 (Initialize DES Key Storage)」、「PKA 鍵ストレージの初期化 (Initialize PKA Key Storage)」、または「AES 鍵ストレージの初期化 (Initialize AES Key Storage)」ウィンドウが表示されます。
3. 鍵ストレージ・ファイルの記述を入力してください。
4. 「初期化 **(Initialize)**」をクリックします。鍵ストレージ・データ・セットの名前の入力を求めるプロンプトが出されます。
5. ファイルの名前を入力して保存します。鍵ストレージ・ファイルがホスト上に作成されます。

   注**:** 同じ名前のファイルが存在する場合は、選択を確認するよう求めるプロンプトが出されます。この理由は、鍵ストレージを初期化するとファイルが変更され、そのファイルに鍵がある場合は、それらが消去されるためです。

保管された鍵の再暗号化**:** 新規マスター・キーでストレージ内の鍵を再暗号化するには、以下の手順を実行します。

1. 「鍵ストレージ **(Key Storage)**」メニューから「**DES** 鍵ストレージ **(DES Key Storage)**」、「**PKA** 鍵ストレージ **(PKA Key Storage)**」、または「**AES** 鍵ストレージ **(AES Key Storage)**」を選択します。
2. 結果のサブメニューから「管理 **(Manage)**」をクリックします。「DES 鍵ストレージ管理 (DES Key Storage Management)」、「PKA 鍵ストレージ管理 (PKA Key Storage Management)」、または「AES 鍵ストレージ管理 (AES Key Storage Management)」ウィンドウが表示されます。このウィンドウ・パネルに、ストレージ内の鍵のラベルがリストされます。
3. 「再暗号化 **(Reencipher)**」をクリックします。鍵は、現行マスター・キー・レジスターの鍵で再暗号化されます。

保管された鍵の削除**:** 保管された鍵を削除するには、以下の手順を実行します。

1. 「鍵ストレージ **(Key Storage)**」から、「**DES** 鍵ストレージ **(DES Key Storage)**」、「**PKA** 鍵ストレージ **(PKA Key Storage)**」、または「**AES** 鍵ストレージ **(AES Key Storage)**」をクリックします。
2. 結果のサブメニューから「管理 **(Manage)**」をクリックします。「DES 鍵ストレージ管理 (DES Key Storage Management)」、「PKA 鍵ストレージ管理 (PKA Key Storage Management)」、または「AES 鍵ストレージ管理 (AES Key Storage Management)」ウィンドウが表示されます。このウィンドウに、ストレージ内の鍵のラベルがリストされます。

   フィルター基準を設定して、ストレージ内の鍵のサブセットをリストすることができます。例えば、フィルター基準に `*.mac` を入力してリストをリフレッシュすると、サブセットは、.mac で終了するラベルをもつ鍵に制限されます (アスタリスクはワイルドカード文字です)。
3. 削除する鍵の鍵ラベルを強調表示します。
4. 「削除 **(Delete)**」をクリックします。確認メッセージが表示されます。
5. 「はい **(Yes)**」をクリックします。保管された鍵が削除されることを確認します。

鍵ラベルの作成**:** 鍵ラベルを作成するには、以下の手順を実行します。

1. 「鍵ストレージ **(Key Storage)**」メニューから「**DES** 鍵ストレージ **(DES Key Storage)**」、「**PKA** 鍵ストレージ **(PKA Key Storage)**」、または「**AES** 鍵ストレージ **(AES Key Storage)**」をクリックします。
2. 結果のサブメニューから「管理 **(Manage)**」をクリックします。「DES 鍵ストレージ管理 (DES Key Storage Management)」、「PKA 鍵ストレージ管理 (PKA Key Storage Management)」、または「AES 鍵ストレージ管理 (AES Key Storage Management)」ウィンドウが表示されます。このウィンドウに、ストレージ内の鍵のラベルがリストされます。

   フィルター基準を設定して、ストレージ内の鍵のサブセットをリストすることができます。例えば、フィルター基準に `*.mac` を入力してリストをリフレッシュすると、サブセットは、.mac で終了するラベルをもつ鍵に制限されます (アスタリスクはワイルドカード文字です)。
3. 「新規 **(New)**」をクリックします。鍵ラベルの入力を求めるプロンプトが出されます。
4. 「ロード **(Load)**」をクリックします。鍵ラベルがストレージにロードされます。

1 次 DES KEK の作成と保管

鍵暗号鍵 (KEK) はデータ暗号化規格 (DES) マスター・キーの下で暗号化され、ローカルで使用するために DES 鍵ストレージに保管されます。

KEK の作成に使用される鍵パーツは、ランダムに生成したり、平文テキスト情報として入力したりできます。これらのパーツは、他のノードに移送したり、ローカル KEK を再作成したりするために、平文テキストでディスクまたはディスケットに保存することもできます。

注**:** 暗号化ノード管理 (CNM) ユーティリティーは、ノード間の鍵の移送には、DES KEK のみをサポートします。アプリケーションは、公開鍵ベースまたは (AES) ベースの鍵の配布に必要なサービスを提供するために、CCA API を使用することができます。

1 次 DES KEK (または他の 2 倍長の操作キー) を作成し、保管するには、以下の手順に従ってください。

1. 「鍵 **(Keys)**」メニューから、「**1** 次 **DES** 鍵暗号鍵 **(Primary DES Key-encrypting keys)**」をクリックします。「1 次 DES 鍵暗号鍵 (Primary DES Key-encrypting keys)」ウィンドウが表示されます。

   「新規 **(New)**」をクリックするといつでも、すべてのデータ・フィールドを消去して、すべてのラジオ・ボタンをデフォルト設定にリセットすることができます。
2. 最初のパーツ **(First Part)**、中間のパーツ **(Middle Part)**、または最後のパーツ **(Last Part)** の中から、入力する鍵パーツのラジオ・ボタンを選択します。
3. 以下のいずれかのアクションを実行して、「鍵パーツ **(Key Part)**」フィールドにデータを入力します。
   - 「開く **(Open)**」をクリックして、「保存 **(Save)**」コマンドを使用して以前にディスク上に保管した、以前から存在する「鍵パーツ **(Key Part)**」、「制御ベクトル **(Control Vector)**」、および「鍵ラベル **(Key Label)**」のデータを取り出します。
   - 「生成 **(Generate)**」をクリックして、コプロセッサーの生成した乱数を「鍵パーツ **(Key Part)**」フィールドに入力します。
   - 手動で「鍵パーツ **(Key Part)**」フィールドにデータを入力します。各「鍵パーツ **(Key Part)**」フィールドは、4 桁の 16 進数を受け入れます。
4. 鍵の制御ベクトルを選択します。
   - デフォルトの KEK 制御ベクトルを使用するには、適切な「デフォルト・インポーター **(Default Importer)**」または「デフォルト・エクスポーター **(Default Exporter)**」ラジオ・ボタンを選択します。
   - カスタム制御ベクトルを使用するには、「カスタム **(Custom)**」ラジオ・ボタンを選択します。2 倍長キーとして、制御ベクトルの左半分または右半分を「制御ベクトル **(Control Vector)**」フィールドに入力します。key-part ビット (ビット 44) がオンになっており、制御ベクトルの各バイトが偶数パリティーになっていなければならないことに注意してください。

     制御ベクトルについて詳しくは、「IBM CCA Basic Services Reference and Guide for the *IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルを参照してください。
5. 鍵ストレージ内で鍵トークンを識別するために鍵ラベルを入力します。
6. 「ロード **(Load)**」をクリックして鍵パーツをコプロセッサーにロードし、結果の鍵トークンを鍵ストレージに保管します。
7. 「保存 **(Save)**」をクリックして、暗号化されていない鍵パーツとその鍵パーツに関連付けられた制御ベクトルおよび鍵ラベルの値をディスクに保存します。
8. ステップ 2 から 7 に従って、残りの鍵パーツ情報をディスクに保存するか、鍵ストレージにロードします。単一鍵の各パーツには、必ず同じ鍵ラベルを使用してください。

CNI ユーティリティーを使用した他のノードの作成

CCA Node Initialization (CNI) ユーティリティー用の CNI リストを作成することにより、ターゲット・ノード上で CNM ユーティリティーを実行せずに、ディスクに保管されている鍵とアクセス制御データを他の暗号化ノードにロードすることができます。

CNI ユーティリティーを使用してノードをセットアップするには、以下の手順を実行します。

1. **csufcnm** コマンドを入力して、CCA Node Management ユーティリティーを開始します。CNM ユーティリティーのロゴとメインパネルが表示されます。
2. 他のノードにインストールするアクセス制御データと鍵を、ホストまたはポータブル・メディア (ディスケットなど) に保存します。ターゲット・ノードで CNI ユーティリティーを実行すると、このユーティリティーは全く同じディレクトリー・パスで各ファイルを検索します。次に例を示します。
   - 確立されたノード・ディレクトリー `c:\IBM4764\profiles` にユーザー・プロファイルを保存すると、CNI ユーティリティーはターゲット・ノード・ディレクトリー `c:\IBM4764\profiles` を検索します。
   - ディスケット・ディレクトリー `a:\profiles` にユーザー・プロファイルを保存すると、CNI ユーティリティーはターゲット・ノード・ディレクトリー `a:\profiles` を検索します。
3. 「ファイル **(File)**」メニューから「**CNI** エディター **(CNI Editor)**」をクリックします。図 6 のような「CCA Node Initialization Editor」ウィンドウが表示されます。

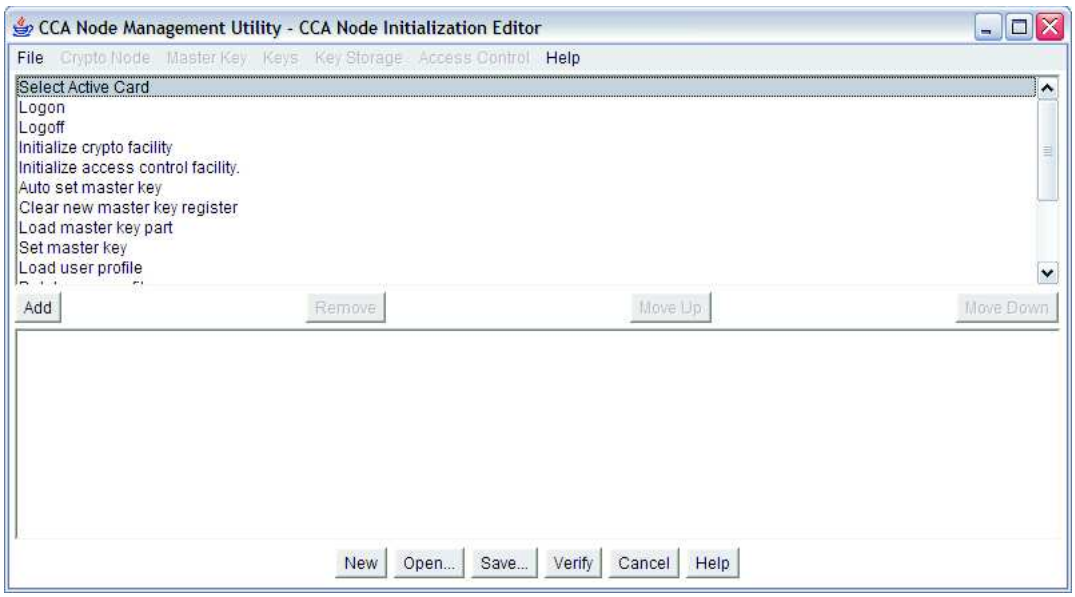

図 6. 「*CCA Node Initialization Editor*」ウィンドウ

ウィンドウ上部のペインに、CNI リストに追加できる機能が表示されます。下のペインには、現行の CNI リストに含まれている機能がリストされます。リスト内のマスター・キーの参照は、DES および PKA マスター・キーを指します。

4. 必要な機能を追加します。CNI リストに機能を追加するには、以下の手順を実行してください。
   a. 機能を強調表示します。
   b. 「追加 **(Add)**」をクリックします。その機能が CNI リストに追加されます。

注**:** 選択した機能がデータ・オブジェクト (鍵パーツ、鍵ストレージ・ファイル、ユーザー・プロファイル、またはロールなど) をロードする場合、ロードするオブジェクトのファイル名または ID の入力を求めるプロンプトが表示されます。

5. 「上へ移動 **(Move Up)**」および「下へ移動 **(Move Down)**」ボタンを使用して、CNM ユーティリティーを使用するときと同じ順序を反映するように機能を編成します。例えば、アクセス制御データをロードしている場合などです。
6. 「検証 **(Verify)**」をクリックして、オブジェクトが正しく作成されたことを確認します。
7. 「保存 **(Save)**」をクリックします。CNI リスト・ファイルの名前とディレクトリー・ロケーションの選択を求めるプロンプトが出されます。
8. CNI リスト・ファイルを保存します。このリスト・ファイルには、CNI リストに指定されたデータ・オブジェクトは含まれていません。
9. CNI ユーティリティーに必要なファイルを、ソース・ホスト上のロケーションをミラーリングする、ターゲット・ホストのディレクトリー・ロケーションにコピーします。ファイルをポータブル・メディアに保存した場合は、そのメディアをターゲット・ノードに挿入します。
10. ターゲット・ノードから **csufcni** コマンドを入力して、CNI ユーティリティーを使用するリストを実行します。

    CNI リストにログオンが含まれている場合は、(ファイル名を指定せずに) コマンド行に **csulcni** または **csuncni** と入力します。CNI ユーティリティーのヘルプ情報で、ID とパスフレーズを入力するための構文について説明します。

    CNI ユーティリティーは、CNI リストの指定に従って、ホストまたはポータブル・メディアからコプロセッサーにファイルをロードします。

CCA API で使用するためのアプリケーションの作成

Common Cryptographic Architecture (CCA) API で使用できるアプリケーションを作成できます。

サンプル・ルーチンのソース・コードは、ソフトウェアに含まれています。コプロセッサーとサポート・プログラムをテストするために、含まれたサンプルを使用することができます。

注**:** このセクションで参照するファイル・ロケーションは、デフォルト・ディレクトリー・パスです。

CCA verb の概要

アプリケーション・プログラムとユーティリティー・プログラムは CCA verb を呼び出して、暗号化コプロセッサーにサービス要求を出します。「*verb*」という用語は、アプリケーション・プログラムが開始できるアクションを暗黙指定します。次に、オペレーティング・システム・コードが、コプロセッサーの物理デバイス・ドライバー (PDD) を呼び出します。この API からアクセスされるハードウェアとソフトウェアは、それら自体が統合サブシステムです。

verb 呼び出しは、C プログラミング言語の標準構文で作成されており、エントリー・ポイント名、verb パラメーター、およびこれらのパラメーターの変数を含んでいます。

CCA セキュリティー・アプリケーション・プログラミング・インターフェース (API) 用にプログラムする際に使用できる verb、変数、およびパラメーターの詳細なリストについては、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルを参照してください。

C プログラム構文での CCA verb の呼び出し

すべてのオペレーティング・システム環境で、標準 C プログラミング言語構文を使用して CCA API verb 呼び出しをコーディングすることができます。

すべての CCA セキュリティー API verb の関数呼び出しのプロトタイプは、ヘッダー・ファイルに含まれています。ファイルとそれらのデフォルト配布ロケーションは、次のとおりです。

AIX /usr/include/

これらの verb 宣言を組み込むには、プログラム内で次のコンパイラー指示を使用してください。

AIX
```
#include "csufincl.h"
```

CCA セキュリティー API verb への呼び出しを出すには、大文字で verb エントリー・ポイント名をコーディングしてください。パラメーター ID はコンマで分離し、括弧で囲んでください。呼び出しの最後にはセミコロン文字を付けてください。次に例を示します。

```
CSNBCKI (&return_code,
         &reason_code,
         &exit_data_length,    /* exit_data_length */
         exit_data,            /* exit_data        */
         clear_key,
         key_token);
```

注: CCA 呼び出しの 3 番目と 4 番目のパラメーター *exit_data_length* と *exit_data* は、現在 CCA 暗号化コプロセッサー・サポート・プログラムではサポートされていません。これらのパラメーターに Null のアドレス・ポインターをコーディングすることは許容されますが、*exit_data_length* パラメーターを使用して、0 の値になる長整数を指定することを推奨します。

CCA アプリケーション・プログラムのコンパイルとリンク

CCA 暗号化コプロセッサー・サポート・プログラムには、C 言語のソース・コードとサンプル・プログラムの Make ファイルが含まれています。

このファイルとそのデフォルト配布ロケーションは以下のとおりです。

AIX /usr/lpp/csufx.4765/samples/c.

CCA を使用するアプリケーション・プログラムをコンパイルし、コンパイルされたプログラムを CCA ライブラリーにリンクします。このライブラリーとそのデフォルト配布ロケーションは以下のとおりです。

AIX /usr/lib/libcsufcca.a.

サンプル C ルーチン: MACの生成

このトピックでは、CCA verb 呼び出しの実際の適用方法を示すために、CCA 暗号化コプロセッサー・サポート・プログラムに含まれているサンプル C プログラミング言語ルーチンについて説明します。

製品 Web サイトにもサンプル・プログラムがあります。そのサンプル・プログラムは、CCA 実装のパフォーマンスを理解するのに役立てることができます。

サンプル・ルーチンは、テキスト・ストリングにメッセージ認証コード (MAC) を生成して、MAC を検証します。MAC を生成して検証するために、ルーチンは以下のことを行います。

1. **Key_Generate**(CSNBKGN) verb を呼び出して、MAC および MACVER 鍵ペアを作成します。
2. **MAC_Generate** (CSNBMGN) verb を呼び出して、MAC 鍵によりテキスト・ストリングに MAC を生成します。

3. **MAC_Verify** (CSNBMVR) verb を呼び出して、MACVER 鍵によりテキスト・ストリング MAC を検証します。

図 7 にサンプル・ルーチンを示します。verb とそのパラメーターの説明については、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」を参照してください。これらの verb は、以下の表にリストされています。

表 5. サンプル・ルーチンによって呼び出される *verb*

verb	エントリー・ポイント名
Key_Generate	CSNBKGN
MAC_Generate	CSNBMGN
MAC_Verify	CSNBMVR

図 7. サンプル *C* ルーチン: *MAC* の生成

```
/*********************************************************************/
/*                                                                   */
/* Module Name: mac.c                                                */
/*                                                                   */
/* DESCRIPTIVE NAME: Cryptographic Coprocessor Support Program       */
/*                   C language source code example                  */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/* Licensed Materials - Property of IBM                              */
/*                                                                   */
/* (C) Copyright IBM Corp. 1997-2010 All Rights Reserved             */
/*                                                                   */
/* US Government Users Restricted Rights - Use duplication or        */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/*        NOTICE TO USERS OF THE SOURCE CODE EXAMPLES                */
/*                                                                   */
/* The source code examples provided by IBM are only intended to     */
/* assist in the development of a working software program. The       */
/* source code examples do not function as written: additional       */
/* code is required. In addition, the source code examples may       */
/* not compile and/or bind successfully as written.                  */
/*                                                                   */
/* International Business Machines Corporation provides the source   */
/* code examples, both individually and as one or more groups,       */
/* "as is" without warranty of any kind, either expressed or         */
/* implied, including, but not limited to the implied warranties of  */
/* merchantability and fitness for a particular purpose. The entire  */
/* risk as to the quality and performance of the source code         */
/* examples, both individually and as one or more groups, is with    */
/* you. Should any part of the source code examples prove defective, */
/* you (and not IBM or an authorized dealer) assume the entire cost  */
/* of all necessary servicing, repair or correction.                 */
/*                                                                   */
/* IBM does not warrant that the contents of the source code         */
/* examples, whether individually or as one or more groups, will     */
/* meet your requirements or that the source code examples are       */
/* error-free.                                                       */
/*                                                                   */
/* IBM may make improvements and/or changes in the source code       */
/* examples at any time.                                             */
/*                                                                   */
/* Changes may be made periodically to the information in the        */
/* source code examples; these changes may be reported, for the      */
```

```
/* sample code included herein, in new editions of the examples.     */
/*                                                                   */
/* References in the source code examples to IBM products, programs, */
/* or services do not imply that IBM intends to make these           */
/* available in all countries in which IBM operates. Any reference   */
/* to the IBM licensed program in the source code examples is not    */
/* intended to state or imply that IBM's licensed program must be    */
/* used. Any functionally equivalent program may be used.            */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/* This example program:                                             */
/*                                                                   */
/* 1) Calls the Key_Generate verb (CSNBKGN) to create a MAC (message */
/*    authentication code) key token and a MACVER key token.         */
/*                                                                   */
/* 2) Calls the MAC_Generate verb (CSNBMGN) using the MAC key token  */
/*    from step 1 to generate a MAC on the supplied text string      */
/*    (INPUT_TEXT).                                                  */
/*                                                                   */
/* 3) Calls the MAC_Verify verb (CSNBMVR) to verify the MAC for the  */
/*    same text string, using the MACVER key token created in        */
/*    step 1.                                                        */
/*                                                                   */
/*********************************************************************/
#include <stdio.h>
#include <string.h>

#ifdef _AIX
  #include <csufincl.h>
#elif __WINDOWS__
  #include "csunincl.h"
#else
  #include "csulincl.h"     /* else linux */
#endif

/* Defines */
#define KEY_FORM            "OPOP"
#define KEY_LENGTH          "SINGLE  "
#define KEY_TYPE_1          "MAC     "
#define KEY_TYPE_2          "MACVER  "
#define INPUT_TEXT          "abcdefghijklmn0987654321"
#define MAC_PROCESSING_RULE "X9.9-1  "
#define SEGMENT_FLAG        "ONLY    "
#define MAC_LENGTH          "HEX-9   "
#define MAC_BUFFER_LENGTH   10

void main()
{
  static long          return_code;
  static long          reason_code;
  static unsigned char key_form[4];
  static unsigned char key_length[8];
  static unsigned char mac_key_type[8];
  static unsigned char macver_key_type[8];
  static unsigned char kek_key_id_1[64];
  static unsigned char kek_key_id_2[64];
  static unsigned char mac_key_id[64];
  static unsigned char macver_key_id[64];
  static long          text_length;
  static unsigned char text[26];
  static long          rule_array_count;
  static unsigned char rule_array[3][8];       /* Max 3 rule array elements  */
  static unsigned char chaining_vector[18];
  static unsigned char mac_value[MAC_BUFFER_LENGTH];
```

```
/* Print a banner */
printf("Cryptographic Coprocessor Support Program example program.\n");

/* Set up initial values for Key_Generate call */
return_code = 0;
reason_code = 0;
memcpy (key_form,        KEY_FORM,   4);     /* OPOP key pair              */
memcpy (key_length,      KEY_LENGTH, 8);     /* Single-length keys         */
memcpy (mac_key_type,    KEY_TYPE_1, 8);     /* 1st token, MAC key type    */
memcpy (macver_key_type, KEY_TYPE_2, 8);     /* 2nd token, MACVER key type */
memset (kek_key_id_1,  0x00, sizeof(kek_key_id_1));  /* 1st KEK not used   */
memset (kek_key_id_2,  0x00, sizeof(kek_key_id_2));  /* 2nd KEK not used   */
memset (mac_key_id,    0x00, sizeof(mac_key_id));    /* Init 1st key token */
memset (macver_key_id, 0x00, sizeof(macver_key_id)); /* Init 2nd key token */

/* Generate a MAC/MACVER operational key pair */
CSNBKGN(&return_code,
        &reason_code,
        NULL,                                /* exit_data_length           */
        NULL,                                /* exit_data                  */
        key_form,
        key_length,
        mac_key_type,
        macver_key_type,
        kek_key_id_1,
        kek_key_id_2,
        mac_key_id,
        macver_key_id);

/* Check the return/reason codes. Terminate if there is an error.          */
if (return_code != 0 || reason_code != 0) {
  printf ("Key_Generate failed: ");             /* Print failing verb      */
  printf ("return_code = %ld, ",  return_code); /* Print return code       */
  printf ("reason_code = %ld.\n", reason_code); /* Print reason code       */
  return;
}
else
  printf ("Key_Generate successful.\n");

/* Set up initial values for MAC_Generate call */
return_code = 0;
reason_code = 0;
text_length = sizeof (INPUT_TEXT) - 1;          /* Length of MAC text      */
memcpy (text, INPUT_TEXT, text_length);         /* Define MAC input text   */
rule_array_count = 3;                           /* 3 rule array elements   */
memset (rule_array, ' ', sizeof(rule_array));   /* Clear rule array        */
memcpy (rule_array[0], MAC_PROCESSING_RULE, 8); /* 1st rule array element  */
memcpy (rule_array[1], SEGMENT_FLAG,        8); /* 2nd rule array element  */
memcpy (rule_array[2], MAC_LENGTH,          8); /* 3rd rule array element  */
memset (chaining_vector, 0x00, 18);             /* Clear chaining vector   */
memset (mac_value, 0x00, sizeof(mac_value));    /* Clear MAC value         */

/* Generate a MAC based on input text */
CSNBMGN ( &return_code,
         &reason_code,
         NULL,                                  /* exit_data_length        */
         NULL,                                  /* exit_data               */
         mac_key_id,                            /* Output from Key_Generate*/
         &text_length,
         text,
         &rule_array_count,
         &rule_array[0][0],
         chaining_vector,
         mac_value);

/* Check the return/reason codes. Terminate if there is an error.          */
if (return_code != 0 || reason_code != 0) {
```

```
    printf ("MAC Generate Failed: ");             /* Print failing verb      */
    printf ("return_code = %ld, ",  return_code); /* Print return code       */
    printf ("reason_code = %ld.\n", reason_code); /* Print reason code       */
    return;
  }
  else {
    printf ("MAC_Generate successful.\n");
    printf ("MAC_value = %s\n", mac_value);    /* Print MAC value (HEX-9)    */
  }

  /* Set up initial values for MAC_Verify call */
  return_code = 0;
  reason_code = 0;
  rule_array_count = 1;                        /* 1 rule array element       */
  memset (rule_array, ' ', sizeof(rule_array));/* Clear rule array           */
  memcpy (rule_array[0], MAC_LENGTH, 8);       /* Rule array element         */
                                               /*  (use default Ciphering    */
                                               /*  Method and Segmenting     */
                                               /*  Control)                  */
  memset (chaining_vector, 0x00, 18);          /* Clear the chaining vector  */

  /* Verify MAC value */
  CSNBMVR (&return_code,
           &reason_code,
           NULL,                               /* exit_data_length           */
           NULL,                               /* exit_data                  */
           macver_key_id,                      /* Output from Key_Generate   */
           &text_length,                       /* Same as for MAC_Generate   */
           text,                               /* Same as for MAC_Generate   */
           &rule_array_count,
           &rule_array[0][0],
           chaining_vector,
           mac_value);                         /* Output from MAC_Generate   */

  /* Check the return/reason codes. Terminate if there is an error.          */
  if (return_code != 0 || reason_code != 0) {
    printf ("MAC_Verify failed: ");              /* Print failing verb      */
    printf ("return_code = %ld, ", return_code);  /* Print return code       */
    printf ("reason_code = %ld.\n", reason_code); /* Print reason code       */
    return;
  }
  else                                         /* No error occurred          */
    printf ("MAC_Verify successful.\n");
}
```

CCA および 4765 コプロセッサーでのスループットの向上

CCA API を使用する場合、ホスト・アプリケーション・プログラムの特性が 4765 のパフォーマンスとスループットに影響します。4765 コプロセッサーで最良のパフォーマンスを得るためには、マルチスレッド化とマルチプロセッシングに基づいて、ならびにデータ暗号化規格 (DES)、公開鍵アルゴリズム (PKA)、および Advanced Encryption Standard (AES) 鍵のキャッシュに基づいてアプリケーションを評価および設計します。

マルチスレッド化とマルチプロセッシング

4765 内で実行される CCA アプリケーションは、複数の CCA 要求を同時に処理することができます。コプロセッサーには、RSA (Rivest-Shamir-Adleman アルゴリズム) エンジン、DES (データ暗号化規格) エンジン、CPU、乱数発生ルーチン、および PCI-X (Peripheral Component Interconnect-X) 通信インターフェースなど、複数の独立したハードウェア・エレメントが含まれています。これらのエレメントはすべて同時に機能して、異なる CCA verb のパーツを処理することができます。コプロセッサーは同時に複数の verb を処理することによってすべてのハードウェア・エレメントをビジー状態にし、全体のシステム・スループットを最大化することができます。

この機能を利用するためには、ホスト・システムは、各 CCA 要求が終了するのを待ってから次の CCA 要求を送信するのではなく、複数の CCA 要求をコプロセッサーに送信する必要があります。複数の要求を送信する一番良い方法は、各スレッドが個別に CCA 要求をコプロセッサーに送信できる、マルチスレッド化されたホスト・アプリケーション・プログラムを設計することです。例えば、Web サーバーは、ネットワークで受信した各要求ごとに新規スレッドを開始できます。これらのスレッドはそれぞれ、他のスレッドが何をしているかに関係なく、必要な暗号要求をコプロセッサーに送信します。マルチスレッド化されたモデルでは、コプロセッサーが十分に活用されることが保証されます。もう 1 つの選択肢は、コプロセッサーを同時にすべてが使用する複数の独立したホスト・アプリケーション・プログラムを持つことです。

DES、PKA、および AES 鍵のキャッシュ

4765 用 CCA ソフトウェアは、最近使用された DES、PKA、および暗号化された (平文テキストでない) AES 鍵のコピーをセキュア・モジュール内のキャッシュに保存します。これらの鍵は、復号と検証の済んだ形式で保管されているため、すぐに使用できます。後の CCA 要求で同じ鍵が再使用されると、4765 はキャッシュされたコピーを使用して、鍵トークンの復号と検証に関連したオーバーヘッドを避けることができます。さらに、保管 PKA 鍵の場合は、このキャッシュにより、内部フラッシュ消去可能プログラマブル読み取り専用メモリー (EPROM) メモリーから鍵を取り出すためのオーバーヘッドを取り除くことができます。

その結果、共通の鍵セットを再使用するアプリケーションは、トランザクションごとに異なる鍵を使用するアプリケーションよりずっと高速で実行することができます。ほとんどの共通アプリケーションは、DES 鍵、PKA 秘密鍵、および暗号化された AES 鍵の共通セットを使用するため、キャッシングはスループットを向上させるのに有効です。PKA 公開鍵と AES クリア・キー (処理オーバーヘッドはほとんどありません) はキャッシュされません。

初期デフォルト・ロール・コマンド

このトピックでは、コプロセッサーの初期化後にほかにアクセス制御データが存在しない場合のデフォルト・ロールの特性について説明します。また、使用可能になっているアクセス制御コマンドもリストします。

初期デフォルト・ロール・コマンドの場合、ロール ID はデフォルトで、認証強度はゼロです。デフォルト・ロールは、一日中いつでも、また一週間毎日有効です。許可される機能は、アクセス制御データのロードに必要な機能のみです。

重要**:** 非認証ユーザーがデフォルト・ロールを使用してアクセス制御データをロードできる場合、暗号モードはセキュアではありません。これらのコマンドは、選択された監視ロールに限定してください。

51 ページの表 6 は、CCA ソフトウェアの最初のロード時、および CCA ノードの初期化時に、デフォルト・ロールで使用可能なアクセス制御コマンドをリストしたものです。

表 6. 初期デフォルト・ロール・コマンド

コード	コマンド名
X'0107'	One-Way Hash, SHA-1
X'0110'	Set Clock
X'0111'	Reinitialize Device
X'0112'	Initialize Access Control System
X'0113'	Change User Profile Expiration Date
X'0114'	Change User Profile Authentication Data
X'0115'	Reset User Profile Logon-Attempt-Failure Count
X'0116'	Read Public Access Control Information
X'0117'	Delete User Profile
X'0118'	Delete Role
X'0119'	Load Function-Control Vector
X'011A'	Clear Function-Control Vector

機械可読ログの内容

CLU ユーティリティーは 2 つのログ・ファイルを作成します。1 つのログ・ファイルは読み取り用で、もう一方はプログラムへの入力用の可能性があるログ・ファイルです。

機械可読 (MRL) ログ・ファイルには、コプロセッサーにサブミットされた各種コマンドに応答した、コプロセッサーからのバイナリー出力が含まれます。

MRL の内容に関する詳細情報は、IBM 4764 および IBM 4765 開発チームから入手できます。IBM 製品 Web サイト (*http://www.ibm.com/security/cryptocards*) で「Support and downloads」タブを使用して、IBM にお問い合わせください。

デバイス・ドライバー・エラー・コード

コプロセッサーのデバイス・ドライバーは、コプロセッサーおよびコプロセッサー・ハードウェア状況レジスターとの通信の状況をモニターします。

コプロセッサーがリセットされるたび、そしてそのリセット原因が障害または改ざんイベントでない場合、コプロセッサーは miniboot、パワーオン自己診断テスト (POST)、コード・ローディング、および状況ルーチンを実行します。このプロセスの間、コプロセッサーはホスト・システムのデバイス・ドライバーとの調整を試みます。コプロセッサーのリセット操作は、パワーオン、デバイス・ドライバーから送信された **reset** コマンド、またはコード更新の完了などのコプロセッサーの内部アクティビティーが原因で発生する可能性があります。

コプロセッサーの障害または改ざん検出回路も、コプロセッサーをリセットする場合があります。

Coprocessor Load Utility (CLU) および CCA サポート・プログラムなどのプログラムは、デバイス・ドライバーから 4 バイトの戻りコードの形式で異常状況を受信することができます。

可能な 4 バイト・コードの形式は、X'8xxxxxx' です。よく取得されるコードについては、 52 ページの表 7 で説明します。XX'8340xxxx' または X'8440xxxx' の形式のコードを検出し、そのコードが表に含まれていない場合は、IBM 製品 Web サイト (*http://www.ibm.com/security/cryptocards*) のサポート・ページから E メールで IBM 暗号チームまでご連絡ください。

表 7. *X'8xxxxxxx'* クラスのデバイス・クラス・ドライバー・エラー・コード

4 バイトの戻りコード **(16** 進数**)**	理由	説明
8040FFBF	外部侵入	この侵入は、コプロセッサーへのオプションの電気的接続により発生します。この状態はリセットできます。
8040FFDA	バッテリー切れ	バッテリーの電力が使い尽くされたままとなっているか、取り外されました。コプロセッサーはゼロ化され、もはや機能していません。
8040FFDB	X 線改ざん、またはバッテリー切れ	コプロセッサーはゼロ化され、もはや機能していません。
8040FFDF	X 線、またはバッテリー切れ	コプロセッサーはゼロ化され、もはや機能していません。
8040FFEB	温度の改ざん	高温または低温の制限を超えました。コプロセッサーはゼロ化され、もはや機能していません。
8040FFF3	電圧の改ざん	コプロセッサーはゼロ化され、もはや機能していません。
V8040FFF9	メッシュ改ざん	コプロセッサーはゼロ化され、もはや機能していません。
8040FFFB	リセット・ビットがオン	低電圧が検出されたか、コプロセッサーの内部動作温度が制限を超えたか、ホスト・ドライバーがリセット・コマンドを送信しました。コプロセッサーを PCI-X バスから取り外して、再挿入を試行します。
8040FFFE	バッテリー警告	バッテリー残量がわずかです。バッテリーの交換手順については、「*IBM 4764 PCI-X Cryptographic Coprocessor Installation Manual*」を参照してください。
804xxxxx (例えば、**80400005)**	一般的な通信問題	前の X'8040xxxx' コード以外に、ホストとコプロセッサー間の通信でさらなる状態が発生します。ホスト・システムに実際にコプロセッサーが存在することを判別します。コプロセッサーを PCI-X バスから取り外して、再挿入を試行します。CLU 状況コマンド (ST) を実行します。問題が解決しない場合は、IBM 製品 Web サイト (*http://www.ibm.com/security/cryptocards*) のサポート・ページから E メールで IBM 暗号チームまでご連絡ください。
8340xxxx	Miniboot-0 コード	このクラスの戻りコードは、最も低いレベルのリセット・テストが原因で発生する。このクラスのコードが発生する場合は、IBM 製品 Web サイト (*http://www.ibm.com/security/cryptocards*) のサポート・ページから E メールで IBM 暗号チームまでご連絡ください。
8340038F	乱数生成障害	乱数発生ルーチンの継続モニターで、問題のある可能性が検出されました。実際に進行中の問題を示さずにこのイベントが発生する可能性は、統計的に非常に低いです。 CLU 状況 (ST) コマンドを少なくとも 2 回実行して、この条件を解消できるかどうか判別してください。
8440xxxx	Miniboot-1 コード	このクラスの戻りコードは、置き換え可能な POST およびコード・ローディング・コードが原因で発生します。
844006B2	無効な署名	miniboot が、CLU ユーティリティーから miniboot に送信されたデータの署名を検証できませんでした。正しいファイルを使用していることを確認してください (例えば、**CR1**xxxxx.clu に対して **CE1**xxxxx.clu など)。問題が解決しない場合は、CLU 状況レポートの出力を取得し、実行したいタスクの説明と一緒にそのレポートを、IBM 製品 Web サイト (*http://www.ibm.com/security/cryptocards*) のサポート・ページから E メールで IBM 暗号チームに転送してください。

マスター・キーの複製

このセクションでは、マスター・キーを複製する方法を説明し、複製時のアクセス制御に関する考慮事項を示します。

マスター・キーの複製の概要

複製手順では、暗号化ノード管理 (CNM) ユーティリティーを使用して 1 つのコプロセッサーから別のコプロセッサーにマスター・キーを複製する方法を概説します。

注**:** CNM ユーティリティーが 、複製手順に使用するすべてのシステムで同一レベルになっていることを確認してください。

マスター・キーの複製手順では、次の目的に使用されるコプロセッサーがどのサーバーに含まれているかについて何の想定も行っていません。

- Share 管理 (**SA** ノード)
- マスター・キー・ソース (**CSS** coprocessor share-signing ノード)
- マスター・キー・ターゲット (**CSR** coprocessor share-receiving ノード)

注**:** AES マスター・キーの複製はサポートされていません。

SA 鍵は、CSS 鍵または CSR 鍵のいずれかと同じコプロセッサーに常駐するか、別のコプロセッサー・ノードに常駐できます。CCA のロードされた複数のコプロセッサーがある場合、コプロセッサーは一緒に同じサーバーに常駐することができます。

オペレーターのログオンおよびログオフするアクションは、インストール環境で使用されている特定のロールに依存するため、この手順では無視されます。1 台のサーバー内で複数のコプロセッサーを使用している場合は、コプロセッサー間を切り替えることができます。

手順は、表 8 で概説する複数のフェーズに分割されます。

表 *8.* マスター・キーの複製手順フェーズの概要

フェーズ	ノード	タスク
1	SA	Share 管理ノードを確立します。SA データベースを作成し、SA 鍵を生成して、その公開鍵とハッシュを SA データベースに保管します。
2a	ソース	ソース・ノードを確立します。CSS 鍵を生成して、公開鍵を SA データベースに追加します。SA 公開鍵をインストールします。
2b	SA	CSS 鍵を認証して、証明書を SA データベースに保管します。
各ターゲット・ノードごとに、フェーズ 3 の手順を繰り返してください。		
3a	ターゲット	ターゲット・ノードを確立します。CSR データベースを作成し、CSR 鍵を生成し、公開鍵をこのノードの CSR データベースに追加します。SA 公開鍵をインストールします。
3b	SA	CSR 鍵を認証し、証明書をターゲット・ノードの CSR データベースに保管します。
3c	ソース	Share と現行のマスター・キー検証情報を取得します。
3d	ターゲット	Share をインストールして、新規マスター・キーを確認します。マスター・キーを設定します。

マスター・キーの複製手順を開始する前に、 54 ページの表 9 および 55 ページの図 8 にあるフォームの記入を完了することをお勧めします。

表 9. 複製の責任、プロファイル、およびロール

タスク	ノード	プロファイル	ロール	責任者
アクセス制御の監査	SA			
SA 鍵の生成	SA			
SA 鍵ハッシュの登録	SA			
SA 鍵の登録	SA			
アクセス制御の監査	CSS			
CSS 鍵の生成	CSS			
CSS マスター・キーの取得	CSS			
SA 鍵ハッシュの登録	CSS			
SA 鍵の登録	CSS			
CSS 鍵の認証	SA			
アクセス制御の監査	CSR1			
CSR 鍵の生成	CSR1			
SA 鍵ハッシュの登録	CSR1			
SA 鍵の登録	CSR1			
CSR1 鍵の認証	SA			
Share の取得	CSS			
Share のインストール	CSR1			
新規 CSR の検証	CSR1			
CSR マスター・キーの設定	CSR1			
アクセス制御の監査	CSR2			
CSR 鍵の生成	CSR2			
SA 鍵ハッシュの登録	CSR2			
SA 鍵の登録	CSR2			
CSR2 鍵の認証	SA			
Share の取得	CSS			
Share のインストール	CSR2			
新規 CSR の検証	CSR2			
CSR マスター・キーの設定	CSR2			

ノード情報

ノード	マシン	セレクター番号	コプロセッサーのシリアル番号	データベースのパスと名前
SA ノード制御				(sa.db)
CSS ノードソース				(sa.db)
CSR ノードターゲット 1				(csr1.db)
CSR ノードターゲット 2				(csr2.db)

SA 鍵ハッシュ

Share の数

最小: “m"	最大: “n"

Share の配布

取得元:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CSR-1 にインストール:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

取得元:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CSR-2 にインストール:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

図 *8.* 複製情報ワークシート

マスター・キーの複製のためのフェーズ 1: Share 管理ノードの確立

コプロセッサーを Share 管理 (SA) ノードとして使用するには、 56 ページの表 10 で記述されているマスター・キーの複製の手順に従います。このコプロセッサーはマスター・キー・ソース・ノードまたはマスター・キー・ターゲット・ノードとしても機能している場合があります。

前提条件**:** この手順を実行する前に、セクション 24 ページの『シナリオ: DES または PKA マスター・キーの複製』に記載されたステップと、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」マニュアルのマスター・キーの概要と管理に関する章を熟知しておいてください。

SA ノードを確立するには、以下の表の手順を実行してください。

表 *10.* マスター・キーの複製手順*: SA* ノードの確立

フェーズ	タスク	✓
1.1	アクセス制御の妥当性を監査します。	
1.2	時刻の同期を実行し、権限 (`fcv_td4kECC521.crt`) がインストールされていることを確認します。	
1.3	マスター・キーを確認 (またはインストール) します。	
1.4	オペレーティング・システムの機能を使用して、SA データベース・メディアから前の SA データベースをすべて削除します。	
1.5	環境 ID (EID) がまだ確立されていない場合は、以下の手順を実行して入力します。 • 「暗号化ノード **(Crypto Node)**」>「環境 **ID** の設定 **(Set environment ID)**」をクリックします。 • EID を入力し、「ロード **(Load)**」をクリックします。	
1.6	SA 鍵を生成します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の作成 **(Create Keys)**」>「**Share** 管理鍵 **(Share Administration Key)**」をクリックします。 • デフォルトの SA 公開鍵と秘密鍵のラベルを受け入れて、SA データベース (`sa.db`) のロケーションを入力します。 • 「作成 **(Create)**」をクリックします。 • この手順の後で使用するために SA 鍵ハッシュの値を記録します。	
1.7	SA 公開鍵ハッシュを登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の作成 **(Create Keys)**」>「**Share** 管理鍵 **(Share Administration Key)**」>「**Share** 管理鍵の登録 **(Register Share Administration Key)**」>「**SA** 鍵ハッシュ **(SA Key Hash)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力します (またはデフォルトを受け入れます)。 • SA 鍵ハッシュを入力し、「登録 **(Register)**」をクリックします。	
1.8	SA 公開鍵を登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の作成 **(Create Keys)**」>「**Share** 管理鍵 **(Share Administration Key)**」>「**Share** 管理鍵の登録 **(Register Share Administration Key)**」>「**SA** 鍵ハッシュ **(SA Key Hash)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力し (またはデフォルトを受け入れて)、「登録 (Register)」をクリックします。	

マスター・キーの複製のためのフェーズ 2: ソース・ノードの確立

マスター・キーのソース・ノードとして指定されたコプロセッサーを使用して、 57 ページの表 11 で説明されているマスター・キーの複製のためのステップに従ってください。このコプロセッサーは SA ノードとしても機能している場合があります。

表 11. マスター・キーの複製: ソース (CSS) ノードの確立

フェーズ	タスク	✓
2a.1	アクセス制御の妥当性を監査します。	
2a.2	時刻の同期を実行し、`fcv_td4kECC521.crt` 権限がインストールされていることを確認します。	
2a.3	コプロセッサーのシリアル番号を確認します。 • 「暗号化ノード **(Crypto Node)**」>「状況 **(Status)**」をクリックします。 • 「アダプター **(Adapter)**」をクリックします。 • コプロセッサーのシリアル番号をメモして、「取消 **(Cancel)**」をクリックします。	
2a.4	マスター・キーを確認 (またはインストール) します。	
2a.5	現行のマスター・キー検証情報を取得します。 • 「マスター・キー **(Master Key)**」>「検証 **(Verify)**」>「現行 **(Current)**」をクリックします。 • トランスポート・メディアに「保存 **(Save)**」して、「取消 **(Cancel)**」をクリックします。	
2a.6	環境 ID (EID) がまだ確立されていない場合、入力します。 • 「暗号化ノード **(Crypto Node)**」>「環境 **ID** の設定 **(Set environment ID)**」をクリックします。 • EID を入力し、「ロード **(Load)**」をクリックします。	
2a.7	まだ確立されていない場合は、Share 数の m 値と n 値を設定します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** の数の設定 **(Set Number of Shares)**」をクリックします。 • 必要な Share の最大数と最小数を設定して、「ロード **(Load)**」をクリックします。	
2a.8	CSS 鍵を生成します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の作成 **(Create Keys)**」>「**CSS** 鍵 **(CSS Key)**」をクリックします。 • CSS 鍵ラベル (例えば、CSS.KEY) を入力します。 • コプロセッサーのシリアル番号を確認します。 • SA データベース名とロケーションを確認または入力します。 • 「作成 **(Create)**」をクリックします。	
2a.9	SA 公開鍵ハッシュを登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** 管理鍵の登録 **(Register Share Administration Key)**」>「**SA** 鍵ハッシュ **(SA Key Hash)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力します (またはデフォルトを受け入れます)。 • SA 鍵ハッシュを入力し、「登録 **(Register)**」をクリックします。	
2a.10	SA 公開鍵を登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** 管理鍵の登録 **(Register Share Administration Key)**」>「**SA** 鍵 **(SA Key)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力し (またはデフォルトを受け入れて)、「登録 **(Register)**」をクリックします。	

マスター・キーの複製のためのフェーズ 3: ターゲット・ノードの確立とマスター・キーの複製

指定されたノードを使用して、 58 ページの表 12 で説明されているマスター・キーの複製ステップに従ってターゲット・ノードを確立し、マスター・キーを複製します。このコプロセッサーは SA ノードとして

も機能している場合があります。

表 *12.* マスター・キーの複製: *CSR* ノードの確立とマスター・キーの複製

フェーズ	ノード	タスク	✓
ターゲット・ノードで			
3a.1	ターゲット	アクセス制御の妥当性を監査します。	
3a.2	ターゲット	時刻の同期を実行し、`fcv_td2k.crt` 権限がインストールされていることを確認します。	
3a.3	ターゲット	コプロセッサーのシリアル番号を確認します。 • 「暗号化ノード **(Crypto Node)**」>「状況 **(Status)**」をクリックします。 • 「アダプター **(Adapter)**」をクリックします。 • コプロセッサーのシリアル番号をメモして、「取消 **(Cancel)**」をクリックします。	
3a.4	ターゲット	(一時) マスター・キーが間違いなく存在していることを確認します。	
3a.5	ターゲット	環境 ID (EID) がまだ確立されていない場合、入力します。 • 「暗号化ノード **(Crypto Node)**」>「環境 **ID** の設定 **(Set environment ID)**」>「暗号化ノード **(Crypto Node)**」をクリックします。 • EID (例えば、CSR1 NODE) を入力し、スペースで 16 個の入力文字まで拡張します。 • 「ロード **(Load)**」をクリックします。	
3a.6	ターゲット	まだ確立されていない場合は、Share 数の *m* 値と *n* 値を設定します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** の数の設定 **(Set Number of Shares)**」をクリックします。 • 必要な Share の最大数と最小数を設定します。 • 「ロード **(Load)**」をクリックします。	
3a.7	ターゲット	オペレーティング・システムの機能を使用して、`csr.db` データ・ファイルを消去します。	
3a.8	ターゲット	CSR 鍵を生成します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の作成 **(Create Keys)**」>「**CSR** 鍵 **(CSR Key)**」をクリックします。 • CSR 鍵ラベル (例えば、CSR1.KEY) を入力します。 • コプロセッサーのシリアル番号を確認します。 • 鍵サイズを選択します。 • CSR データベースの名前とロケーション (例えば、CSR1.DB) を提供します。 • 「作成 **(Create)**」をクリックします。	
3a.9	ターゲット	SA 公開鍵ハッシュを登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** 管理の登録 **(Register Share Administration)**」>「**SA** 鍵ハッシュ **(SA-Key Hash)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力します (またはデフォルトを受け入れます)。 • SA 鍵ハッシュを入力し、「登録 **(Register)**」をクリックします。	

表 12. マスター・キーの複製: *CSR* ノードの確立とマスター・キーの複製 (続き)

フェーズ	ノード	タスク	✓
3a.10	ターゲット	SA 公開鍵を登録します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** 管理の登録 **(Register Share Administration)**」>「**SA** 鍵 **(SA Key)**」をクリックします。 • SA データベース・ファイル名とロケーションを入力し、「次へ **(Next)**」を選択します。 • SA 公開鍵ラベルを入力し (またはデフォルトを受け入れて)、「登録 **(Register)**」をクリックします。	
SA ノードで			
3b.1	SA	CSS 鍵を認証します (必要な場合)。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の認証 **(Certify Keys)**」>「**CSS** 鍵 **(CSS Key)**」をクリックします。 • SA データベースの名前とパスを入力し、「次へ **(Next)**」をクリックします。 • CSS 鍵ラベル、コプロセッサーのシリアル番号、および SA 環境 ID を確認します。 • 「認証 **(Certify)**」をクリックします。	
3b.2	SA	CSR 鍵を認証します。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「鍵の認証 **(Certify Keys)**」>「**CSS** 鍵 **(CSS Key)**」をクリックします。 • SA データベースと CSR データベースの名前とパスを入力し、「次へ **(Next)**」をクリックします。 • SA 鍵ラベル、CSR 鍵ラベル、および SA 環境 ID を確認します。 • CSR シリアル番号を入力します。 • 「認証 **(Certify)**」をクリックします。	
ソース・ノードで			
3c.1	ソース	少なくとも m 個および n 個の Share を取得します。Share ごとに以下のサブステップを実行します。各 Share を取得するためにログオンとログオフが必要な場合がありますので注意してください。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** の取得 **(Get Share)**」をクリックします。 • Share を選択します。Share の追加セットを取得している場合は、「配布済み (Distributed)」メッセージは無意味な可能性がありますので注意してください。 • SA データベースと CSR データベースの名前とパスを入力し、「次へ **(Next)**」をクリックします。 • CSS 鍵ラベル、CSS コプロセッサーのシリアル番号、および CSR コプロセッサーのシリアル番号を確認します。 • 「**Share** の取得 **(Get Share)**」をクリックします。 必要に応じて繰り返してください。	
ターゲット・ノードで			

表 12. マスター・キーの複製: *CSR* ノードの確立とマスター・キーの複製 (続き)

フェーズ	ノード	タスク	✓
3d.1	ターゲット	m 個および n 個の Share をインストールします。Share ごとに以下を実行して、応答を監視します。応答は、いつ新規マスター・キーを形成するのに十分な Share がインストールされたかを示します。各 Share をインストールするためにログオンとログオフが必要な場合があることに注意してください。 • 「暗号化ノード **(Crypto Node)**」>「**Share** 管理 **(Share Administration)**」>「**Share** のロード **(Load Share)**」をクリックします。 • Share を選択します。 • CSR データベースと SA データベースの名前とパスを入力して、「次へ **(Next)**」を選択します。 • CSS 鍵ラベル、CSS コプロセッサーのシリアル番号、および CSR コプロセッサーのシリアル番号を確認します。 • 「**Share** のロード **(Load Share)**」をクリックします。 応答を監視します。十分な Share をロードすると、新規マスター・キーが完了します。 必要に応じて繰り返してください。	
3d.2	ターゲット	新規マスター・キーを確認します。 • 「マスター・キー **(Master Key)**」>「検証 **(Verify)**」>「新規 **(New)**」をクリックします。 • 「比較 **(Compare)**」をクリックするか、ファイルを選択するか、「**OK**」をクリックするか、「取消 **(Cancel)**」をクリックします。	
3d.3	ターゲット	`csr.db` データ・ファイルを削除します。これはセキュリティーの問題ではなく、マスター・キーの複製操作中の混乱を避けるためのものです。	
3d.4	ターゲット	必要に応じて、マスター・キーを設定します。 • 「マスター・キー **(Master Key)**」>「設定 **(Set)**」をクリックします。 • 「**OK**」をクリックします。	

複製時のアクセス制御に関する考慮事項

複製操作に関して検討が必要な 3 種類のロールがあります。

- Share 管理 (SA) ノードでのロール
- ソース・ノードでのロール: Coprocessor Share Signing (CSS) ノード
- ターゲット・ノードでのロール: Coprocessor Share Signing (CSS) ノード

セキュリティー・ポリシーでは、以下の権限をどのユーザーがもつかを定義する必要があります。

- ソース・ノードでランダム・マスター・キーを生成する。
- マスター・キーを設定する。新規マスター・キーを動作させるためのアクション。マスター・キーを変更した場合は、そのマスター・キーによって暗号化された鍵を更新する必要があります。
- 保存 Rivest-Shamir-Adleman (RSA) 鍵を生成してソース・ノードとターゲット・ノードの公開鍵 (SA 鍵) を認証し、ソース (CSS) ノードとターゲット (CSR) ノードで保存鍵を生成する。
- SA 鍵とハッシュを登録して、これが分割責任になるかどうかを判別する。

このほかに、マスター・キーの複製に何台のノードの協力が必要かを決定する必要があります。これは、共謀を防ぐために当然選択する必要があります。

m 値と n 値を決定する場合は、いつ複製を実行するか、および (Share 破損、あるいは Share の取得またはインストールが可能な個人が 1 人以上不在であるなどの理由で) ソース・ノードから取得した合計数より少ない数の Share からマスター・キーを再設定しなければならないかどうかを検討します。

注: 暗号化ノード管理 (CNM) ユーティリティーは、ノードからすべての Share を csr.db ファイルに入れます。各 Share は、固有の、3 倍長のデータ暗号化規格 (DES) 鍵 (これ自体、ターゲット・ノードの CSR 公開鍵によって暗号化されている) の下で暗号化されます。

表 13 には、複製に関連したロールに適用されるアクセス権を選択するためのガイダンスが記載されています。

表 13. マスター・キーの複製に関連する *CCA* コマンド

コード	コマンド名	**verb** 名	考慮事項
X'001A'	Set Master Key (マスター・キーの設定)	**Master_Key_Process**	重大。このロールは、新規マスター・キー・レジスターの内容とマスター・キーの変更の影響を認識している必要があります。
X'001D'	Compute Verification Pattern (検証パターンの計算)	**Many**	すべて
X'0020'	Generate Random Master Key (ランダム・マスター・キーの生成)	**Master_Key_Process**	これによって新規マスター・キー・レジスターが埋められることを除き、重大ではありません。
X'0032'	Clear New Master Key Register (新規マスター・キー・レジスターのクリア)	**Master_Key_Process**	このロールは、マスター・キーを設定できるロールに割り当てられます。このロールは、収集された Share をオーバーライドできます。「Generate Random Master Key (ランダム・マスター・キーの生成)」コマンドと同時に使用することはできません。
X'0033'	Clear Old Master Key Register (古いマスター・キー・レジスターのクリア)	**Master_Key_Process**	一般に、使用されません。
X'008E'	Generate Key (鍵の生成)	**Key_Generate** **Random_Number_Generate**	すべて
X'0090'	Reencipher to Current Master Key (現行のマスター・キーへの再暗号化)	**Key_Token_Change**	このロールは、マスター・キーによって暗号化された、機能している鍵を誰が更新するかによって異なります。
X'0100'	PKA96 Digital Signature Generate (PKA96 デジタル署名の生成)	**Digital_Signature_Generate**	このロールは SA、CSS、および CSR 鍵を認証します。
X'0101'	PKA96 Digital Signature Verify (PKA96 デジタル署名の検証)	**Digital_Signature_Verify**	すべて
X'0102'	PKA96 Key Token Change (PKA96 鍵トークンの変更)	**PKA_Key_Token_Change**	このロールは、マスター・キーによって暗号化された、機能している鍵を誰が更新するかによって異なります。
X'0103'	PKA96 PKA Key Generate (PKA96 PKA 鍵の生成)	**PKA_Key_Generate**	このロールは SA、CSS、および CSR 鍵を生成するために必要です。
X'0107'	One-Way Hash, SHA-1 (片方向ハッシュ、SHA-1)	**One_Way_Hash**	すべて

表 13. マスター・キーの複製に関連する CCA コマンド (続き)

コード	コマンド名	verb 名	考慮事項
X'0114'	Change User Profile Authentication Data (ユーザー・プロファイル認証データの変更)	**Access_Control_Initialization**	このロールは、すべてのプロファイル内のパスフレーズの変更を許可します。慎重に使用してください。
X'0116'	Read Public access control Information (公開アクセス制御情報の読み取り)	**Access_Control_Maintenance**	すべて
X'011C'	Set EID (EID の設定)	**Cryptographic_Facility_Control**	このロールは CSS ノードと CSR ノードをセットアップするために必要です。
X'011D'	Initialize Master Key Cloning (マスター・キーの複製の初期化)	**Cryptographic_Facility_Control**	このロールは CSS ノードと CSR ノードに m-of-n 値をセットアップするために必要です。
X'0200'	PKA Register Public Key Hash (PKA 公開鍵ハッシュの登録)	**PKA_Public_Key_Hash_Register**	このロールは CSS ノードと CSR ノードで、SA 鍵が認識可能なことを保証するために使用する必要があります。X'0201' との分割責任。
X'0201'	PKA Public Key Register (PKA 公開鍵レジスター)	**PKA_Public_Key_Register**	このロールは CSS ノードと CSR ノードで、SA 鍵が認識可能なことを保証するために使用する必要があります。X'0200' との分割責任。
X'0203'	Delete Retained Key (保存鍵の削除)	**Retained_Key_Delete**	このロールは、差し替え済みの SA、CSS、および CSR 鍵を除去するために使用されます。サービス妨害に注意してください。
X'0204'	PKA Clone Key Generate (複製鍵の生成)	**PKA_Key_Generate**	このロールは CSS および CSR 鍵を生成するために必要です。
X'0211' - X'021F'	Clone-info (Share) Obtain (複製情報 (Share) の取得)	**Master_Key_Distribution**	このロールは、分割責任を実施するために、各 Share のプロファイルとロールを割り当てます。
X'0221' - X'022F'	Clone-info (Share) Install (複製情報 (Share) のインストール)	**Master_Key_Distribution**	このロールは、分割責任を実施するために、各 Share のプロファイルとロールを割り当てます。
X'0230'	List Retained Key (保存鍵のリスト)	**Retained_Key_List**	すべて

デジタル署名サーバーへの脅威に関する考慮事項

IBM 4765 を IBM Common Cryptographic Architecture (CCA) サポート・プログラムと一緒に、デジタル署名アプリケーションにデプロイする際のさまざまな脅威について検討します。議論の多くは、コプロセッサーを適用する可能性のあるほかの環境に適用できます。

認証局 (CA)、登録局 (RA)、Online Certificate Status Protocol (OCSP) レスポンダー、または操作へのタイム・スタンプ・サービスを実施する組織は、さまざまな脅威に対してどのような取り組みがインストール環境で行われているかを検討する必要があります。 63 ページの表 14 では、潜在的な脅威をリストし、これらの多くの脅威に対する製品設計および実装ソリューションを提示しています。注には、問題にさらされている弱点をさらに軽減するために検討すべきステップが記載されています。

コプロセッサーのデプロイ時に使用できるアクション、考慮すべきポリシー、組み込まれるアプリケーション機能については、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors* 」マニュアルを参照してください。

インストール環境の初期の決定を行った後、表 14 の内容をお読みください。

表 *14.* デジタル署名サーバーへの脅威に関する考慮事項

脅威に関する議論	脅威の軽減
コプロセッサーへの物理的攻撃に関連する脅威	
コプロセッサーの物理的プローブ 攻撃者が、コプロセッサーの物理的プローブを実行して、設計情報や運用内容を明らかにする可能性があります。そのようなプローブには電子的機能が含まれる場合がありますが、それにはコプロセッサー内部機能との直接接触が必要なため、ここでは物理的プローブと呼びます。物理的プローブでは 、IC 障害分析および IC リバース・エンジニアリングの取り組みで一般に用いられる手法を使用したコプロセッサーからのデータの読み取りを伴う場合があります。攻撃者の目的は、ハードウェアのセキュリティー・メカニズム、アクセス制御メカニズム、認証システム、データ保護システム、メモリー・パーティショニング、または暗号化プログラムなどの設計詳細を識別することです。また、初期化データ、パスワード、PIN、または暗号鍵などのソフトウェア設計の判別も目的である可能性があります。	コプロセッサーの電子技術には、高度なアクティブ改ざん検出センサーと応答メカニズムのセットが組み込まれています。高温と低温、電圧レベルとシーケンス制御 、放射、および物理的侵入のセンサーが、異常な環境状態を検知するように設計されています。 すべての機密電子機器は、物理的にシールドされたパッケージに入れられています。潜在的な改ざんイベントを検出すると、コプロセッサーはすぐにすべての内部 RAM メモリーを消去します。これにより、フラッシュ・メモリーから機密の永続データをリカバリーするために使用される鍵のゼロ化も行われます。また、独立した状態コントローラーもリセットされ、コプロセッサーがもはや工場認証状態でないことを示します。 各種の改ざんセンサーは、コプロセッサーの製造時から存続期間の終わりまで電力供給されます。コプロセッサーは照会応答にデジタル署名します。この署名を検証することにより、そのコプロセッサーが本物で、改ざんされていないことを確認することができます。 コプロセッサー内のメインプロセッサーで稼働するほとんどすべてのソフトウェアは Web 上で使用可能なため、リバース・エンジニアリングの対象になります。ただし、コプロセッサーは、攻撃者によって変更されたコードがコプロセッサーにロードされないように、受け入れるよう要求されたコードのデジタル署名を検証します。提供されたコードの検証に使用される公開鍵は、改ざんイベントが認識されると破壊されます。 設計と実装は、FIPS PUB 140-2 レベル 4 規格を使用して、米国国立標準技術研究所 (USA NIST) による独立した評価および認証が行われています。 **注:** コプロセッサーの状態とコード内容の検証が必要です。
コプロセッサーの物理的変更 攻撃者が、設計またはセキュリティーに関連した情報を漏えいするために、物理的にコプロセッサーを変更する可能性があります。この変更は、ハードウェアの障害分析およびリバース・エンジニアリング活動で一般に用いられる手法を使用して達成できる可能性があります。この目的は、ハードウェア・セキュリティー・メカニズム、アクセス制御メカニズム、認証システム、データ保護システム、メモリー・パーティショニング、または暗号化プログラムなどの設計詳細を識別することです。また、初期化データ、パスワード、または暗号鍵などのソフトウェア設計の判別も目的である可能性があります。	機密電子機器はすべて、コプロセッサー上にマウントされた改ざん対応パッケージ内にまとめられています。機密電子機器の変更過程で 、コプロセッサーの工場証明書が破壊され、デバイスは使えなくなります。 **注:** 特定の、シリアル番号の付いたコプロセッサーが使用されていることを確認し、その状況照会応答を監査して、そのコプロセッサーが適切なソフトウェアのロードされた、未変更の IBM コプロセッサーのままであることを確認してください。

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
コプロセッサーの環境操作 攻撃者は、コプロセッサーの仕様環境条件の域を越える環境条件を使用して、不正にコプロセッサーを使用するためにデータまたはプログラム・フローを取得または変更する可能性があります。この変更としては、電気系統やクロック速度を操作したり、コプロセッサーを高温、低温、および放射線にさらしたりといったことが考えられます。その結果、コプロセッサーが、命令が正しく実行されない状態になる可能性があります。その結果、コプロセッサーのセキュリティー要件に反して、セキュリティーに重要なデータが変更されたり、公開されたりする可能性があります。	コプロセッサーには、誤作動を引き起こす可能性のある環境ストレスを検出するためのセンサーが搭載されています。異常条件が発生すると、装置がゼロ化される可能性があります。
置換プロセス コプロセッサーへの要求、またはコプロセッサーからの応答が代替実装環境に誘導されて、攻撃者が結果に影響を与えることが可能になる場合があります。代替実装環境が、異なるセキュリティー機能に置換される可能性があります。例えば、秘密鍵の生成とデジタル署名の作成が、秘密鍵の機密漏れを可能にする代替実装環境で行われる可能性があります。	**注:** 1. 監査員は、署名鍵が間違いなく、適切なコプロセッサー内に保存されているのを確認するように、記載されたプロセスを完了する必要があります。 2. ホスト・システムのセキュリティー手段と正しい動作を信頼できるように、ホスト・システムへのアクセスを監視する必要があります。
コプロセッサーへの論理攻撃に関連する脅威	
フォールトの挿入 攻撃者は、選択したデータを繰り返し挿入してその結果を観察することによって、セキュリティーに重要な情報を割り出す可能性があります。選択した入力を挿入して、その後の変化について出力をモニターするのは、比較的よく知られた暗号装置の攻撃方法です。この目的は、選択した入力に対してコプロセッサーがどのように応答するかに基づいて情報を判別することです。この脅威は、入出力操作に関連した物理的特性であるランダムな選択または操作に対して、意図的で反復した入力データの選択と操作が行われることによって識別されます。	コプロセッサーの電子設計により、スマート・カード攻撃への古典的方法は実行不可能になります。 **注:** 組織は、重要なセキュリティー・ステップとして、論理的かつ物理的に、ホスト・システムの監視とシステムへのアクセスの制御を実行する必要があります。
強制リセット 攻撃者は、選択した操作を不正に終了することにより、強制的にコプロセッサーを無保護状態にする可能性があります。コプロセッサーでの無保護状態の生成は、割り込み機能の挿入やインターフェース機能の改ざんにより、コプロセッサーとホスト間のトランザクションまたは通信を未完了で終了することによって試行される可能性があります。	コプロセッサーは、トラップおよびリセット条件が発生すると、必ず初期パワーオン・シーケンスを実行するように設計されています。アプリケーション・レベルの各要求は、別々の作業単位として扱われ、定義されている単一の初期条件のセットから処理されます。
無効な入力 攻撃者、またはコプロセッサーの許可ユーザーが、無効な入力を注入して、コプロセッサーのセキュリティー機能を危うくする可能性があります。無効な入力は、正しくフォーマットされていない演算、レジスター制限を超える情報の要求、または資料に記載のないコマンドの検出と実行の形式を取ることがあります。そのような攻撃の結果、セキュリティー機能が危険にさらされたり、攻撃に利用できるエラーが演算で生成されたり、保護データが流出したりすることがあります。	トランザクション要求は、呼び出し元のドメインで適用され、コプロセッサーによって検証される認証情報を持っています。各要求は、定義済みの条件をもつ、単一の既知の状態から処理されます。コプロセッサー・ソフトウェアは、それぞれの要求の特性を検証して、誤用シナリオに対処します。

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
データ・ロードの誤動作 攻撃者は、セットアップ・データで故意にエラーを生成して、コプロセッサーのセキュリティー機能を危うくする可能性があります。コプロセッサーへの特殊キー、ロールの ID などをコプロセッサーにロードするを伴う、コプロセッサーの準備段階中に、そのデータ自体が、意図された情報から変更されていたり、破損したりしている可能性があります。いずれの場合も、許可されない方法でコプロセッサーのセキュリティー機能に侵入したり、セキュリティーをあばこうとする試みである可能性があります。	**注:** 監査員の手順で概説しているように、インストール済みのコプロセッサー・ソフトウェアを確認するとともに、アクセス制御のセットアップを確認する必要があります。
無許可のプログラム・ロード 攻撃者は、未許可プログラムを使用して、コプロセッサーのセキュリティー機能の侵入または変更を行う場合があります。未許可プログラムには、通常の運用では使用が意図されていない正当なプログラムの実行や、明確にセキュリティー機能の侵入または変更をターゲットとしたプログラムの無許可ロードなどが含まれる可能性があります。	コプロセッサーは、署名が検証された後に、デジタル署名されたソフトウェアのみを受け入れます。IBM のソフトウェア構築と署名手順の独立評価、およびコプロセッサーの設計は、ロードされたソフトウェアの身元にかける信頼を確約します。 **注:** 監査員は手順に従って、指定されたソフトウェアが使用されていることを確認する必要があります。
アクセスの制御に関連した脅威	
無効なアクセス コプロセッサーのユーザーまたは攻撃者が、ロール・プロファイルに定義されたアクセス権をもたずに、情報またはサービスにアクセスする可能性があります。各ロールには、コプロセッサーの選択されたサービスのみへのアクセスを許可する、定義済みの特権があります。これらの指定されたサービスを越えてアクセスした場合、保護情報が漏えいすることがあります。	監査員は、確立された各ロールに付与されたアクセス権と各ロールに関連付けられたユーザー・プロファイルのセットを確認することができます。コプロセッサー・ソフトウェア実装の独立評価とテストにより、アクセス制御実装の整合性が確認されています。
初回使用での不正 攻撃者が、まだインストールされていない新規のコプロセッサーを無許可で使用することにより、コプロセッサー情報にアクセスする可能性があります。攻撃者は、製造プロセス中、またはその直後にコプロセッサーにアクセスして、不正ソフトウェアをコプロセッサーにロードしたり、お客様に出荷される前の、製造および工場初期化処理中にコプロセッサー内に保管された重要なデータを変更したりする可能性があります。	IBM の製造および物流の慣習により、工場認証以前には、コプロセッサーのエンド・ユーザーは不明であり、未割り当てであることが保証されます。 工場でインストールされたソフトウェアは、デジタル署名の検査により、妥当性検査が行われています。 **注:** 1. 標準のインストール立ち上げプロセスにより、すべてのランタイム・コプロセッサー・ソフトウェアが置換されます。 2. 実動用にコプロセッサー・ソフトウェアをロードする前に、必ずセグメント 2 と 3 が UNOWNED になっていることを確認してください。このアクションにより、後続操作に影響を与える残留データが残っていないことが保証されます。
なりすまし 攻撃者が、コプロセッサーの許可ユーザーを装って、コプロセッサー情報またはサービスにアクセスする可能性があります。コプロセッサーは、必要な認証メカニズム、およびロールが使用を許可されるサービスを含めて、特定のロールを定義する必要があります。攻撃者は、定義されたロール内で操作する許可ユーザーを装って情報にアクセスしようとしたり、許可ユーザーに許されているサービスを実行しようとしたりする可能性があります。	次の 2 つのユーザー・クラスがあります。 1. (IBM) コプロセッサー・コード署名者: コードを作成および署名するための IBM の手順の独立評価は、ユーザーの監査員によって正当なコードと識別され得ることを保証します。 2. CCA アクセス制御設計は、ユーザー・プロセスのドメインからコプロセッサーへの、ユーザーのアクセス制御パスフレーズの整合性と機密性を保護します。正しいパスフレーズとプロファイル ID によって、ロールの使用が許可されます。 **注:** 指定されたユーザーのパスフレーズがセキュアであることを保証するために、ホスト・システムのセキュリティー・ポリシー、ホスト・システムのアプリケーション設計ポリシー、および管理ポリシーが必要です。
予期しない相互作用に関連した脅威	

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
不許可アプリケーション機能の使用 攻撃者が、アプリケーション間の対話を利用して、機密のコプロセッサー・データまたはユーザー・データをあばく可能性があります。対話に、実行中の特定のアプリケーションで必要ない、または許可されていないコマンドの実行が含まれている可能性があります。例としては、マスター・キーの管理に関連した機能、あるいは対称暗号方式または金融サービスに関連した機能の使用などがあります。これらの機能が、デジタル署名アプリケーションに必要なコプロセッサー機能に何らかの悪影響を与えることはありません。	コプロセッサー設計には、アクセス制御セットアップの構成が必要です。CCA ソフトウェアは検査されて、必要なコマンドが使用可能になっていない場合は、必ず機能を不許可にするようになっています。 **注:** 1. アクセス制御構成では、操作可能フェーズに必要な機能のみがこのフェーズで呼び出されるように、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」Redbooks 資料の付録 H で説明されている原則に従う必要があります。 2. デジタル署名アプリケーションについて、非常に限られた機能をもつロールのセット、およびデジタル署名に不可欠なものにコプロセッサー機能を制限するセットアップ手順に関するガイドラインを確立してください。 一部のインストール済み環境では、ロールに対して異なるアプローチを実行するか、追加のアプリケーションの機能を検討する (またはその両方を実行する) ことが望ましい場合があります。これらの場合は、ご使用環境への適用可能性について、必ず「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」Redbooks 資料の付録 H にあるガイドラインと確認をレビューしておいてください。
暗号機能に関する脅威	
暗号攻撃 攻撃者が、アルゴリズムへの暗号攻撃、または総当り攻撃により、セキュリティー機能を打破する可能性があります。この攻撃には、署名生成と検証機能、または乱数発生ルーチンのいずれかが含まれる可能性があります。	コプロセッサーでは、確立して標準化された暗号機能が実装されています。 乱数生成の実装は、米国国立標準技術研究所 (USA NIST) およびドイツ情報セキュリティー庁 (German Information Security Agency) (German Bundesamt fur f³r Sicherhert in der Informations Technik または German BSI) によって公開された基準の下でなされた広範な評価によっています。 保管 (retained) 秘密鍵に与えられている機密状態は、独立評価の対象になります。これらの設計および実装ステップは、暗号攻撃に対する保証を提供します。 **注:** デジタル署名サーバーについては、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」Redbooks 資料の付録 H のガイドラインを参照してください。
デジタル署名に関する脅威	
署名データの偽造 攻撃者が、署名者または第三者が検出できないように、コプロセッサーによってデジタル署名されたデータを変更する可能性があります。この攻撃は、セキュア・ハッシュ関数のぜい弱性、署名エンコード方式のぜい弱性、または偽の署名の生成に使用された暗号アルゴリズムのぜい弱性を使用する可能性があります。	コプロセッサーでは、確立して標準化された暗号機能が実装されています。 **注:** 1. 「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」Redbooks 資料の付録 H に記載されている、CCA の使用における予防措置を順守する必要があります。 2. ユーザーは、暗号アルゴリズムの強度およびそれらが使用するプロセスについて (オープン) フォーラムで議論されているぜい弱性に注意する必要があります。

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
署名前のデータの偽造 攻撃者が、コプロセッサー内で署名が生成される前に、コプロセッサーによってデジタル署名されるデータを変更する可能性があります。この攻撃では、コプロセッサーが実際に署名を計算する前に署名のためにコプロセッサーに伝送されたデータを、攻撃者が変更することを可能にする実装環境のぜい弱性が使用される可能性があります。	ユーザー・ホスト・アプリケーション・プロセス・メモリーからの要求には、ハッシュをデジタル署名に取り込む前にコプロセッサーが確認する整合性検査の値が含まれています。 **注:** ユーザーはホスト・システムおよびホスト・アプリケーション・プログラムのセキュリティーを確認して、コプロセッサー内に受け入れられた認証済みハッシュ値が漏えいしておらず、保護されるデータの代表であることを保証する必要があります。
署名機能の悪用 攻撃者が、コプロセッサーの署名作成機能を悪用して、コプロセッサーが署名することになっていないデータに署名する可能性があります。 攻撃者は、デジタル署名を生成する前に実行されるコプロセッサーの許可検査をパスせずに、データをコプロセッサーにサブミットしたり、データに署名させたりしようとする可能性があります。 あるいは代替方法として、攻撃者はコプロセッサー機能を使用したり、コプロセッサー内のデータが変更されるようにコプロセッサーに影響を与えようとしたりして、コプロセッサー内のデータを改ざんしようとする可能性があります。	コプロセッサー・ソフトウェアの独立検査では、以下の事項を確認することが求められます。 • デジタル署名生成サービスでは、ロールに適切なアクセス権が必要である。 • 要求の処理と設計の整合性により、データ変更が回避される。 **注:** 1. コプロセッサーとそのコードの整合性は、コプロセッサー状況照会を検査する監査員によって確認される必要があります。 2. 監査員は、無許可ユーザー をデジタル署名機能の使用から排除する適切なアクセス制御ロールとプロファイルが確立されていることを確認する必要があります。
署名の検証機能の偽造 偽の署名が有効な署名として受け入れられるように、攻撃者が署名の検証の機能を変更する可能性があります。この攻撃では、検証にこの偽の署名が提示されたときにコプロセッサーが成功メッセージを戻すように、署名の検証機能や、検証される署名データの変更が試行される可能性があります。	ここで最も重要な署名検証機能は、コプロセッサーの (Miniboot 内での) コード・ロード・プロセスで発生します。この製品の場合: • Miniboot コードは、制御プログラムおよび (CCA) アプリケーション・プログラムのコードと同様に、コプロセッサーが署名付きコードの署名を検証した場合にのみコプロセッサー内に受け入れられます。 • 工場でロードされた初期 Miniboot コードも、デジタル署名検証の対象となります。 • 署名には、標準化された暗号プロセスが使用されます (SHA-1、RSA、ISO 9796)。 • コードの作成および署名プロセスは、独立検査の対象になります。
秘密 RSA 署名鍵の開示 攻撃者が、秘密 RSA 署名鍵を開示する機能を使用する可能性があります。	独立評価では、CCA サポート・プログラムに、保管秘密鍵の値を出力したり漏えいしたりする機能が含まれていないことを確認することが求められます。認証された評価では、制御プログラムがコプロセッサーの永続ストレージに保存されているデータを出力したり、そのようなストレージを読み取るための下位機能がないことをはっきり示すことが求められます。

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
秘密 RSA 署名鍵の削除 攻撃者が、秘密 RSA 署名鍵の削除機能の使用許可なしで、また物理的にコプロセッサーの改ざんを行わずに、この鍵の削除機能を使用する可能性があります。	独立評価では、保管秘密鍵の削除が次の環境でのみ行われることを確認することが求められます。 1. CCA 制御下で、Retained_Key_Delete verb を使用する 2. コプロセッサー CCA ソフトウェアをロードする* 3. コプロセッサー CCA ソフトウェアを除去する 4. 改ざんイベントを引き起こす **注:** これらの機密漏れに対処するには、以下のアクションを実行します。 1. 保存鍵の削除 (Delete Retained Key) コマンド、X'0203' を選択的に使用可能にする。 2. ホスト・システム・アクセス制御を使用して、CLU の使用を管理する。 3. コプロセッサーへの物理的アクセスを管理する。 * CEXxxxxx.clu などのファイルをコプロセッサー・ソフトウェアに再ロードしても、永続ストレージの内容はゼロになりません。ファイル CNWxxxxx.clu は永続ストレージをゼロ化します。 8 ページの『コプロセッサーへのソフトウェアのロードとアンロード』を参照してください。
情報モニターの脅威	
情報漏えい 攻撃者が、通常使用中にコプロセッサーから漏えいした情報を利用する可能性があります。情報の漏えいは、放射、電力使用量の変動、入出力特性、クロック周波数を通じて、または処理時間要件の変更によって起こる可能性があります。この漏えいは、裏チャネル (Covert Channel) 伝送として解釈される可能性がありますが、それよりも作動パラメーター の測定に関連しています。これは、直接 (接触による) 測定、または放射の測定のいずれかから得られ、その後実行されている特定操作に関連付けられる可能性があります。	情報漏えいを判断するための実用的手段は、民間および政府機関の研究所での進行中の研究テーマです。徹底した防御手段としては、暗号環境へのアクセスの制限、および暗号環境内およびその近くでの専門機器の使用の制限などを挙げることができます。
複数の観察の結合 攻撃者が、リソースまたはサービスの複数の使用を観察し、それらの観察を結合することにより、重大なセキュリティー情報を明らかにする情報を導き出す可能性があります。一定期間にわたってコプロセッサーの多くの使用を観察してそれらを組み合わせたり、さまざまな操作を観察して得た知識を統合したりすることによって、情報が明らかになる可能性があります。攻撃者はこの情報を使用することにより、直接情報を学んだり、コプロセッサーが秘密にしておく必要のある情報をさらに明らかにする可能性のある攻撃を計画したりすることができます。	**注:** 1. 暗号機器の使用を制御する必要があります。さらに、「*IBM CCA Basic Services Reference and Guide for the IBM 4765 PCIe and 4764 PCI-X Cryptographic Coprocessors*」Redbooks 資料の付録 H のガイドラインにも従ってください。 2. 攻撃者は、署名付きデータと署名にアクセスできる可能性があります。したがって、ユーザーが任意の署名要求を実行依頼する能力を制限するための制御を導入する必要があります。 3. 標準化された暗号手順を使用し、プロセス (SHA-1、RSA、ISO 9796、X9.31、HMAC、および triple-DES) のぜい弱性に関する暗号コミュニティーの認識をモニターすることにより、セキュア操作を保証できます。
その他の脅威	

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
被リンク攻撃 攻撃者は連続攻撃を実行する可能性があります。その結果、コプロセッサーが不安定になったり、セキュリティー機能のある局面が低下したりします。それにより、その後の攻撃の実行が成功する可能性があります。環境ストレスの存在するところで入力を操作しながら出力をモニターするというのは、被リンク攻撃の例です。	**注:** 1. 暗号システムの使用は、コプロセッサーのアクセス制御とホスト・システム制御の使用を通じて実施される、許可された状態に制限する必要があります。 2. ホスト・システム制御と組織のポリシーでは、モニタリングのためのシステムへのアクセスと、任意要求の実行依頼を制限する必要があります。
反復攻撃 攻撃者が、検出されない侵入試行を繰り返し使用して、メモリー内容をあばいたり、コプロセッサー内のセキュリティーに重要なエレメントを変更したりする可能性があります。コプロセッサーのセキュリティーへの侵入を対話形式で効果的に開発するために、ここに述べられている他の脅威の一部またはすべてに関連した反復試行が使用される可能性があります。いずれの場合も、これらの攻撃を検出できないでいる場合、ぜい弱性の増大に関する警告は出されません。	**注:** 暗号システムの使用は、コプロセッサーのアクセス制御とホスト・システム制御の使用を通じて実施される、許可された状態に制限する必要があります。ホスト・システム制御と組織のポリシーでは、モニタリングのためのシステムへのアクセスと、任意要求の実行依頼を制限する必要があります。
複製 (クローン作成) 攻撃者が、機能するコプロセッサーの一部またはすべてを複製して、さらなる攻撃の計画を立てる可能性があります。コプロセッサーの一部またはすべての複製に成功するのに必要な情報は、コプロセッサー自体の詳細な検査、または設計情報の不正使用から得られる可能性があります。	**注:** 監査員は、デジタル署名鍵、適切なコード、およびアクセス制御の管理体制が、許可されたコプロセッサーに常駐していることを確認する必要があります。
オペレーティング環境によって対処される脅威	
コプロセッサーの変更と再利用 攻撃者は、情報資産に不正にアクセスできるようにするために、変更されたコプロセッサーを使用してオリジナルのコプロセッサーと見せかける可能性があります。そのコプロセッサーの除去、変更、およびホスト・システムへの再挿入が、そのような組み合わせをオリジナルとして通用させるために使用される可能性があります。その後、これは秘密署名鍵や、保護されているその他のセキュリティーに重要な情報をアクセスしたり変更したりするために使用される可能性があります。	**注:** 1. 監査員は、コプロセッサーの署名した照会応答を調べることにより、デバイスが本物であり、適切なコードがロードされていることを確認します。 2. また監査員は、デジタル署名鍵がコプロセッサー内の保存鍵であることも確認する必要があります。
特権ユーザーによる不正使用 不注意な管理者、故意に怠慢な管理者、悪意のある管理者、あるいはその他の特権ユーザーが、セキュリティー機能や保護データを漏えいするアクションを実行して、コプロセッサーの資産を危険にさらす可能性があります。特権ユーザーまたは管理者が、ここに記載されたいずれかの脅威に基づいて攻撃を直接実装したり、手助けしたりする可能性があります。	**注:** 組織は、暗号システムに対して単一の個人が持つアクセス権を制限するポリシーを確立、実施、および監査する必要があります。セットアップ手順では、必ず単一ユーザーが不正システムを稼働する機会を得られないようにする必要があります。
データの変更 コプロセッサーによって署名されるデータが、正当なユーザーによって承認された後、そのデータが署名のためにコプロセッサーにサブミットされるまでの間に、攻撃者、または作動環境での障害によって変更される可能性があります。署名するよう正当なユーザーによって承認されたデータは、そのデータが正当なユーザーによって承認された後、署名のためにコプロセッサーに転送されるまでの間に、攻撃者、偽または悪意のあるプログラム、あるいは環境エラー (例えば、送信エラー) によって変更される可能性があります。	**注:** そのような攻撃を阻止するためには、ホスト・システムのセキュリティー予防措置、および組織のポリシーが、定義、実施、および監査される必要があります。

表 14. デジタル署名サーバーへの脅威に関する考慮事項 (続き)

脅威に関する議論	脅威の軽減
データの検証 コプロセッサーによって検証される署名データが、署名検証のためにコプロセッサーにサブミットされる前に、コプロセッサーの応答にその署名の妥当性が反映されないように、攻撃者、または作動環境での障害によって変更される可能性があります。ユーザーによってサブミットされた署名データが、検証のためにコプロセッサーに渡される前に、コプロセッサー環境内で変更される可能性があります。その結果、検証の必要なデジタル署名の実際の妥当性を反映しない応答がコプロセッサーから戻される可能性があります。 また、コプロセッサーの応答が、署名検証を要求したユーザーに渡される前に、コプロセッサー環境内で変更される可能性もあります。	コプロセッサーは、コードと特定のコード・ロード・コマンドの署名を検証します。独立評価では、これを迂回できないことを確認することが求められます。 CCA の設計は、コプロセッサーとホスト・システム内の CCA コードの最上層との間の要求と応答の整合性の検証をサポートしています。 **注:** ホスト・システムのセキュリティー手段が、要求の入力と出力の変更のブロッキングに対処している必要があります。

IBM 暗号化コプロセッサーの特記事項

IBM 暗号化コプロセッサーの特記事項には、電子コンポーネントの安全な廃棄のためのガイドラインを提供する 3 つの特記事項があります。

製品のリサイクルと廃棄

この装置には、使用終了時に特別な処理および廃棄を必要とする、鉛や銅/ベリリウム合金が使われている回路ボード、ケーブル、電磁適合性ガスケットやコネクターなどの材料が含まれています。この装置を廃棄する前に、それらの部品を取り外し、該当する規定に従ってリサイクルするか廃棄する必要があります。IBM では、いくつかの国で製品回収プログラムを提供しています。製品リサイクル・オファリングについては、IBM のインターネット・サイト (http://www.ibm.com/ibm/environment/products/prp.shtml) を参照してください。IBM では情報技術 (IT) 機器の所有者に、機器が必要なくなったときに責任を持って機器のリサイクルを行うことをお勧めしています。また、機器の所有者による IT 製品のリサイクルを支援するため、さまざまなプログラムとサービスを提供しています。製品リサイクル・オファリングについては、次の IBM インターネット・サイトを参照してください。

http://www.ibm.com/ibm/environment/products/prp.shtml

注: WEEE マークは EU 諸国とノルウェーにのみ適用されます。この機器には、EU 諸国に対する廃電気電子機器指令 2002/96/EC (WEEE) のラベルが貼られています。この指令は、EU 諸国に適用する使用済み機器の回収とリサイクルの骨子を定めています。このラベルは、使用済みになった時に指令に従って適正な処理をする必要があることを知らせるために種々の製品に貼られています。

バッテリー回収プログラム

この製品には、密封された鉛酸、ニッケル・カドミウム、ニッケル水素、リチウム、またはリチウム・イオンのバッテリーが含まれている場合があります。特定のバッテリー情報については、お手元のユーザー・マニュアルまたはサービス・マニュアルを参照してください。バッテリーは、正しくリサイクルするか廃棄する必要があります。リサイクル施設がお客様の地域にない場合があります。米国以外の国におけるバッテリーの廃棄については、http://www.ibm.com/ibm/environment/products/batteryrecycle.shtml を参照するか、またはお客様の地域の廃棄物処理施設にお問い合わせください。米国では、IBM は、IBM 装置からの使用済みの IBM の密封された鉛酸バッテリー・パック、ニッケル・カドミウム・バッテリー・パック、ニッケル水素バッテリー・パック、その他のバッテリー・パックの再利用、リサイクル、または適切な廃棄

のための回収プロセスを確立してあります。これらのバッテリーの正しい廃棄については、IBM 1-800-426-4333 にお問い合わせください。お問い合わせの前に、バッテリー上に記載されている IBM 部品番号をご用意ください。

IBM 暗号化コプロセッサー・カードの回収プログラム

本マシンには、水銀を含有するポリウレタン材が使用されているオプション・フィーチャー、暗号化コプロセッサー・カードが含まれることがあります。このカードの廃棄にあたっては、地方自治体の条例または規則に従ってください。IBM では、一部の IBM 暗号化コプロセッサー・カードの回収プログラムを用意しています。詳しい情報については、以下を参照してください。

http:/www.ibm.com/ibm/environment/products/prp.shtml

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの websites サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年).

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらのCookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、『IBM オンラインでのプライバシー・ステートメントのハイライト』(http://www.ibm.com/privacy/jp/ja/)、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。 現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Windows は、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

［マ行］

［ヤ行］

［ラ行］

［数字］

A

C

D

E

K

M

S

V

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21